Михаэль Кофлер

Linux

полное руководство

Debian
Fedora
openSUSE
Ubuntu

PEARSON
Addison Wesley

ПИТЕР

Michael Kofler

Linux 2010

Debian
Fedora
openSUSE
Ubuntu

Boston San Francisco New York
London Toronto Sydney Tokyo Singapore Madrid
Mexico City Munich Paris Cape Town Hong Kong Montreal

Михаэль Кофлер

Linux

полное руководство

Debian
Fedora
openSUSE
Ubuntu

ПИТЕР®

Москва · Санкт-Петербург · Нижний Новгород · Воронеж
Ростов-на-Дону · Екатеринбург · Самара · Новосибирск
Киев · Харьков · Минск
2011

ББК 32.973.2-018.2
УДК 004.451
К74

Кофлер М.
К74 Linux. Полное руководство. — СПб.: Питер, 2011. — 800 с.: ил.

ISBN 978-5-459-00508-0

Эта книга — перевод девятого издания фундаментального руководства Михаэля Кофлера, уже ставшего классикой.

Михаэль Кофлер открыл путь в мир свободных операционных систем для нескольких поколений пользователей Linux. Журнал Linux-Magazin причисляет его к 15 наиболее влиятельным специалистам в данной области.

Книга представляет собой справочник на тему «Как это делается в Linux», она будет полезна и актуальна для всех, кто хочет работать с Linux на ПК или на сервере.

ББК 32.973.2-018.2
УДК 004.451

ISBN 978-3-8273-2877-9 нем.

978-5-459-00508-0

Краткое содержание

Оглавление

Предисловие

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

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

Об этой книге

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

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

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

В добрый путь!

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

Михаэль Кофлер
www.kofler.cc

От издательства

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

Мы будем рады узнать ваше мнение!

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

1 Виртуализация и Wine

В последние годы популярностью стали пользоваться различные технологии, связанные с виртуализацией. Под виртуализацией понимается создание на компьютере одной или нескольких виртуальных машин, на которых работают различные операционные системы. Существуют разнообразные возможности применения виртуализации.

- **Одновременная работа в Windows и Linux** — никогда не стоит полагаться на одну из операционных систем больше, чем на другую. Благодаря виртуализации можно одновременно использовать две операционные системы на одном и том же компьютере. Исходной системой может быть как Windows, так и Linux. Тогда в определенном окне будет работать та или иная операционная система
- **Компьютер для технической поддержки и разработки** — часто разработчику программ для системы Linux необходимо тестировать плоды своего труда в различных версиях разных дистрибутивов. Схожие требования предъявляются и к сотрудникам службы поддержки: многие вопросы пользователей касаются определенного дистрибутива. Вместо того чтобы каждый раз заново перезапускать систему или задействовать несколько компьютеров, можно установить несколько дистрибутивов операционной системы в одной среде разработки и по мере надобности активизировать или останавливать те или иные системы. Таким образом экономятся время и ресурсы.
- **Виртуализация сервера** — возможности современных серверов далеко не ограничиваются выполнением одной серверной функции. Системы виртуализации позволяют размещать на одной физической машине несколько виртуальных серверов. Эта возможность имеет определяющее значение, так как виртуальные серверы работают полностью независимо друг от друга. Проблема, связанная с безопасностью, или неправильная конфигурация одного сервера не влияют на остальные серверы.

В этой главе значительное внимание уделяется программе *VirtualBox* производства Sun (или Oracle), которая в настоящее время является наилучшим свободно распространяемым решением, предназначенным для виртуализации. С помощью VirtualBox можно задействовать Linux в Windows, Windows в Linux или параллельно работать с несколькими дистрибутивами. Кроме того, будет рассмотрена система виртуализации *KVM/QEMU*, популярность которой неуклонно растет.

Wine. Между прочим, совсем необязательно проделывать массу работы и виртуализировать операционную систему целиком. Если нужно всего лишь запустить

определенную программу для Linux в системе Windows, то на помощь приходит программа *Wine* или ее усовершенствованный коммерческий вариант CrossOver. Сокращение Wine расшифровывается как Wine is not an emulator (Wine — не эмулятор), то есть Wine — никакой не эмулятор, а собрание библиотек, содержащее все самые важные функции из библиотек Windows.

1.1. Основы виртуализации

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

Технологии виртуализации

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

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

- **Полная виртуализация (виртуальные машины, эмуляция).** В данном случае программа имитирует работу виртуального аппаратного обеспечения, то есть компьютера, состоящего из процессора, ОЗУ, жесткого диска, сетевой карты и т. д. Гостевые системы «считают», что виртуальное аппаратное обеспечение является реальным. Чтобы такая система функционировала, работающая на «хозяине» программа виртуализации должна отслеживать код «гостя» и заменять определенные команды другими фрагментами кода. Эту задачу выполняет гипервизор (также называемый монитором виртуальных машин, или сокращенно VMM). Такая программа-гипервизор также отвечает за события, связанные с хранением информации и управлением процессами.

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

  Недостатки: сравнительно медленно работает.

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

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

  Преимущества: высокая эффективность.

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

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

- **(Пара)виртуализация с поддержкой аппаратного обеспечения.** Современные процессоры производства Intel и AMD содержат аппаратные функции, предназначенные для упрощения процессов виртуализации. Intel именует такую технологию Intel-VT (ранее — Vanderpool), а AMD окрестил свои функции AMD-V (ранее — Pacifica).

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

  Недостатки: требуются специальные процессоры.

  Программы/фирмы: KVM, Xen.

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

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

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

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

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

- http://virt.kernelnewbies.org/TechOverview;
- http://wiki.openvz.org/Introduction_to_virtualization.

Поддержка процессора. Чтобы установить, будет ли данный процессор полезен при виртуализации аппаратного обеспечения (Intel-VT или AMD-V), выполните команду `egrep`. Показанный ниже результат выдан двухъядерным процессором

Intel. Если вывод пуст, то ваш процессор не поддерживает виртуализацию либо соответствующая функция была деактивизирована в BIOS.

```
user$ egrep '^flags.*(vmx|svm)' /proc/cpuinfo
flags : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat pse36
        clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx lm constant_tsc pni
        monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr lahf_lm
flags : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat pse36
        clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx lm constant_tsc pni
        monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr lahf_lm
```

Виртуальное аппаратное обеспечение

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

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

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

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

Большинство программ виртуализации позволяют присвоить каждому виртуальному CD или DVD ISO-файл. Тогда «гость», вместо того чтобы пользоваться реальным приводом, обращается к такому файлу. Это исключительно полезно в тех случаях, когда необходимо многократно устанавливать одни и те же программы, и работа протекает эффективно и тихо. При необходимости вы можете без особого труда сами извлечь ISO-файл с CD или DVD (см. также раздел. 3.6).

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

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

Звуковые функции. Большинство программ виртуализации предоставляют гостевой системе виртуальную звуковую карту и перенаправляют звуковой вывод на аудиосистему «хозяина». Если вы не выдвигаете экстраординарных требований

к аудиосистеме (например, вам не требуется эффект «звук вокруг»), то такого механизма вполне достаточно.

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

Виртуальные машины и проблемы сетевых соединений

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

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

- **Сетевой мост** — при применении механизма сетевого моста (bridged networking) «гость» является дополнительным клиентом в локальной сети. Этот вариант является идеальным на тот случай, если в сети (но не на машине-«хозяине») присутствует DHCP-сервер либо если машина-«хозяин» подсоединена к ADSL- или WLAN-маршрутизатору. В таком случае сетевые конфигурации сообщаются гостевым машинам через подобный сервер или маршрутизатор и гостевые машины получают доступ как к локальной сети, так и к Интернету. Если компьютер-«хозяин» оборудован несколькими сетевыми интерфейсами, то следует указать, какой из них будет использоваться для создания сетевого моста (через этот интерфейс будет обеспечиваться соединение с Интернетом).
- **NAT** — при использовании механизма NAT (преобразование сетевых адресов) сама система виртуализации работает по отношению к своим гостевым машинам как DHCP-сервер и выполняет функции маскарадинга (см. также раздел 17.3). Таким образом, гостевые машины могут использовать соединение с Интернетом, настроенное на машине-«хозяине». Доступ к локальной сети в таком случае становится невозможным по причине несовпадения адресных пространств локальной сети и сети NAT системы виртуализации.
- **Сетевая модель «Только хозяин»** — при использовании такого варианта «гость» может обмениваться через сеть информацией только с «хозяином», но не с другими компьютерами, подключенными к данной сети или Интернету. Целесообразно применять такой вариант в тех случаях, когда вы хотите построить на нескольких виртуальных машинах испытательную систему, недоступную извне.

- **Внутренняя сеть** — программа виртуализации создает виртуальную сеть, по которой могут сообщаться только виртуальные машины. При использовании такого варианта виртуальные машины не имеют доступа ни к локальной сети, ни к Интернету.

Обмен данными между «хозяином» и «гостем»

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

Сетевые каталоги

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

Область обмена данными (буфер обмена)

Многие программы виртуализации позволяют обмениваться выделенным текстом через область обмена данными. К сожалению, такая система обычно далека от совершенства и работает с ошибками. Виновата в этом так называемая графическая система X, которая должна отличать буфер обмена, созданный специально для данного фрагмента выделенного текста, и буфер обмена, созданный для текста, скопированного с помощью сочетания клавиш Ctrl+C. Иногда непонятно, какой из этих областей обмена данных пользуется система виртуализации. Кроме того, функциональность буфера обмена ограничивается копированием только текстовой информации. Если вам потребуется быстро скопировать диаграмму Excel с гостевой машины на машину-«хозяина», на которой открыт документ, созданный в редакторе OpenOffice, то вы будете разочарованы.

Программы для виртуализации

Спектр предлагаемых инструментов виртуализации как в коммерческом сегменте, так и среди свободно распространяемого ПО необозримо велик. В следующем перечне коротко поименованы важнейшие флагманы рынка виртуализации. В скобках указано, является ли данный продукт коммерческим или распространяется сво-

бодно, какая фирма занимается разработкой и реализует на рынке соответствующую продукцию.

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

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

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

- **KVM/QEMU (свободно распространяемая программа, Red Hat).** Собственно, KVM — это просто модуль ядра, который радикально ускоряет работу эмулятора QEMU при использовании современных процессоров, при том что раньше этот эмулятор работал достаточно медленно. С тех пор как KVM официально вошел в состав ядра, а Red Hat купил Qumranet — фирму, разработавшую KVM, значение модуля KVM резко выросло и он уже считается стандартным виртуализационным решением в дистрибутивах Fedora, Ubuntu и, конечно же, для версии 6 Red Hat Enterprise Linux. KVM одинаково хорош для применения как на ПК, так и на сервере. И все же по таким показателям, как понятность для пользователей, совместимость и скорость, KVM пока не может конкурировать с аналогичными коммерческими программами — Vmware, VirtualBox и Xen. В качестве системы-«хозяина» поддерживается только Linux.

- **Xen (частично бесплатная программа, производитель — Citrix).** Xen — это гипервизор, работающий без операционной системы. Виртуализированные гостевые системы работают в так называемых доменах (domU), причем первый домен (domU) имеет особые привилегии и в определенном смысле сравним с системой-«хозяином» в других программах виртуализации. Во многих

практических ситуациях Xen значительно эффективнее, чем другие системы виртуализации. Однако в то же время настройка и конфигурирование гостевых систем (доменов) требуют гораздо больших усилий. Это не в последнюю очередь объясняется тем, что расширения ядра, необходимые для правильной работы Xen, очень объемны и, несмотря на все приложенные усилия, пока не входят в состав официальной версии ядра. Если же вы готовы вложить в работу с Xen много времени, то достигнете выдающихся результатов, но для использования от случая к случаю Xen не годится.

- **OpenVZ, Virtuozzo (частично бесплатные программы, производитель — Parallels), а также Linux-VServer (свободно распространяемое ПО).** OpenVZ, базирующийся на его основе коммерческий продукт Virtuozzo и технически сходное виртуализационное решение Linux-VServer позволяют обустраивать много изолированных сред в одном дистрибутиве Linux. OpenVZ или Virtuozzo при работе исходят из того, что в системах «хозяина» и его «гостей» работает одна и та же версия Linux. Эта концепция отлично подходит для тех случаев, когда необходимо виртуализировать несколько (много!) аналогичных серверов. Такая система частично используется провайдерами интернет-хостинга, которые предлагают недорогие виртуальные корневые серверы.
- **Hyper-V (коммерческая программа, Microsoft).** Корпорация Microsoft поначалу не успела поучаствовать в разделе рынка виртуализации, но сейчас прилагает титанические усилия, чтобы сделать собственное виртуализационное решение — Hyper-V — конкурентоспособным. Hyper-V воспринимает систему Windows Server как систему-«хозяина», но при этом может поддерживать Linux в качестве гостевой системы. Компания даже разработала для этой цели собственные драйверы ядра Linux, причем эти драйверы — с открытым кодом (такой шаг дался Microsoft с большим трудом, так как эта корпорация очень долго представляла Стандартную Общественную Лицензию в самом черном свете).

Немного из собственного опыта

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

Когда речь заходит о тестировании реального аппаратного обеспечения, то, разумеется, и возможности VirtualBox исчерпываются достаточно быстро. Информативные испытания скоростных характеристик, пробы современных аппаратных компонентов или параллельная установка нескольких версий Windows и Linux, как

и ранее, проводятся только на реальном оборудовании. Поэтому все дистрибутивы, описанные в данной книге, тестировались не только виртуально, но и на реальных машинах. Роль испытательного полигона сыграли три ноутбука и два ПК.

Межсистемная совместимость

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

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

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

1.2. VirtualBox

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

В общем, функции VirtualBox и рабочей станции VMware весьма сходны, но VirtualBox предоставляется частным пользователям бесплатно, а в некоторых «урезанных» вариантах даже имеет открытый код. Кроме того, система VirtualBox обеспечена технической поддержкой значительно лучше, чем рабочая станция VMware. Каждый год выходит по несколько новых версий VirtualBox, не считая ежемесячных обновлений, поэтому можно быть уверенным, что система будет

совместима и с новейшими доступными на сегодняшний день версиями ядра и X-версиями.

Система VirtualBox сначала принадлежала компании InnoTek, которая разработала вместе с другой фирмой (Connetix) продукт Virtual PC. Корпорация Microsoft приобрела эту виртуализационную программу в 2003 году и по-прежнему предлагает Virtual PC, однако официально поддерживает только виртуализацию для систем Microsoft. С 2004 года VirtualBox и Virtual PC развиваются независимо друг от друга. В феврале InnoTek вошла в состав Sun, и сразу после этого Sun стала частью Oracle (возможно, в связи с этим еще возникнут проблемы антимонопольного плана). Однако, судя по всему, Oracle в дальнейшем останется владельцем VirtualBox. Как это отразится на будущем продукта, неизвестно.

В данном разделе мы поговорим о VirtualBox версии 3.0.8. Актуальная информация и подробная документация об этой системе находится по адресу www.virtualbox.org.

Версии. На сайте VirtualBox можно скачать две различные версии программы.

- **Версия с исходным кодом (Open Source Edition, OSE).** В ней содержится исходный код к VirtualBox, в данном случае используется Стандартная Общедоступная Лицензия (GPL). В рамках указанной лицензии эта версия VirtualBox используется бесплатно. При этом в большинстве дистрибутивов Linux применяются готовые пакеты, имеющие открытый код, благодаря чему установка программы упрощается.
- **Двоичная версия.** Она бесплатно предоставляется для использования в частных целях, в университетах и для испытательных целей. Однако для коммерческого использования обязательно необходимо получить лицензию от Sun/Oracle!

  В двоичной версии по сравнению с версией с открытым кодом содержатся следующие дополнительные функции: вы можете обращаться к виртуальным машинам через USB-устройства и iSCSI-сервер, а также удаленно управлять виртуальными машинами с другого компьютера в сети по протоколу RDP (протокол для удаленных дисплеев).

Установка VirtualBox

Установка пакетов версии OSE в Linux

Как уже было сказано, в большинстве дистрибутивов предлагаются пакеты OSE-версии программы VirtualBox. В Fedora потребуется обратиться к версии источника пакетов rpmfusion. Установка сама по себе также не очень сложна.

В системе-«хозяине» VirtualBox обращается к трем модулям ядра: `vboxdrv`, `vboxnetadp` и `vboxnetfit`. В некоторых дистрибутивах эти модули также поставляются вместе в виде пакета, который обновляется при каждом обновлении ядра. Если такой механизм не задействован, то модули ядра должны заново компилироваться с помощью следующей команды после установки и каждого обновления ядра (VirtualBox указывает на эту команду при каждом запуске виртуальной машины, когда система обнаруживает, что при загрузке в наличии нет модулей ядра):

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

Исходный код для модулей ядра устанавливается вместе с VirtualBox. Однако для компиляции также необходимы C-компиляторы gcc и файлы заголовков ядра. В Ubuntu эти требования выполняются по умолчанию, в других дистрибутивах нужно устанавливать соответствующие пакеты (см. раздел 15.1).

Установка двоичной версии в Linux

Частные пользователи имеют возможность установить не пакеты VirtualBox, поставляемые вместе с дистрибутивом, а двоичную версию virtualbox.org. Такой вариант обладает двумя преимуществами: во-первых, предлагаемая по адресу http://www.virtualbox.org/wiki/Linux_Downloads версия почти всегда новее той, что поставляется вместе с дистрибутивом, и во-вторых, она содержит некоторые дополнительные функции, например поддержку USB.

VirtualBox предлагается для загрузки в различных форматах: в виде пакетов RPM или Debian для различных дистрибутивов, а также в виде универсального установщика, который запускается следующим образом:

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

Если вы пользуетесь двоичной версией, то вам в любом случае придется самостоятельно компилировать модули ядра с помощью команды /etc/init.d/vboxdrv setup. Пользователям Ubuntu и Fedora следует установить для VirtualBox пакет dkms. В таком случае DKMS управляет модулями VirtualBox и при обновлении ядра автоматически совершает новую компиляцию (см. раздел 15.1). Однако во многих инсталляциях VirtualBox этот механизм не всегда работает надежно.

Источник пакетов APT

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

```
debhttp://download.virtualbox.org/virtualbox/debianjauntynon-free  (для Ubuntu9.04)
debhttp://download.virtualbox.org/virtualbox/debiankarmicnon-free  (для Ubuntu9.10)
debhttp://download.virtualbox.org/virtualbox/debianlennynon-free  (для Debian5)
```

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

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

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

```
root# apt-get install virtualbox-3.0
```

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

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

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

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

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

Подготовительные работы

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

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

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

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

Установка для Windows

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

К сожалению, не существует готовых скомпилированных версий OSE для Windows. Коммерческие пользователи, не желающие платить лицензионных взносов, вынуждены сами компилировать версию OSE. Однако для этого нужны специальные инструменты разработки (например, Microsoft Visual Studio или Microsoft Visual C++), а также определенные навыки.

Настройка виртуальной машины в Linux

В этом разделе рассказывается, как настроить в рамках VirtualBox виртуальную машину в Linux. При этом не важно, в какой операционной системе будет использоваться VirtualBox: в Windows или Linux.

При настройке виртуальной машины вспомогательные функции выполняет специальный ассистент. После выбора типа операционной системы (например Linux с ядром 2.6n), желаемого объема оперативной памяти и настройки виртуального жесткого диска VirtualBox покажет обобщенную информацию по аппаратным компонентам машины. Здесь при необходимости можно выполнять дальнейшие настройки, например изменить параметры доступа к сети или указать в качестве источника данных для CD/DVD-привода файл ISO (рис. 1.1). Выбор файлов ISO несколько сложен: при нажатии кнопки открывается окно, в котором для начала необходимо добавить нужный файл к списку файлов ISO, известных программе VirtualBox. Только после этого можно выбирать файлы из окна списка.

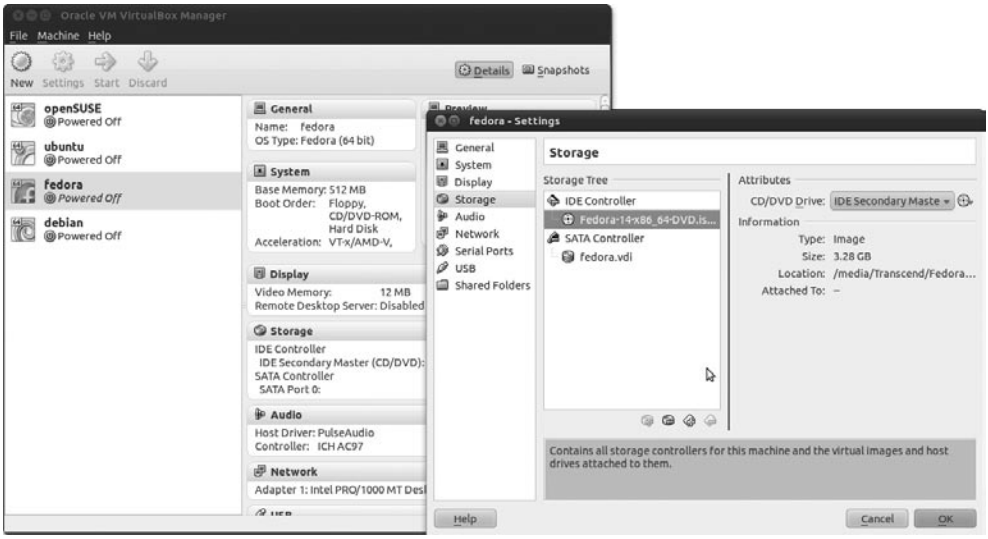

Рис. 1.1. Конфигурация виртуальной машины в VirtualBox

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

Сетевое соединение между «хозяином» и «гостем» по умолчанию осуществляется через NAT. При этом гостевая машина хотя и имеет соединение с Интернетом, однако не может обмениваться никакими данными по локальной сети. В подразделе «Виртуальные машины и проблемы сетевых соединений» раздела 1.1 описаны другие сетевые варианты, которые поддерживаются в VirtualBox. Я использовал такой вариант настройки, при котором машина-«хозяин» была соединена с маршрутизатором ADSL. Для сетевой конфигурации виртуальных машин я применил сетевой мост. С помощью такого моста все виртуальные машины становятся элементами локальной сети, благодаря чему упрощается обмен данными между системой-«хозяином» и гостевыми системами (SSH, NFS, Samba и др.).

Виртуальная машина автоматически оказывается в фокусе для работы с мышью и клавиатурой, когда вы нажимаете определенную клавишу. По умолчанию окно

выводится из фокуса нажатием правой клавиши Ctrl. В основном окне VirtualBox можно задать и другую хост-клавишу: Файл ▸ Глобальные настройки ▸ Ввод. К сожалению, сочетания клавиш здесь использовать нельзя. В качестве хост-клавиши я использую левую клавишу Windows, так как в системе Linux за ней не закреплено никакой определенной функции. В табл. 1.1 представлены различные сочетания горячих клавиш и хост-клавиши. Еще некоторые горячие клавиши перечислены в меню Машина окна VirtualBox.

Таблица 1.1. Горячие клавиши в VirtualBox

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

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

Для установки вставьте требуемый CD/DVD и выполните команду Устройства ▸ Установить гостевые расширения. Интегрировав компакт-диск с гостевыми расширениями в файловую систему виртуальной машины (как правило, это делается щелчком кнопкой мыши на значке CD, иногда — с помощью команды `mount`, например `mount/dev/sr0/media/cdrom`), выполните следующие команды:

```
root# sh /media/cdrom/VBoxLinuxAdditions-x86.run   (32-Bit-Gast)
root# sh /media/cdrom/VBoxLinuxAdditions-amd64.run (64-Bit-Gast)
```

СОВЕТ

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

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

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

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

Если вам требуется узнать действующую версию Fedora, будьте внимательны: есть две разные версии ядра и только в одной из них поддерживается PAE (рис. 1.2). Вы можете узнать, какой именно версией пользуетесь, с помощью команды `uname -r`. Если в результате получается последовательность символов, содержащая `pae`, то ядро работает с поддержкой PAE. В таком случае необходимо установить `kernel-devel` как пакет `kernel-PAE-devel`. Только тогда будет возможно скомпилировать совместимый модуль на работающем ядре.

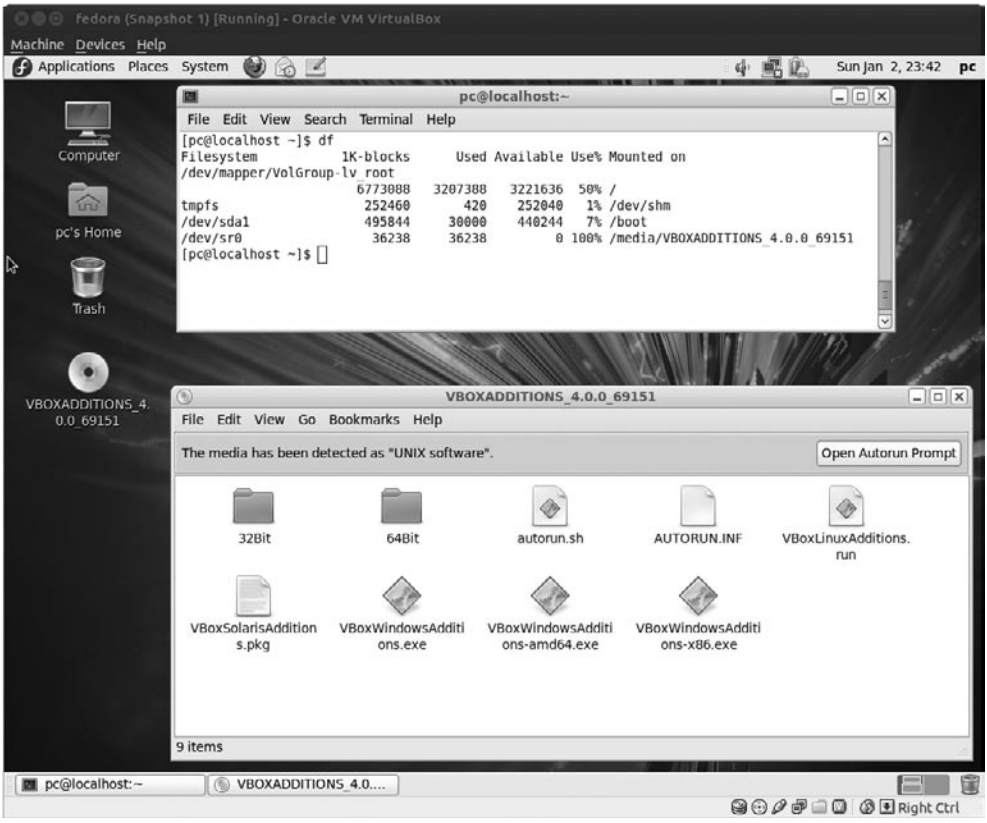

Рис. 1.2. Альфа-версия Fedora 12 с виртуальной машиной VirtualBox

Некоторые дистрибутивы, например openSUSE, предлагают готовые пакеты с гостевыми расширениями для VirtualBox. Кроме того, установка в этих дистрибутивах частично сходится. Когда я тестировал эту концепцию (неплохую саму по себе) на openSUSE 11.1, испытания не удались, но вот с openSUSE 11.2 повезло больше: после инсталляции в режиме live гостевые расширения VirtualBox имелись в распоряжении с самого начала и работали без дополнительной установки. При обычной установке с DVD потребовалось всего лишь заново инсталлировать файл `xorg.conf`:

```
root# cp /etc/X11/xorg.conf.install /etc/X11/xorg.conf
```

Установка виртуальной машины в Windows

Если у вас есть установочный диск, соответствующий файл ISO, а также действующая лицензия и ключ к ней, то вы можете установить VirtualBox и в системе Windows. Во время проведенных мной испытаний установка VirtualBox с гостевыми расширениями в Windows XP и Windows 7 прошла без проблем (рис. 1.3).

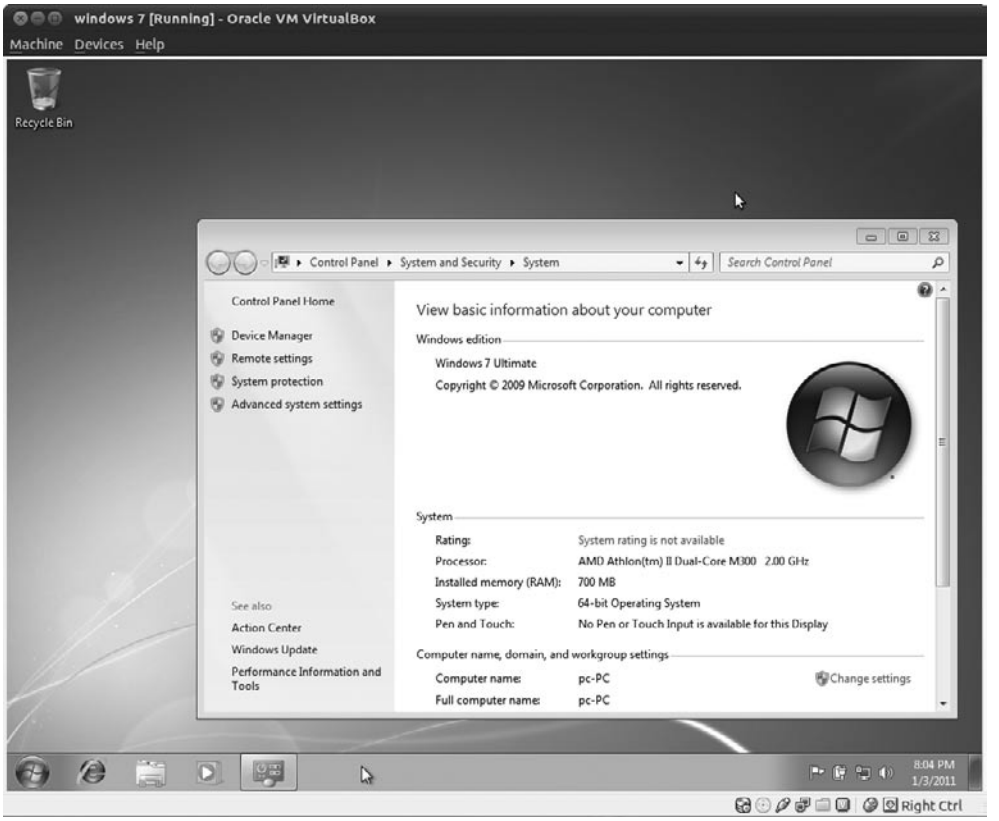

Рис. 1.3. 64-битная версия Windows 7 RC в виртуальной машине VirtualBox

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

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

Дополнительные функции VirtualBox

Организация совместно используемого каталога

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

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

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

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

Использование USB-устройств в виртуальных машинах

Если вы используете двоичную версию VirtualBox (а не OSE-версию), то можете работать в виртуальных машинах и с USB-устройствами. Эта функция работает только в том случае, если в системе-«хозяине» *не* применяется USB-устройство. Обычно в системе-«хозяине» USB-носитель автоматически включается в файловую систему. В таком случае вам потребуется вновь удалить USB-устройство из системы.

Еще одно необходимое условие заключается в том, что пользователь, работающий с VirtualBox, должен входить в группу vboxusers (см. подраздел «Установка VirtualBox» данного раздела). Наконец, следите за тем, чтобы контроллер USB 2.0 на вкладке окна USB был активизирован при настройке виртуальной машины. В данном окне можно также задать фильтр, который будет присваивать USB-устройство определенной виртуальной машине. Правда, последнее требование не является

обязательным. Кроме того, после включения USB-устройства его можно динамически присвоить статусной строке VirtualBox с помощью значка USB.

На многих сайтах написано, что USB-функции VirtualBox работают только тогда, когда в дерево каталогов системы-«хозяина» внедрена файловая система usbfs (например, по адресу https://help.ubuntu.com/community/VirtualBox/USB). Однако, когда я испытывал на виртуальных машинах устройства, подключаемые через USB (сканер и цифровой фотоаппарат), они безупречно работали в виртуальных машинах, может быть, только чуть медленнее, чем в системе-«хозяине». Измерения скорости при этом не проводились.

Импорт и экспорт виртуальных машин

Одно из наиболее серьезных преимуществ рабочей станции VMware по сравнению с VirtualBox заключается в том, что каталог виртуальной машины с файлами VMware можно без проблем копировать с одного компьютера на другой. В VirtualBox еще недавно дело обстояло сложнее: нужно было сначала скопировать файл жесткого диска, затем установить новую виртуальную машину, которая имела бы доступ к этому файлу.

С появлением версии VirtualBox 2.2 процесс упростился, но он все еще не так элегантен, как у VMware. Теперь в VirtualBox нужно выбрать меню Файл ▸ Экспорт модуля. Так создается *виртуальный модуль*, то есть виртуальная машина, предназначенная для переноса данных, состоящая, как правило, из двух файлов: в OVF-файле содержится описание виртуальной машины, а в VMDK-файле — заархивированный образ жесткого диска. Такую виртуальную машину можно снова установить с помощью команды меню Файл ▸ Импорт модуля при установке другого экземпляра VirtualBox. Хотя формат для виртуального модуля и является стандартизированным, переход с VMware на VirtualBox и наоборот, к сожалению, невозможен. Причина заключается в различиях виртуального аппаратного обеспечения, применяемого в каждой из систем.

1.3. KVM/QEMU

Собственно, KVM — это всего лишь небольшой модуль ядра, работающий только вместе с программой-эмулятором QEMU. Для работы KVM нужен процессор с поддержкой функций виртуализации аппаратного обеспечения, создающий из эмулятора QEMU виртуальную аппаратную систему. Элегантность KVM заключается в том, что он выполняет обычные задачи гипервизора (например, управление хранением информации и процессами) не самостоятельно, а с помощью стандартных функций ядра Linux, поэтому можно реализовать модуль ядра KVM, написав сравнительно небольшой объем кода.

QEMU

Поскольку QEMU является основой для KVM, в этом разделе мы сначала рассмотрим QEMU и лишь затем KVM. QEMU эмулирует различные процессоры и элементарные компоненты аппаратного обеспечения обычного компьютера (сетевая карта, CD-привод и т. д.). Кроме того, QEMU может эмулировать процес-

соры, несовместимые с процессором машины-«хозяина» (ARM, Sparc, PowerPC, MIPS и т. д.). От этого, разумеется, снижается скорость работы. Здесь мы рассмотрим эмулирование x86-совместимых процессоров. Подробное описание QEMU, в том числе его функции и границы возможностей, описаны по адресу http://www.nongnu.org/qemu/.

Пока мы рассмотрим работу с QEMU и KVM с помощью командной строки. Правда, для работы с этими программами существует и специальный пользовательский интерфейс, но он немного недоработан. В принципе, работа строится так: сначала с помощью команды `qemu-img` создается файл образа виртуального жесткого диска, а затем этот файл и различные параметры виртуального аппаратного обеспечения передаются QEMU в качестве параметров команды.

В команде `qemu-img` параметр `-f` указывает, какой формат вы собираетесь использовать. Стандартными форматами QEMU являются QCOW и QCOW2 (COW означает copy on write — «копирование при записи»). Файл увеличивается по мере того, как задействуются всё новые сектора виртуального жесткого диска. Еще допустимы для использования форматы файлов RAW и VDMK (совместимые с рабочей станцией VMware 3 и 4). Размер файла указывается в мегабайтах [nM] или гигабайтах [nG].

```
user$ qemu-img create -f qcow2 qemu.img 10G
```

Сам эмулятор запускается командой `qemu` или `qemu-xxx`. При этом `qemu` эмулирует x86-совместимый процессор, а `qemu-system-x86_64` — его 64-битный вариант. Синтаксис таков:

```
user$ qemu [параметры] image-файл
```

В следующем списке обобщены важнейшие параметры команды `qemu`. Есть и многочисленные другие параметры, предназначенные для выбора виртуального сетевого аппаратного обеспечения, виртуальной графической карты, применения компонентов USB и т. д. Они описаны в документации к команде `qemu` и на сайте www.qemu-buch.de (на немецком языке).

- `-boot a/c/d` — указывает загрузочное устройство (дисковод для гибких дисков, жесткий диск или CD/DVD-привод). Буквам соответствуют буквенные обозначения дисков, применяемые в MS-DOS и Windows. По умолчанию QEMU загружается с жесткого диска.
- `-cd-rom` *ISO-файл* — использует указанный файл ISO в качестве источника данных для виртуального CD/DVD-привода.
- `-k` *сокращенное название языка в соответствии с ISO* — задействует указанную раскладку клавиатуры. Среди допустимых языковых сокращений: `de` (немецкий) и `en-us` (американский английский). Этот параметр требуется очень редко, когда QEMU не может правильно интерпретировать коды клавиатуры на машине-«хозяине».
- `-localtime` — инициализирует виртуальные CMOS-часы гостевой системы с местным временем (а не с временем UTC, как это установлено по умолчанию).
- `-m` *n* — настраивает размер оперативной памяти виртуальной машины (в мегабайтах). По умолчанию размер оперативной памяти виртуальных машин составляет 128 Мбайт.

- `-net` *параметры* — конфигурирует виртуальное сетевое аппаратное обеспечение. Этот параметр требуется только в том случае, когда стандартные настройки QEMU (сетевая карта, совместимая с NE2000, сетевой стек в пользовательском режиме) не подходят.
- `-no-acpi` — деактивизирует поддержку ACPI[1].
- `-snapshot` — выполняет QEMU, не внося в файл образа жесткого диска долговременных изменений.

QEMU представляет эмулируемую систему в простом окне без меню и каких-либо других элементов управления. Для управления многочисленными дополнительными функциями применяются горячие клавиши. Важнейшие сочетания клавиш обобщены в табл. 1.2. На мониторе QEMU можно выполнять команды для управления виртуальной машиной. В файле справки коротко описаны все имеющиеся команды.

Таблица 1.2. Горячие клавиши KVM/QEMU

Сочетание клавиш	Функция
Ctrl+Alt	Снятие фокуса клавиатуры
Ctrl+Alt+F	Включение/выключение полноэкранного режима
Ctrl+Alt+1	Показать стандартный вывод
Ctrl+Alt+2	Показать монитор QEMU
Ctrl+Alt+3	Показать удаленный пульт администратора
Ctrl+Alt+S	Сохранить изменения в файле образа диска (только в режиме мгновенного снимка)
Ctrl+Alt+X	Окончание работы программы

Испытания QEMU. Стандартная версия QEMU способна с приемлемой скоростью эмулировать современные операционные системы с графическим интерфейсом только на очень мощной аппаратуре, поэтому для проведения первых тестов рекомендую применять дистрибутив Linux Mini. Если вы тоскуете по старым добрым временам господства MS-DOS, то можете провести пробный прогон на FreeDOS (рис. 1.4). Файл ISO с установочными файлами этой системы можно скачать по адресу http://freedos.org.

Обе следующие команды создают файл образа виртуального жесткого диска и запускают QEMU, при этом образ диска с системой FreeDOS используется в качестве загрузочного носителя.

```
user$ qemu-img create -f qcow2 qemu.img 500M
user$ qemu -m 16 -boot d -cdrom fdbasecd.iso qemu.img
```

После того как установка успешно завершится, образ компакт-диска вам больше не понадобится. Для запуска системы будет использоваться виртуальный жесткий диск.

```
user$ qemu -m 16 qemu.img
```

[1] ACPI — усовершенствованный интерфейс конфигурации и управления питанием.

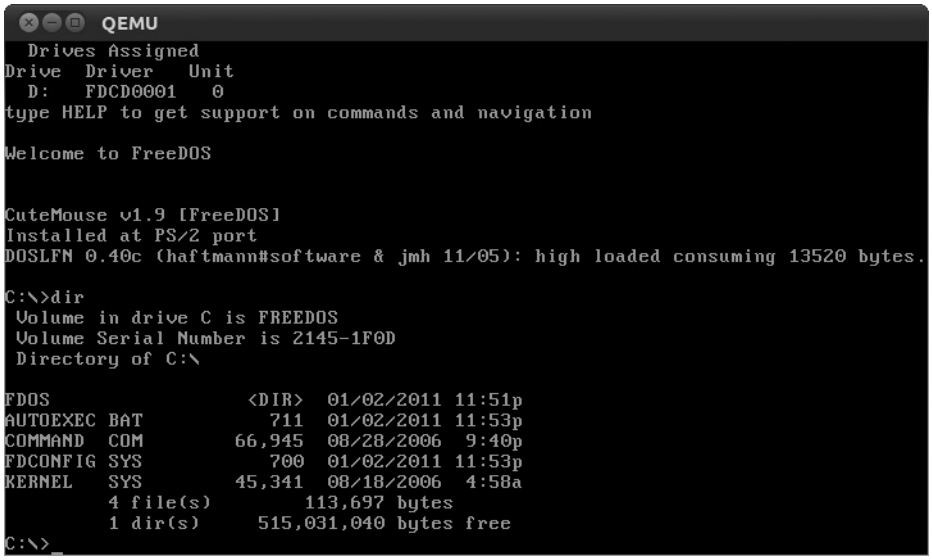

Рис. 1.4. FreeDOS в качестве гостевой системы на виртуальной машине от QEMU

Установка с QEMU «настоящих» операционных систем, например Windows и Linux, в принципе возможна, но даже на самых быстрых компьютерах занимает несколько часов. Дело идет существенно быстрее, если при виртуализации используется процессор и можно применять KVM.

KVM

Это вариант QEMU, обращающийся к модулям ядра KVM, чтобы виртуализация проходила более эффективно. Функции KVM предоставляются в виде трех модулей ядра: основные функции находятся в модуле `kvm`, функции, специфичные для intel-VT, — в модуле `kvm-intel`, а функции, характерные для AMD-V, — в модуле `kvm-amd`. Чтобы работать с KVM, нужно сначала загрузить модуль, подходящий для аппаратного обеспечения вашего компьютера. При этом `kvm` автоматически догружается.

```
root# modprobe kvm-intel (для процессоров Intel-VT)
root# modprobe kvm-amd   (для процессоров AMD-V-Prozessoren)
```

Использование функций KVM происходит через файл-устройство `/dev/kvm`. Применять KVM могут только пользователи, обладающие правами на чтение и запись информации. Во многих дистрибутивах работать с KVM может только root. Чтобы и другие пользователи могли применять KVM, следует соответствующим образом настроить права доступа для файла `/dev/kvm`. Более подробная информация о KVM находится на сайте http://www.linux-kvm.org/.

В зависимости от дистрибутива KVM запускается командой `kvm` или `qemu-kvm`. При этом применяются те же параметры, что и с `qemu`. Скорость работы KVM существенно выше, чем у QEMU.

KVM на практике. Для испытания KVM можно установить любой дистрибутив Linux. Я попробовал Ubuntu 9.04, и никаких проблем не возникло (рис. 1.5). Необходимые команды выглядят так:

```
user$ kvm create -f qcow2 ubuntu.img 8G (Создание файла образа)
user$ qemu -m 384 -boot d -cdrom ubuntu.iso ubuntu.img (Установка Ubuntu)
user$ qemu -m 384 ubuntu.img            (Выполнение Ubuntu)
```

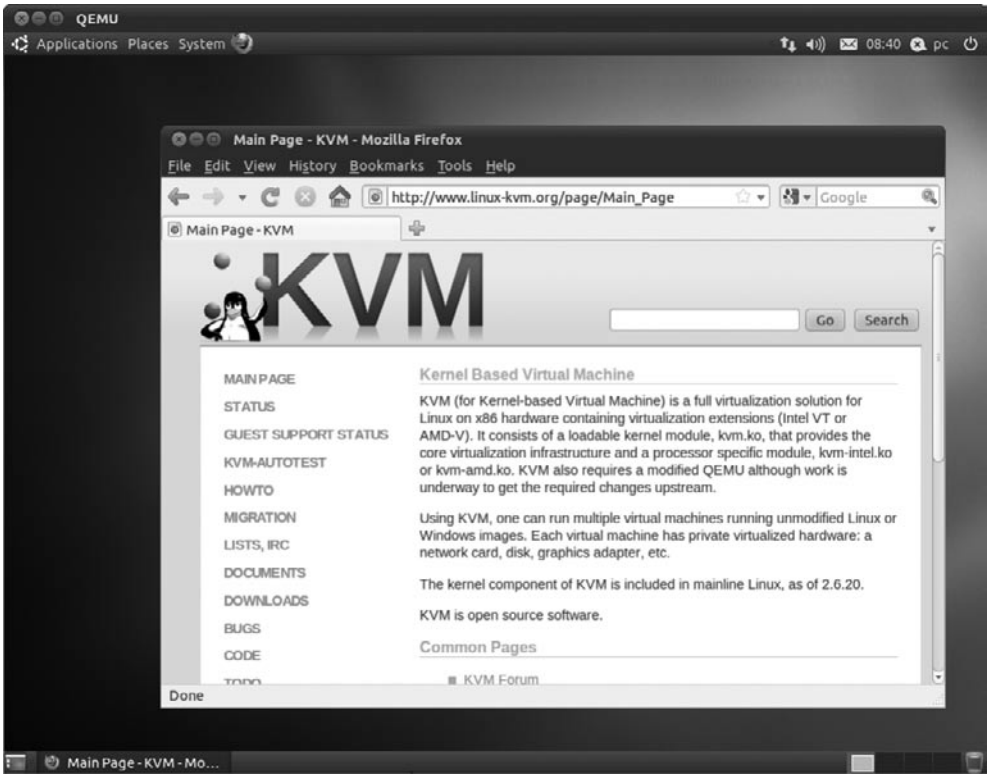

Рис. 1.5. Ubuntu в качестве гостевой системы на виртуальной машине KVM

Таким же образом можно установить и выполнить Windows XP SP2. Кроме всего прочего, в таком случае скорость работы будет достаточно высока.

```
user$ qemu-img create -f qcow2 winxp.img 10G (Файл образа)
user$ kvm -m 256 -boot d -cdrom win-install-cd.iso -localtime winxp.img (Установка)
user$ kvm -m 256 -localtime winxp.img (Применение)
```

Менеджер виртуализации. Для установки, запуска виртуальной машины KVM и управления ею можно также применять менеджер виртуализации, разработанный Red Hat. Хотя этот пользовательский интерфейс и поставляется с несколькими дистрибутивами в течение последних лет, он еще, к сожалению, не совсем доработан. Чтобы программа хорошо функционировала, нужно запустить ее с root-правами, что нежелательно с точки зрения безопасности.

1.4. Wine

Вообще программы, написанные для Windows, не работают в Linux, так как в последней отсутствуют необходимые библиотеки. С помощью программ виртуализации, рассмотренных выше в этой главе, данная проблема решается — нужно просто полностью установить Windows на виртуальной машине. Пусть этот механизм и хорошо функционирует, но он требует приобретения дорогой лицензии Windows и достаточно современного аппаратного обеспечения.

Почти во всех дистрибутивах предоставляются пакеты Wine, которые, однако, как правило, не устанавливаются автоматически. В этом разделе будет рассмотрена версия Wine 1.1.29. Более подробная информация по Wine имеется на следующих сайтах:

- http://www.winehq.com/;
- http://www.wine-wiki.com.

Частные советы, касающиеся использования Wine в Ubuntu, находятся по адресу https://help.ubuntu.com/community/Wine.

Wine. Если требуется лишь выполнить одну-единственную программу Windows, то можно сэкономить с помощью Wine много денег и ресурсов. Ведь Wine — это собрание библиотек с открытым кодом, позволяющих пользоваться важнейшими функциями операционной системы Windows. По причинам, связанным с лицензированием, в ходе разработки Wine, разумеется, нельзя было прямо копировать в программу код Windows. Однако многие функции просто писались на основе Windows. Это обстоятельство позволяет понять, почему Wine отстает от Windows на много лет. Часто Wine позиционируется как 32-битная версия Windows XP. Поддержка 64-битной версии пока в разработке, уже функционирует, но нестабильно. Одним словом, лучше всего учиться работать с Wine, запуская в ней маленькие, немного устаревшие программы для Windows.

CrossOver. Это коммерческий вариант Wine производства фирмы CodeWeavers, существенно упрощающий установку и запуск программ для Windows. CrossOver специально оптимизирован для того, чтобы обеспечивать работу в Linux некоторых популярных программ, разработанных для Windows. К таким программам относятся, в частности, Adobe Photoshop и различные версии пакета Microsoft Office. Подробнее о CrossOver рассказано в разделе 1.5.

Ограничения

Прежде чем воскликнуть «Ура!», познакомьтесь с некоторыми ограничениями. Wine и CrossOver функционируют только в том случае, когда дистрибутив Linux, с которым они используются, работает на оборудовании с процессором x86. Обе программы не могут приспособить машинный код к процессорам с другой архитектурой. Такие же ограничения действуют и для большинства других программ виртуализации.

Кроме того, ограничена совместимость между Wine, CrossOver и Windows. Многие программы не работают либо способны выполнять лишь элементарные функции. Скорость работы обычно остается удовлетворительной и даже более высокой, чем характерно для Windows. Не хватает современных 3D-библиотек.

ПРИМЕЧАНИЕ

Кроме использования виртуализационных программ и Wine/CrossOver терминальные серверы предоставляют еще одну возможность запуска программ для Windows на компьютерах с Linux. Вместо выполнения программы Windows на локальном компьютере они запускаются на внешнем компьютере, на котором установлена система Windows. Перенос данных, отображаемых на мониторе, — с одной стороны, и ввода, производимого с клавиатуры и с помощью мыши, — с другой, выполняется по сетевому протоколу (VNC, NX и т. д.). В данном случае локальный компьютер работает только в качестве терминала.

Такой вариант представляется интересным в первую очередь в больших сетях: сервер Windows с необходимыми параметрами обеспечивает доступ к программам Windows для всех пользователей Linux, которым такой доступ может понадобиться. По сравнению с другими вариантами на начальном этапе требуется выполнить относительно большую работу по конфигурированию системы. Кроме того, необходимо решить вопросы с приобретением лицензии. Как правило, нужна специальная многопользовательская лицензия, чтобы коммерческая программа могла применяться несколькими пользователями одновременно.

Конфигурация

Wine обычно работает без предшествующей конфигурации. Если возникнут проблемы, то вам поможет программа winecfg (рис. 1.6): с ее помощью можно задавать различные аудио- и графические настройки, позволяющие связать буквы дисководов Windows с определенными каталогами системы и т. д. На вкладке Приложения вы указываете, какую версию Windows будет эмулировать Wine (обычно Windows XP). Однако для отдельных программ вы можете установить соответствие другим версиям Windows.

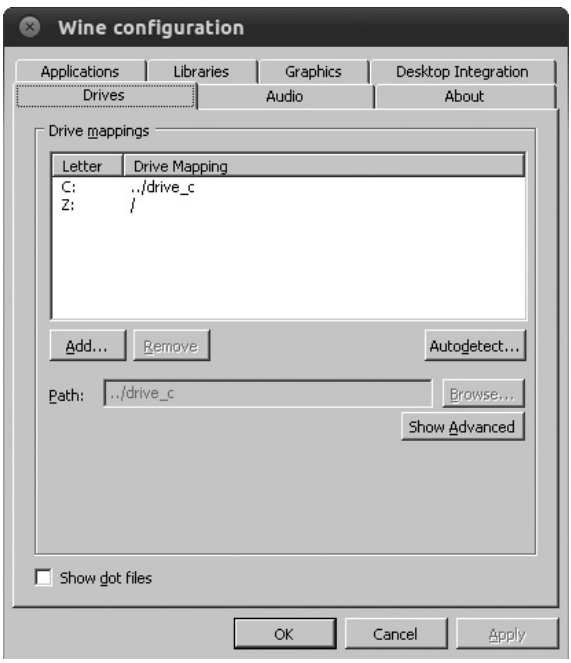

Рис. 1.6. Конфигурирование Wine с помощью программы winecfg

Установка и выполнение программ для Windows

После установки Wine можно запустить любую программу Windows с помощью команды `wine programmname.exe`. В качестве первого испытания предлагаю запустить программу-блокнот, поставляемую вместе с Wine:

```
user$ wine /usr/lib32/wine/notepad.exe.so
```

Разумеется, этот метод работает, если программа уже установлена. Если какая-либо программа для Windows уже есть у вас на компьютере в определенном сегменте диска, то ничего не выйдет, так как для выполнения программ Windows, как правило, требуются дополнительные библиотеки и записи в локальном системном реестре. Итак, желаемая программа должна быть установлена специально для Wine:

```
user$ cd каталог_с_файлами_установки/
user$ wine setup.exe
```

Если вам повезет, то установка пройдет без сообщений об ошибках. Обычно все проходит так хорошо, если вы устанавливаете не слишком новые, не слишком крупные и популярные программы. Если же программный пакет новый, очень большой и экзотический, то установка, напротив, часто прерывается непонятными сообщениями об ошибках. Проблема заключается в том, что для работы требуется дополнительная функция или целая библиотека Windows (DLL), которая пока отсутствует в Wine. При определенных обстоятельствах можно разработать обходные маневры, поискав информацию в Интернете. Но, бывает, приходится просто смириться, что эта программа в Wine (пока!) не работает. Если вы поместите сообщение об ошибке по адресу http://bugs.winehq.org, то поможете разработчикам Wine в будущем устранить эту проблему.

Если уже в процессе установки вы хотите проверить, заработает ли программа для Windows, то можете обратиться к базе данных приложений для Wine по адресу http://appdb.winehq.org. В этой базе содержится информация примерно о 13 000 программ для Windows, в частности устанавливаются ли они под Wine, а если устанавливаются, то как и в какой мере.

Если установка пройдет успешно, то Wine закончит работу, не сообщая никакой информации о том, где программа была установлена и где находится запускающий ее EXE-файл. Вам на помощь придет команда `find`, находящая все EXE-файлы, измененные в течение последних 60 минут. Конечно, вы можете получить и несколько результатов, если одновременно были установлены несколько подпрограмм. Тогда нужно найти нужную программу:

```
user$ find ~/.wine -cmin -60 -iname '*.exe'
```

Обратите внимание, что в Wine нет Рабочего стола Windows с его меню Пуск, Панелью задач, Проводником и т. д.

Обмен информацией между Wine и Linux

По умолчанию диску C: системы Windows присваивается каталог `~/wine/drive_c`, а дисководу Z: — корневой каталог Linux /. Если понадобится, можно определить

с помощью программы winecfg дополнительные буквы дисков и изменить пути к соответствующим дисководам.

При передаче данных через буфер обмена вы по аналогии с Windows можете попытаться сделать это с клавиатуры, нажав Ctrl+C и Ctrl+V. Но в Linux не применяется буфер для передачи выделенного текста! Поэтому в программах для обработки текста обычные сочетания клавиш и указатель мыши не работают.

Wine позволяет программам Windows пользоваться всеми принтерами, определенными для работы с Linux. Однако, когда я пытался печатать информацию из приложений Windows на принтерах для Linux, иногда случались накладки. В свою очередь, не возникало никаких проблем с использованием сетевых функций под Windows. Если ваш компьютер, работающий с Linux, имеет выход в Интернет, этим доступом можно пользоваться и при работе с программами, установленными для Windows.

Установка Internet Explorer

Вы, возможно, спросите, зачем устанавливать самую ненадежную программу из мира Windows, который и так надежностью не отличается? На то есть две причины: во-первых, Microsoft уже многие годы использует Internet Explorer в качестве основы для установки многочисленных расширений операционной системы. Многие программы для Windows можно установить только при условии, что вы пользуетесь актуальной версией Internet Explorer. Во-вторых, веб-разработчики попросту обязаны тестировать свои сайты в Internet Explorer. Если установите этот браузер на локальном компьютере, то избавите себя от постоянных переходов из одной операционной системы в другую. На данный момент Wine хорошо поддерживает Internet Explorer 6, Internet Explorer 7 — с ограничениями, а Internet Explorer 8 вообще не поддерживает.

Устанавливать Internet Explorer (далее просто IE) несколько утомительно. Сценарий ies4linux способен сделать за вас большую часть работы. Он устанавливает IE6, а также плагин для Flash-9. В качестве варианта можно установить IE7, но эта функция пока не доработана. Когда я проводил соответствующие испытания, установка удавалась, но IE7 отказывал уже при первом запуске. Более подробная информация содержится на сайте, посвященном сценарию ies4linux, по адресу http://www.tatanka.com.br/ies4linux/.

Для работы этого сценария необходимо сначала установить Wine и программу cabextract, которая предназначена для распаковки архивов CAB (архивы, запакованные таким методом, чаще всего применяются в Microsoft). Не забывайте, что установка IE разрешена только в том случае, если вы пользуетесь лицензионной копией Windows!

Сама установка происходит с помощью приведенных ниже команд. Когда вы выберете версию IE, которую собираетесь инсталлировать, установочные файлы этой версии будут скачаны прямо с веб-сервера Microsoft и без дальнейших запросов система устанавливается сама ошеломляюще быстро.

```
user$ wget http://www.tatanka.com.br/ies4linux/downloads/ies4linux-latest.tar.gz
user$ tar zxvf ies4linux-latest.tar.gz
user$ cd ies4linux-*
user$ ./ies4linux
```

Кроме того, запуск IE в зависимости от версии производится командой `~/bin/ie6` или `~/bin/ie7` (рис. 1.7).

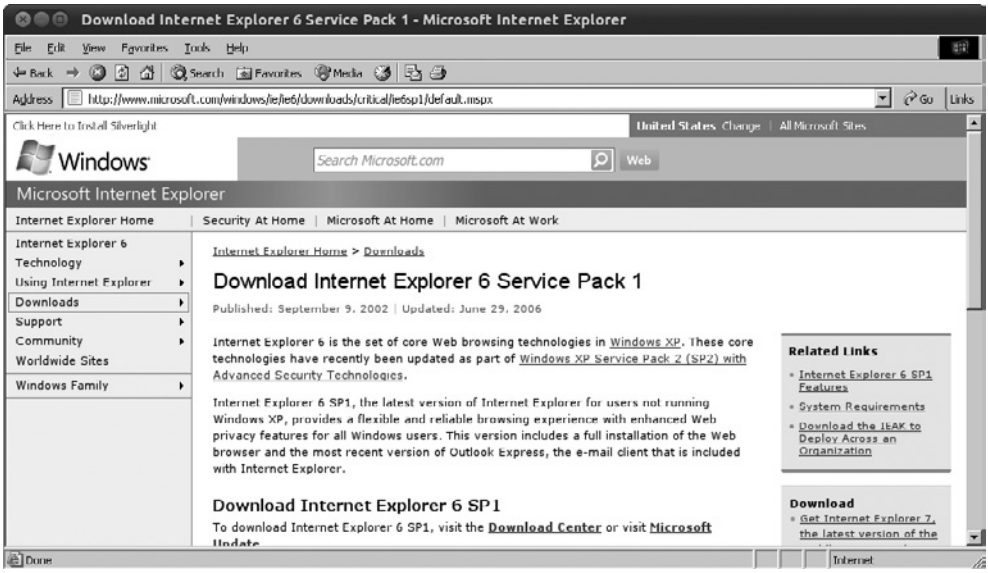

Рис. 1.7. Internet Explorer 6 в Linux

Практическое испытание

Чтобы протестировать, как Wine ведет себя с более экзотическими программами, в качестве эксперимента я установил два электронных словаря. Эти программы сохранились со времен отпуска, проведенного в Швеции, и относятся к числу тех немногих программ для Windows, которых в Linux иногда очень недостает.

Обе программы установились с ходу. Достаточно сложно оказалось найти соответствующие EXE-файлы. Всего при установке двух программ в каталоги Program Files и BC31 установилось девять таких файлов. Однако в итоге обнаружилось, что пользоваться обеими программами удается с большим трудом.

Резюме

В Wine можно работать с некоторыми программами для Windows. Работа с Wine рассчитана прежде всего на опытных пользователей Linux. Если вы не готовы немного поэкспериментировать и поискать в Интернете решения для частных проблем, то Wine вряд ли вам понравится. В целом программа оставляет впечатление некоторой сыроватости, которое глубоко не согласуется с огромными усилиями, потраченными на ее разработку (ведь в проекте Wine более миллиона строк кода!). Таким образом, программа подходит скорее для любителей-умельцев, чем для офисных работников, которые хотят время от времени пользоваться программами для Windows.

1.5. CrossOver

CrossOver — это коммерческий, немного улучшенный вариант Wine. Эта улучшенность проявляется в первую очередь в удобной поддержке при установке многих популярных программ. При этом CrossOver весьма значительно облегчает установку программ. К числу поддерживаемых приложений относятся в том числе версии Microsoft Office 97, 2000, XP и 2007, Adobe Photoshop, FrameMaker, Lotus Notes 6.5.1, а также браузерные плагины Quicktime, Shockwave и Проигрыватель Windows Media версии 6.4. Данные по CrossOver содержатся на сайте http://www.codeweavers.com/products/cxoffice/, посвященном этой программе.

Существуют различные виды продукции CrossOver. В последнее время относительно недорогую стандартную версию можно было купить всего за 37 евро. Она позволяет выполнять программы прямо на клиентском компьютере. В CrossOver Professional имеются дополнительные функции, в частности возможность клиентско-серверной эксплуатации, так что эту версию можно использовать и на терминальном сервере. CrossOver Games — это еще одна специальная версия, позволяющая играть в разные игры. Наконец, существуют версии CrossOver для Mac OS X, которые, однако, не являются темой этой книги.

Проект CrossOver продолжает тесное сотрудничество с Wine, в частности, в CrossOver заняты многие разработчики Wine. Улучшения и оптимизация варианта системы Wine CrossOver идут на пользу и первоначальному проекту. Правда, оказание поддержки при установке — это ресурс с закрытым кодом.

Конечно же, CrossOver не умеет творить чудеса. Хотя, например, многие пакеты Microsoft Office поддерживаются хорошо, совместимость со специальными функциями уже оказывается небезупречной: некоторые макросы VBA выполняются не совсем правильно, OLE (технология связывания и внедрения объектов) функционирует только с ошибками и т. д. Вообще старые программы обычно работают лучше новых, поэтому прежде чем приобретать CrossOver, обязательно попробуйте поработать с бесплатной 30-дневной тестовой версией.

Установка

Обычно CrossOver загружается в виде пакета для Debian или RPM с сервера CodeWeaver и устанавливается. Например, вы работаете с 64-битным дистрибутивом. Если этот дистрибутив основан на Debian, то сначала необходимо установить пакет ia32-libs. Если дистрибутив основан на RPM, для команды `rpm` нужно указать параметр `--ignorearch`, так как на данный момент существуют только 32-битные пакеты. Если дистрибутив не поддерживает ни Debian, ни RPM, то в нем предусмотрен собственный установочный сценарий.

Приложение установки добавляет в интерфейс Gnome и KDE собственное CrossOver-меню. Если оно не будет у вас работать, запустите программу CrossOver с помощью команды `/opt/cxoffice/bin/cx«имя»`.

Установка программы для Windows

Чтобы установить программу для Windows, выберите меню CrossOver ▸ Установить программу для Windows или выполните команду `cxinstallwizard`. Кроме того, выбе-

рите в списке желаемую программу и вставьте в привод установочный диск или укажите, где находятся установочные файлы (рис. 1.8). Далее установка протекает так же, как это обычно бывает в Windows, за исключением того, что перезагрузка системы, к счастью, всего лишь имитируется. Разумеется, для установки требуются и регистрационные ключи, поставляемые производителем. Итак, для установки коммерческих программ под CrossOver нужна лицензия, точно как для установки под Windows.

Рис. 1.8. Установка программ Windows с помощью CrossOver

Когда программа, которую необходимо установить, на выбор не предлагается, нужно установить флажок Установить неподдерживаемое программное обеспечение. После выбора файлов для установки CrossOver организует собственную рабочую среду, которая иногда называется bottle («бутылка»). За исключением этого инсталляция проходит так же, как и при работе с поддерживаемыми программами, но вероятность ошибок выше.

После удачного окончания установки можно с удобством запускать программы через новое меню Приложения Windows. Доверенная Wine операция по поиску нужного EXE-файла заканчивается успешно.

Удаление программ

Разумеется, программы Windows можно не только устанавливать, но и удалять (деинсталлировать). Для этого выполните команду CrossOver ▸ Конфигурация, а в раскрывающемся списке выберите программу, которую необходимо удалить.

Изменение размера шрифта меню

К сожалению, CrossOver игнорирует настройки разрешения экрана (DPI-настройки) и всегда отображает меню шрифтом одного и того же размера. Если вам требуется увеличить или уменьшить шрифт, добавьте в файл `~/.cxoffice/default/drive_c/windows/win.ini` две следующие строки:

```
[Desktop]
menufontsize=15
```

Если отдельные программы не установлены в стандартной рабочей среде, следует изменить файл `win.ini` и в других каталогах вида `~/.cxoffice/name`.

Практический тест

Испытание я провел на Office 2000. Установка программного пакета, а также обоих доступных пакетов обновлений удалась сразу. Кроме того, были загружены различные тестовые файлы, в том числе 1000-страничный документ Word. Далее я немного с ним поработал. Скорость меня абсолютно устроила. Самые простые VBA-макросы в Excel также удалось выполнить.

В принципе CrossOver работает отлично. Чтобы узнать, подходит ли программа для выполнения рутинных задач, которые обычно возлагаются на Windows, и достаточно ли она для этого совместима, потребуется провести значительно более долгие испытания, причем именно с вашими программами и в условиях вашего рабочего процесса. Я провел такие испытания у себя и остановился на одном из решений, связанных с виртуализацией.

2 Работа с консолью

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

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

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

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

В этой главе будет сделан лишь вводный обзор способов работы с консолью. В следующей главе соответствующие команды Linux мы рассмотрим более подробно. Эти команды служат, например, для управления файловыми системами (`ls`, `cp`, `mv`, `ln`, `rm`) для поиска файлов (`find`, `grep`, `locate`), для управления сетевыми функциями (`ping`, `ifconfig`, `ssh`) и т. д. Кроме того, мы подробнее изучим многие особенности Linux.

В главе 8 рассказывается об оболочке bash. Это программа, которая обычно работает с любой консолью и принимает команды, вводимые через нее. Bash может

применяться и для программирования. В разделе 9.2 описывается конфигурация текстовой консоли. Если у вас возникнут проблемы с раскладкой клавиатуры, отображением специальных символов и т. д., здесь будут предложены решения таких проблем. Наконец, в главе 8 содержится справочник важнейших команд Linux — они расположены по алфавиту.

2.1. Текстовые консоли и окна консолей

Текстовые консоли

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

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

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

Таблица 2.1. Сочетания клавиш для активизации текстовых консолей

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

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

Окно консоли (командное окно)

Конечно же, излишне переходить из работы в графическом режиме к текстовой консоли только лишь для выполнения команд. Для этого вполне достаточно окна консоли (рис. 2.1). Такие окна еще называются командными, или окнами терминала.

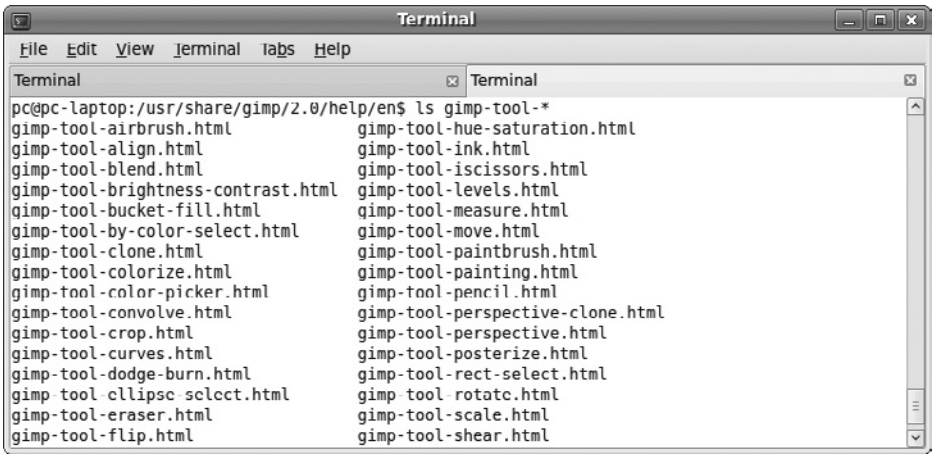

Рис. 2.1. Окно консоли

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

- Fedora 11 (Gnome) — Приложения ▸ Системные Инструменты ▸ Терминал;
- openSUSE 11.1 (KDE): Программы ▸ Система ▸ Терминалы ▸ Программа терминала;
- Ubuntu9.10: Приложения ▸ Стандартные ▸ Терминал.

В некоторых версиях Gnome окно консоли открывается еще удобнее — через контекстное меню на рабочем столе. Для этого требуется установить дополнительный пакет `nautilus-open-terminal`.

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

Важные сочетания клавиш

При работе с текстовыми консолями или окнами консолей могут пригодиться горячие сочетания клавиш, предназначенные для быстрого ввода команд. В следующей таблице обобщены важнейшие из них. Они работают только тогда, когда при конфигурации по умолчанию в качестве стандартной оболочки применяется

bash — так обстоит дело в большинстве дистрибутивов. Если вы работаете с Gnome в окне консоли, то нужно выполнить Правка ▸ Комбинации клавиш и снять флажок Активизировать все буквы горячих клавиш в меню.

Таблица 2.2. Сочетания клавиш для ввода команд в оболочке bash

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

Огромное количество времени экономится благодаря автозавершению команд клавишей Tab. Кроме того, нажимайте Tab, когда имя файла уже можно однозначно предугадать — если возможность окончания только одна, система дополнит команду за вас. Если дважды нажать Tab, то откроется список всех файлов, названия которых начинаются с уже введенной последовательности букв (подробнее этот механизм описан в начале раздела 8.3).

Мышь

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

Чтобы мышь работала в текстовых консолях, необходимо запустить программу `gpm`. Советы по конфигурированию этой программы даются в подразделе «Gpm-конфигурация (мышь)» раздела 9.2.

Выполнение команд

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

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

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

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

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

```
root# mkinitrd /boot/initrd-2.6.29.4-162.fc11.i686.PAE.img \
      2.6.29.4-162.fc11.i686.PAE
```

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

Выполнение команд на фоне работы программы

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

Работа с привилегиями администратора

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

Советы о том, как перевести выполнение команды в фоновый режим при работе другой программы, как просмотреть список всех выполняемых команд (процессов) и т. д., даются в разделе 4.1. В разделе 4.2 демонстрируются различные возможности выполнения программ при работе с привилегиями администратора, не входя при этом в систему в качестве root. В связи с этим также рекомендую

почитать главу 8: там сообщается вводная информация об оболочке bash и демонстрируются среди прочего возможности связывания друг с другом нескольких команд, передача результатов выполнения одной команды другой команде и т. д.

2.2. Просмотр и редактирование текстовых файлов

Команда less

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

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

Таблица 2.3. Клавиши и их сочетания для работы с командой less

Названия клавиш	Функция
Стрелки	Перемещение текста вверх или вниз
Home, End	Перейти к началу/концу текста
G, Shift+G	Перейти к началу/концу текста
/ *образец* Enter	Поиск по направлению вперед
? *образец* Enter	Поиск по направлению назад
N	Повторить поиск вперед (next)
Shift+N	Повторить поиск назад
Q	Завершить (выход)
H	Показать текст справки с другими обозначениями клавиш

СОВЕТ

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

Препроцессор

В большинстве дистрибутивов `less` может показывать не только обычные текстовые, но и заархивированные файлы, например в формате TAR. Для обеспечения

этого препроцессор (программа первичной обработки данных) анализирует данные, предназначенные для обработки, и передает результат команде `less`. В деталях этот процесс отличается от дистрибутива к дистрибутиву.

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

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

Текстовые редакторы

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

Emacs, Jove, Jed, Jmacs

Особая роль среди программ-редакторов отводится GNUEmacs и совместимому с ним XEmacs (начинайте с emacs или xemacs). В этих редакторах имеются невероятно широкие наборы функций, и такая программа с успехом может заменить программисту целую среду разработки. В табл. 2.4 обобщены только простейшие команды. Эти же команды работают в редакторах Jove, Jed и Jmacs. При этом я говорю о наиболее упрощенной версии Emacs, совместимой только с самыми элементарными функциями.

Таблица 2.4. Сочетания клавиш для работы с Emacs

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

Vi,Vim и Elvis

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

В Vi имеется не меньше функций, чем в Emacs, однако научиться работать с ним сложнее. При этом Vi сравнительно компактен и обычно доступен в экстренных системах. К тому же Vi есть практически в любых системах, созданных на основе UNIX. Программа является неофициальным стандартом в области UNIX/Linux, и многие другие программы автоматически вызывают ее для редактирования.

Важнейшее фундаментальное отличие от других редакторов заключается в том, что с Vi предусмотрено несколько режимов работы. Ввод текста осуществляется только в режиме вставки (табл. 2.5). Ввод большинства команд происходит в режиме complex command, который активизируется символом : (табл. 2.6). Перед этим необходимо выйти из режима вставки, нажав клавишу Esc. Редактору Vi в книге посвящена глава 7.

Таблица 2.5. Сочетания клавиш для работы с Vi

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

Таблица 2.6. Сочетания клавиш для работы в режиме complex command

Сочетание клавиш	Функция
:w *имя*	Сохраняет текст под новым именем
:wq	Сохраняет результаты работы и выключает редактор Vi
:q!	Выключает редактор Vi без сохранения информации
:help	Вызывает онлайн-справку

Joe. Это очень простой редактор. Сочетания клавиш аналогичны тем, что используются в программе для обработки текста Wordstar (табл. 2.7). Подробное описание всех команд выводится при выполнении на консоли команды `man joe`. Программа также иногда запускается под именами Jmacs или Jpico. При этом действуют другие сочетания клавиш, совместимые, соответственно, с Emacs или с Pico.

Таблица 2.7. Сочетания клавиш для работы с Joe

Сочетание клавиш	Функция
Ctrl+K, H	Открывает/закрывает окно справки
Ctrl+K, E	Загружает новый файл
Ctrl+K, D	Сохраняет файл (возможно сохранение под новым именем)
Ctrl+Y	Удаляет строку
Ctrl+Shift+–	Отменяет удаление
Ctrl+C	Завершает работу Joe (с запросом о необходимости сохранения информации)

Nano, Pico

Редактор Nano (или его вариант Pico) обладает небогатым набором команд, но зато прост в использовании. В этом редакторе в двух последних строках, выводимых на экране, дается обзор команд, имеющихся в вашем распоряжении (рис. 2.2). В большинстве современных дистрибутивов установлен только редактор Nano. Pico ранее был более широко распространен, однако его лицензия не полностью соответствовала принципам свободно распространяемого ПО и теперь этот редактор в Linux практически не применяется. Nano совместим с Pico, но не имеет проблем с лицензией.

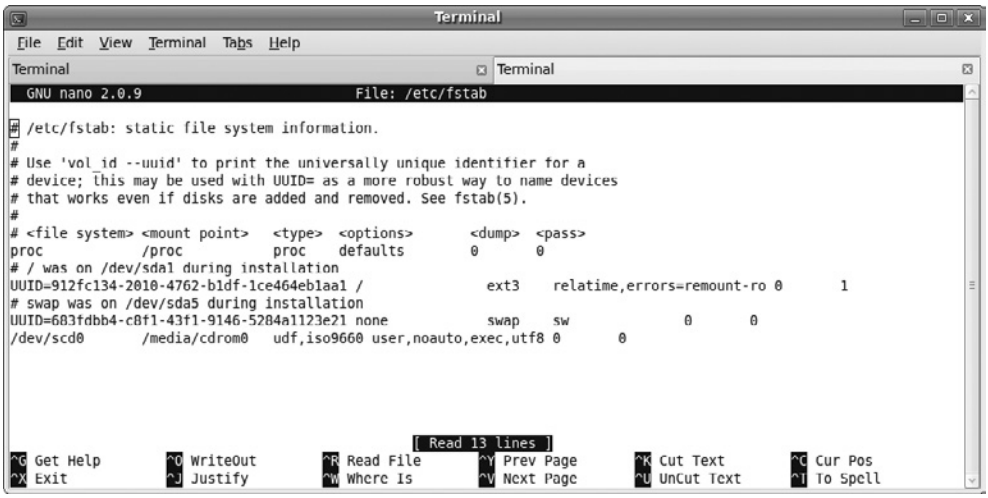

Рис. 2.2. Редактор Nano в окне консоли

Настройка текстового редактора по умолчанию

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

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

2.3. Онлайн-справка

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

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

Команда man

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

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

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

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

- 1 — пользовательские команды;
- 2 — системные вызовы;
- 3 — функции языка программирования C;

- 4 — форматы файлов, файлы-устройства;
- 5 — файлы конфигурации;
- 6 — игры;
- 7 — разное;
- 8 — команды для системного администрирования;
- 9 — функции ядра;
- n — новые команды.

Тексты файлов справки демонстрируются командой less, поэтому для навигации по тексту справки подходят и сочетания клавиш, обобщенные в табл. 2.3. Чтобы указать, из каких каталогов будут читаться тексты справки, нужно настроить переменную окружения MANPATH или придать управляющему файлу вид /etc/manpath.config.

В KDE и Gnome файлы справки можно читать из любого веб-обозревателя или электронного справочника. На следующих примерах демонстрируется, как показать страницу man по команде ls и список всех страниц man:

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

В программе konqueror системы KDE тексты man можно также вызывать с помощью *#команда*.

Команда help

Вызов справки по некоторым командам осуществляется не с помощью man, а с использованием help. Это касается всех команд, выполняемых прямо из оболочки. Оболочка — это интерпретатор команд, принимающий ваш ввод. Подробная информация по стандартной Linux-оболочке bash содержится в главе 8.

Программа info

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

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

К сожалению, достоинство, связанное с четкой структурой, часто оборачивается недостатком: навигация по этим текстам оставляет желать лучшего, кроме того, отсутствует механизм поиска, который позволял бы находить информацию в более широком диапазоне, чем страница, открытая в настоящий момент.

В табл. 2.8 приводится список клавиш для работы с info.

Таблица 2.8. Клавиши для работы с info

Названия клавиш	**Функция**
Пробел	Прокрутка текста по направлению вниз
Backspace	Прокрутка текста по направлению вверх
B, E	Перейти к началу или концу данного раздела справки
Tab	Переход к следующей перекрестной ссылке
Enter	Переход к другой части справочной информации
N	Следующая часть справки на том же уровне иерархии
P	Предыдущая часть справки на том же уровне иерархии
U	Переход на один уровень иерархии выше
L	Назад к последнему показанному фрагменту текста
H	Подробное практическое руководство
?	Обзор команд
Ctrl+O	Закрывает текущее подокно
Q	Завершает работу справки info

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

3 Управление файлами

В данной главе описаны способы управления файлами. Будут подробно рассмотрены следующие темы:

- файлы, каталоги и ссылки;
- копирование, перемещение и удаление файлов и каталогов;
- сжатие и архивация файлов;
- поиск файлов;
- запись CD и DVD;
- резервное копирование;
- права доступа к файлам (включая список контроля доступа);
- структура каталогов в Linux;
- файлы-устройства.

В главе 13 обсуждается администрирование файловых систем. Она в определенной степени представляет собой продолжение этой главы, но там будут рассматриваться вопросы, более важные не для пользователей, а для системных администраторов, а именно:

- какие файловые системы существуют;
- как файлы интегрируются в систему [`/etc/fstab`, `mount`-параметры];
- как можно применять систему программ массива независимых жестких дисков;
- что такое LVM;
- как можно зашифровать целую файловую систему.

3.1. Работа с файлами и каталогами

Коротко перечислю важнейшие фактические сведения об именах файлов.

- В ОС Linux имя файла должно быть не длиннее 255 символов.
- Имена файлов чувствительны к регистру!
- В именах файлов допускается указание международных символов, однако при использовании различных кодировок могут возникать проблемы (например, если разные кодировки одновременно применяются в одной и той же сети).

С недавнего времени почти во всех дистрибутивах Linux стандартной кодировкой считается UTF-8. О том, какие еще существуют кодировки и на что необходимо обращать внимание при их смене, рассказано в разделе 9.5, который посвящен проблеме интернационализации.

Для ядра Linux имя файла — это просто байтовая последовательность, в которой не может присутствовать символ или код 0. Интерпретация этой байтовой последовательности зависит от применяемой кодировки.

- В имени файла может содержаться сколько угодно точек. Имя файла `README.bootutils.gz` является совершенно тривиальным и означает, что мы имеем дело с файлом `README` по теме «Загрузочные утилиты».
- Файлы, имена которых начинаются с точки, считаются скрытыми (см. подраздел «Скрытые файлы» этого раздела). Как правило, скрытые файлы не отображаются в программе `ls` и в различных файловых менеджерах.
- Имена файлов, которые невозможно однозначно опознать как таковые после ввода команды (в частности, имена файлов, содержащие пробелы) должны даваться в кавычках (например, `"a b"`).

Размер файлов в современных версиях Linux практически ничем не ограничен и в зависимости от файловой системы обычно исчисляется терабайтами.

Каталоги

Дерево каталогов

Дерево каталогов в Linux начинается с корневого каталога /. Указывать диск, например C:, в Linux не только невозможно, но и бессмысленно (раздел 13.5). В этой книге все остальные каталоги считаются *подчиненными*. Таким образом, если представить дерево каталогов наглядно, то корневой каталог расположен на самой его верхушке. В некоторых книгах используется прямо противоположная номенклатура, которая больше напоминает дерево (корни внизу, ветви вверху), но не соответствует обычному значению терминов.

Одна из наиболее серьезных проблем в начале работы с Linux/UNIX заключается в том, чтобы найти определенный файл в широко разветвленной системе каталогов. Обзор этой проблемы дается в разделе 3.9.

Личный каталог

После входа в систему вы сразу же оказываетесь в так называемом личном, или «домашнем», каталоге. В вашем распоряжении все находящиеся в нем файлы и подкаталоги. Другие пользователи (кроме того, который обладает привилегиями администратора) не могут ни изменять, ни удалять файлы вашего личного каталога (а при определенных настройках даже не могут их читать).

Личный каталог обычно находится в дереве каталогов Linux по адресу `/home/loginname/` (только личный каталог администратора называется `/root`). Поскольку было бы неудобно каждый раз писать `/home/loginname/`, название собственного домашнего каталога сокращается до символа тильды (~). Кроме того, доступ к личным каталогам других пользователей возможен через запись `~loginname`.

Каталоги . и ..

В каждом каталоге имеется два особых подкаталога, необходимыех для формального управления иерархией каталогов. Каталог с именем, начинающимся на `.`, представляет собой ссылку на актуальный каталог, а каталог с именем, начинающимся на `..`, — на каталог, расположенный уровнем выше (табл. 3.1).

Таблица 3.1. Специальные символы, используемые при работе с каталогами

Символ	Значение
~	Домашний каталог
.	Текущий каталог
..	Каталог, расположенный на один уровень выше, чем текущий

Обе следующие команды копирования показывают, как можно использовать эти каталоги (другие `cp`-примеры даются в следующем разделе). Первая команда копирует файл `/etc/fstab` в актуальный каталог. Если каталог называется `/home/name`, то новый файл будет иметь имя `/home/name/fstab`.

```
user$ cp /etc/fstab .
```

Во втором примере мы сначала активизируем с помощью команды `cd` каталог `~/linux8`. Затем команда копирования `cp` создает резервную копию файла `fileuse.tex` (в котором содержится текст этой главы). Резервная копия имеет имя `~/fileuse.tex.bak`.

```
user$ cd ~/linux8
user$ cp fileuse.tex ../fileuse.tex.bak
```

Если домашний каталог называется `/home/name`, то вы сейчас создали резервную копию `/home/name/linux8/fileuse.tex`. Следовательно, полное название резервной копии: `/home/name/fileuse.tex.bak`.

Простейшие команды для работы с каталогами

Хотя при работе с KDE и Gnome в вашем распоряжении есть современные файловые менеджеры, опытные пользователи Linux охотнее работают с текстовыми командами. В табл. 3.2 приведены самые важные из таких команд.

Таблица 3.2. Команды для работы с каталогами

Команда	Функция
cd	Смена актуального каталога
cp	Копирование файлов
less	Постраничный показ текстовых файлов
ls	Показ всех файлов каталога
mkdir	Создание нового каталога
mv	Перемещение файлов или изменение их имен
rm	Удаление файлов
rmdir	Удаление каталогов

Перечисление файлов

Команда `ls` возвращает список всех файлов данного каталога. Если вы также хотите увидеть скрытые файлы, то задайте дополнительный параметр `-a`. Если вас интересует не только имя файла, но и другая информация о нем, например размер, владелец и т. д., то вам поможет параметр `-l`. По умолчанию вывод информации по `ls` производится в алфавитном порядке. Чтобы сортировать файлы по времени внесения изменений, размеру или расширению, используйте параметры `-t`, `-S` или `-X`. Параметр `-r` позволяет отсортировать информацию в обратном порядке. Следующая команда показывает все файлы с расширением TEX в каталоге linuxbuch, расположенные в порядке уменьшения размера:

```
user$ ls -l -S linuxbuch/*.tex
...
-rw-r--r-- 1 kofler kofler 30113 2009-05-11 09:09 linuxbuch/intro.tex
-rw-r--r-- 1 kofler kofler 63173 2009-01-29 08:05 linuxbuch/kde.tex
-rw-r--r-- 1 kofler kofler 76498 2009-06-08 15:43 linuxbuch/kernel.tex
...
```

Приведу несколько замечаний по интерпретации вывода команды `ls`: десять первых символов в начале строки указывают тип файла и биты доступа. В описание типа файла входят дефис (`-`) для обычного файла, `d` для каталога (directory), `b` или `c` для файла устройства (block или char) или `l` для символьной ссылки. Три следующих символа (`rwx`) указывают, что пользователи могут делать с файлом: читать, записывать в него информацию или выполнять его. Кроме того, сообщается аналогичная информация для членов группы, а также для других пользователей системы. Число, следующее за десятью символами, означающими тип и уровень доступа, обозначает, сколько жестких ссылок указывает на файл (о том, что такое ссылки, рассказано в разделе 3.2; подробности об управлении доступом к файлам Linux описаны в разделе 3.7). В следующих столбцах задано, к какой группе относится файл и кто его владелец (здесь в обоих случаях `kofler`), размер файла, дата его последнего изменения и, наконец, имя файла.

В большинстве дистрибутивов команда `ls` конфигурирована так, что файлы и каталоги распределяются по типу и окрашиваются разными цветами в зависимости от типа. Если в вашем дистрибутиве это не так, подобного эффекта можно достичь добавлением дополнительного параметра `--color`.

Обычно `ls` учитывает только файлы, находящиеся в каталоге, открытом в настоящий момент. Если вы хотите учесть также те файлы, которые находятся в подкаталогах, используйте параметр `-R`. Он, кстати, используется и со многими другими командами.

Следующая команда позволяет перечислить все без исключения файлы, находящиеся во всех подкаталогах (в том числе скрытые каталоги и файлы). Как правило, этот список достаточно длинный. Из него команда `| less` передает результат от `ls` к `less`, так что теперь вы можете пролистывать полученный программный вывод.

```
user$ ls -lR | less
```

Копирование файлов

Команда `cp` *имя1 имя2* копирует файл *имя1*. Копия называется *имя2*. Чтобы скопировать несколько файлов, вызовите команду вида `cp` *имя1 имя2 ... целевой каталог*. Следующие команды сначала переводят в активное состояние каталог linuxbuch, затем создают подкаталог bak, а потом копируют туда все текстовые TEX-файлы.

```
user$ cd linuxbuch
user$ mkdir bak
user$ cp *.tex bak/
```

Для того чтобы копировать целые каталоги вместе со всем их содержимым, используйте команду `cp -a`. Далее `cd`, не требуя дополнительных аргументов, делает домашний каталог актуальным. Вторая команда создает абсолютную копию каталога linuxbuch под именем linuxbuch-bak.

```
user$ cd (Активизировать домашний каталог)
user$ cp -a linuxbuch linuxbuch-bak
```

Удаление файлов и каталогов

Команда `rm` позволяет безвозвратно удалить указанный файл. Как правило, `rm` применяется для удаления файлов, а не каталогов. Для удаления каталогов предусмотрена команда `rmdir` *каталог*, которая, правда, работает лишь в том случае, если указанный каталог пуст. На практике для удаления каталогов обычно применяется команда `rm` с параметром `-rf`. Это означает, что все каталоги и находящиеся в них подкаталоги и файлы рекурсивно удаляются без запроса о подтверждении удаления. Понятно, что команда `rm -rf` очень опасна! Следующая команда удаляет созданную выше резервную копию каталога linuxbuch.

```
user$ rm -rf linuxbuch-bak/
```

Как узнать, сколько памяти нужно для размещения всех файлов и каталогов

С помощью команды `ls -l` невозможно узнать, насколько велик файл. Однако зачастую требуется знать, сколько места занимают все файлы, находящиеся в каталоге, сколько свободного места еще есть на диске и т. д. Для этого вам пригодятся две команды: `df` и `du`.

Узнаем, сколько свободного места на диске

Команда `df` показывает для всех сегментов файловой системы, или носителей данных, сколько всего места есть на этих носителях и сколько еще свободно. Параметр `-h` позволяет показать все данные о доступном дисковом пространстве в удобочитаемых числах, в килобайтах, мегабайтах или гигабайтах (а не блоками по 1 Кбайт, как это задано по умолчанию). Команда `df` также показывает различные

файловые системы, необходимые только для внутреннего управления ресурсами, а не для сохранения обычных файлов.

```
user$ df -h
Файл. система  Размер  Использ.  Дост.  Использ.%  Прикреплено к
/dev/sda3         14G     4,7G   8,5G         36%  /
/dev/sda2        942M      47M   849M          6%  /boot
/dev/sda6         28G     7,7G    19G         30%  /home
...
```

Команда `df` также позволяет установить, в каком сегменте диска физически находится файл. В следующем примере каталог `/home/kofler` находится в сегменте `/dev/sda6`, который, в свою очередь, расположен в дереве каталогов на месте `/home`.

```
user$ df -h /home/kofler/
/dev/sda6         28G     7,7G    19G         30%  /home
```

Установление размера каталога

Команда `du` дает возможность установить, сколько дискового пространства требует данный каталог, в том числе все содержащиеся в нем подкаталоги. Параметр `-h` опять же позволяет вывести результат в удобочитаемой форме, а не в виде блоков по одному килобайту. Параметров для сортировки результатов команды `du` не существует (однако в Gnome имеется программа `baobag`, позволяющая графически представлять размер каталогов; в KDE подобная наглядная возможность предусмотрена в файловом менеджере Konqueror).

```
user$ du -h fotos/2008
74M    fotos/2008/2008-03-ostern
162M   fotos/2008/2008-08-korsika
66M    fotos/2008/2008-11-diverse
...
2,0G   fotos/2008
```

Джокерные символы

При ежедневном обращении с файлами часто требуется обработать целые группы файлов, например все файлы с расширением TEX. Чтобы такие операции были возможны, при вводе команд Linux предусмотрены так называемые джокерные символы (табл. 3.3).

Таблица 3.3. Джокерные символы для имен файлов

Символ	Значение
?	Любой символ
*	Сколько угодно любых символов (а также ни одного)
[abc]	Один из символов, указанных в скобках
[a–f]	Один из символов, относящийся к указанному диапазону
[!abc] или [^abc]	Все символы, указанные в скобках, должны отсутствовать

Символы * и ?

Символ ? служит для указания *любого* символа, а символ * — для указания любого количества символов (в том числе ни одного). Пользователи, которые умеют работать с MS-DOS, на первый взгляд могут решить, что отличий от этой системы нет. Однако это впечатление обманчиво.

- Символ * включает практически любые символы, в том числе точки (кроме точек, с которых начинается имя файла). Если вы хотите обработать все файлы, то в Linux потребуется указать *, а не *.*! (Замечания относительно скрытых файлов даются ниже.)
- Если указать много джокерных символов, это никак не повлияет на работу Linux. Например, можно искать с помощью выражения *graf* все файлы, в имени которых содержится graf, — в том числе grafik.doc, apfelgraf и README.graf.

Символы [] и [!]

Если указания символов * и ? недостаточно, можно ужесточить ограничение, добавив квадратные скобки. Например, [abc] — это подстановочный символ для одной из трех букв: a, b или c. Если в квадратных скобках между двумя буквами или цифрами стоит дефис, то имеется в виду символ «между». Таким образом, [a-f]* охватывает все файлы, начинающиеся с любой из букв между a и f включительно.

Выражение *[_.-]* означает все файлы, в названии которых содержится минимум одна точка, нижнее подчеркивание или специальный символ, а * .[hc] означает все файлы, которые заканчиваются на .c или .h.

Джокерные символы можно применять и при работе с каталогами. Выражение */*.tex означает все TEX-файлы, которые находятся в подкаталогах текущего каталога (только на один уровень ниже, то есть не включая подкаталоги подкаталогов актуального каталога). Выражение /usr/ * bin/* означает все файлы, находящиеся в каталогах /usr/bin и /usr/sbin.

Интерпретацией джокерных символов занимается не команда, вызываемая для обработки, а та оболочка, из которой вызывается команда. Оболочка bash, которая используется в Linux чаще всего, кроме уже указанных джокерных символов распознает и множество других специальных символов, оказывающих специфическое влияние при выполнении команды (раздел 8.6).

Пример. Следующая команда копирует все C-файлы из каталога project в текущий каталог:

```
user$ cp project/*.c .
```

Сложности при использовании джокерных символов

Работа с джокерными символами на первый взгляд кажется проще, чем она есть на самом деле. Если у вас возникают сложности с джокерными символами, проведите несколько экспериментов с командой echo jokerzeichen. Она показывает все имена файлов, охватываемые комбинацией с джокерным символом, и выводит эти имена на экран, не изменяя при этом имен файлов.

Проблема заключается в том, что символ * относится не только к файлам, но и к каталогам, поэтому команда `ls*` показывает не только все файлы текущего каталога, но и файлы, находящиеся в подкаталогах, которые также охватываются символом *. При использовании команды `ls` эту проблему можно устранить с помощью параметра `-d`, но при работе с другими командами он недоступен.

Обработка каталогов с помощью */

Если вы хотите обработать все каталоги (но не файлы), вам поможет джокерная комбинация */. Она охватывает все «файлы», которые содержат ссылку на самих себя как на подкаталог (а такая ситуация возможна только с каталогами). Внутри системы каталоги считаются особой разновидностью файлов, поэтому я ставлю в данном случае кавычки.

```
user$ echo */.
```

Проблемы с *.окончанием

Тот факт, что обработкой джокерных символов занимается не какая-либо программа, а сама оболочка, таит в себе не только преимущества, но и недостатки. Например, как оказывается, невозможно произвести поиск TEX-файлов в подкаталогах с помощью команды `ls -R *.tex` (параметр `-R` для команды `ls` вызывает рекурсивный поиск по подкаталогам).

Причина этого проста: оболочка расширяет схему `*.tex` для текущего каталога и передает список найденных файлов к `ls`. Эта команда выводит информацию по данным файлам. Если у вас нет каталогов с окончанием `*.tex`, то команда `ls` завершает работу. В данном случае не поможет даже параметр `-R`. Рекурсивный поиск будет применен только к тем каталогам, имена которых были переданы в качестве параметра.

В Linux для поиска по файлам предусмотрена гораздо более гибкая команда `find`. В следующем примере показан список всех TEX-файлов в текущем каталоге и во всех его подкаталогах. Основы работы и примеры по `find` приводятся в подразделе «Команда find и grep» раздела 3.4.

```
user$ find . -name '*.tex'
```

Переименование файлов

К сожалению, в Linux нельзя переименовать все файлы `*.x` в файлы `*.y` с помощью команды `mv *.x *.y`. Причина такого ограничения такая же, как и в вышеописанных случаях: оболочка заменяет запись `*.x` списком всех файлов, соответствующих данной схеме. Для `*.y` соответствующие имена файлов отсутствуют. При этом команде `mv` передается список из нескольких файлов и выражение `*.y` — в этом случае `mv` «не знает», что делать с такими аргументами.

Вот конкретный пример: предположим, что в текущем каталоге находятся только файлы `markus.x`, `peter.x` и `ulrike.x`. Если выполнить в данном случае команду `mv *.x *.y`, то оболочка заменит схему `*.x` тремя файлами, указанными выше. Кроме того, оболочка не найдет файлов, подходящих для `*.y`, и передаст схему

в неизменном виде. Только сейчас будет запущена команда mv. Она получит следующие параметры, с которыми, как и следовало ожидать, будет невозможна любая работа.

```
user$ mv markus.x peter.x ulrike.x *.y
```

Даже если передать команде в качестве списка параметров запись markus.x peter.x ulrike.x markus.y peter.y ulrike.y, мы получим не тот результат, который нужен. Команда mv в принципе не может переименовать сразу несколько файлов. В таком случае или несколько файлов будут перемещены в другой каталог, или будет переименован только один файл.

Переименование с помощью sed

Разумеется, эксперты из области UNIX нашли решение и для этой проблемы: в таких случаях они стали применять потоковый редактор sed. Поскольку обслуживать sed достаточно сложно, примеры, подобные приведенному ниже, подходят только при программировании оболочки.

Коротко опишу принцип работы: команда ls возвращает список файлов, которые необходимо переименовать, и передает их sed. Редактор sed образует вместе с командой s (регулярное нахождение и замена) список cp-команд и, в свою очередь, передает этот список новой оболочке sh, которая, наконец, выполняет команды. Строка, показанная ниже, позволяет скопировать все файлы *.xxx в *.yyy.

```
user$ ls *.xxx | sed 's/\(.*\)\.xxx$/cp & \1.yyy/' | sh
```

Еще одна альтернатива — написать небольшой цикл. С помощью приведенной ниже команды делаются копии всех файлов с расширением *.tex. Эти копии получают окончание tex ~ (окончание ~ часто применяется для обозначения резервных копий).

```
user$ for i in *.tex; do cp $i $i~; done
```

Скрытые файлы

Файлы, имя которых начинается с точки, в Linux являются скрытыми, поэтому символ * не позволяет учесть все файлы, находящиеся в каталоге: скрытые файлы (часто ими являются файлы конфигурации, которые должны оставаться невидимыми) игнорируются.

И если вы думаете, что можете включить в выборку все скрытые файлы, поставив подстановочный символ .*, вас ожидает серьезная проблема: таким образом в выборку будут включены не только невидимые файлы, имена которых начинаются с ., но и каталоги, названия которых начинаются с . и .. (то есть текущий каталог и каталог, находящийся в иерархии на одну ступень выше). Если определенная команда сможет изменить целые каталоги, то последствия могут оказаться катастрофическими.

Эту проблему можно обойти, задав схему поиска .[!.]*. Она охватывает все имена файлов, первым символом в которых является точка, далее следует минимум

один символ, не являющийся точкой, а затем любое количество символов (в том числе ни одного).

```
user$ echo .[!.]*
```

С командой `ls` можно использовать параметр `-a`, который позволяет найти все файлы в данном каталоге (включая скрытые). Такой метод не допускает использования масок (например, `*rc*`). Параметр `-a` работает только в том случае, когда `ls` сама ищет себе файлы, а не передает выполнение этой задачи оболочке.

В данном случае универсальным вариантом является только команда `find`. Следующая команда находит все скрытые файлы в текущем каталоге, а также во всех его подкаталогах:

```
user$ find . -name '.*'
```

Особые виды файлов (файлы-ссылки, файлы-устройства)

Кроме обычных в Linux различается ряд специфических видов файлов, например каталоги, ссылки (раздел 3.2), файлы устройств для доступа к компонентам аппаратного обеспечения (раздел 3.10) и т. д. (табл. 3.4). В команде `ls -F` такие специфические файлы обозначаются дополнительным символом.

```
user$ ls -lF
... 13. Apr 11:31 bak/
... 11. Apr 12:21 grepalltex*
...
```

Таблица 3.4. Идентификация специальных файлов

Символ	Значение
/	Каталог
*	Исполняемый файл
@	Символьная ссылка
-	Символьное устройство
+	Блочное устройство
=	Pipe, FIFO

3.2. Ссылки

Ссылки — это указания на файлы. С помощью ссылок из различных участков структуры каталогов можно получать доступ к определенному файлу без необходимости сохранять несколько копий файла в различных местах жесткого диска. Таким образом, ссылки очень важны и помогают избежать избыточности в системе. В файловой системе Linux ссылки чаще всего встречаются в каталогах

/bin и /lib (рекомендую внимательнее рассмотреть, например, /usr/bin или /usr/lib с ls -l).

Чтобы лучше понять, что такое ссылки, приведу пример. Допустим, в каталоге test находится файл abc. Логично, что команда ln abc xyz должна создавать новый файл xyz. Но на самом деле abc и xyz — это всего лишь две ссылки на один и тот же файл. Чтобы проверить, не имеем ли мы дело именно с такой ситуацией, необходимо воспользоваться командой ls с параметром -l. Во втором столбце вывода команды сообщается, сколько ссылок указывает на один и тот же файл (в данном примере — две). Если дополнительно используется параметр -i, то ls также называет индексный узел файла, идентичный у ссылок, указывающих на файл (и только у них).

```
user$ ls -li
 59293 -rw-r--r-- 1 root root 1004 Oct 4 16:40 abc
user$ ln abc xyz
user$ ls -li
 59293 -rw-r--r-- 2 root root 1004 Oct 4 16:40 abc
 59293 -rw-r--r-- 2 root root 1004 Oct 4 16:40 xyz
```

Если теперь вы измените любой из двух файлов (все равно, какой), то автоматически изменится и другой файл (так как в действительности существует только один файл). Если удалить один из этих двух файлов, то просто уменьшится количество ссылок.

ПРИМЕЧАНИЕ

При обработке связанных ссылок в текстовом редакторе иногда могут получаться странные результаты: после первого сохранения ссылка указывает на резервный файл, а при повторном сохранении — в пустоту.

Причина такова: при сохранении некоторые редакторы создают файл резервной копии, переименовывая при этом имеющийся файл. Например: файл abc и копия abc~. Измененный файл сохраняется заново, получает новый индексный узел и поэтому не связан со ссылками. Чтобы с этим справиться, используйте символьные ссылки.

Символьные ссылки. В Linux различаются два вида ссылок. В предыдущем примере были показаны жесткие ссылки в таком виде, как они обычно создаются командой ln. Если же, напротив, применяется параметр -s, то команда создает символьные ссылки. Символьные ссылки имеют определенное достоинство по сравнению с жесткими ссылками — они могут указывать из файловой системы одного физического жесткого диска на файлы и каталоги другого диска. (Как правило, с жесткими ссылками невозможно ни то, ни другое. Особую категорию образуют жесткие ссылки на каталоги. Ставить их можно, но эта возможность предоставляется только суперпользователям.)

Команда ls при работе с символьными ссылками позволяет указать, где находится исходный файл. В любом случае команда не предоставляет счетчика, с помощью которого можно было бы указать, из какого количества мест поставлены ссылки на исходный файл.

Внутреннее отличие между жесткими и символьными ссылками заключается в том, что в первом случае сохраняются данные об индексном узле, а во втором

случае — имя файла или (если ставится ссылка на файл, расположенный за пределами каталога) путь к этому файлу.

```
user$ ln -s abc efg
user$ ls -li
 59293 -rw-r--r-- 2 root root 1004 Oct 4 16:40 abc
 59310 lrwxrwxrwx 1 root root 3 Oct 4 16:52 efg -> abc
 59293 -rw-r--r-- 2 root root 1004 Oct 4 16:40 xyz
```

СОВЕТ

Перед тем как создавать символьную ссылку, всегда необходимо перейти в тот каталог, в котором она будет содержаться, иначе ссылка может указать не туда, куда вы ожидали.

Символьные ссылки работают немного иначе, нежели жесткие. При удалении исходного файла (в том числе `abc`) ссылка на этот файл не изменяется, но теперь `efg` указывает в пустоту. Если же, напротив, удаляется символьная ссылка, то на исходный файл это никак не влияет.

Символьные ссылки могут указывать не только на файлы, но и на каталоги. В данном случае возможна некоторая путаница — одна символьная ссылка может вызвать эффект «дублирования» целого дерева каталогов. Однако на самом деле символьная ссылка представляет собой всего лишь дополнительный путь к тем же самым файлам и подкаталогам.

Как правило, при выставлении ссылок необходимо пользоваться только относительными указаниями путей и избегать абсолютных. Таким образом вы избавитесь от проблем, возникающих при подключении каталогов через NFS или при их перемещении.

Итак, определенные достоинства есть и у жестких, и у символьных ссылок. С символьными ссылками проще работать. Однако жесткие ссылки занимают меньше места на диске и работают быстрее.

3.3. Типы файлов (MIME)

Когда вы выбираете ссылку на MP3-файл в браузере или менеджере файлов, этот файл автоматически воспроизводится в проигрывателе. Если такая связь работает, это значит, что типы MIME в системе сконфигурированы правильно.

Аббревиатура MIME означает Multipurpose Internet Mail Extensions — Многоцелевые расширения электронной почты. Первоначально типы MIME разрабатывались для приложений к электронным письмам. Если в приложении к письму отсылался, например, файл в формате PostScript или JPEG, то клиент электронной почты сразу «знал», с помощью какой программы следует просмотреть или обработать этот файл. Чтобы этот механизм работал, требовалось настроить конфигурацию MIME.

Однако со временем область применения типов MIME существенно расширилась: когда вы переходите к файлу по ссылке, стоящей в браузере или в файловом менеджере, программа должна «знать», как поступить с данным файлом. Итак, правильная конфигурация типов MIME важна в любых программах, которые должны уметь работать с файлами различных типов.

Конфигурация MIME

Linux не была бы Linux'ом (или UNIX'ом), если бы в системе имелось только центральное место для проведения конфигурации MIME. Такие места, конечно, есть, но они не централизованы, и их очень много. Данные MIME для программ KDE, программ Gnome, различных браузеров, системы печати CUPS и т. д. управляются по отдельности. Кроме того, есть и центральная система конфигурации MIME для всех тех программ, которые не располагают собственными файлами MIME.

Есть несколько причин тому, что конфигурация MIME в Linux была распределена по нескольким местам. При работе KDE и Gnome применяется концепция, в соответствии с которой для обработки различных типов данных используются разные компоненты. Если в файловом менеджере программы KDE необходимо открыть файл-изображение в формате PNG, то в систему просто загружаются, а затем выполняются соответствующие компоненты. Поскольку библиотеки KDE и Gnome, как правило, несовместимы друг с другом, попытка файлового менеджера KDE выполнить компоненты Gnome (и наоборот) привела бы к катастрофическим последствиям. Во избежание такой ситуации KDE и Gnome используют собственные отдельные базы данных по MIME. По сходным причинам собственная конфигурация MIME присваивается и другим программам.

При работе со многими конфигурационными файлами MIME необходимо различать глобальную и индивидуальную конфигурацию, то есть общие настройки и настройки для отдельно взятого пользователя. Далее показана базовая конфигурация компонентов MIME в системе Linux. Пользовательские компоненты, связанные с MIME, описаны в других главах, например конфигурация типов MIME для KDE — в главе о KDE и т. п.

Общая конфигурация MIME. Общие файлы конфигурации MIME используются только теми программами, которые не располагают собственными файлами MIME. Настройки разделены на два файла, один из которых представляет собой глобальную версию, а другой — пользовательскую (табл. 3.5).

Таблица 3.5. Конфигурационные файлы MIME

Файл	Значение
/etc/mime.types	Глобальная конфигурация для типов файлов
/etc/mailcap	Глобальная конфигурация для программ
~/.mime.types	Локальная конфигурация для типов файлов
~/.mailcap	Локальная конфигурация для программ

Файл `mime.types` содержит список, в котором представлено соответствие между типами файлов (первый столбец) и расширениями файлов (все остальные столбцы). Например, типу `application/pdf` соответствует расширение PDF. В списке `mime.types` различаются текстовые приложения и X-приложения, типы MIME которых имеют вид `application/x-name`.

```
# in /etc/mime.types
...
application/pdf pdf
```

Файл mailcap указывает, какая программа должна применяться для отображения или обработки файла того или иного типа. В следующей строке сообщается, что для отображения файлов PDF следует использовать программу evince. В отличие от mime.types, столбцы mailcap должны разделяться точками с запятой; %s — это подстановочный символ для имен файлов.

```
# in /etc/mailcap
application/pdf; evince %s
```

Магические файлы для распознавания типа файла

Тип MIME предназначен для того, чтобы те или иные файлы открывались или обрабатывались программами, предназначенными специально для этого. Но как вообще удается установить тип обрабатываемого файла? Как правило, тип файла определяется по его расширению. Например, расширение *.ps означает, что перед нами файл PostScript.

Если файл не имеет такого идентификатора, программа file или соответствующие программы систем KDE или Gnome пытаются вычленить эквивалент типа файла по первому байту, содержащемуся в этом файле, либо по характерным для него последовательностям символов. Метод распознавания основывается на информацию о том, какие байты и типичные последовательности символов может содержать файл; эта информация встраивается в команду file при компиляции. В некоторых дистрибутивах стандартную конфигурацию можно изменять с помощью файлов /etc/magic или in~/.magic.

3.4. Поиск файлов

В системе Linux предусмотрено множество возможностей поиска файлов (табл. 3.6). Выбор команды, лучше всего подходящей для конкретного случая, зависит от того, о файле какого типа идет речь (текстовый файл, программа и т. д.) и какая информация о файле известна (фрагменты названия, фрагменты содержимого и т. д.).

Таблица 3.6. Инструменты для поиска файлов

Команда	Функция
beagle-query	Поиск файлов по их содержимому
grep	Поиск текста в текстовом файле
find	Поиск файлов по имени, дате, размеру и т. д.
locate	Поиск файлов по имени
whereis	Поиск файлов в заданных каталогах
which	Поиск программ в PATH-каталогах

Команды which и whereis

Команда which ищет указанную команду. Она возвращает полное имя команды, которая была бы выполнена, если бы вы набрали ее имя без указания пути. Команда which осуществляет поиск только по каталогам, указанным в PATH, работая при

этом исключительно быстро. PATH содержит список каталогов, в которых находятся программы. Однако необходимо учитывать, что PATH для администратора содержит больше каталогов, чем для обычного пользователя. Итак, если вы ищете системные команды, необходимо войти в систему с правами администратора.

```
user$ which emacs
/usr/bin/emacs
```

Команда whereis просматривает все пути, обычно используемые для двоичных файлов, файлов конфигурации, man-страниц и исходного кода. Для этого ее нужно ввести после имени файла. Команда whereis охватывает больше файлов, чем which, при этом поиск не ограничивается одними только программами. Она не может искать в каталогах, которые специально не указаны программе как места для поиска (см. man whereis).

```
user$ whereis fstab
fstab: /etc/fstab /usr/include/fstab.h /usr/share/man/man5/fstab.5.gz
```

Команда locate

Команда locate ищет файлы, в полном названии которых (путь + имя файла) содержится указанная схема поиска. Поиск осуществляется очень быстро: locate не ищет по файловой системе, а обращается к базе данных, в которой содержится список имен всех файлов из файловой системы. В зависимости от дистрибутива locate показывает все файлы, к которым пользователь может получить доступ. Если вы ищете системные файлы, войдите в систему как администратор и выполните команду locate. Использовать ее можно только в том случае, когда в системе установлен соответствующий пакет, который по умолчанию имеется не во всех дистрибутивах.

Примеры

Следующая команда ищет конфигурационный X-файл xorg.conf:

```
user$ locate xorg.conf
/etc/X11/xorg.conf
/etc/X11/xorg.conf.backup
/etc/X11/xorg.conf~
/usr/share/man/man5/xorg.conf.5x.gz
```

Поиск dvips (возможен, если установлен и этот пакет, и LaTEX) выдает много результатов, так как результат поискового запроса имеется во многих названиях каталогов. Все эти результаты не отображаются, а подсчитываются с помощью параметра wc.

```
user$ locate dvips | wc -l
   421
```

Количество результатов значительно уменьшится, если искать только те файлы, названия которых оканчиваются на dvips:

```
user$ locate '*dvips'
/usr/bin/dvips
```

```
/usr/bin/odvips
/usr/bin/opdvips
/usr/bin/pdvips
/usr/local/texmf/dvips
/usr/local/texmf/fonts/map/dvips
...
```

Команда updatedb

Качество результатов поиска зависит от актуальности базы данных, используемой командой `locate`. В большинстве дистрибутивов эта база ежедневно обновляется с помощью команды `updatedb`. Разумеется, `updatedb` можно выполнять вручную всякий раз, как это потребуется. Для этого необходимо обладать правами администратора.

Детали, характерные для определенных дистрибутивов

Внедрение команд `locate` и `updatedb` отличается от дистрибутива к дистрибутиву. В Debian, Fedora и Ubuntu эти команды входят в состав стандартного пакета mlocate. Файловая база данных находится в файле `/var/lib/mlocate/mlocate.db` и ежедневно обновляется программой Cron-Job `/etc/cron.daily/mlocate`. Конфигурационный файл `/etc/updatedb.conf` определяет, какие каталоги и файловые системы не будут учитываться при поиске (например, CD, DVD, различные буферные каталоги).

В стандартной сборке openSUSE команда `locate` отсутствует. Прежде чем вы сможете пользоваться этой поисковой командой, потребуется установить пакет `findutils-locate` и один раз выполнить `updatedb`, будучи в системе с привилегиями администратора. В дальнейшем эта команда будет ежедневно выполняться программой Cron-Job `/etc/cron.daily/suse.de-updatedb`. Конфигурация осуществляется через `/etc/sysconfig/locate`.

Команды find и grep

Команда `find` очень мощная, но не менее сложная, чем описанные выше команды для поиска файлов. Она учитывает различные поисковые критерии (схемы поиска по именам файлов, размеру файлов, датам создания файла или последнего обращения к нему и т. д.). Полная справка по параметрам этой команды содержится в `man find`. Однако лучше всего будет продемонстрировать работу с `find` на примерах, приведенных ниже. Учитывайте, что команда `find` работает сравнительно медленно, так как она просматривает всю файловую систему каталог за каталогом.

Команда find

Если не указывать дополнительные параметры, `find` выдает список всех файлов, находящихся в текущем каталоге и всех его подкаталогах:

```
user$ find
...
```

Следующая команда ищет все файлы в текущем каталоге и во всех подкаталогах, названия которых начинаются с .e:

```
user$ find -name '.e*'
./.evolution
./.emacs
./.emacs~
./.esd_auth
...
```

Команда find производит поиск, начиная с каталога /usr/share/texmf, и ищет все файлы вида *.tex в каталоге, название которого оканчивается на latex.

```
user$ find /usr/share/texmf -path '*latex/*.tex'
/usr/share/texmf/ptex/platex/base/plnews03.tex
/usr/share/texmf/ptex/platex/base/kinsoku.tex
...
```

В следующем примере find ищет все каталоги, находящиеся в /etc/. Обычные файлы, находящиеся в /etc/., среди результатов не показываются. Список результатов упорядочивается по алфавиту с помощью команды sort (по умолчанию такой сортировки не происходит).

```
root# find /etc -type d | sort
/etc
/etc/acpi
/etc/acpi/actions
...
```

В следующем примере find ищет все файлы в (под)каталогах /home, принадлежащих пользователям группы users, причем искомые файлы должны были каким-либо образом быть изменены в течение последних пяти дней (содержание, права доступа и т. д.). Параметр -ctime+5 позволяет находить файлы, измененные ранее, чем пять дней назад, а -ctime 5 — те файлы, которые были изменены именно пять дней назад.

```
root# find /home -group users -ctime -5
...
```

Следующая команда удаляет все резервные копии, содержащиеся в данном каталоге и во всех подкаталогах. При этом find строит список всех сомнительных файлов и передает его команде rm через подстановку команды ($(*команда*)).

```
user$ rm $(find . -name '*~')
```

Если речь идет об *очень* большом количестве файлов, то при выполнении вышеуказанной команды возникает ошибка: запись со всеми *~-файлами может получиться такой длинной, что превысит размер командной строки. В таких случаях следует воспользоваться либо параметром -exec- команды find, либо командой xargs.

Команда grep

Эта команда ищет в текстовом файле соответствия для заданной поисковой схемы. В зависимости от того, какие параметры настроены в конкретном случае, команда может дополнительно показывать найденные фрагменты текста или же просто сообщать, в каком количестве строк была найдена заданная поисковая схема. Поисковая схема является так называемым регулярным выражением.

Следующая команда просматривает все TEX-файлы текущего каталога в поисках последовательности символов `emacs`. Список всех найденных строк (перед каждой из которых указывается имя файла) отображается на экране.

```
user$ grep emacs *.tex
...
```

Команда `grep` определяет, как часто применяется функция `arctan` в указанных C-файлах.

```
user$ grep -c arctan\(.*\) *.c
```

Команда `grep` с параметром `-v` возвращает в качестве результата все строки, в которых отсутствует заданный шаблон поиска. В приведенном далее примере `grep` удаляет из `configfile` все строки, которые начинаются с символа # (то есть все комментарии). Последующая команда `cat` дополнительно удаляет все пустые строки. Конечный результат сохраняется в файле `nocomments`. Эта команда очень удобна, когда всего несколько строк конфигурационного кода приходятся на сотни или тысячи строк комментариев.

```
user$ grep -v '^#' configfile | cat -s > nocomments
```

Комбинирование find и grep

Команды `find` и `grep` можно комбинировать, чтобы выполнять расширенный поиск. В следующем примере команда `find` просматривает все файлы на предмет того, нет ли в них последовательности символов `emacs`. Если такая последовательность обнаруживается, то название файла выводится на экран. Обратите внимание на то, что нельзя указывать параметр `-print` перед `-exec` (в отличие от предыдущего примера, команда `grep emacs *.tex` учитывает все файлы с расширением TEX независимо от глубины вложения подкаталога, в котором они могут находиться).

```
user$ find -name '*.tex' -type f -exec grep -q emacs {} \; -print
...
```

Следующая команда просматривает в текущем каталоге все файлы размером менее 10 Кбайт на предмет наличия в них регулярного выражения `case.*in`. Список найденных файлов сохраняется в файле `resultat`. Ограничение размера файла вводится для того, чтобы исключить из поиска двоичные файлы (обычно они гораздо больше 10 Кбайт).

```
user$ find -name '*' -maxdepth 1 -size -10k -exec grep -q \
> case.*in {} \; -print > ergebnis
```

Поисковики для персональных компьютеров

В последнее время в Linux, как и в других системах, стали популярны системы для локального поиска (поиска по ПК). Наибольшее распространение получили программы Beagle и Tracker. Компания Google также позаботилась о создании специальной версии локального поисковика Google Desktop для Linux, но пока эта программа остается в арьергарде. По следующим адресам можно найти популярные поисковики для персональных компьютеров:

- http://www.beagle-project.org/;
- http://projects.gnome.org/tracker/;
- http://desktop.google.com/linux/.

По сравнению с рассмотренными выше текстовыми поисковыми командами системы локального поиска обладают целым букетом преимуществ.

- Поиск быстр, так как применяется механизм, сходный с работой команды `locate`, — анализируются заранее созданные индексные файлы.
- Поисковый индекс собственных файлов обновляется при каждом изменении файла, поэтому он всегда актуален.
- Поисковые алгоритмы приспособлены к работе с важнейшими двоичными форматами файлов, поэтому поисковые программы находят и тексты, содержащиеся в документах OpenOffice, и PDF-файлы, и электронные письма.
- Работа с такими программами не составляет труда даже для новичков, так как эти поисковики хорошо интегрированы в KDE или Gnome.

Есть, конечно, и недостатки: постоянное обновление поискового индекса затрудняет работу системы, и программы, занятые обработкой большого объема информации, могут существенно тормозить. Кроме того, индексные файлы локального компьютера занимают сравнительно много места. Я так и не смог сделать окончательный выбор между Beagle и Tracker.

Программа Beagle

С Beagle вы встретитесь прежде всего в дистрибутивах SUSE, в которых система поиска устанавливается по умолчанию. В Gnome для поиска применяется пользовательский интерфейс `beagle-search`, в KDE — программа `kerry`. Даже те пользователи, которые работают с консолью, могут применять Beagle с помощью команды `beagle-query`.

По умолчанию Beagle выдает результаты поиска, содержащие результаты *всех* заданных поисковых запросов, содержащиеся в имени или содержании файла. Кроме того, существуют различные возможности формулирования и ограничения поиска, например *слово1* `OR` *слово2* ищет документы, в которых содержится как минимум один поисковый запрос из двух; *слово1* `-`*слово2* ищет документы, в которых содержится первый запрос, а второй отсутствует; `.pdf` означает, что поиск ведется только в PDF-документах.

Следует упомянуть еще о некоторых технических деталях: поисковик Beagle написан на C#, поэтому предназначен для установки с Mono. Файловая система, по которой проводится поиск, должна поддерживать *расширенные атрибуты*. Это означает, что для каждого файла сохраняется дополнительная информация (атрибуты) — подробнее об этом в разделе 3.8. С помощью таких атрибутов Beagle помечает, какие каталоги и файлы были проиндексированы и не изменились ли они с тех пор.

Начиная с версии ядра 2.6, расширенные атрибуты предоставляются по умолчанию для систем ext2/3/4, xfs и NFS4. Конечно же, сама по себе поддержка расширенных атрибутов еще не означает, что эта функция действительно активна во всех системах. Например, в системах ряда ext для активизации необходим параметр `user_xattr`. Взгляните на четвертый столбец в файле `/etc/fstab`. Если этого параметра там нет, добавьте его. Изменения вступят в силу только после новой привязки файловой системы или после перезагрузки. Справочная информация по `/etc/fstab` содержится в разделе 13.5.

Программа Tracker

Считается, что Tracker немного быстрее и эффективнее, чем Beagle, не в последнюю очередь потому, что эта программа обходится без Mono. После установки необходимо выйти из системы и снова войти. После того как Tracker потратит некоторое время на индексирование файловой системы, начинайте поиск с помощью инструмента `tracker-search-tool` (рис. 3.1).

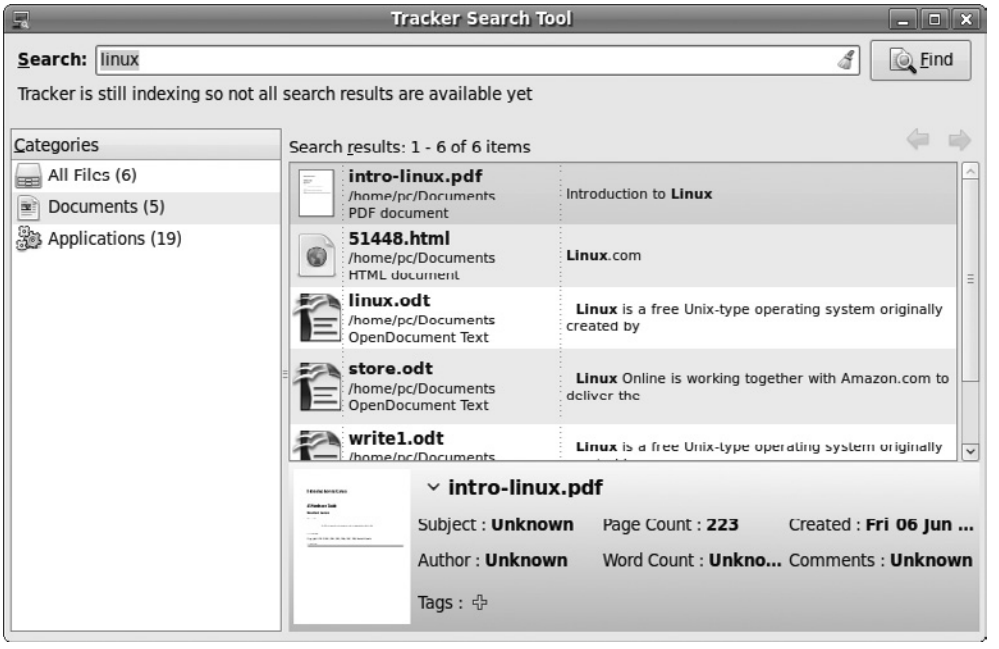

Рис. 3.1. Программы Tracker для локального поиска

Tracker сохраняет свои индексные файлы в каталоге ~/.cache/tracker. По умолчанию поиск производится только по файлам из домашнего каталога. Если вы хотите включить в поиск определенные каталоги или, наоборот, исключить их, измените конфигурацию в tracker-preferences.

Программа Strigi

Еще одна система локального поиска, созданная специально для KDE, — это Strigi. Во многих дистрибутивах Strigi необходимо сначала установить (пакеты strigi*), а потом активизировать с помощью параметра STRIGI-FILE-INDEXER в специальном системном управляющем модуле. Сам поиск осуществляется не самой интересной программой, которая называется strigiclient. Использование Strigi в Gnome обеспечивается минипрограммой Deskbar. Я признаюсь, что ничего хорошего о программе Strigi сказать не могу и не рекомендую ею пользоваться.

3.5. Резервное копирование

Винчестеры не вечны, ноутбуки иногда крадут, а если нечаянно выполнить команду rm-rf, то ущерб также может причинить большой вред. По этим и другим причинам вам никак не обойтись без регулярного резервного копирования всей персональной информации. В этом разделе вашему вниманию предлагается целый спектр команд и программ, которые в различной форме помогают архивировать файлы и предотвращать потери информации.

Я, например, рекомендую такую стратегию резервного копирования: все мои компьютеры связаны с центральным сервером, по крайней мере иногда такая связь существует. Раз в неделю я с помощью команды rsync синхронизирую каталоги со своими личными файлами с определенным каталогом на сервере. С самыми важными файлами, с которыми я работаю ежедневно, такое резервное копирование выполняется также ежедневно. Таким образом, все важные файлы находятся на двух разных компьютерах, а значит, на двух независимых друг от друга жестких дисках. Примерно раз в месяц целый сегмент диска сервера синхронизируется с внешним USB-носителем, который хранится отдельно от жестких дисков (в другом помещении). И вот за последние 15 лет мне пришлось попрощаться с несколькими винчестерами из-за неисправностей оборудования, но значительные объемы файлов не терялись никогда.

Сжатие и архивирование файлов

В Windows для сжатия одного или нескольких файлов обычно применяется архиватор WinZIP или совместимая с ним программа. В Linux также есть подобные пользовательские интерфейсы: например file-roller (Gnome) или ark (KDE). В этом разделе будут продемонстрированы некоторые альтернативы, позволяющие архивировать файлы с помощью специальных команд, которые собраны в табл. 3.7.

Таблица 3.7. Инструменты для сжатия и архивирования файлов

Название устройства	Значение
gzip	Архивирует файл
gunzip	Извлекает файл из архива
bzip2	Архивирует файл (сильное, но более медленное сжатие)
bunzip2	Извлекает файл из архива bzip2
tar	Создает или извлекает файловый архив
zip	Создает совместимый с Windows ZIP-архив
unzip	Извлекает ZIP-архив
zipinfo	Указывает информацию по определенному ZIP-архиву

Команды gzip и gunzip

Команда `gzip` архивирует файлы, заданные в виде параметров, и переименовывает их по принципу *имя*`.gz`, где *имя* — имя файла. Команда `gunzip` выполняет противоположную операцию. Обе команды используют так называемый алгоритм Лемпеля-Зива (LZ77), особенно удобный при работе с текстовыми файлами (но непригоден для работы с аудио- и видеофайлами). Разумеется, архивация происходит без потери данных, то есть при извлечении из архива восстановленный файл имеет тот же вид, что и до архивации. На следующих примерах показана работа этих команд:

```
user$ ls -l filesystem.tex
... 178794 1. Aug 17:43 filesystem.tex
user$ gzip filesystem.tex
user$ ls -l filesystem.tex.gz
... 57937 1. Aug 17:43 filesystem.tex.gz
user$ gunzip filesystem.tex.gz
```

Команды bzip2 и bunzip2

Это команды, альтернативные `gzip`/`gunzip`. Достоинство данных команд заключается в более сильной архивации, но они немного медленнее работают. Файлы, заархивированные таким образом, имеют расширение `.bz2`.

```
user$ bzip2 filesystem.tex
user$ ls -l filesystem.tex.bz2
... 47105 1. Aug 17:43 filesystem.tex.bz2
user$ bunzip2 filesystem.tex.bz2
```

Команда tar

Команда `tar` обычно применяется в Linux для того, чтобы объединять несколько файлов в архиве, создаваемом, как правило, командами `gzip` или `bzip2`. Сначала `tar` предназначалась для записи файлов на стримерный накопитель и считывания информации с него. Поскольку такие устройства сегодня применяются уже достаточно редко, я опишу только то, как эта команда используется с файловыми архивами.

Следующая команда помещает все файлы из каталога buch в сжатый архивный файл `buch.tgz`. Коротко расскажу о буквенных названиях параметров: `c` означает

«создать» (create), то есть таким образом tar создает архив; z означает zip, то есть архив должен создаваться с помощью gzip; f означает «файл» (file), то есть tar должна создать файл архива (а не записывать архив на кассету стримера). В дополнение к этому параметру указывается имя файла. Обычно такие файлы имеют расширение .tar.gz, или коротко .tgz.

```
user$ tar -czf meinarchiv.tgz buch/
```

Команда tar -tzf возвращает содержимое архива. Данные в архиве располагаются произвольно. В большинстве дистрибутивов less сконфигурирована так, что содержимое архива можно просмотреть, просто задав команду less имя.tgz.

```
user$ tar -tzf meinarchiv.tgz
linux8/
linux8/lanserver.tex
linux8/security.tex~
linux8/buch.tex
linux8/u4.txt~
...
```

Команда tar -xzf распаковывает архив и извлекает содержащиеся в нем файлы:

```
user$ cd другой_каталог/
user$ tar -xzf meinarchiv.tgz
```

В следующем примере tar извлекает из архива только TEX-файлы. Обратите внимание на апостроф рядом со схемой файла, предотвращающий мгновенную обработку таких файлов оболочкой.

```
user$ tar -xzf meinarchiv.tgz '*.tex'
```

Если вы хотите создать архив с помощью bzip2, а не с помощью gzip, измените параметр z на j.

Команда zip

В мире UNIX/Linux файлы TAR — это доминирующий формат для передачи файловых архивов. Однако если вы работает с пользователями Windows, то ZIP-архив — это оптимальный выбор. Следующая команда помещает все HTML-файлы, переданные в виде параметров, в архив myarchive.zip.

```
user$ zip meinarchiv.zip *.html
```

Если вы хотите заархивировать содержимое целых каталогов, укажите параметр -r:

```
user$ zip -r meinarchiv.zip mywebsite/
```

Содержимое ZIP-файла можно просмотреть с помощью команды zipinfo:

```
user$ zipinfo meinarchiv.zip
Archive: test.zip 143677915 bytes 1899 files
-rw-r--r-- 2.3 unx 78039 tx defN 10-Jul-06 11:27 linux8/lanserver.tex
-rw-r--r-- 2.3 unx 115618 tx defN 7-Apr-05 15:58 linux8/security.tex~
-rw-r--r-- 2.3 unx 3899 tx defN 28-Jul-06 16:38 linux8/buch.tex
-rw-r--r-- 2.3 unx 752 tx defN 11-Feb-04 12:06 linux8/u4.txt~
...
```

Для извлечения архива используйте следующие команды:

```
user$ cd другой_каталог/
user$ unzip meinarchiv.zip
```

Синхронизация каталогов

Команда rsync первоначально разрабатывалась для синхронизации сетевых каталогов. Специальные возможности использования этой команды при работе с сетями перечислены в подразделе «RSYNC» раздела 6.3. Здесь мы поговорим лишь о том, как синхронизировать два локальных жестких диска.

Следующая команда копирует все JPG-файлы из одного каталога в другой. В отличие от работы с командой cp, те файлы, которые уже имелись в системе при последнем копировании и не изменились с тех пор, заново не копируются. Если вы делаете небольшую резервную копию, этим можно пренебречь, но при синхронизации гигантских систем каталогов размером во многие гигабайты разница более чем существенна!

```
user$ rsync каталог1/*.jpg каталог2/
```

Чтобы синхронизировать целый каталог вместе со всеми его подкаталогами, используйте параметр -a, который является сокращенным вариантом целого ряда других параметров (-rlptgoD). Параметр вызывает рекурсивную обработку всех подкаталогов и гарантирует, что информация (владелец, принадлежность к определенной группе, время последнего изменения и т. д.) сохранится в максимально полном объеме. Если *каталог2* не существует, то этот каталог создается.

```
user$ rsync -a каталог1/ каталог2/
```

По умолчанию rsync копирует или обновляет все новые или измененные файлы, но ничего не удаляет. Если вы хотите, чтобы файлы, удаленные из *каталог1*, также были удалены из *каталог2*, дополнительно укажите параметр --delete. Само собой разумеется, что этот параметр опасен: если случайно удалить каталог, то именно он удалится при следующей операции резервного копирования и со второго жесткого диска! Однако в долгосрочной перспективе вам без --delete не обойтись. Если этим параметром не пользоваться, со временем накопится масса уже ненужной информации.

Если вы хотите обеспечить надежную подачу данных о том, какие файлы были изменены и сколько файлов (с какой скоростью и куда именно) было перенесено, дополнительно укажите параметр -v.

Автоматизация работы rsync. Чтобы не вводить команду rsync многократно, лучше всего запаковать ее в небольшой сценарий командного процессора. Тогда вы гарантированно сможете избежать опечаток. Основы программирования сценариев будут рассмотрены в разделе 8.8.

```
#!/bin/sh
mount /backup # Привязка резервного каталога NFS к файловой системе
rsync -av --delete /folder1/* /backup/rechnerX/folder1/
rsync -av --delete /folder2/* /backup/rechnerX/folder2/
```

Резервное копирование будет еще удобнее и надежнее, если автоматизировать его запуск программой Cron-Job. Введение в Cron-Job дается в разделе 4.6.

Инкрементное резервное копирование

Интересной альтернативой rsync является команда rdiff-backup. Ее важнейшее отличие от rsync заключается в том, что она сохраняет в резервном каталоге для измененных файлов и старые версии этих файлов. Для экономии места можно сохранять не копию измененного файла, а только внесенные изменения (которые также можно архивировать). Кроме того, rdiff-backup позволяет без особых усилий организовать инкрементное резервное копирование: из получаемых таким образом копий можно восстанавливать также более старые варианты файлов. В принципе функционально rdiff-backup аналогична Time Machine из Apple Mac OS X — отсутствует только симпатичный пользовательский интерфейс.

Выполнение резервного копирования

Проще всего применять rdiff-backup с двумя локальными каталогами. Если целевой каталог еще не существует, он создается.

```
root# rdiff-backup /home /home-backup
```

Команда rdiff-backup создает в каталоге для резервного копирования подкаталог rdiff-backup-data. В нем сохраняются различные статистические файлы и статусная информация. Кроме того, в каталоге increments содержатся старые версии тех файлов, которые были изменены или удалены. При этом сохраняются только изменения (.diff), которые дополнительно архивируются. В имя файла интегрируется дата создания последней версии. Таким образом, получаются сравнительно сложные имена файлов вида имя.2009-04-03T08:37:58+02:00.diff.gz.

Доступ к резервным копиям

Если вы хотите обратиться к резервной копии, то вам нужен каталог /home-backup, отражающий состояние каталога /home на момент последнего резервного копирования (за исключением дополнительного резервного каталога rdiff-backup-data — именно такой результат вы бы получили, если бы выполнили резервное копирование с помощью команды cp-a или rsync -a --delete). Таким образом, открыть последнюю резервную копию совсем не сложно. Конечно, резервную копию можно восстановить и с помощью rdiff-backup. Для этого применяется параметр -r и указание времени now. Следующая команда позволяет восстановить резервную копию во временном каталоге:

```
root# rdiff-backup -r now /home-backhup /tmp/home-aktuell
```

Если вы хотите обратиться к старой версии файла или к недавно удаленному файлу, дело обстоит сложнее: нужно по порядку восстановить все DIFF-файлы (начиная с самого нового), пока вы не получите ту (прошлой) версиюю, которая вам нужна.

Конечно, это необязательно делать вручную — для этой цели существует команда rdiff-backup. Следующая команда восстанавливает каталог /home в таком виде, в котором он был десять дней назад:

```
root# rdiff-backup -r 10D /home-backup/ /tmp/home-historisch
```

Момент резервного копирования на ваш выбор можно задавать или в абсолютном виде (например, 2009-12-31), или в относительном (в часах (h), днях (D), неделях (W) и т. д.). Обратите внимание, что при восстановлении старых файлов с возрастанием количества версий увеличивается нагрузка на процессор, соответственно, работа системы замедляется!

Зачастую вам может потребоваться восстановить старую версию только одного файла или подкаталога. При этом можно указать уже несуществующий файл или удаленный каталог.

```
root# rdiff-backup -r 10D /home-backup/file file-historisch
root# rdiff-backup -r 10D /home-backup/folder/ folder-historisch
```

Удаление всех резервных копий. Если вы регулярно пользуетесь командой rdiff-backup, то каталог с резервными копиями со временем серьезно разрастается. Чтобы удалить все резервные копии, созданные более четырех месяцев назад, сделайте так:

```
root# rdiff-backup --remove-older-than 4M --force /home-backup/
```

Вместо конкретного временного пункта вы также можете указать, сколько времени должны сохраняться заархивированные резервные версии. Следующая команда уменьшает количество резервных версий до трех (независимо от того, насколько давно они были созданы):

```
root# rdiff-backup --remove-older-than 3B --force /home-backup/
```

Сетевое резервирование

Во всех предыдущих примерах мы исходили из того, что исходный и конечный каталоги находятся в файловой системе локального компьютера. Однако команда rdiff-backup может обращаться через сеть и к удаленным каталогам. В отличие от случая работы с rsync, команда rdiff-backup должна быть установлена и на удаленном компьютере! Передача данных осуществляется через SSH. Специальной конфигурации rdiff-backup не требуется.

При указании внешних каталогов применяется практически такой же синтаксис, как при работе с rsync. Единственное отличие заключается в том, что после имени хоста необходимо поставить *два* двоеточия:

```
root# rdiff-backup user@firma-abc.de::/home /home-backup
```

Подробное описание того, как работать с rdiff-backup, а также соответствующие примеры находятся на сайте http://www.nongnu.org/rdiff-backup/.

Программа Duplicity

Если вам, в принципе, нравится идея команды rdiff-backup, но также требуется закодировать резервную копию и обеспечить закачку информации через SSH или

FTP на внешний сервер, обратите внимание на программу Duplicity, записанную на языке Python. Функционально она напоминает `rdiff-backup`, но создает только архивы в формате TAR. В настоящий момент программа проходит бета-тестирование и активно разрабатывается. Информацию можно просмотреть на сайте http://duplicity.nongnu.org/.

Пользовательские интерфейсы для резервного копирования

Компания Apple своей программой Time Machine доказала, что даже приложение для резервного копирования может вызвать восторг пользователя. Linux, к сожалению, не может составить Apple конкуренцию в данном отношении: хотя выбор команд и инструментов для резервного копирования и необозрим, пользователи персональных компьютеров все еще ждут появления гениального и простого инструмента. В Gnome и KDE программа резервного копирования по умолчанию вообще отсутствует. Это тем более поразительно, так как существует несколько исключительно интересных и простых в использовании приложений. В этом разделе мы коротко рассмотрим три следующие программы:

- Grsync (http://www.opbyte.it/grsync/);
- PyBackPack (http://andrewprice.me.uk/projects/pybackpack/);
- DéjàDup (http://mterry.name/deja-dup/, https://launchpad.net/deja-dup).

В настоящее время ведется активная разработка всех трех программ. Однако ситуация может быстро измениться: именно среди программ, предназначенных для резервного копирования, существует множество брошенных проектов, которые не обновляются годами. Старайтесь не пользоваться «островными» решениями и при необходимости своевременно меняйте свой инструмент для резервного копирования!

Grsync

На самом деле Grsync не является программой для резервного копирования в полном смысле этого слова. Это всего лишь очень простой пользовательский интерфейс для работы с командой `rsync`. После установки вы связываете с компьютером внешний жесткий диск или USB-флешку, запускаете Grsync и копируете содержимое вашего домашнего каталога в каталог на внешнем жестком диске. При первом запуске следует скопировать все файлы, в ходе последующих запусков — только измененные или новые файлы.

Многие параметры, по сути, могут быть оставлены в таком виде, как они настроены по умолчанию. Однако о двух параметрах следует рассказать подробнее.

- Флажок Удалить в целевом каталоге определяет, должна ли программа Grsync синхронизировать процессы удаления. Если после первого резервного копирования вы удалите файл в вашем домашнем каталоге, то этот файл при следующем прогоне будет удален и в резервном каталоге. Если необходимо защитить резервный каталог от нечаянного удаления, то этот флажок можно не устанавливать (такая настройка также сделана по умолчанию). Его необходимо установить

лишь в том случае, когда содержимое вашего каталога и страхующего его резервного каталога должны полностью совпадать.

- Установка флажка Не покидать файловую систему означает, что Grsync будет синхронизировать только те файлы, которые находятся в файловой системе исходного каталога. Обычно стоит устанавливать этот флажок.

Основными достоинствами Grsync являются простота в использовании и тот факт, что ваши файлы будут копироваться в резервный каталог «один к одному». Кроме того, для обращения к резервной копии не требуются никакие специальные инструменты.

PyBackPack

Программа предназначена для того, чтобы сохранять резервную копию домашнего каталога на CD или DVD в любом каталоге (например, на внешнем диске) или на сервере, доступ к которому обеспечивается через SSH. Интерфейс этой программы существует только на английском языке, состоит из трех вкладок. Если вы хотите просто делать резервные копии вашего домашнего каталога и сохранять их на CD или DVD, выберите на вкладке Home вариант Go.

Если вы хотите просто создать резервные копии определенных каталогов или вам нужно другое место для резервного копирования, перейдите на вкладку Backup. Кнопка New позволяет создать новый набор резервных копий. При этом вы указываете, какие каталоги и где именно хотите сохранить, а какие каталоги должны быть исключены из резервного копирования. Нажатие кнопки Backup запускает резервное копирование. Если PyBackPack не сможет прочитать отдельные файлы по причине отсутствия необходимых прав доступа, будет показано соответствующее предупреждение, но копирование остальных файлов будет продолжено (рис. 3.2).

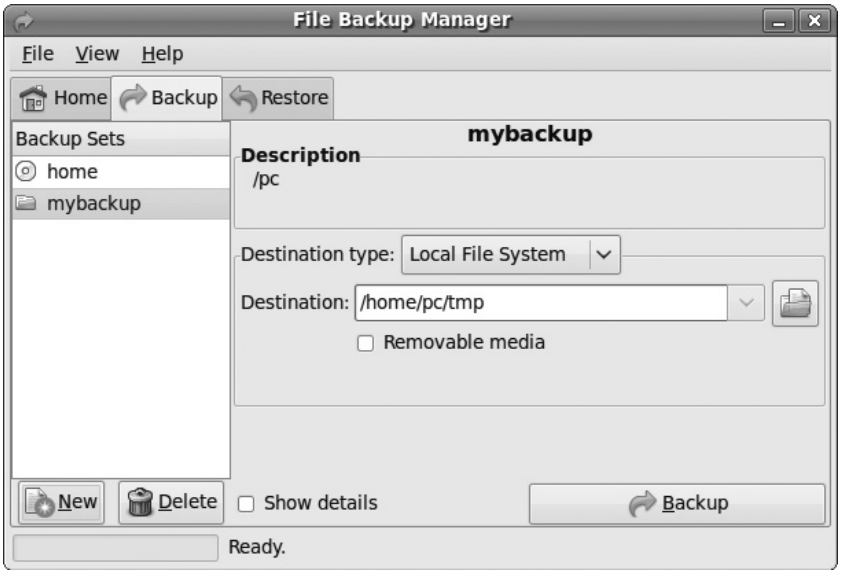

Рис. 3.2. Окно программы PyBackPack

На фоне выполнения программы PyBackPack выполняется уже известная вам команда `rdiff-backup`. Однако в PyBackPack не предусмотрена возможность удаления устаревших резервных копий, поэтому при каждом следующем резервном копировании на этот каталог требует все больше места. Простейший способ справиться с этой проблемой — создавать при каждом резервном копировании новый каталог, а старый удалять.

DéjàDup

Эта программа, как и PyBackPack, разработана для максимально простого создания резервной копии в каталоге на локальной машине или в сети. В качестве места для резервного копирования очень хорошо подходит сервер Amazon-S3. С помощью команды Правка ▸ Настройки можно указать, где следует сохранить каталог и на какие каталоги будет распространяться резервное копирование, а на какие — нет. Резервное копирование запускается нажатием кнопки Создание резервной копии.

Если установить флажок Регулярно создавать резервные копии, DéjàDup на ваш выбор будет осуществлять копирование ежедневно, еженедельно, раз в две недели или раз в месяц. В любом случае резервное копирование осуществляется только тогда, когда вы вошли в систему. Запуск фоновой программы, выполняющей сохранение, инициируется вводом логина.

По сравнению с другими программами резервного копирования DéjàDup, к сожалению, затрачивает на первое копирование данных несравнимо больше времени. Причина этого заключается в том, что по умолчанию резервная копия при создании сжимается и зашифровывается. Если шифрование вам не требуется, то соответствующий флажок необходимо снять! При создании следующих копий программа сохраняет только сделанные изменения и время экономится. На настоящий момент DéjàDup не умеет удалять ненужные резервные копии.

Кнопка Восстановить позволяет полностью восстановить резервную копию. При этом можно задать нужную версию резервной копии и указать, куда должны быть скопированы резервные файлы. Если вы хотите восстановить лишь один файл или каталог, щелкните на нем правой кнопкой мыши в менеджере Nautilus и выполните команду контекстного меню Восстановить старую версию.

В перспективе DéjàDup должна стать стандартной программой для резервного копирования в Ubuntu. Работа программы основана на созданном на языке Python сценарии для резервного копирования Duplicity. Недостаток заключается в том, что файлы резервных копий создаются в очень специфическом формате, так что восстанавливать их можно только с помощью программы DéjàDup или сценария Duplicity.

3.6. Запись CD и DVD

Если вы записываете диски лишь иногда, от случая к случаю, то используйте программы Brasero и K3B — в них вы найдете все необходимые функции, к тому же с ними легко работать. Если хотите работать с большим комфортом, обратите внимание на коммерческую программу Nero, то есть на ее версию для Linux. Если же

вы ищете пользовательскую программу для работы с текстовой консолью, то вам очень понравится команда `burncdda`.

В этом разделе будут рассмотрены команды, которые выполняются на фоне основной программы. Они могут быть интересны, например, в тех случаях, когда вам нужен автоматизированный сценарий для записи CD с резервными копиями. Как и во многих других случаях, понимание принципа работы этих команд поможет вам лучше понимать саму концепцию работы Linux.

Названия устройств. Прежде чем начать работу с командами, описанными ниже, нужно знать название устройства; по такому названию вы будете обращаться к приводу. Как правило, верное название устройства таково: `/dev/scd0`, `/dev/scd1` и т. д. или `/dev/sr0`, `/dev/sr1` и т. д. Если ни ваш компьютер, ни версия установленного на нем дистрибутива Linux не являются новейшими, правильное название устройства, напротив, будет следующим: `/dev/hda`, `/dev/hdb` и т. д. Обзор номенклатуры устройств дается в разделе 13.10.

Для некоторых команд название устройства необходимо указывать в виде тройки чисел (например, `dev=3,0,0`). В эту тройку входят: номер шины SCSI (обычно `0`), SCSI-ID устройства и, наконец, *логический номер устройства* (коротко ЛНУ, обычно также `0`). Верную комбинацию чисел для вашего привода вам проще всего будет узнать с помощью команды `readcd -scanbus`.

ПРИМЕЧАНИЕ

Независимо от того, какой именно диск вы записываете — CD или DVD, вам пригодится следующее указание: убедитесь, что дистрибутив Linux не встраивает носитель с данными в дерево каталогов и не обращается к нему каким-либо иным способом! Механизмы автоматического связывания CD и DVD с системой, отличающиеся от дистрибутива к дистрибутиву, часто не позволяют записывать диски с носителей данных вручную.

Применение внешних носителей (USB, Firewire), по моему опыту, гораздо чаще чревато проблемами, нежели использование внутренних приводов. Это справедливо не только для Linux, но и для Windows. Если все же приходится работать с внешним записывающим устройством, нужно значительно снизить рабочую скорость записи.

Создание и тестирование ISO-образов

Прежде чем вы сможете записать данные на CD или DVD, вам потребуется так называемый *образ диска* (ISO). Это файл, содержащий другие файлы, подготовленные для записи во внутреннем формате оптического носителя данных. Как правило, для создания образов дисков используется команда `genisoimage` (ранее — `mkisofs`). Когда же речь идет о копировании данных с имеющихся носителей, интересную альтернативу представляют собой команды `dd` или `readcd`, коротко описанные в конце этого раздела.

Команда genisoimage

Команда `genisoimage` позволяет записать на диск содержимое одного или нескольких каталогов. Формат ISO-9660, предусмотренный для CD, оперирует собственным очень ограниченным набором символов, в котором допускаются лишь немногие символы, кроме тех, что относятся к кодировке ASCII. Чтобы справиться с этим

недостатком, существует множество расширений стандарта ISO, наиболее распространенными из которых являются следующие два. Оба они поддерживаются genisoimage.

- Расширение Rockridge, обычное в системах UNIX/Linux, обеспечивает сохранение длинных названий файлов в форме любых, нуль-терминированных последовательностей символов. Кроме того, это расширение позволяет сохранять права доступа (UID, GID, биты доступа).

  В любом случае диски, записанные с помощью Rockridge, не содержат информации о том, в какой кодировке создавался образ диска. Это может вызывать проблемы, если позже носитель данных будет использоваться на компьютере, который поддерживает другую кодировку. Проще всего объяснить это на примере: пару лет назад в Linux еще широко применялась кодировка Latin-1. Диски с данными, создававшиеся тогда с помощью расширения Rockridge, записывались именно в этой кодировке. Если сегодня вы используете такие диски на одном из современных дистрибутивов, в которых применяется кодировка Unicode (UTF-8), то символы, которые встречаются в названиях файлов и не относятся к ASCII, будут интерпретированы неправильно. Если уже перед подготовкой ISO-образа известно, что создаваемый диск будет использоваться на компьютере с другой кодировкой, то можно настроить нужную кодировку с помощью параметра -output-charset.

- Расширение Joliet, которое часто используется и в системах с Windows, также позволяет сохранять длинные названия файлов, при этом применяется кодировка Unicode (UTF-16).

Примеры. Следующая команда записывает все файлы, содержащиеся в каталоге /master, в файл /tmp/master.iso. Сам каталог master *не* хранится в образе диска. Образ диска использует расширение Rockridge (параметр -r), а также расширение Joliet (параметр -J) и получает название Linux (параметр -V). Если вы записываете ISO-образ на CD, то данная последовательность символов становится именем диска.

```
user$ genisoimage -o /tmp/master.iso -r -J -V Linux /master
```

Второй пример напоминает первый, но теперь мы создаем загрузочный диск:

```
user$ genisoimage -o /tmp/master.iso -r -J /master -b images/boot.img -c boot.
catalog
```

В третьем примере каталог master сам становится каталогом в образе диска (параметр -graft-points):

```
user$ genisoimage -o /tmp/master.iso -r -graft-points /master=/master
```

СОВЕТ

Если вы не работаете с параметром –r, следите за тем, чтобы все файлы каталога master относились к корневому каталогу и прочитывались!

```
user$ chown -R root.root /master
user$ chmod -R a+r /master
```

Команда dd

Если вы хотите скопировать без изменений CD или DVD с данными (но не аудиодиск!), вам хватит одной команды dd, чтобы создать необходимый для этого файл образа диска. Вместо /dev/cdrom укажите имя устройства вашего CD- или DVD-привода, которое отличается в зависимости от дистрибутива.

```
user$ dd if=/dev/cdrom of=/usr/local/iso.img bs=2048
```

Команда readcd

Альтернативой для dd является команда readcd или readom. При этом для считывания информации с CD используются команды SCSI, это дает те же результаты, что и применение dd. Я проводил испытания на двух разных компьютерах, и в обоих случаях readcd выдавала бесчисленные сообщения об ошибках. В отличие от работы с dd, требовалось указать привод CD или DVD с помощью тройки чисел. Правильную комбинацию чисел для вашего привода можно узнать с помощью команды readcd -scanbus.

```
user$ readcd dev=0,0,0 f=iso.img
```

Достоинство readcd заключается в том, что она также может создавать TOC-файл (при работе с аудиодисками — с параметром -clone) и в зависимости от указанного параметра будет по-разному обращаться с ошибками считывания (параметры -noerror и -noclone). С помощью параметра -w можно применять readcd и для записи CD.

Тестирование ISO-образа

С помощью так называемого *петлевого устройства* ядра Linux можно рассмотреть любой файл как файловую систему и, используя команду mount, связать его с деревом каталогов. Функция петлевого устройства находится в модуле ядра loop, который входит в состав всех наиболее распространенных дистрибутивов. Если этот модуль не загружается автоматически, попробуйте вариант с modprobe (см. подраздел «Команды для управления модулями» раздела 15.1). Следующая команда связывает файловую систему образа диска, содержащуюся в файле master.iso, с деревом каталогов в режиме «только для чтения»:

```
root# mkdir /iso-test
root# mount -t iso9660 -o loop,ro /tmp/master.iso /iso-test/
```

Теперь можно просмотреть в каталоге iso-test содержимое будущего CD.

Запись CD

Программа cdrecord и команда wodim

Уже более десяти лет команда cdrecord используется для записи CD. С лета 2006 года разработчик cdrecord Йорг Шиллинг (Jörg Schilling) использует при работе с некоторыми составляющими одноименного пакета лицензию компании Sun под названием CDDL. Многие другие разработчики считают, что эта лицензия несовместима с GPL. Именно поэтому случилось так называемое ветвление, то есть раскол проекта: последняя версия cdrecord, соответствующая GPL, послужила основой

для команды wodim (write data to optical disk media)[1], относящейся к новому проекту cdrkit. Параллельно с cdrecord существует еще два ветвления, две новые команды: из mkisofs получилась genisoimage, а из cdda2wav — icedax.

Во всех нынешних дистрибутивах применяются команды wodim, genisoimage и icedax. Однако из соображений совместимости cdrecord можно вызывать по старому названию cdrecord; /usr/bin/cdrecord — это только ссылка на wodim.

Перед тем как записать компакт-диск с данными, вам, как правило, потребуется записать ISO-образ с помощью genisoimage. Следующие команды сначала симулируют запись CD с данными (-dummy), а потом осуществляют его на самом деле:

```
root# wodim -dummy -v speed=16 dev=/dev/scd0 iso.img
root# wodim -v speed=16 dev=/dev/scd0 iso.img
```

Если компьютер достаточно быстрый, вы можете связать genisoimage и wodim с помощью символа вертикальной линии. Так можно сэкономить место для ISO-образа:

```
root# genisoimage -r /master | wodim -v speed=16 dev=/dev/scd0 -
```

Следующая команда создает аудиодиск. Исходными данными являются файлы в формате WAW. Они обрабатываются в алфавитном порядке. Если вы хотите задать другой порядок, нужно перечислить файлы желаемым образом.

```
root# wodim -v speed=16 dev=0,5,0 -pad -dao -audio *.wav
```

Команда cdrdao

Это альтернатива для wodim. Возможности cdrdao не так широки, но она предлагает гораздо больше параметров для считывания и записи аудиодисков. Название команды подсказывает, что информация создается в режиме односеансовой записи (disk at once, сокращенно — DAO).

На практике команда cdrdao чаще всего используется для копирования аудиодисков. Первая команда cdrdao создает файлы data.bin (содержимое компакт-диска) и data.toc (оглавление). Вторая команда записывает эти данные на CD. Сначала нужно сообщить устройству записи дисков нужную тройку чисел. Узнать комбинацию, соответствующую вашему приводу, проще всего с помощью команды readcd -scanbus.

```
user$ cd tmp/
user$ cdrdao read-cd --device 0,0,0 data.toc
user$ cdrdao write --device 0,0,0 --buffers 64 data.toc
```

Верификация дисков с данными

Следующая команда сравнивает содержимое CD с оглавлением каталога master файл за файлом и байт за байтом. Все найденные отличия записываются в файл diff.log, находящийся в домашнем каталоге. Вместо /media/cdrom необходимо указать каталог в вашей файловой системе, к которому будет привязан CD.

```
root# diff -qrd /master /media/cdrom/ >& ~/diff.log
```

[1] Запись данных на оптические диски.

Во втором окне (или во второй консоли) можно проследить процесс создания файла `diff.log` с помощью `tail`. Поскольку в данном случае используются символьные ссылки, не обойдется без сообщений об ошибках, так как символьные ссылки уже не будут указывать в нужное место CD. Однако по-настоящему беспокоиться стоит в том случае, когда отдельные файлы вообще не удается прочитать (ошибка ввода-вывода) или когда оглавление файлов отличается от необходимого (а вы уверены, что не вносили в файл изменений).

```
root# tail -f ~/diff.log
```

Если вы хотите просто протестировать, можно ли прочитать все данные, содержащиеся на диске (вне зависимости от того, что это за данные), выполните следующую команду. Такой тест оправдан, например, в тех случаях, когда вы получили компакт-диск, который, возможно, неисправен (например, это касается установочного диска Linux).

```
root# dd if=/dev/cdrom of=/dev/null
```

Запись DVD

При записи DVD вам на выбор предлагается целый ряд команд или пакетов.

- Самая популярная команда — `dvd+rw-tools`, которую мы кратко рассмотрим в данном подразделе.
- Если с этой командой возникнут проблемы, можно попытать счастья с `wodim`. Она подходит для записи обычных DVD-R и DVD+R, причем ее синтаксис не отличается от применяемого при записи CD. В любом случае `wodim` предлагает меньше параметров для записи на носители DVD+RW или DVD-RW.

Команда dvd+rw-tools

Все команды, показанные далее в этой главе, входят в состав пакета `dvd+rw-tools`. Первоначально он обеспечивал только поддержку форматов DVD+R и DVD+RW (отсюда и название). Однако теперь с его помощью можно записывать также DVD-R и DVD-RW, а также Blu-ray-диски (последнего варианта я не испытывал). Пакет `dvd+rw-tool` по умолчанию установлен во всех дистрибутивах, распространенных в настоящее время. Более подробная информация находится по адресу http://fy.chalmers.se/~appro/linux/DVD+RW/.

Команда growisofs

Это основная команда пакета `dvd+rw-tools`. Она записывает диски DVD+R, DVD+RW, DVD-R, DVD-RW и Blu-ray. Далее я предлагаю некоторую общую информацию по разным типам носителей.

- DVD+R, DVD-R — данные можно записать на диск как при мультисессии. При первой сессии используется команда `growisofs -Z`, при всех остальных сессиях — `growisofs -M`. Удалить записанные данные уже нельзя. Форматирование диска произвести невозможно.
- DVD+RW, DVD-RW — перед первым применением носитель необходимо отформатировать с помощью команды `dvd+rw-format`. Кроме того, можно добавлять

данные на DVD+R/DVD-R в несколько этапов. Если вы хотите записать данные на место информации, уже содержащейся на диске, то просто запустите новый цикл сессий с помощью команды `growisofs -Z`. В отличие от записи CD-RW, заново форматировать DVD в данном случае не требуется.

При записи DVD-RW в зависимости от форматирования поддерживаются режимы *последовательных добавлений* и *ограниченной перезаписи*. Поскольку при работе данной команды используется команда `genisoimage`, большинство параметров этих команд идентичны.

Следующая команда сохраняет содержимое каталога data на DVD. Параметры команды `genisoimage -r` и `-J` влияют на то, что DVD получает длинные названия, необходимые для работы с расширениями Rockridge и Joliet. Вместо имени устройства `/dev/sr`*n* нужно в зависимости от дистрибутива указывать `/dev/scd`*n*.

```
user$ growisofs -r -J -Z /dev/sr0 data/
```

Вторая сессия производится следующим образом (параметр `-M` вместо `-Z`):

```
user$ growisofs -r -J -M /dev/sr0 moredata/
```

ПРИМЕЧАНИЕ

Обратите внимание, что перед началом новой сессии необходимо извлечь DVD, а потом вновь поместить его в привод!

Чтение мультисессионных DVD может протекать с проблемами на некоторых приводах. При работе с DVD-RW необходимо использовать режим ограниченной перезаписи.

Некоторые DVD-ROM-приводы вообще не могут работать с DVD+RW (как с моносессионными, так и с мультисессионными). С помощью команд, указанных ниже, эти проблемы иногда удается устранить. Команда dvd+rw-format записывает на DVD+RW область для вывода (DVD не форматируется, все данные сохраняются!), а dvd+rw-booktype изменяет на диске информацию о типе (Book-type).

```
user$ dvd+rw-format -lead-out /dev/sr0
user$ dvd+rw-booktype -dvd-rom -media /dev/sr0
```

Обычно команда `growisofs` сообщает все параметры, кроме `-Z` или `-M`, команде `genisoimage`, а затем записывает результат выполнения `genisoimage` прямо на DVD. Если вы хотите записать уже имеющийся образ диска (ISO), синтаксис будет такой: `-Z device=isofile`:

```
user$ growisofs -Z /dev/sr0=data.iso
```

Формат dvd+rw-format

DVD+RW и DVD-RW (когда использовать режим ограниченной перезаписи, рассказано ниже) перед первым применением необходимо отформатировать. Эту функцию выполняет команда `dvd+rw-format`:

```
user$ dvd+rw-format /dev/sr0
```

Детали процесса форматирования немного отличаются в зависимости от типа носителя.

- DVD+RW — в данном случае форматируется только начальная область болванки. Ее размер определяется в зависимости от записывающего устройства. Поэтому когда процесс форматирования завершается на отметке около 11,5 (или любой другой отметке меньше 100) — это не ошибка! Форматирование за преде-

лами начальной области автоматически выполняется приводом, как только информация, записываемая на DVD, заполняет всю начальную область (отформатированную предварительно).

- DVD-RW — по умолчанию такие диски форматируются командой `dvd+rw-format` для ограниченной перезаписи. В этом режиме можно перезаписать на диск новую информацию на место уже имеющейся. При этом не требуется форматировать DVD-RW перед каждой новой сессией записи.

Если DVD-RW имеет параметр `-blank`, его можно форматировать и для записи в режиме последовательных добавлений. Этот режим особенно хорош для записи Video-DVD и улучшает совместимость с некоторыми программами-плеерами. В таком режиме команда `growisofs` не может перезаписывать данные. Для этого DVD каждый раз нужно заново форматировать — очень длительный процесс.

Итак, для оптимального взаимодействия с `growisofs` необходимо обязательно форматировать DVD-RW с помощью команды `dvd+rw-format` и без параметра `-blank`!

При форматировании не происходит физического удаления данных. Если вам требуется именно физическое удаление, например из соображений информационной безопасности, лучше выполнить команду вида `-Z device=/dev/zero`. Таким образом информация на носителе стирается полностью.

Команда dvd+rw-mediainfo

Если у вас есть диск и вы не знаете, какого он типа, заполнен ли уже этот диск информацией, и если да, то в каком режиме и с каким количеством сессий, эту информацию можно узнать с помощью команды `dvd+rw-mediainfo`:

```
user$ dvd+rw-mediainfo /dev/sr0
INQUIRY:                [_NEC ][DVD_RW ND-1300A ][1.07]
GET [CURRENT] CONFIGURATION:
 Mounted Media:         1Ah, DVD+RW
GET PERFORMANCE:
 Speed Descriptor#0:    00/221280 Reading@7.8x Writing@2.3x
READ DVD STRUCTURE[#0h]:
 Media Book Type:       92h, DVD+RW book [revision 2]
 Media ID:              RICOHJPN/W01
 Legacy lead-out at:    221280*2KB=453181440
...
```

3.7. Права доступа, пользователи и принадлежность к группам

Linux изначально создавалась как многопользовательская система, поэтому ей требуются механизмы, управляющие тем, кто и к каким данным имеет доступ, кто может изменять такие права и т. д. Основу системы доступа образуют принципы управления пользователями и группами, описанные в разделе 9.4.

Начиная с версии ядра 2.6, Linux поддерживает и расширенный вариант управления правами с помощью *списков контроля доступа* (ACL), которые подробно описаны в разделе 3.8. В этом разделе мы поговорим только о традиционном управлении правами, которые действуют в UNIX-системах уже не одно десятилетие.

Права доступа к файлу

О каждом файле или каталоге сохраняется следующая информация:

- владелец файла;
- группа, к которой относится файл;
- девять битов доступа (`rwxrwxrwx` для `read/write/execute` — для владельца файла, членов группы владельца и всех остальных);
- еще несколько дополнительных битов для выполнения специальных функций.

Как правило, владельцем файла является его создатель. В качестве группы обычно используется основная группа владельца.

Информация о доступе (`r`, `w` и `x`) описывает, кто имеет право читать файл, записывать в него информацию (то есть вносить изменения) и выполнять его. Таким образом, владелец файла получает больше прав, чем другие пользователи. Обычно эта информация называется битами доступа, так как внутри системы она сохраняется как число с разрядной кодировкой.

Биты доступа, владелец файла, а также отнесенность этого файла к той или иной группе можно просмотреть с помощью команды `ls -l`. Для обычного текстового файла `ls` возвращает примерно такой результат:

```
michael$ ls -l файл.txt
-rw-r--r--   1 michael   users    3529 Oct    4 15:43 файл.txt
```

Коротко объясню эти данные: первый символ указывает тип файла (в данном случае для обычного файла, для каталога ставится `d` (directory), `l` — для символьной ссылки (link) и т. д.). Поскольку мы имеем дело с текстовым файлом, первый `x`-бит деактивизирован, то есть файл нельзя выполнять. Все остальные пользователи, независимо от того, относятся ли они к группе users, могут читать этот файл, но не могут его изменять.

Если michael пожелает, чтобы этот файл был доступен для чтения только пользователям группы users и был недоступен для пользователей, не входящих в эту группу, нужно будет деактивизировать последний `r`-бит. Для этого применяется команда `chmod`.

```
michael$ chmod o-r файл.txt
michael$ ls файл.txt -l
-rw-r-----   1 michael  users    3529 Oct    4 15:43 файл.txt
```

Допустим, требуется предоставить доступ к чтению файла *файл*`.txt` только двум пользователям: michael и kathrin. Для этого можно создать новую группу, к которой будут относиться оба этих пользователя. Если michael и kathrin вдвоем работают в группе по разработке документации в определенной фирме, можно назвать эту

группу, например, dokuteam. Дополнительно изменяем групповую отнесенность с помощью команды chgrp:

```
michael$ chgrp dokuteam файл.txt
michael$ ls файл.txt -l
-rw-r-----   1 michael   dokuteam   3529 Oct   4 15:43 файл.txt
```

Восьмеричное представление

Можно не использовать запись rwxrwxrwx, а представить восемь битов доступа и еще три специальных бита в восьмеричном виде: биты доступа для пользователей, группы и всех остальных пользователей соответственно представляются в виде одной цифры. Каждая цифра составляется из величин 4, 2 и 1 для r, w и x соответственно. Таким образом, 660 означает rw-rw----, 777 означает rwxrwxrwx. Три специальных бита Setuid, Setgid и Sticky имеют восьмеричные значения 4000, 2000 и 1000.

С помощью команды chmod можно указывать биты доступа в восьмеричной форме, опытные пользователи предпочитают ею пользоваться, чтобы не печатать лишнего:

```
user$ chmod 640 файл.txt
```

Поразительно, но команда ls не может представлять биты доступа в восьмеричном виде. Помочь в данном случае может следующая команда (к сожалению, абсолютно нечитабельная):

```
user$ ls -l | awk '{k=0;
                    for(i=0;i<=8;i++)k+=((substr($1,i+2,1)~/[rwx]/)*2^(8-i));
                    if(k)printf("%0o ",k);print}'
755 drwxr-xr-x 17 kofler kofler 4096 2008-10-28 15:34 php53-beispiele
550 dr-xr-x---  2 kofler kofler 4096 2008-10-17 10:33 Private
755 drwxr-xr-x 10 kofler kofler 4096 2007-10-10 18:17 samples
644 -rw-r--r--  1 kofler kofler 1747 2009-05-25 15:16 sun_vbox.asc
...
```

Права доступа к каталогам

В принципе, девять битов доступа применимы и при работе с каталогами, но в таком случае их значение несколько отличается: r-бит позволяет другим пользователям просмотреть содержимое каталога с помощью команды ls. Кроме того, x-бит позволяет перейти в этот каталог с помощью cd. Если поставить и x, и w, то в каталоге можно создавать новые файлы.

Права доступа к устройствам

Доступ к различным аппаратным компонентам, например жестким дискам, приводам CD и DVD, интерфейсам и т. д., осуществляется в Linux через так называемые файлы устройств (раздел 3.10). Чтобы иметь возможность управлять тем, какой пользователь может обращаться к каким устройствам, они распределяются между

различными группами пользователей. Например, устройства /dev/ttyS* назначены для работы с серийными устройствами в системе Ubuntu в группе dialout:

```
root# ls -l /dev/ttyS1
crw-rw----    1 root    dialout    5, 65 Jul 18   /dev/ttyS1
```

Если системному дминистратору понадобится предоставить пользователю hubert права работы с серийным интерфейсом, то hubert будет добавлен в группу dialout:

```
root# usermod -a -G dialout hubert
```

Специальные биты

Понять значение трех троек битов доступа rwxrwxrwx совсем не сложно. С их помощью можно сохранить дополнительную информацию о файлах и каталогах. Как правило, об этих специальных битах должны знать только системные администраторы.

Бит Setuid

Он иногда сокращенно называется suid. С его помощью программа всегда выполняется так, как если бы пользователь запускал ее сам. Итак, если владельцем программы является администратор (это случается довольно часто), то любой пользователь может работать с этой программой так же, как если бы он был администратором. Внутри системы для выполнения программы применяется пользовательский идентификационный номер владельца файла, а не UID пользователя, работающего с файлом в данный момент.

Бит Setuid используется для того, чтобы присваивать обычным пользователям дополнительные права, действующие лишь при выполнении данной конкретной программы. Правда, это чревато угрозой для безопасности — в особенности тогда, когда при выполнении данной программы запускаются другие программы. Это означает, что по возможности следует избегать применения бита Setuid. Одно из немногих исключений — команда mount. Вместо бита Setuid можно использовать команду sudo.

В таких программах команда ls -l показывает для пользовательского бита доступа букву s (а не x). Восьмеричное значение этого бита (для chmod) составляет 4000.

```
root# ls -l /bin/mount
-rwsr-xr-x   1 root   root   68508 Feb 25   01:11   /bin/mount
```

Бит Setgit

Этот бит работает с программами почти так же, как и Setuid. Правда, в данном случае при выполнении программы используется идентификационный номер группы, к которой относится файл (а не групповой идентификационный номер пользователя, работающего с программой). В таких программах команда ls -l выдает для групповых битов доступа букву s (а не x). Восьмеричное значение этого бита составляет 2000.

При работе с каталогами бит Setgid действует так, что новые файлы, создаваемые в том или ином каталоге, сразу же относятся к той же группе, что и этот каталог (а не к группе пользователя, создавшего файл, как это бывает обычно).

Бит Sticky

В тех каталогах, где файлы может изменять любой пользователь, бит Sticky гарантирует, что каждый из пользователей может удалять только свои файлы, но не файлы коллег. Этот бит ставится, например, для каталога /tmp. Здесь любой пользователь может сохранять временные файлы. Однако необходимо избегать таких случаев, в которых любой пользователь мог бы по собственному усмотрению переименовывать или удалять файлы других пользователей.

Команда ls -l в таких программах выдает для всех действительных битов доступа букву t (а не x). Восьмеричное значение этого бита составляет 1000. Не забывайте при этом, что значение бита Sticky является специфичным для Linux. В других системах UNIX значение этого бита может быть другим (или может вообще отсутствовать).

```
user$ ls -ld /tmp/
drwxrwxrwt   18 root   root   4096 Jun 14   15:34   /tmp/
```

Специальные биты в ls

При выполнении команды ls -l в определенной ситуации могут отображаться специальные биты S и T. При этом мы не имеем дело с новыми специальными битами, а получаем указание на то, что биты Setuid, Setgid или Sticky были использованы неправильно.

- S — бит Setuid или Setgid поставлен, но не поставлен бит доступа x (тогда Setuid или Setgid не будут работать).
- T — бит Sticky поставлен, но не поставлен бит доступа x для группы others.

Функция или тип файла

В системе Linux вместе с битами доступа и специальными битами также сохраняется информация о том, какую функцию выполняет файл. Например, это может быть обычный файл, каталог, ссылка, блочное устройство и т. д.

Большинство файловых менеджеров скрывают эту дополнительную информацию. Однако существуют и такие программы, которые отображают эти данные в виде числовых значений. Тогда полная спецификация (указание) для обычного файла равняется 10000 плюс *x*000 (специальные биты) плюс *xxx* (биты доступа), то есть, к примеру, 100760. Числовые коды, соответствующие файлам различных типов, можно узнать с помощью команды man 2 stat.

Владелец, группа и биты доступа для новых файлов

В этом подразделе будет рассмотрен вопрос о том, какие факторы определяют информацию о доступе к новым файлам. Чтобы просто опробовать этот механизм,

воспользуйтесь командой touch. Она создает новый пустой файл, если указанный файл не существует.

Пример. Пользователь michael создает новый файл myFile1. Не удивляйтесь тому, что этот файл, опять же, принадлежит пользователю michael, ведь именно michael его создал. В качестве индикатора групповой принадлежности автоматически используется michael. Это основная группа пользователя michael (в некоторых дистрибутивах пользователю не выделяется отдельная группа, а все пользователи относятся к группе users).

```
michael$ touch myFile1
michael$ ls -l myFile1
-rw-r--r--    1 michael   michael      0 Jun 14 16:45 myFile1
```

Пользователь michael также принадлежит к ряду других групп (команда groups). Чтобы создать файл, не относящийся к основной группе, сначала нужно сменить активную группу (команда newgrp):

```
michael$ groups
michael adm admin cdrom dokuteam dialout lpadmin plugdev sambashare
michael$ newgrp dokuteam
michael$ touch myFile2
michael$ ls -l myFile2
-rw-r--r--   1 michael    dokuteam      0 Jun 14 17:02 myFile2
```

Разумеется, myFile2 можно было бы создать и без применения newgrp. В таком случае групповую отнесенность файла следовало бы постепенно изменить с помощью команды chgrp. Команду newgrp удобно использовать тогда, когда создается несколько новых файлов, которые должны автоматически присваиваться определенным группам.

Владелец и групповая отнесенность

Из двух показанных выше примеров понятно, что создаваемые файлы автоматически присваиваются своему автору. Как правило, эти файлы определяются в ту же группу, к которой относится создавший их пользователь. Из этого правила есть два исключения.

- Если пользователь с помощью команды newgrp сделает активной другую группу, к которой он также принадлежит, то новый файл будет относиться к этой группе.
- Если в каталоге будет поставлен бит Setgid (см. предыдущий подраздел), то файлы, создаваемые в данном каталоге, автоматически попадают в ту же группу, что и этот каталог. Активная группа пользователя в расчет не принимается.

Биты доступа

С битами доступа дело обстоит немного сложнее. В Linux предусмотрено, что новые файлы получают биты доступа rw-rw-rw (восьмеричное значение 666), то есть такие файлы может читать и изменять любой пользователь. Новые программные файлы, создаваемые компилятором, автоматически получают биты доступа rwxrwxrwx (777), то есть могут выполняться кем угодно.

Для практической работы с несколькими пользователями эта базовая настройка в любом случае была бы слишком бесхитростной, поэтому все оболочки Linux (то есть интерпретаторы команд) содержат так называемую umask-настройку. Она представляет собой числовое значение, указывающее биты, которые могут быть вычтены из стандартных битов доступа.

В Linux обычно применяется значение `umask`, равное `022` (`----w--w-`). Таким образом, новые файлы получают биты доступа `666-022=644` (`rw-r--r--`), новые программы — биты доступа `777-022=755` (`rwxr-xr-x`). Текущее значение настройки `umask` можно узнать (а также изменить) с помощью одноименной команды:

```
michael$ umask
0022
```

Базовая настройка значения `umask` производится в файлах конфигурации соответствующей оболочки. Что касается bash (самой популярной оболочки Linux), `umask` обычно настраивается в файле `/etc/profile` или `/etc/bashrc`. В большинстве дистрибутивов отдельные пользователи могут поставить в файле `~/.bashrc` специальную настройку, отличающуюся от задаваемой по умолчанию. Если, например, вы хотите, чтобы создаваемые вами файлы могли читать только члены группы, но не все остальные пользователи, применяйте следующую настройку:

```
# in ~/.bashrc
umask 027
```

Тогда новые файлы получают права доступа `rw-r-----`, а новые программы — `rwxr-x---`.

После того как файл будет создан, при работе другого пользователя с этим файлом не будут изменены ни данные о владельце, ни биты доступа. Изменять владельца файла может только администратор (таким образом, исключена возможность того, что владелец файла как бы «подарит» этот файл другому владельцу).

3.8. Списки контроля доступа и расширенные атрибуты

Списки контроля доступа (ACL). Типичное для UNIX управление пользователями и группами, а также основанные на нем права доступа к каталогам и файлам сохраняются в неизменном виде уже не одно десятилетие. Концепция настолько проста, что ее начинаешь понимать после пары часов изучения. Правда, встречаются случаи, когда этой простой системы недостаточно.

Именно поэтому была разработана более «мелкоячеистая» система управления правами доступа, основанная на так называемых *списках контроля доступа* (ACL). ACL позволяют установить для файла или каталога любое количество правил, касающихся того, какие пользователи или группы имеют право читать или изменять файл или группу файлов, находящихся в каталоге, а какие пользователи *не имеют такого права* — эта возможность в системах UNIX уже не предусмотрена. Таким образом, ACL дополняют стандартные права доступа и могут вводить дополнительные права или отменять существующие.

Списки контроля доступа по умолчанию входят в состав ядра Linux, начиная с версии 2.6. Для более ранних версий ядра существуют соответствующие заплатки. В файловых системах jfs и xfs списки контроля доступа активны в любом случае. В файловых системах ext, напротив, необходимо использовать mount-параметр `acl`, чтобы активизировать списки контроля доступа. В большинстве дистрибутивов эта функция не выполняется автоматически. Samba (раздел 20.3) — это важнейшая из тех программ, в которой очень выгодно использовать списки контроля доступа. Благодаря ACL эта программа способна во многом экстраполировать права доступа, существующие в Windows, на Linux.

Ограничения. Тот факт, что ACL предоставляют дополнительные возможности, ни в коем случае не отменяет традиционного управления правами! Для опытных администраторов крупных сетей списки контроля доступа могут обеспечить дополнительную безопасность или как минимум упростить управление сетью, но для большинства пользователей Linux обычного метода управления правами будет вполне достаточно. Если неправильно использовать сложную систему списков контроля доступа, то могут возникнуть всевозможные бреши в сфере безопасности сети. Поэтому в настоящее время и не существует дистрибутивов, в которых списки контроля доступа использовались бы по умолчанию.

Основная проблема заключается в том, что многие программы и команды Linux неправильно работают с ACL. Вполне может случиться так, что в скопированном файле вдруг не окажется информации об ACL, которая присутствует в оригинале. Большинство файловых менеджеров также не могут правильно отображать и изменять списки контроля доступа (Konqueror — приятное исключение из этого правила).

Расширенные атрибуты (EA). Расширенные атрибуты находятся в тесном родстве со списками контроля доступа (EA). Они позволяют сохранять для любых файлов дополнительные сведения в форме пар «атрибут — значение». Так, например, вы можете присвоить текстовому файлу атрибут `charset` с настройкой `utf8`, чтобы сохранить таким образом применяемую кодировку. Конечно, с выгодами это связано только тогда, когда существуют программы, способные интерпретировать такую информацию. В зависимости от того, какой файловой системой вы пользуетесь, расширенные атрибуты необходимо активизировать с помощью соответствующего `mount`-параметра; например, в случае с файловой системой ext2/3 это параметр `user_xattr`. Расширенные атрибуты используются, в частности, поисковой программой Beagle.

Более подробная информация о применении списков контроля доступа и расширенных атрибутов содержится на страницах справки `man` (`acl`, `getfacl`, `setfacl`, `attr(5)`, `getfattr` и `getsattr`), а также на следующих сайтах:

- http://acl.bestbits.at/ (на немецком языке);
- http://www.suse.de/~agruen/ (на немецком языке);
- http://www.vanemery.com/Linux/ACL/linux-acl.html;
- http://www.heise.de/kiosk/archiv/ct/2003/23/218 (на немецком языке; платно).

Предпосылки. В следующих примерах я буду исходить из того, что у вас установлен пакет `attr` с командами `attr`, `getfattr` и `setfattr` и что вы работаете с файловой системой, в которой активизированы списки контроля доступа и расширенные

атрибуты. Если вы работаете с файловой системой ext4, то результат выполнения команды mount должен выглядеть так:

```
user$ mount
...
/dev/sdc5 on /test type ext4 (rw,acl,user_xattr)
...
```

В иных случаях в следующих примерах вы получите сообщение об ошибке, которое будет гласить примерно следующее: *«Операция не может быть выполнена»*. Чтобы справиться с этим, попробуйте изменить параметр mount в /etc/fstab и заново привязать файловую систему. Почитайте в главе 13 о том, как администрировать файловую систему. Информация об изменении параметров mount в /etc/fstab находится в подразделе «Автоматическое подключение файловых систем (/etc/fstab)» раздела 13.5.

Списки контроля доступа

Команда getfacl

В системах, где установлены списки контроля доступа, обычно все равно действуют стандартные принципы управления доступом, которые часто называются «минимальный список контроля доступа». Команда getfacl отображает эти права в виде ACL:

```
user$ touch файл1
user$ getfacl файл1
# file: файл1
# owner: kofler
# group: kofler
user::rwgroup::
r--
other::r--
user$ ls -l файл1
-rw-r--r-- 1 kofler kofler ... файл2
```

Команда setfacl

С помощью setfacl можно определить новые права доступа. Следующие команды предоставляют пользователю gabi и всем членам группы docuteam доступ к файлу с правом читать и изменять этот файл, однако закрывают пользователю kathrin какой-либо доступ к файлу:

```
user$ setfacl -m gabi:rw файл1
user$ setfacl -m g:docuteam:rw файл1
user$ setfacl -m kathrin:- файл1
```

Список прав команды getfacl немного длиннее. Теперь ls показывает на месте прав доступа для членов группы специальную ACL-маску. За буквами, означающими доступ, следует символ +, указывающий на наличие правил ACL.

```
user$ getfacl файл1
# file: файл1
```

```
# owner: kofler
# group: kofler
user::rwuser:
gabi:rwuser:
kathrin:---
group::r--
group:docuteam:rwmask::
rwother::
r--
user$ ls -l файл1
-rw-rw-r--+ 1 kofler kofler ... файл1
```

Как правило, списки контроля доступа применяются для того, чтобы предоставить определенному пользователю доступ к его файлам, не предоставляя такого доступа всем остальным пользователям. В таком случае вам потребуется попросить администратора, чтобы он создал группу, к которой будете принадлежать вы и те пользователи, вместе с которыми вы хотите обрабатывать файлы. При использовании ACL просто нужно выполнить команду `setfacl -m` *`пользователь`*`:rw` *`файл`*.

Маска ACL

Маска ограничивает права, предоставляемые правилами ACL. Если вы, например, устанавливаете для маски ACL значение `r`, то ни одно правило не может предоставить пользователю права изменять или выполнять файлы. Итак, ACL-маска имеет приоритет над правилами ACL. Но в любом случае она никак не влияет на обычные права доступа, которые предоставляются пользователю или группе пользователей «традиционным» способом.

При изменении правила ACL командой `setfacl` маска автоматически рассчитывается заново так, чтобы могли выполняться все прочие правила ACL. Эта маска отображается с помощью команды `getfacl` и также учитывается командой `ls -l`.

Можно явным образом настроить маску с помощью команды `setfacl -m m:rwx file`, ограничив, таким образом, права ACL. Однако учитывайте, что ваша маска действует лишь до тех пор, пока вы не определите новое правило ACL. В таком случае маска ACL будет автоматически пересчитана (если только вы не предотвратите это с помощью параметра `-n`).

Стандартные списки контроля доступа

При работе с каталогами можно установить второй набор правил для стандартных списков контроля доступа. Стандартные ACL не управляют доступом к каталогу, а служат образцом для новых файлов. Любой файл, создаваемый внутри данного каталога, наследует стандартные списки контроля доступа, указанные для этого каталога. При одновременном применении нескольких списков контроля доступа новый каталог с умело подобранными стандартными ACL может использоваться в работе в качестве отправной точки.

Совместимость ACL

Распространение списков контроля доступа сильно осложняется тем, что многие стандартные команды и практически все пользовательские программы просто

игнорируют ACL. Если вы просто копируете файл с правилами ACL с помощью команды cp, то в скопированном файле правила ACL не сохранятся. То же самое произойдет, если вы откроете файл в текстовом редакторе OpenOffice или Gimp и сохраните его под другим именем. При использовании команды cp будет полезен параметр -p, но у большинства других команд и программ подобные параметры отсутствуют либо эти программы вообще не приспособлены к работе с ACL.

Проблемы возникают и при резервном копировании. Команды tar и rsync элиминируют правила ACL. Файловая система CD и DVD не рассчитана на работу с ACL, так что сохраняемая в них информация теряется. Есть два выхода: во-первых, можно использовать вместо tar версию star, совместимую с ACL, во-вторых, перед резервным копированием можно создавать отдельный текстовый файл, в который заносятся **ACL-правила всех копируемых файлов. После резервного** копирования правил ACL восстанавливаются на основании этого файла.

```
user$ getfacl -R --skip-base . > acl-backup  (Сохранение ACL-правил)
user$ setfacl --restore=acl-backup           (Восстановление ACL-правил)
```

Расширенные атрибуты

Команды setfattr и getfattr

На следующих примерах показано, как сохранять атрибуты с помощью команды setfattr и считывать их с помощью getfattr. Количество атрибутов, которое может иметь один файл, в файловых системах ext3 ограниченно.

```
user$ touch файл2
user$ setfattr -n user.language -v ru файл2
user$ setfattr --name=user.charset --value=utf8 файл2
user$ getfattr -d файл2
# file: файл2
user.charset="utf8"
user.language="ru"
```

Команда getfattr обычно возвращает только те атрибуты, чье название начинается с user. Если вы хотите увидеть другие атрибуты, нужно указывать имя атрибута вместе с меткой -n, а образец атрибута — с меткой -m.

```
user$ getfattr -n security.selinux -d tst
# file: tst
security.selinux="user_u:object_r:user_home_t:s0^000"
```

Совместимость с расширенными атрибутами

К сожалению, пока практически не существует программ, которые сохраняли бы расширенные атрибуты при копировании, архивировании и т. д. Даже команда cp -p игнорирует атрибуты. При создании резервных копий лучше всего поступать так же, как и при работе с ACL, — создавать при копировании текстовый файл со всеми расширенными атрибутами. На основании этого файла можно потом восстановить расширенные атрибуты.

```
user$ getfattr -R . > ea-backup    (Сохранение атрибутов)
user$ setfattr --restore=ea-backup (Восстановление атрибутов)
```

3.9. Структура каталогов в Linux

Стандарт иерархии файловой системы. Типичная UNIX-система состоит из тысяч файлов. В ходе разработки UNIX закрепились определенные правила, описывающие, в каких каталогах какие файлы принято сохранять. Эти правила были приспособлены к особенностям Linux и обобщены в специальном документе, именуемом Стандартом иерархии файловой системы (Filesystem Hierarchy Standard, FHS). Практически все дистрибутивы Linux, за исключением нескольких, построены согласно этому стандарту. Информацию о нем вы можете найти по адресу http://www.pathname.com/fhs/.

Информация, представленная в данном разделе, является лишь поверхностной справкой, помогающей сориентироваться на первом этапе (не больше!). Поэтому я учел здесь не только стандарт FHS, но и другие особенности, издавна характерные для различных популярных дистрибутивов Linux.

Файловая система начинается с корневого каталога. Как правило, в нем нет никаких файлов, только следующие каталоги.

- `/bin` — содержит элементарные команды Linux, предназначенные для управления системой. Эти команды могут выполняться любыми пользователями. Остальные программы находятся в `/usr/bin`.
- `/boot` — хранит файлы, предназначенные для загрузки системы (обычно с помощью GRUB или LILO). В большинстве дистрибутивов здесь же располагается ядро.
- `/dev` — содержит все файлы устройств. Доступ практически к любым компонентам аппаратного обеспечения, будь то серийный интерфейс или сегмент жесткого диска, осуществляется через специальные файлы-устройства (собственно, это не настоящие файлы). Более подробно номенклатура файлов устройств описана в разделе 3.10.
- `/etc` — включает в себя конфигурационные файлы для всей системы. В `/etc` имеется множество подкаталогов, в которых конфигурационные файлы распределяются по группам, например `/etc/X11` для всех X-специфичных файлов. Многие файлы из `/etc` рассмотрены в этой книге в главах, посвященных вопросам конфигурации. Рекомендую также обратиться к предметному указателю (на букву E)!
- `/home` — содержит домашние каталоги всех обычных пользователей Linux. Как вы помните, домашним называется тот каталог, в котором пользователь автоматически оказывается после входа в систему и в отношении файлов которого этот пользователь имеет неограниченные права (как обычно, администратор требует отдельного упоминания — его домашний каталог называется `/root`).
- `/lib[64]` — содержит множество общих библиотек (shared libraries) или символьные ссылки на них. Эти файлы обеспечивают работу программ. Каталог `/lib/modules` включает модули ядра, которые динамически активизируются или деактивизируются без остановки работы системы. Остальные библиотеки находятся в каталоге `/usr/lib[64]`.

В каталоге /lib/firmware находятся встроенные программы (так называемая прошивка) различных компонентов аппаратного обеспечения (например контроллер WLAN).

- /lost+found — обычно этот каталог пуст. Если в нем есть файлы, то они являются фрагментами, которые не удалось упорядочить после восстановления файловой системы (fsck). Иными словами, были найдены сектора, но неизвестно, к какому файлу относится какой сектор. Вместо того чтобы просто удалять такие фрагменты, команда fsck копирует их в каталог /lost+found. Эта команда автоматически выполняется при запуске системы всякий раз, когда работа Linux была завершена с ошибками (перебои с электричеством, фатальный сбой и т. д.) или файловая система не проверялась в течение длительного времени. Цель fsck заключается в том, чтобы привести файловую систему в состояние, которое можно непротиворечиво описать.
- /media — содержит такие подкаталоги, как cdrom или floppy, на месте которых можно привязать внешнюю файловую систему. Раньше этот каталог назывался /mnt, но в последнее время закрепился вариант /media.
- /opt — этот каталог предусмотрен для дополнительных пакетов, но в наиболее распространенных дистрибутивах он используется редко: возможно, потому, что не до конца ясно, чем дополнительные пакеты отличаются от обычных.
- /proc — хранит подкаталоги для всех процессов, выполняемых в настоящий момент. В данном случае речь не идет о настоящих файлах! Каталог /proc просто отражает внутреннее управление процессами в системе Linux (см. раздел 15.3).
- /root — содержит файлы root-пользователя (то есть системного администратора).
- /sbin — включает команды, предназначенные для управления системой. Общий признак всех программ, содержащихся в этом каталоге, заключается в том, что все они могут выполняться только администратором.
- /share — иногда содержит архитектурно зависимые файлы (то есть файлы, не зависящие от процессора). Более правильно располагать такие файлы в каталоге /usr/share.
- /srv — в некоторых дистрибутивах хранит данные по серверным процессам, например /srv/www для всех документов Apache, /srv/ftp для FTP-файлов и т. д.
- /sys — в версиях ядра от 2.6 и выше данный каталог содержит файловую систему sysfs (см. раздел 13.4). Эта система (как и proc) сообщает информацию о состоянии компьютера.
- /tmp — содержит временные файлы. Однако часто временные файлы сохраняются и в /var/tmp.
- /usr — включает все пользовательские программы, полноценную X-систему, исходные коды Linux и т. д. Как правило, содержимое этого каталога изменяется только при установке пакетов и выполнении обновлений. Изменяющиеся файлы располагаются в каталоге /var.

- /var — содержит изменяющиеся файлы. Важнейшими подкаталогами здесь являются adm (административные файлы, отличаются в зависимости от дистрибутива), lock (блокирующие файлы, предотвращающие доступ к устройствам для пользователей, не имеющих на это права), log (файлы регистрации), mail (файлы электронных сообщений, также находятся в каталоге spool/mail) и spool (сохраненные в буфере файлы для вывода на печать, новостные файлы и т. д.).

Итак, понять структуру каталогов, расположенных на корневом уровне, совсем просто.

Проблемы начинаются тогда, когда каталоги /usr и /var дробятся на бесчисленные подкаталоги. В принципе многие каталоги на этом уровне называются так же, как и на корневом, — лишь исполняемые программы в данном случае находятся в каталоге bin.

При этом возникает такая проблема: существует много групп исполняемых программ. Это, например, текстовые команды, X-программы и т. д. Соответственно имеется много возможностей скрывать эти программы. Так сложилось, что с помощью ссылок в системе часто используется много параллельных путей. Например, /usr/bin/X11 ведет к тем же программам, что и /usr/X11R6/bin (оба этих пути логически и исторически обоснованны).

Полностью описать структуру каталогов в принципе невозможно. В табл. 3.8 дается краткое описание подкаталогов, относящихся к /usr.

Таблица 3.8. Каталоги из /usr

Каталог	Содержимое
/usr/bin	Исполняемые программы
/usr/games	Игры или ссылки на /usr/share/games
/usr/include	Файлы C-Include
/usr/lib[64]	Различные библиотеки, а также многочисленные подкаталоги C-компилятора, многие другие языки программирования, большие программные пакеты, например emacs и LATEX
/usr/local	Приложения и файлы, не относящиеся напрямую к дистрибутиву Linux либо установленные позже, чем основная часть системы
/usr/sbin	Программы, исполняемые только администратором
/usr/share	Архитектурно зависимые файлы (например, файлы Emacs-Lisp, кодировки Ghostscript и т. д.), документация (/usr/share/doc)
/usr/src	Исходный код для Linux и, возможно, для других программ

3.10. Файлы-устройства

В файловой системе Linux осуществляется управление не только файлами и каталогами, но и «устройствами». Так называются особым образом обозначаемые файлы, в которых не хранится никакой информации, так как они обеспечивают связь с ядром Linux.

Старший и младший номера устройства

Устройства обеспечивают доступ ко многим компонентам аппаратного обеспечения компьютера, то есть к жестким дискам, флоппи-дисководам, серийным и параллельным интерфейсам, оперативной памяти (RAM) и т. д. Три параметра — *старший номер устройства*, *младший номер устройства* и *тип доступа* (блочный или символьный) — определяют характеристику устройства.

Старший номер определяет, какой драйвер ядра Linux отвечает за управление устройством. Большинство драйверов с их старшими номерами устройств перечислены на сайте http://www.kernel.org/doc/Documentation/devices.txt.

У многих драйверов младший номер служит тем признаком, с помощью которого различаются отдельные (родственные) устройства: такова, например, ситуация с драйверами различных сегментов одного и того же жесткого диска.

Тип доступа указывает, являются ли устройства буферизованными (таковы все блочные устройства, например жесткие диски) или нет (символьно-ориентированные устройства, например серийные или параллельные интерфейсы).

Когда вы просматриваете содержимое каталога `/dev` с помощью команды `ls -l`, вместо размера файла система показывает номера устройства (старший и младший). Первый символ бита доступа — `b` или `c` — означает блочную либо символьную ориентацию.

```
user$ ls -l /dev/sda?
brw-rw---- 1 root root 8, 1 2007-02-02 10:39 /dev/sda1
brw-rw---- 1 root root 8, 2 2007-02-02 10:39 /dev/sda2
...
```

Внутренние свойства

Внутри системы Linux в каталоге `/dev` находятся только так называемые индексные дескрипторы (I-Node). Это мельчайшие управляющие единицы системы, а не настоящие файлы. Новые файлы устройств можно создавать с помощью команды `mknod`. Однако на практике это требуется редко, так как система udev делает такое автоматически.

Старший и младший номера устройств, начиная с версии ядра 2.6, представлены в виде 64-битных чисел (до версии 2.4 для этого применялись 32-битные числа).

Доступ ко многим устройствам (из соображений безопасности) может получить только администратор либо члены определенной группы. Чтобы предоставить другому пользователю доступ к этим устройствам, добавьте его к этой группе.

Некоторые файлы устройств выполняют специфическую функцию: так, `/dev/null` служит «черной дырой» и отправляемые туда данные навсегда исчезают (используется, например, для переадресации командного вывода, если эта информация должна быть скрыта); `/dev/zero` — это неиссякаемый источник нулевых байтов, которые иногда используются для того, чтобы заполнять нулями файлы до определенного (заданного) размера; `/dev/random` и `/dev/urandom` возвращают случайные числа (табл. 3.9).

Таблица 3.9. Важные файлы устройств

Устройство	Значение
/dev/cdrom	Ссылка на привод CD-ROM
/dev/console	Виртуальный терминал, активный в настоящий момент
/dev/disk/*	Дополнительные ссылки на устройства жестких дисков и сегментов
/dev/dri/*	Инфраструктура прямой визуализации (3D-графика с X)
/dev/dsp*	Доступ к звуковой карте (устройству цифрового сэмплирования)
/dev/fb*	Кадровый буфер (графическая карта)
/dev/fd*	Дисководы для гибких дисков
/dev/hd*	IDE-дисководы (жесткие диски, CD- и DVD-приводы)
/dev/ht*	IDE-стримеры с автоматической обратной перемоткой
/dev/input/*	Мышь и джойстик
/dev/kbd	Клавиатура (PS/2)
/dev/kmem	Оперативная память (RAM) с магнитным сердечником (для отладчика)
/dev/lp*	Параллельные интерфейсы для принтера и т. д.
/dev/mapper	Файлы соответствия для программы управления логическими томами (LVM), пути к контейнерам и т. д.
/dev/md*	Мета-устройства (RAID и т. д.)
/dev/mem	Память (RAM)
/dev/mixer*	Доступ к звуковой карте
/dev/modem	Интерфейс модема (ссылка)
/dev/pktcdvd	Пакетная запись для CD и DVD
/dev/psaux	Мышь PS/2
/dev/nht*	IDE-стример без автоматической обратной перемотки
/dev/nst*	SCSI-стример без автоматической обратной перемотки
/dev/port	Порты ввода-вывода
/dev/pts/*	Виртуальные терминалы стандарта UNIX 98
/dev/ptyp*	Виртуальные терминалы для X (типа «ведущий»)
/dev/ram	Виртуальный диск
/dev/raw1394	Непосредственный доступ к Firewire-устройствам
/dev/rmt*	Стример без SCSI
/dev/sd*	Жесткие диски SCSI/SATA/USB/Firewire
/dev/scd*	CD/DVD-приводы типов SCSI/SATA/USB/Firewire
/dev/shm	Совместно используемая память POSIX
/dev/snd	Звук ALSA (ссылка на /proc/asound/dev)
/dev/scd*	CD/DVD-приводы типов SCSI/SATA/USB/Firewire
/dev/st*	SCSI-стример с автоматической обратной перемоткой
/dev/tape*	Стример
/dev/tty*	Виртуальные терминалы для работы в текстовом режиме
/dev/ttyp*	Виртуальные терминалы для X (типа «ведомый»)
/dev/ttyS*	Серийные интерфейсы (модем, мышь и т. д.)
/dev/usb/*	USB-устройства (см. также /proc/bus/usb)

Система udev

Раньше дистрибутивы Linux в ходе установки создавали множество файлов устройств (например, при установке Red Hat 9 создается почти 8000 таких файлов!). На деле же используется всего около сотни таких файлов, но этот набор неодинаков в зависимости от того, на каком компьютере установлена система и каково аппаратное обеспечение этого компьютера.

В данном случае помогает сделать правильный выбор система udev, которая появилась в версии ядра 2.6. Фоновая программа udevd распознает все аппаратные компоненты, подключенные к компьютеру, и создает необходимые файлы устройств. Программа udevd запускается в начале процесса Init-V. Конфигурация осуществляется с помощью файлов, содержащихся в каталоге `/etc/udev`.

Система udev работает просто отлично и способна обращаться с внешними винчестерами, флешками и многими другими аппаратными компонентами, которые могут подключаться к компьютеру и вновь отсоединяться в ходе работы. Основная проблема системы udev заключается в том, что создание файлов устройств при запуске компьютера длится достаточно долго (несколько секунд). Поскольку часть этих файлов необходима для продолжения запуска (в частности, для доступа к жестким дискам и сетевым интерфейсам), отложить выполнение udev или запустить ее в фоновом режиме очень сложно.

Размышления о том, как бы запускать Linux быстрее, навели разработчиков на мысль заменить udev более эффективной системой или (по крайней мере частично) вернуться к статической конфигурации. По адресу http://lwn.net/Articles/331818/ расположена статья, в которой описывается система devtmpfs, доступная начиная с ядра 2.6.32. Остается подождать, чтобы увидеть, насколько широко она будет использоваться в дистрибутивах.

Подробные комментарии относительно называния и нумерации hd- и sd-устройств для устройств IDE- или SATA/SCSI/USB/Firewire даны в разделе 13.2. Полное описание всех устройств, имеющихся в Linux в настоящее время, вместе с соответствующими номерами вы найдете по адресу http://www.kernel.org/doc/Documentation/devices.txt.

4 Управление процессами

В данной главе описано, как в системе Linux организована работа с процессами. На страницах этой главы будут рассмотрены следующие вопросы:

- какие существуют возможности запускать программы и (при необходимости принудительно) завершать их;
- как обычному пользователю получить привилегии администратора для выполнения программы;
- что такое «демон»;
- как можно автоматически запускать программу в заданный момент времени.

4.1. Запуск программ, управление ими и завершение процессов

Программы, команды, процессы, задачи. В этой главе речь пойдет в основном о процессах. Однако практически в любом контексте можно заменить слово «процесс» терминами «программа», «команда» или «задача». Внутри системы Linux нет принципиальных отличий между программой и командой. В обиходе текстовые программы, например `ls`, часто называются «командами». Чтобы с точностью сказать, с чем мы имеем дело — с программой или командой, — необходимо знать о том, какой файл при этом выполняется. Программный файл отличается от остальных файлов только тем, что имеет бит доступа `x` (см. раздел 3.7).

В обоих следующих файлах `server.tex` является информационным файлом, а `sichere` — программой. Точнее говоря, здесь мы имеем дело с обычным сценарием командного процессора, выполняющим резервное копирование. Программирование таких сценариев описано в разделе 8.8. Оба этих файла являются текстовыми, но исполнять можно только один из них — `sichere`, так как в нем установлены биты доступа `x`.

```
user$ ls -l s*
-rw-r--r--   1 kofler   users   180383 Apr 24 10:20 server.tex
-rwxr-xr-x   1 kofler   users      222 Jun  6 10:58 sichere
```

Только при запуске «безжизненного» программного файла он становится «живым» процессом (синоним — задача), а управляет этим процессом ядро Linux. В таком контексте можно было бы переформулировать название этой главы: «Запуск программ и команд, управление процессами и завершение их».

EXE-файлы. Постоянно возникает вопрос: а где в Linux находятся EXE-файлы? Еще несколько лет назад ответ звучал: «Таких файлов нет». Исполняемые программы в Linux характеризуются наличием бита доступа `x`. Такой файл называется исполняемым (executable). Таким образом, применяемое в Windows расширение файлов EXE оказывается ненужным.

Теперь этот ответ уже не настолько бесспорен, так как во многих системах Linux появились отдельные EXE-файлы. Это программы, написанные на языке C#, которые выполняются с помощью библиотеки Mono. Библиотека Mono — это еще одно нововведение с открытым кодом, относящееся к платформе .NET-Framework компании Microsoft. Более подробная информация по Mono дается в разделе 11.4.

Запуск программ

Запуск программ с X. В X программы, как правило, запускаются через меню или при щелчке кнопкой мыши на пиктограмме. В KDE и Gnome присутствует сочетание клавиш (Alt+F2), обеспечивающее быстрый запуск программ.

Текстовая консоль, командное окно. Кроме того, можно запускать программы в командном окне (например, `xterm`, `konsole` и т. д.) или в текстовой консоли. Для этого нужно просто указать название программы и нажать клавишу Enter. Профессионалы от Linux работают именно таким образом, поскольку ввести пару букв гораздо быстрее, чем отыскать программу в очень разветвленном меню.

Как правило, достаточно просто указать название программы. Затем Shell-интерпретатор ищет программу во всех каталогах, перечисленных в переменной окружения `PATH`. В следующих строках показано, как обычно настраивается эта переменная:

```
user$ echo $PATH
   /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
```

Если вы хотите запустить программу, которая находится в каком-то другом каталоге, то необходимо указать путь к ней полностью. Это касается, в том числе, программ текущего каталога! В таком случае путь указывается просто как точка (то есть, например, `./mojaprogramma`).

Приоритетные и фоновые программы

Если запускать программы в X через меню, то они будут работать в виде так называемых фоновых процессов, не мешая друг другу. Эти процессы также могут запускать другие программы еще до завершения инициирующей программы.

Совсем иначе обстоит дело при выполнении программы в текстовой консоли или командном окне. Программа запускается как приоритетный процесс. Прежде чем запустить следующую программу, вам придется дождаться завершения работы текущего процесса.

Однако в командном окне или текстовой консоли все же можно запускать программы в фоновом режиме. Для этого после названия программы нужно просто поставить символ `&`.

```
user$ emacs &
```

Если вы забудете поставить &, можно постепенно перевести выполнение программы в фоновый режим. Прервите выполнение программы, нажав сочетание клавиш Ctrl+Z, а затем возобновите программу с помощью команды bg:

```
user$ emacs
<Ctrl>+<Z>
[1]+ Stopped emacs
user$ bg
[1]+ emacs &
```

Если вместо bg использовать команду fg, то программа будет выполняться как приоритетный процесс.

При выполнении некоторых программ в фоновом режиме вам будут мешать многочисленные текстовые сообщения, выдаваемые при этом. Однако избавиться от них совсем несложно, нужно просто переадресовать их в каталог /dev/null. Например, следующая команда позволяет отформатировать USB-флешку в фоновом режиме:

```
root# mkfs.ext3 /dev/sdc > /dev/null &
```

Список всех текущих процессов

Команда ps

Список всех текущих процессов создается с помощью команды ps. Без дополнительных параметров она отображает только ваши собственные процессы — и только те, которые были запущены из текстовых консолей или командных окон. Команда ps может получать разнообразные параметры. В следующем примере я сократил список процессов ради экономии места.

```
user$ ps ax
 PID  TTY    STAT   TIME  COMMAND
   1  ?      S      0:00  init [5]
   2  ?      SN     0:00  [ksoftirqd/0]
   3  ?      S      0:00  [watchdog/0]
   4  ?      S<     0:00  [events/0]
...
3064  pts/2  S      0:39  emacs command.tex
3151  pts/2  S+     1:23  /bin/sh ./lvauto
3735  pts/4  S      0:00  su -l
3740  pts/4  S+     0:00  -bash
```

Команда top

Обычно top практичнее ps: эта команда упорядочивает процессы обратно пропорционально тому, насколько они загружают процессор, и сначала отображает процессы, протекающие в настоящий момент. Кроме того, программа сообщает, какой объем памяти при этом необходим и т. д. Список процессов обновляется раз в две секунды, пока программа не завершается нажатием клавиши Q. Следующие строки характеризуют работу системы в холостом режиме:

```
top - 14:45:38 up 6:56, 3 users, load average: 0.73, 0.55, 0.32
Tasks: 187 total, 1 running, 186 sleeping, 0 stopped, 0 zombie
```

```
Cpu(s): 2.7%us, 1.1%sy, 0.2%ni, 95.3%id, 0.6%wa, 0.1%hi, 0.1%si, 0.0%st
Mem:  6006012k total, 3043980k used, 2962032k free,  213476k buffers
Swap: 3903480k total,       0k used, 3903480k free, 1269644k cached
  PID  USER    PR   NI  VIRT  RES  SHR S  %CPU  %MEM   TIME+   COMMAND
 6645  kofler  20    0  935m 628m  51m S     4  10.7  5:06.99  VirtualBox
 4118  kofler  20    0  364m  47m  18m S     2  0.8   3:31.16  compiz.real
12602  kofler  20    0  391m  19m  13m S     2  0.3   0:39.50  mplayer
    1  root    20    0  4104  924  632 S     0  0.0   0:00.83  init
    2  root    15   -5     0    0    0 S     0  0.0   0:00.00  kthreadd
...
```

В столбце PID указаны номера процессов. Зная номер процесса, можно принудительно остановить вышедшие из-под контроля программы или фоновые процессы с помощью команды kill.

Процессы могут находиться в различных состояниях. Чаще всего встречаются состояния R (running)[1] и S (sleeping[2], то есть сейчас процесс не выполняет никаких задач и ожидает ввода информации). Кроме того, выполнение программ можно временно прервать, переводя их в состояние T (stopped[3]).

Команда top обладает способностью интерактивного приема команд. При этом процессы можно останавливать (K — kill) или изменять их приоритет (R — renice).

Графические альтернативы top. Разумеется, для текстовой команды top существуют и графические альтернативы, например ksysguard (KDE) или gnome-system-monitor (Gnome) (рис. 4.1).

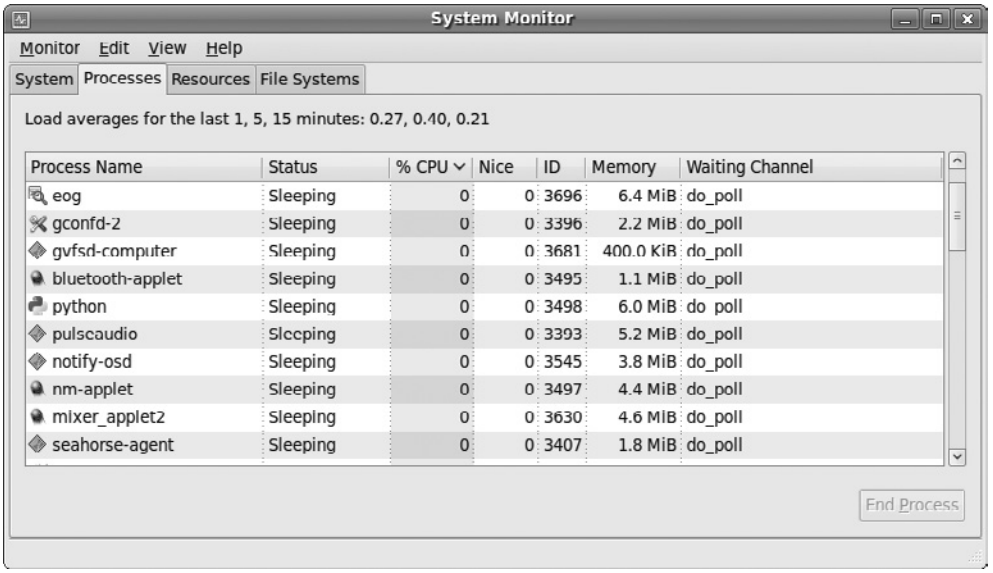

Рис. 4.1. Обзор процессов с помощью программы gnome-system-monitor

[1] Действующий (англ.).
[2] Спящий (англ.).
[3] Остановленный (англ.).

Установление номера процесса

Когда вы знаете названия программ и хотите узнать соответствующий программе номер процесса (PID), поможет команда `pidof`. Если имеется несколько процессов с одинаковым названием, то `pidof` возвращает весь список номеров:

```
root# pidof nscd
1777 1776 1775 1774 1765 1763 1753
```

Иногда бывает полезно установить, какие программы обращаются к тому или иному файлу или каталогу. Соответствующие номера процессов можно узнать с помощью команды `fuser`. Каталог считается задействованным и в том случае, когда в нем была запущена программа. Следующая команда показывает, что оболочка bash использует каталог `/media/dvd`:

```
root# fuser -v /media/dvd
             USER      PID  ACCESS  COMMAND
/media/dvd   kofler   2183  ..c..   bash
             Root   Kernel  mount   /media/dvd
```

Обратите внимание, что факт доступа к файлу может быть установлен лишь в том случае, когда программа действительно открыла файл на достаточно долгий период времени. В случае с текстовым редактором этого, например, не происходит (текстовый редактор открыл файл для загрузки, но потом закрыл; чтобы сохранить файл, он вновь ненадолго его открывает).

PID-файлы. Некоторые фоновые процессы сохраняют в каталоге `/var/run` PID-файл. В первой строке этого файла содержится номер процесса, а в остальных строках может содержаться дополнительная информация, например данные о сетевом интерфейсе. PID-файлы обеспечивают целенаправленное завершение определенного процесса с помощью сценария Init-V (подраздел «Сценарии Init-V для активизации уровней запуска» раздела 14.10), в том числе и тогда, когда существует несколько одноименных процессов.

Иерархия процессов

Внутри системы вместе с каждым процессом сохраняется PID-номер его родительского процесса. Эта информация позволяет построить дерево процессов, на вершине которого всегда располагается процесс `init`. Это первая программа, запускаемая сразу же после загрузки ядра (раздел 14.10).

Лучше всего увидеть иерархию процессов с помощью команды `pstree`. Параметр `-h` позволяет выделить те процессы, которые являются родительскими для процесса, выполняемого в настоящий момент. На рис. 4.2 команда `pstree` оболочки bash выполнена в окне konsole.

Принудительное завершение процессов

Как правило, с окончанием программы завершается и процесс. Но, к сожалению, и в Linux программы могут содержать ошибки, не позволяющие им остановиться, из-за чего такие процессы все сильнее потребляют ресурсы оперативной памяти и процессора. В таких случаях процесс необходимо принудительно завершать.

Рис. 4.2. Обзор процессов с помощью команды pstree

При работе с текстовыми командами для принудительного завершения достаточно нажать Ctrl+C. Программа сразу завершается.

Команда kill. Эта команда посылает действующему процессу сигналы, специфицируемые благодаря номеру `PID` (его можно узнать с помощью команды `top` или `ps`). Чтобы «аккуратно» завершить программу, используется сигнал 15 (`kill` использует этот сигнал по умолчанию). Если это не помогает, необходимо применить сигнал 9 (в данном случае — для процесса 2725):

```
user$ kill -9 2725
```

Команду `kill` можно применять только к собственным процессам. Администратор имеет право завершать процессы и других пользователей.

Команда top. Можно завершать процессы и с помощью команды top: просто нажмите клавишу K и дополнительно — номер процесса, а также желаемый сигнал.

Команда killall. Эта команда гораздо удобнее, так как при ее использовании можно указывать не номер процесса, а название программы. Правда, в данном случае будут завершены все процессы с таким именем.

```
user$ kill -9 2725
```

Команда xkill. В X все еще удобнее. Запустите в командном окне команду xkill и просто щелкните кнопкой мыши в окне с программой, которую хотите завершить. Процессу опять же будет отправлен сигнал 9.

В KDE можно запускать xkill и с помощью сочетания клавиш Ctrl+Alt+Esc. Если вы нажмете его по ошибке, то можно прервать выполнение xkill с помощью клавиши Esc.

Сложные случаи. Иногда xkill закрывает окно, а процесс или его части продолжают работать. Убедитесь в том, что команда действительно прекратила работу, выполнив команду top или ps. При необходимости воспользуйтесь командой kill -9 n.

Блокировка мыши и клавиатуры. И уж совсем неприятно, когда X-программа не просто остается висеть, но и переводит на себя фокус мыши или клавиатуры либо каким-то образом блокирует X. В таком случае компьютер почти не реагирует на ввод.

В таких случаях иногда помогает волшебное сочетание Ctrl+Alt+F1, позволяющее перейти в первую текстовую консоль. Оттуда можно войти в систему, найти подвисшую программу и завершить ее с помощью команды top.

Когда клавиатура полностью заблокирована, всегда остается возможность зайти в систему из сети через SSH и выполнить kill таким образом. Разумеется, этот вариант возможен лишь тогда, когда вы работаете в локальной сети, а на локальном компьютере запущена программа sshd.

Если сама X окажется заблокированной после остановки программы, попытайтесь принудительно завершить работу или, наконец, выполнить команду shutdown. Все эти варианты лучше, чем нажатие Reset, так как в последнем случае можно потерять данные!

Ограничение размера процесса. При работе с программами, запускаемыми из оболочки (например, со всеми командами, которые выполняются в командном окне), можно воспользоваться оболочковой командой ulimit, которая позволяет ограничить максимальный размер потребляемой памяти, создаваемых файлов и т. д. Обычно ulimit настраивается с помощью файла /etc/profile.

Распределение машинного времени (продолжительности вычислений)

При повседневной работе с Linux вычислительной мощности машины обычно более чем достаточно, чтобы выполнять все текущие процессы без задержек. Однако если Linux занята выполнением работы, требующей большого объема вычислений,

например компилирует крупную программу, то система пытается справедливо распределить имеющееся в распоряжении вычислительное время между всеми процессами.

Иногда бывает целесообразно выделить определенному процессу заметно больше или несколько меньше времени, нежели остальным. Для этого предназначена команда `nice`, с помощью которой можно запускать программы с повышением или понижением приоритета. В таком случае команде `nice` сообщается приоритет, значение которого находится в диапазоне между `19` (совсем низко) и `-20` (очень высоко). По умолчанию процессы запускаются с приоритетом `0`. В следующем примере программа резервного копирования запускается с низким приоритетом, чтобы она не влияла на выполнение других процессов (ведь неважно, сколько времени будет длиться резервное копирование: на три секунды дольше или короче).

```
user$ nice -n 10 sichere
```

С помощью команды `renice` можно изменять приоритет процессов, выполняемых в настоящий момент. В качестве параметра необходимо указать ID процесса (который вы ранее узнали с помощью команды `top` или `ps`). Подробная информация о `renice` содержится на соответствующей странице справки `man`. Команда `top` также может интерактивно менять приоритет процесса. В любом случае только администратор может выполнять программы с более высоким приоритетом, чем `0`, либо повышать приоритет процесса, который уже выполняется.

Многопоточность. Linux не только может параллельно выполнять несколько процессов, но и различает в пределах одного процесса отдельные подпроцессы (потоки). Способность подразделять функции на несколько потоков характерна прежде всего для серверных приложений. Это помогает значительно повысить производительность, в особенности тогда, когда в распоряжении имеется несколько процессоров или многоядерный процессор. При управлении потоками и обменом информацией между ними помогают функции ядра. Начиная с версии ядра 2.6, библиотека Native POSIX Thread Library поддерживает потоки в соответствии со стандартом POSIX.

Переадресация ввода и вывода, программный канал

Практически все текстовые программы (команды) ожидают ввода через так называемый стандартный канал ввода (по умолчанию это клавиатура) и посылают результаты работы через стандартный канал вывода (текст отображается в консоли или в командном окне). И ввод, и вывод можно переадресовывать, благодаря чему открываются широкие дополнительные возможности. Например, следующая команда сохраняет список всех файлов, находящихся в каталоге `xy`, в файл `z`:

```
user$ ls xy > z
```

С помощью так называемых программных каналов вывод одной команды может быть интерпретирован как данные ввода для другой команды. В следующем при-

мере команда grep отфильтровывает из списка всех установленных пакетов те, в которых содержится последовательность символов mysql без учета регистра. Команда sort сортирует полученный список.

Иными словами, благодаря использованию программных каналов (обозначаемых символом |) вывод команды rpm переадресовывается команде grep, а вывод этой команды переадресовывается команде sort (второй символ |). Более подробное объяснение и примеры переадресации ввода и вывода, а также информация об использовании программных каналов приводятся в разделе 8.4.

```
user$ rpm -qa | grep -i mysql | sort
mysql-5.1.32-1.fc11.i586
mysql-connector-java-3.1.12-7.fc11.i586
...
```

4.2. Выполнение процессов от имени другого пользователя (su)

При выполнении программы обычным пользователем действует два ограничения.

- Рядовые пользователи могут выполнять только такие процессы, в которых это допускается в соответствии с правами доступа (пользователь, группа, биты доступа r и x). При работе с обычными программами это ограничение никак не проявляется. Но, к примеру, в каталоге /usr/sbin есть отдельные команды, предназначенные для системного администрирования, которые могут запускаться только администратором.
- Как я уже говорил, процессы принадлежат тому пользователю, который их запустил. Это означает, что процесс имеет доступ к тем же файлам, что и пользователь, запустивший его (то есть ваши программы не имеют доступа к тем файлам, которые вы как пользователь не имеете права изменять). Новые файлы, созданные процессом, также принадлежат пользователю, запустившему программу (см. раздел 3.8).

По этим причинам обычный пользователь не может выполнять многие административные задачи. Очевидно, что самое простое решение — войти в систему с правами администратора. Однако, как я уже не раз указывал, не стоит постоянно работать в качестве администратора: слишком велика опасность случайного причинения вреда. Поэтому в некоторых дистрибутивах в принципе отключена возможность входа с привилегиями администратора. Это касается, например, Ubuntu.

В этом разделе рассказывается, как можно выполнять административные задачи с помощью su и ssh, оставаясь в статусе обычного пользователя. В следующем разделе мы рассмотрим альтернативный метод с sudo, закрепившийся, главным образом, в Ubuntu. Наконец, в разделе 4.4 будет описана программа PolicyKit, предоставляющая принципиально новые возможности выполнения административных задач.

Как обычно, я мог бы рассказать гораздо больше, чем позволяют размеры этой книги. Более подробная информация содержится в документах Security-HOWTO и Remote-X-Mini-HOWTO, расположенных по адресам http://www.tldp.org/HOWTO/Security-HOWTO/index.html и http://www.tldp.org/HOWTO/Remote-X-Apps.html соответственно. Оба этих документа уже достаточно старые, но содержащиеся в них советы по-прежнему сохраняют актуальность.

Команда su. Часто требуется получить привилегии администратора в первую очередь для того, чтобы быстро выполнить команду, — покидать ради этого X было бы очень неудобно. Самая простая возможность изменить пользователя, не покидая окна X-Shell, — воспользоваться командой `su name`. Если вы выполняете эту команду без привилегий администратора, то система запросит у вас пароль соответствующего пользователя. После ввода пароля вы сможете выполнять команды в командном окне (`xterm`, `konsole`) под измененным именем, пока не вернетесь в обычный режим с помощью команды `exit` или сочетания клавиш Ctrl+D.

В следующих строках показано, как обычный пользователь ненадолго входит в систему с привилегиями администратора, пользуясь этими привилегиями, осуществляет привязку сегмента жесткого диска дереву каталогов, а затем выходит из системы (как администратор) и продолжает работать в обычном режиме:

```
user$ su -l root
Password: xxx
root# mount -t ext2 /test /dev/hdc7
root# <Ctrl>+<D>
logout
user$ ls /test
```

Для того чтобы команда `su` была полноценной заменой для входа в систему в качестве администратора, необходимо пользоваться параметром `-l`! Таким образом гарантируется, что стартовые файлы, обеспечивающие вход в систему (например, необходимые для определения `PATH`), будут прочитаны.

Команда kdesu. В KDE для запуска программ X лучше всего использовать команду `kdesu`. Программа показывает окно для пароля администратора.

```
user$ kdesu kate /etc/fstab
```

Команда `kdesu` действует лишь в том случае, когда запущен демон `kdesud`. Обычно этот демон запускается вместе с KDE. Обобщенная справка по всем параметрам `kdesu` предоставляется командой `kdesu -helpall`. В некоторых дистрибутивах `kdesu` интегрирована прямо в меню KDE. Итак, когда вы запускаете программу, требующую привилегий администратора, на экран автоматически выводится форма входа в систему, управляемая `kdesu`.

Команда gksu. Аналог `kdesu` из системы Gnome называется `gksu`. Эта команда работает без дополнительного фонового процесса. Обобщенная справка по параметрам команды дается в `man gksu`.

Сценарий su-to-root. Debian (но только не Ubuntu!) использует для запуска административного инструментария из меню Gnome или KDE независимый от локального компьютера сценарий `su-to-root`. Он входит в состав пакета menu. Справка `man su-to-root` обобщает немногочисленные параметры этого сценария.

Команда consolehelper. В Fedora и Red Hat используется разновидность gksu под названием consolehelper. Эта программа также открывает окно, но работает совершенно по-другому. Основная идея заключается в том, что используемый инструментарий администратора устанавливается в каталог /usr/sbin, где этими инструментами может пользоваться только администратор. Для обычных пользователей в каталоге /usr/bin предусмотрена символьная ссылка на consolehelper. Следующая команда демонстрирует соответствующую конфигурацию system-config-network (сетевая конфигурация для Red Hat или Fedora):

```
user$ ls -l /usr/sbin/system-config-network /usr/bin/system-config-network
-rwxr-xr-x ... root root /usr/sbin/system-config-network
lrwxrwxrwx ... root root /usr/bin/system-config-network -> consolehelper
```

Если теперь администратор выполнит команду system-config-network, то запустится механизм /usr/sbin/system-config-network. Напротив, если обычный пользователь выполнит system-config-network, то запустится consolehelper. Если пользователь укажет правильный пароль администратора, то consolehelper запустит нужную программу system-config-network.

Программа ssh. В большинстве дистрибутивов su работает только с текстовыми командами. На это может быть много причин: во-первых, для запуска X-программы требуется переменная окружения DISPLAY, которая должна содержать имя компьютера, где будет отображаться программа (export DISPLAY=localhost:0). Во-вторых, из соображений безопасности X может запретить пользователям со стороны запускать программы (в данном случае вам пригодится команда xhost). В-третьих, сетевой порт может быть закрыт для обмена информацией с X-сервером.

Если у вас в распоряжении нет kdesu, gksu или consolehelper, то ssh предлагает простейшее решение всех рассмотренных проблем: просто выполните нужную команду по следующему образцу:

```
user$ ssh -X -l user localhost
```

Биты доступа suid и guid. Биты доступа предлагают еще одну возможность помечать определенные программы так, чтобы любой пользователь мог их выполнять, как если бы он был администратором. Важнейшее отличие от sudo заключается в том, что биты доступа suid и guid действительны для *всех* пользователей (а при работе с sudo таких пользователей нужно явно перечислить). Более подробная информация по битам доступа suid и guid находится в подразделе «Специальные биты» раздела 3.7.

4.3. Выполнение процессов от имени другого пользователя (sudo)

Программа sudo применяет при работе принципиально иной подход по сравнению с описанными выше вариантами su. После проведения конфигурации она предоставляет определенным пользователям права администратора для выполнения

определенных программ. Из соображений безопасности указывать при этом нужно *свой* пароль (а не пароль администратора!).

В таком случае sudo выполняет соответствующие программы так, как если бы они были запущены другим пользователем (по умолчанию администратором). Таким образом, отдельные пользователи могут брать на себя выполнение административных задач либо выполнять команды, критически важные для работы системы, не зная при этом пароля администратора. Программа sudo протоколирует все выполненные команды (в том числе неудачные попытки) и обычно заносит эту информацию в файл /var/log/messages.

Программа sudo запоминает пароль на 15 минут. Если в течение этого времени вы выполните с помощью sudo другую команду, вам не будет направляться запрос о пароле (заданное время можно изменить в файле /etc/sudoers с помощью ключевого слова timestamp_timeout).

Конфигурация

Конфигурация sudo осуществляется с помощью файла /etc/sudoers. Проще говоря, в этом файле в трех столбцах описывается, какие пользователи с каких компьютеров имеют право выполнять отдельные программы. Следующая запись означает, что пользователь kathrin, работающий на компьютере uranus, имеет право выполнять команду /sbin/fdisk (ключевое слово ALL означает, что kathrin имеет право выполнять команду под любой учетной записью, то есть как root, news, lp и т. д.).

```
# in /etc/sudoers
kathrin uranus=(ALL) /sbin/fdisk
```

Если в первом столбце перед sudoers поставить символ %, то запись будет действительна для всех членов указанной группы. Многочисленные прочие варианты синтаксиса описаны в файле справки man sudoers.

ВНИМАНИЕ

Из соображений безопасности рекомендую редактировать /etc/sudoers исключительно с помощью visudo (см. также соответствующий раздел справки!). Перед сохранением visudo проверяет синтаксис и гарантирует, что вы не совершили ошибок, которые могли бы не позволить вам выполнять другие административные задачи. Это особенно важно в таких дистрибутивах, как Ubuntu, где не предусмотрен вход в систему с привилегиями администратора.

Применение

Пользователь kathrin теперь может выполнить команду fdisk следующим образом. Вводим пароль пользователя kathrin. При работе с fdisk нужно указать путь целиком, если программа не находится в одном из каталогов PATH пользователя kathrin. Команда fdisk будет автоматически выполнена с привилегиями администратора. Чтобы выбрать другую учетную запись для выполнения, нужно указать ее так: sudo -u «учетная запись».

```
kathrin$ sudo /sbin/fdisk /dev/sda
Password: xxxxxx
```

sudo без пароля. Можно позволить определенному пользователю выполнять sudo, не указывая пароля. Для этого внесите в sudoers строку, построенную по следующему образцу:

```
kofler ALL=(ALL) NOPASSWD: ALL
```

Конечно, такой метод небезопасен, но если вам приходится часто выполнять административные задачи, то вы по достоинству оцените этот удобный способ. Обратите внимание, что метка NOPASSWD действительна лишь в том случае, когда отсутствуют другие строки sudoers, требующие, чтобы пользователь указал пароль. Это же правило действительно для групповых записей, то есть, например, %admin ....

Выполнение нескольких команд с помощью sudo. При решении объемных административных задач со временем станет обременительно ставить перед каждой командой слово sudo. Гораздо более элегантное решение — перейти с помощью sudo -s в режим администратора. Все последующие команды будут выполняться с привилегиями администратора. Чтобы выйти из этого режима, нажмите сочетание клавиш Ctrl+D.

Программа gksudo. При запуске административных программ под X во многих дистрибутивах используется gksudo. Эта программа позволяет ввести пароль в диалоговом окне, а затем запускает желаемую программу (рис. 4.3).

Рис. 4.3. Запуск административной программы с применением программы gksudo

Конфигурация /etc/sudoers предусматривает гораздо больше синтаксических вариантов, чем описано выше. Обязательно прочтите страницы man-справки по sudo и sudoers! Еще подробнее о sudo рассказано на сайте программы: http://www.courtesan.com/sudo/.

sudo в Ubuntu

В Ubuntu и некоторых других дистрибутивах пароль администратора не предусмотрен. Поэтому войти в систему с привилегиями администратора невозможно. Команды su или ssh -l root также не работают. В данном случае можно выполнять

административные команды только с помощью sudo. В файле /etc/sudoers есть только три строки:

```
# Конфигурация по умолчанию в /etc/sudoers в Ubuntu
Defaults   env_reset
root    ALL=(ALL) ALL
%admin  ALL=(ALL) ALL
```

Первая строка возвращает все переменные окружения к исходным значениям в случае смены пользователя. Вторая строка предоставляет администратору неограниченный доступ к любым программам. В Ubuntu эта строка в принципе является бессмысленной, так как вход в систему с привилегиями администратора невозможен. Самая важная строка третья; она позволяет всем членам группы admin обращаться к любым программам.

В Ubuntu по умолчанию членом группы admin является только первый пользователь (тот, кто проводил установку). Если вы создаете в системе учетные записи других пользователей, то их нужно определить в группу Администратор (Система ▸ Управление системой ▸ Пользователь, кнопка Добавить пользователя).

Следующая дополнительная строка в /etc/sudoers позволяет пользователю kofler выполнять без пароля команду apt-get и программу Synaptic. Таким же образом можно устанавливать обновления, не указывая пароль.

```
# Дополнение в /etc/sudoers в Ubuntu
kofler ALL=NOPASSWD: /usr/sbin/synaptic, /usr/sbin/synaptic
```

sudo в SUSE

В SUSE роль sudo значительно скромнее, чем в Ubuntu. Если вы все же хотите воспользоваться sudo, обратите внимание на некоторые особенности конфигурации, задаваемой по умолчанию (при необходимости такие установки можно изменить).

```
# Конфигурация, задаваемая по умолчанию в /etc/sudoers
# в дистрибутиве SUSE версии 11.1 и выше
Defaults always_set_home  # Запрещает выполнять X-программы
Defaults env_reset        # Запрещает выполнять X-программы
Defaults env_keep = "LANG LC_ADDRESS LC_CTYPE ..."
Defaults targetpw         # sudo запрашивает пароль целевого пользователя
ALL ALL=(ALL) ALL         # Если пароль правильный, то всем можно всё
root ALL=(ALL) ALL
```

Первые три строки из соображений безопасности не позволяют выполнять X-программы с помощью sudo. Если sudo должна поддерживать непосредственный запуск таких программ, эту строку нужно удалить или поставить перед ней символ #.

Выражение Defaults targetpw означает, что в принципе необходимо указать пароль для учетной записи, с которой будет выполняться команда (как правило, это пароль администратора). Наконец, строка ALL ALL=(ALL) ALL позволяет любым пользователям выполнять любые команды, если известен пароль нужной учетной записи.

4.4. Выполнение процессов от имени другого пользователя (PolicyKit)

Концепция

Основная концепция PolicyKit заключается в том, что при работе программы подразделяются на два компонента: первый содержит пользовательский интерфейс и работает с обычными пользовательскими правами. Вторая часть программы, именуемая в номенклатуре PolicyKit *«механизм»*, предназначена для вмешательства в работу системы и работает с правами администратора. Такое разделение связано с принципиальным преимуществом — требуется запускать не огромную программу, обладающую правами администратора, а всего лишь небольшие компоненты. Это снижает потенциальный риск для безопасности системы. Кроме того, теоретически существует возможность, что различные пользовательские интерфейсы (например, программы для Gnome и KDE) смогут обращаться к одинаковому набору механизмов.

Обмен информацией между обоими компонентами осуществляется через систему шин (как правило, через шину D-Bus, см. подраздел «Система горячего подключения» раздела 9.6). Решение о возможности выполнения того или иного механизма принимается функциями, которые содержатся в библиотеке PolicyKit и обращаются к центральной базе данных, где хранится информация о правах. При принятии решения учитываются три критерия:

- **субъект** — кто (какой пользователь) хочет внести изменения в систему;
- **объект** — какой объект требуется изменить (например, файл, сегмент диска или сетевое соединение);
- **действие** — что должно быть сделано (например, привязка сегмента к файловой системе).

С точки зрения пользователя

Во многих случаях пользователь даже не замечает работы PolicyKit. Например, стандартная конфигурация большинства дистрибутивов позволяет связывать с файловой системой файловый менеджер и внешние носители данных. Поскольку дальнейшей аутентификации не требуется, процесс протекает автоматически, как только вы подсоедините носитель данных.

При использовании второго варианта, например, при обновлении системы с помощью PackageKit или при доступе к сегменту локального диска по правилам PolicyKit требуется авторизация. Для этого выводится специальное окно, показанное на рис. 4.4. Следует отметить, что при определенной конфигурации PolicyKit запоминает данные, введенные в ходе аутентификации, и сохраняет их в файлах `*.auths` в каталоге `/var/lib/PolicyKit/`. Иначе говоря, если пользователь однажды войдет в систему, чтобы выполнить определенные операции, PolicyKit больше не будет запрашивать авторизацию.

Третий вариант используется при работе с различными административными инструментами Gnome, которые применяются, например, в Ubuntu: здесь кнопка Разблокировать открывает окно аутентификации. Только после указания пароля пользователя или администратора можно будет внести изменения в систему.

Рис. 4.4. Аутентификация с помощью PolicyKit

Конфигурация и администрирование

Конфигурация PolicyKit производится в трех следующих местах:

```
/etc/PolicyKit/PolicyKit.conf         (Глобальная конфигурация)
/usr/share/PolicyKit/policy/*.policy  (Действия)
/var/lib/PolicyKit/*.auths            (Явно установленные права)
```

Базовая конфигурация имеет отличительные особенности, характерные для конкретных дистрибутивов. Например, в системах Ubuntu файл `PolicyKit.conf` дает всем пользователям группы admin права администратора. Это требуется потому, что в Ubuntu, как правило, не предусмотрена учетная запись администратора, следовательно, нет и пароля администратора.

Пользовательские права в PolicyKit также можно настраивать централизованно, через удобный интерфейс. Для этого используется программа `polkit-gnome-authorization` (рис. 4.5). Можно также пользоваться различными командами, позволяющими изменять базу данных с правами пользователей (`polkit-auth`, `polkit-action` и др.). В данном случае обнаруживается второе значительное преимущество PolicyKit по сравнению с другими инструментами аутентификации: это

первый инструмент, который без малейших сложностей позволяет передавать обычным пользователям некоторые права, связанные с управлением системой. Для этого не нужно ни раскрывать пароль администратора, ни затевать неудобных и чреватых ошибками манипуляций, связанных с распределением пользователей по группам.

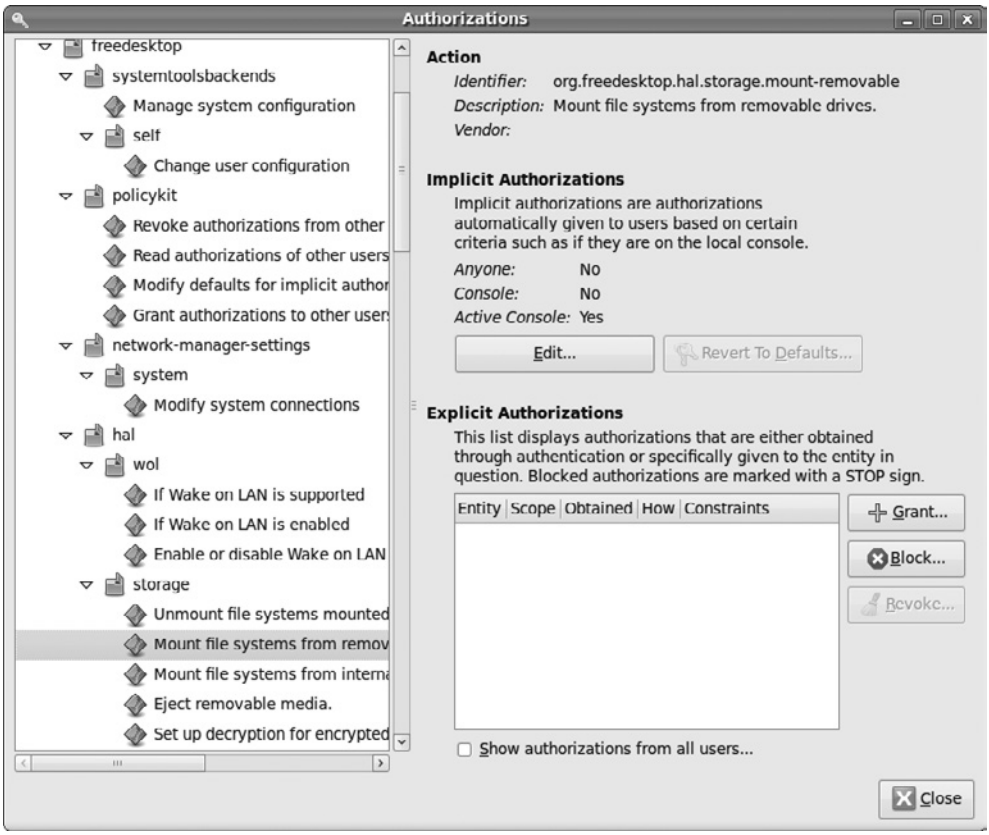

Рис. 4.5. Администрирование с помощью PolicyKit

4.5. Системные процессы (демоны)

Демоны (от англ. daemons) — это фоновые процессы, предназначенные для управления системой (табл. 4.1). Обычно эти процессы запускаются при включении компьютера в рамках процесса Init-V. Если вы хорошо знакомы с терминологией Windows, то подскажу: *демоны* Linux соответствуют *службам* Windows. По мере описания программ на страницах этой книги я буду давать ссылки на посвященные им сайты. Однако не забывайте, что названия демонов могут отличаться от дистрибутива к дистрибутиву (например, `httpd` или `apache2` — демон веб-сервера Apache).

Таблица 4.1. Важные системные процессы

Процесс	Значение
atd	Запуск других программ в заданное время (подобно cron)
avahi-daemon	Автоматическая конфигурация сети (ZeroConf, Rendezvous, Bonjour)
bluetoothd	Управление технологией Bluetooth
cron	Запуск других программ в заданное время
cupsd	Диспетчер печати
dbus-daemon	Обмен данными через шину D-BUS
dhclient	DHCP-клиент
dhcpd	Присваивание другим компьютерам сетевого IP-адреса
dhcpcd	Считывание сетевого IP-адреса
gdm	Менеджер учетных записей Gnome
gpm	Управление мышью для текстовых консолей
hald*	Управление аппаратными компонентами
hpiod	Функции печати и сканирования HP
httpd	Веб-сервер (например, Apache)
kdm	Менеджер учетных записей KDE
klogd	Протоколирование сообщений ядра
lockd	Функция NFS-Locking
lpd	Обычный диспетчер печати на основе BSD-LPD
mdnsd	Автоматическая конфигурация сети (ZeroConf, Rendezvous, Bonjour)
mysqld	Сервер базы данных MySQL
named	Сервер доменных имен
nm-system-settings	Устройство управления сетью
nmbd	Сервер имен для Windows/Samba
nscd	Кэш для хранения информации о пользователях, группах и именах компьютеров
ntpd	Установка времени с помощью сетевого протокола синхронизации времени (Network Time Protocol)
pppd	VPN-клиент, доступ к Интернету
pptpd	PPTP-сервер для VPN
portmap	Часть NFS-сервера
postfix	Почтовый сервер для рассылки электронных сообщений
rpc.*	Сетевые службы RPC (удаленного вызова процедур), обычно для NFS
sdpd	Управление технологией Bluetooth
sendmail	Почтовый сервер для рассылки электронных сообщений
smartd	Система оценки состояния жесткого диска SMART
smbd	Файловый сервер Windows/Samba
squid	Сетевые прокси и сетевой кэш
sshd	Сервер SSH
syslogd	Протоколирование системных сообщений
syslog-ng	Протоколирование системных сообщений
udevd	Управление устройствами
vsftpd	FTP-сервер
xdm	Дисплейный менеджер для X
xinetd	Запуск других сетевых демонов

Потоки ядра

Кроме обычных серверных служб, например `httpd` (Apache), существует ряд фоновых процессов, которые, однако, являются не настоящими программами, а потоками (threads) ядра. Эти процессы можно опознать по тому, что рядом с их названиями в квадратных скобках записывается `ps axu`. Некоторые из таких потоков сопровождаются номером, указывающим на процессор. Таким образом, `kblockd/0` управляет буфером блочного устройства для первого процессора, `kblockd/1` — для второго процессора и т. д.

Большинство потоков ядра занимаются выполнением низкоуровневых задач операционной системы (управление памятью, процессами, ЦПУ и т. д.). Большинство потоков запускаются уже в ходе инициализации системы, в начале ее запуска. Для выполнения обычных требований дополнительного конфигурирования не требуется. Функции важнейших потоков ядра обобщены в табл. 4.2.

Таблица 4.2. Важные потоки ядра

Поток ядра	Значение
aio	Асинхронное управление вводом/выводом (например, для сетевых процессов)
events	Управление событиями и программными прерываниями
kacpid	Функции ACPI (усовершенствованного интерфейса конфигурации и управления питанием)
kblockd	Управление буфером блочных устройств
khelperd	Загрузка или удаление модулей ядра для пользовательских программ
khubd	Управление подключением и извлечением USB-устройств
kjournald	Осуществление журналирования для файловых систем ext3/4
knfsd	NFS-сервер
kthread	Управление потоками
nfsd	NFS-сервер
kscand	Управление памятью
kseriod	Обмен информацией с серийными устройствами
ksoftirqd	Управление аппаратными прерываниями
kswapd	Свопинг
lockd	NFS-Locking
migration	Определение, какие процессы выполняются на каком процессоре
pccardd	Управление картами PCMCIA
pdflush	Физическое сохранение изменений, вносимых в файлы
rpciod	NFS
scsi_eh	Управление ошибками и перерывами SCSI
watchdog	Проверка того, реагирует ли еще система на команды

Запуск и завершение работы демонов

В большинстве дистрибутивов многие из вышеперечисленных демонов запускаются так называемой системой Init-V. Базовая и подробная информация по этой системе приводится в разделе 14.10. Там же будет рассказано, как вы можете интегрировать в систему новые сценарии.

В некоторых новых дистрибутивах вместо Init-V используется Upstart (см. раздел 14.11), поэтому методы работы, хорошо знакомые по Init-V, в дальнейшем также не устареют. Некоторые команды Upstart действуют лишь в том случае, когда демон управляется непосредственно системой Upstart (а не совместимым с ним аналогом из Init-V).

Здесь я очень сжато объясню, как можно запустить или остановить демон вручную, что сделать, чтобы демон автоматически запускался при старте системы, либо как избежать автоматического запуска демона. Эта информация особенно пригодится вам при создании и конфигурировании сетевых служб. Обратите внимание, что в разных дистрибутивах различаются не только команды, но и названия служб. Например, в Debian, Ubuntu и SUSE сценарий для запуска веб-сервера Apache называетсйя `apache2`, а в Fedora и Red Hat — `httpd`.

Запуск вручную

Чтобы запустить демон, сетевую службу или сервер, выполните следующую команду. Обратите внимание, что новые сетевые службы в некоторых дистрибутивах запускаются автоматически сразу после установки пакета (например, это касается Debian и Ubuntu), в то время как в других дистрибутивах запуск необходимо производить вручную.

```
root# /etc/init.d/имя start  (Debian, Fedora, Red Hat, SUSE, Ubuntu)
root# invoke.rc имя start    (Debian, Ubuntu)
root# service имя start      (Fedora, Red Hat)
root# rcимя start            (SUSE)
root# start имя              (Только для процессов Upstart)
```

Остановка вручную

Теперь посмотрите на команды, предназначенные для остановки служб:

```
root# /etc/init.d/имя stop  (Debian, Fedora, Red Hat, SUSE, Ubuntu)
root# invoke.rc имя stop    (Debian, Ubuntu)
root# service имя stop      (Fedora, Red Hat)
root# rc имя stop            (SUSE)
root# stop имя              (Только для процессов Upstart)
```

Перезагрузка/перезапуск

Многие сетевые службы позволяют заново считывать конфигурационную информацию, не останавливая работы системы. Команда `reload` нужна для того, чтобы служба учитывала изменения, внесенные в конфигурационный файл. Службы, которые не поддерживают `reload`, следует полностью перезапускать с помощью команды `restart`.

```
root# /etc/init.d/имя reload/restart (Debian, Fedora, Red Hat, SUSE, Ubuntu)
root# invoke.rc имя reload/restart   (Debian, Ubuntu)
root# service имя reload/restart     (Fedora, Red Hat)
root# rcимя reload/restart           (SUSE)
```

Автоматический старт при запуске компьютера

Когда вы правильно настроите сетевую службу, она будет запускаться автоматически, при загрузке компьютера, а при завершении работы системы служба будет автоматически останавливаться. В некоторых дистрибутивах (например, Debian, Ubuntu) службы конфигурируются требуемым образом уже при установке пакетов. В других дистрибутивах этого не происходит по причинам, связанным с безопасностью, и автоматический старт требуется активизировать вручную.

В Debian команда `update-rc.d` по умолчанию присваивает ссылкам для запуска и остановки Init-V номер 20. Если вы хотите поставить другие номера, укажите их вместо `n1`, `n2` (см. также раздел 14.12).

В Red Hat и Fedora обычно бывает достаточно первой команды `chkconfig`. Вторая команда требуется лишь в тех случаях, когда сценарий Init-V не содержит никаких указаний относительно того, на каком уровне (runlevel) запускается служба (обычно это уровни 3 и 5, (см. также раздел 14.13)).

```
root# update-rc.d имя defaults [n1 n2]  (Debian, Ubuntu)
root# chkconfig --add имя               (Fedora, Red Hat)
root# chkconfig --level 35 имя on       (Fedora, Red Hat)
root# insserv имя                       (SUSE)
```

Предотвращение автоматического запуска

Следующие команды позволяют предотвратить автоматический запуск службы при старте системы. Если служба уже работает, то ее нельзя остановить какой-либо из следующих команд; для этого требуется специальная команда `stop`.

```
root# update-rc.d -f имя remove  (Debian, Ubuntu)
root# chkconfig --del имя        (Fedora, Red Hat)
root# insserv -r имя             (SUSE)
```

4.6. Автоматический запуск процессов (crontab)

Если ваш компьютер вдруг, как кажется, без причины, начнет производить поиск по диску, присылать вам почту и т. д., то, скорее всего, это работа демона `cron`, который автоматически запускает процессы. Демон активизируется раз в минуту, анализирует все файлы `crontab` и запускает указанные в них программы. Он применяется в первую очередь в ходе работ по поддержанию системы — для удаления устаревших файлов регистрации и временных файлов, для обновления каталогов и т. д.

Глобальная конфигурация `cron` производится в файле `/etc/crontab`. Кроме того, пользователи могут задавать собственные задачи для демона `cron`, определяя их в каталоге `/var/spool/cron/tabs/user`.

Права на управление пользовательскими демонами `cron` регулируются в файлах `/var/spool/cron/allow` и `/deny`. Если файл `allow` существует, то команды `cron` могут

выполнять только те пользователи, которые указаны в этом файле. Если существует файл /deny, то указанные в нем пользователи лишены такого права. Если оба файла отсутствуют, выполнение задач зависит от компиляции демона cron, в частности то, смогут ли им пользоваться какие-либо пользователи, кроме администратора.

Файл crontab

Файл /etc/crontab или файлы в каталоге /etc/cron.d содержат построчные записи, в которых перечисляются программы, предназначенные для выполнения (табл. 4.3). Синтаксис таков:

```
# in /etc/crontab
минута час день месяц неделя пользователь команда
```

Таблица 4.3. Столбцы crontab

Столбец	Значение
Минута	Указывает, в какую минуту часа (0–59) должна выполняться программа
Час	Указывает час (0–23)
День	Указывает день месяца (1–31)
Месяц	Указывает месяц (1–12)
Неделя	Указывает день недели (0–7, и 0, и 7 соответствуют воскресенью)
Пользователь	Указывает, какой пользователь должен выполнять команду (обычно это администратор)
Команда	Получает команду, которую необходимо выполнить

Если в любом из первых пяти полей вместо числа стоит символ *, то это поле игнорируется. Например, 15**** означает, что команда должна выполняться через каждые 15 минут после истечения очередного часа, и так каждый час, каждый день, каждый месяц, независимо от дня недели. Запись 29 0 ** 6 означает, что команда должна выполняться каждую субботу в 0:29.

В тех полях, где обозначается время, также допускается запись вида */*n*. Это означает, что команда должна выполняться в каждую *n*-ю минуту (час). Например, */15**** означает, что команда должна выполняться через каждые 15 минут (*n*:00, *n*:15, *n*:30 и *n*:45).

К сожалению, документация по этому демону еще не доработана и из нее не вполне ясно, что означает 0 применительно ко дню месяца или месяцу года (см. man 5 crontab).

Файлы /var/spool/cron/tabs/user имеют тот же формат, что и crontab. Единственное отличие — отсутствует столбец *пользователь*.

Чтобы изменить глобальную конфигурацию cron, можно обработать файл /etc/crontab или файлы в каталоге /etc/cron* прямо в текстовом редакторе. Если к тому же вы хотите внести в cron специфические пользовательские дополнения, для этого следует использовать команду crontab -e (сначала выполните export EDITOR=emacs, если не хотите работать с vi; более подробная информация содержится на справочных сайтах, посвященных cron и crontab).

Каталоги cron.hourly, .daily, .weekly, .monthly

В большинстве дистрибутивов по умолчанию задается такая конфигурация, что в /etc/crontab содержится всего несколько записей, необходимых для ежечасного выполнения сценариев, которые, в свою очередь, расположены в /etc/cron.hourly/*, для ежедневного выполнения сценариев из /etc/cron.daily/* и т. д. В Ubuntu каталог /etc/crontab выглядит так:

```
# /etc/crontab
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || \
                      ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || \
                      ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || \
                      ( cd / && run-parts --report /etc/cron.monthly )
```

Это означает, что демон cron:

- через 17 минут после истечения каждого часа выполняет все сценарные файлы, находящиеся в каталоге /etc/cron.hourly;
- ежедневно в 6:25 выполняет все сценарные файлы из каталога /etc/cron.daily;
- еженедельно в 6:47 в воскресенье выполняет все сценарные файлы из каталога /etc/cron.weekly;
- в первое число каждого месяца в 6:52 выполняет все сценарные файлы из каталога /etc/cron.monthly.

Если установлена программа Anacron, то сценарии из /etc/cron.daily, -.weekly и -.monthly выполняться не будут.

Чтобы самостоятельно регулярно выполнять сценарии, занимающиеся резервным копированием, обновлениями и выполнением других задач, просто поместите соответствующий сценарный файл в один из каталогов: /etc/cron.hourly, -.daily, -.weekly или -.monthly. Не забудьте установить бит execute (chmod a+x *файл*)!

Если вы хотите изменить время, заданное в crontab для работы с каталогами /etc/cron.*xxx*, то, разумеется, можете вписать в /etc/crontab дополнительную строку и указать время, когда должен выполняться ваш сценарий. Еще нагляднее было бы создать в /etc/cron.d соответствующий новый файл. Обратите внимание, что все конфигурационные файлы cron должны заканчиваться разрывом строки, иначе система будет игнорировать последнюю строку!

Если вы занимаетесь срочной и важной работой, например записываете DVD, и хотите обезопасить систему от возможных помех, которые могут возникнуть из-за работы демона cron, просто временно отключите его: /etc/init.d/cron stop.

Планировщик задач Anacron

Наряду с демоном cron в большинстве дистрибутивов также установлен планировщик задач Anacron. Задача Anacron — обеспечивать ежедневное, еженедельное или

ежемесячное выполнение различных задач также и в том случае, когда компьютер используется с перерывами, а по ночам и в выходные бывает выключен.

Anacron однократно (по требованию) выполняет сценарии из каталогов `/etc/cron.daily`, `-.weekly` и `-.monthly`. В отличие от `cron`, он завершает работу после выполнения необходимых сценариев.

ПРИМЕЧАНИЕ

На сервере, который может неделями работать без перерывов, применять Anacron нецелесообразно. Anacron выполняет задачи cron в каталогах /etc/cron.daily, -.weekly и -.monthly только по одному разу, а cron, в свою очередь, игнорирует эти задачи, если на компьютере установлен Anacron. В результате все без исключения задачи будут выполнены только по одному разу, независимо от того, как долго работает сервер!

Anacron сохраняет момент выполнения задачи в файлах каталога `/var/spool/anacron`. Таким образом, исключается ситуация, в которой определенный сценарий выполнялся бы дважды в один и тот же день. Anacron управляется из каталога `/etc/anacrontab`; конфигурация, задаваемая по умолчанию, обычно вполне подходит.

5 Конвертер графических, аудио- и текстовых файлов

В Linux имеется множество команд, с помощью которых можно преобразовывать изображения, тексты и другие файлы из одних форматов в другие: из GIF в JPEG, из PostScript в PDF, из HTML в обычный текст, из MP3 в WAV и т. д.

В этой главе я проведу краткий обзор таких команд и покажу примеры их использования. Если в вашей системе не окажется каких-либо команд, рассмотренных в этой главе, найдите и установите соответствующий пакет (однако обычно таких ситуаций не бывает).

5.1. Графический конвертер

Среди многих графических конвертеров, разработанных для Linux, выделяются два: Image Magick и Netpbm. Оба пакета приспособлены к работе со множеством графических форматов, содержат команды (параметры), упрощающие работу с изображениями (изменение изображения, его фрагмента, улучшение контрастности, уменьшение количества цветов и т. д.). Далее я коротко представлю оба пакета, а также некоторые входящие в их состав команды и библиотеки, применяемые для преобразования файлов.

Image Magick. Программный пакет Image Magick состоит из множества отдельных команд, самая важная из которых — `convert`. Она создает из имеющегося файла изображения новый файл, изменяя при этом формат. Исходный и конечный форматы понятны уже по имени файла. Например, следующая команда создает файл `image.png` в формате PNG:

```
user$ convert bild.jpg bild.png
```

Благодаря наличию более 100 параметров можно одновременно изменять различные свойства изображения (размер, количество цветов, степень сжатия и т. д.):

```
user$ convert -resize 100x100 bild.jpg bild.png
user$ convert -type Grayscale bild.jpg bild.eps
user$ convert -quality 80 bild.bmp bild.jpg
```

Команда mogrify работает подобно convert, но изменяет существующий файл (не создавая нового):

```
user$ mogrify -resize 50% test.jpg
```

Наконец, кратко рассмотрим еще некоторые команды: compare сравнивает два изображения, conjure выполняет команды сценарного языка Magick Scripting Language (MSL). Команда identify возвращает описание файла изображения (формат, размер и т. д.), import создает скриншот и сохраняет изображение в файле, montage создает из нескольких изображений новую картинку.

```
user$ identify -verbose bild.png
Image: bild.png
  Format: PNG (Portable Network Graphics)
  Geometry: 85x100
  Type: TrueColor
  ...
```

Пример небольшого Shell-сценария, создающего эскизы (миниатюрные варианты) всех изображений, файлы с которыми были переданы сценарию в виде параметров, приводится в подразделе «Пример 4: сценарий резервного копирования» раздела 8.8. Подробная документация по всем командам находится на сайте http://www.imagemagick.org/.

Netpbm. По принципу, напоминающему Image Magick, работает пакет Netpbm (ранее — Portable Bitmap Utilites). Для обработки в данном пакете любые файлы необходимо преобразовывать во внутренний формат Pnm или Pbm. В следующем примере показано преобразование файла из формата TIFF в формат EMS с одновременной нормализацией цветового формата изображения (pnmnorm):

```
user$ tifftopnm bild.tif | pnmnorm | pnmtops -noturn -rle -scale 0.5 > bild.eps
```

На сайте http://netpbm.sourceforge.net/doc/ приводится описание около 200 команд пакета Netpbm.

Библиотека libtiff. Пакет libtiff содержит одноименную библиотеку, а также различные команды для обработки и преобразования TIFF-файлов. Важнейшие команды для преобразования изображений — bmp2tiff, gif2tiff, tiff2pdf и tiff2ps. При работе с TIFF-файлами будут особенно полезны команды tiffcp, tiffinfo и tiffsplit.

Библиотека libwmf. Пакет libwmf содержит одноименную библиотеку, а также различные команды для обработки и преобразования WMF- и EMF-файлов (метафайлы среды Windows, или улучшенные метафайлы). Важнейшие команды для преобразования изображений — wmf2eps, wmf2svg и wmf2gd (преобразование в форматы JPEG и PNG).

SVG-конвертер. В зависимости от дистрибутива пакет librsvg2 или librsvg2-bin может содержать команды rsvg и rsvgconvert, позволяющие преобразовывать файлы в формате SVG (масштабируемая векторная графика) в точечные изображения.

EXIF. В некоторых дистрибутивах имеются различные библиотеки и команды для преобразования EXIF-файлов в JPEG, например exif, exiftran или exiv2.

RAW-файлы. Некоторые цифровые фотоаппараты позволяют сохранять снимки без потери качества в формате, заданном производителем (обычно патентованном). При преобразовании таких файлов в обычные изображения поможет команда `dcraw` из одноименного пакета.

5.2. Аудио- и видеоконвертер

В табл. 5.1 перечислены важнейшие команды, позволяющие считывать аудиофайлы с CD либо преобразовывать аудио- и видеофайлы из одного формата в другой. Если имя пакета не входит в название команды, то оно дополнительно указывается в скобках (название одного и того же пакета в различных дистрибутивах может варьироваться).

Таблица 5.1. Аудио- и видеоконвертер

Формат	**Команда (пакет)**
CD→WAV	icedax, cdparanoia
MP3→WAV	mpg123, mpg321, madplay
WAV→MP3	lame
OGG→WAV	oggdec (vorbis-tools)
WAV→OGG	oggenc (vorbis-tools)
MP3→OGG	mp32ogg
AAC→WAV	faad
WAV→AAC	faac
WAV↔FLAC	flac
Аудио↔аудио	sox
Аудио↔аудио	sfconvert (audiofile)
Аудио/видео↔аудио/видео	Ffmpeg (общий аудио- и видеоконвертер)

Далее я сообщу дополнительную информацию и приведу примеры по самым важным командам. Ради экономии места я не буду отдельно останавливаться на параметрах этих команд (если вам нужна дополнительная информация, то обратитесь к `man` *`имя_команды`*).

CD-риппер. В этом разделе мы поговорим, как лучше всего переписывать аудиотреки с компакт-диска на винчестер с максимальной эффективностью и минимальными потерями качества. Среди наиболее популярных команд, которые разрабатывались специально для этого, необходимо отметить `cdparanoia` и `icedax` (ранее `cdda2wav`). Команда `cdparanoia` снискала отличную репутацию при работе с поцарапанными дисками и при решении других сходных проблем. Обе команды обладают большим количеством параметров.

Рассмотрим два примера. Первая команда считывает дорожку 3 с компакт-диска, который находится в SCSI-CD-приводе. Полученный в результате файл называется `audio.wav`:

```
root# icedax -D /dev/scd0 -t 3
```

Следующая команда считывает дорожку 4 с компакт-диска, который находится в том же дисководе. Полученный в результате файл называется cdda.wav и находится в локальном каталоге:

```
root# cdparanoia -d /dev/scd0 "4"
```

MP3-кодер. Из-за проблем, связанных с патентами, сегодня практически не осталось дистрибутивов Linux, вместе с которыми поставляется MP3-кодер. Однако различные кодеры размещены в Интернете в виде дополнительных пакетов. Самая известная подобная программа называется lame (http://lame.sourceforge.net/).

Работать с ней достаточно просто: команда lame input.wav output.mp3 создает из исходного файла в формате WAV соответствующий файл в формате MP3. Ход процесса и, в особенности, желаемый уровень качества MP3-файла зависят от применения разнообразных параметров.

MP3→OGG. Перечисленные выше команды можно комбинировать, чтобы, например, преобразовать MP3-файл в формат Ogg-Vorbis. Однако следует учитывать, что при таких преобразованиях снижается качество материала, поэтому их по возможности следует избегать.

```
user$ mpg321 -s in.mp3 -w - | oggenc - -o out.ogg
```

К сожалению, при преобразовании также теряются информационные метки (ID3). Однако от этого недостатка можно избавиться, применяя сценарий MP3-Ogg-Konverter (mp32ogg с сайта http://faceprint.com/code/).

FLAC. FLAC означает Free Lossles Audio Codec (дословно «свободный аудиокодек без потерь»). Файлы FLAC несколько больше, чем MP3- или OGG-файлы, но они гораздо меньше WAV-файлов. Значительное преимущество по сравнению с MP3 или OGG заключается в том, что с помощью формата FLAC можно кодировать аудиофайлы без потерь. Для кодирования и декодирования используется команда flac.

SoX. SoX означает sound exchange, а команда sox обеспечивает еще одну возможность преобразования файлов из одного формата в другой. Она поддерживает больше форматов, чем sfconvert. Для sox существует X-интерфейс xsox, а также Gnome-интерфейс gsox.

Библиотека Audio File. Пакет audiofile внедряет важнейшие функции одноименной библиотеки, разработанной производителем SGI. Самая интересная команда — sfconvert: она конвертирует форматы аудиофайлов aiff, aifc, next и wave. Команда sfinfo пытается установить, в каком формате записан аудиофайл.

SoundConverter. Если вы хотите выполнять конвертирование аудиофайлов с большим удобством, вам, вероятно, понравится программа soundconverter. Она преобразует все заранее выбранные аудиофайлы в желаемый формат (меню Правка ▸ Настройки, по умолчанию выбран формат Ogg-Vorbis). Новые файлы сохраняются в том же каталоге, что и исходные, однако, конечно же, получают новое расширение (например, *.ogg).

Видеоконвертер (ffmpeg). Команда ffmpeg из одноименного пакета преобразует аудио- и видеофайлы из одного формата в другой. Список поддерживаемых форматов очень велик, его можно узнать с помощью команды ffmpeg -formats.

```
user$ ffmpeg -i in.avi out.mpg
```

Кроме того, эта команда подходит для записи аудио- и видеофайлов на DVD. Соответствующий пример приводится в `man ffmpeg`.

5.3. Текстовые конвертеры (кодировка и разрывы строк)

В этом разделе будут представлены команды `recode`, `iconv`, `unix2dos` и `dos2unix`. Они применяются для изменения кодировки и символов разрыва строки в обычных текстовых файлах. Это требуется в тех случаях, когда текстовые файлы передаются между системами, в которых отличаются кодировки или конвенции текстовых форматов.

Команда recode. Эта команда преобразует набор символов из кодировки 1 в кодировку 2. Следующая команда преобразует DOS-файл `dosdat` в файл с кодировкой «латиница-1»:

```
user$ recode ibmpc..latin1 < dosdat > linuxdat
```

Команда `recode` считывает текстовый файл `latin1dat` с кодировкой «латиница-1» и сохраняет его в формате UTF-8 (Unicode):

```
user$ recode latin1..u8 < latin1dat > utf8dat
```

Команда iconv. Популярная альтернатива для `recode` — команда `iconv`. Правда, эта команда не может изменять символы разрыва строки. В следующем примере опять же создается файл UTF-8 из соответствующего файла в кодировке «латиница-1».

```
user$ iconv -f latin1 -t utf-8 latin1dat > utf8dat
```

Команды dos2unix, unix2dos. Эти команды изменяют символы разрыва строки при преобразовании файла из формата, типичного для MS-DOS/Windows (CR плюс LF) в формат, типичный для UNIX/Linux (только LF). Команды пригодны для работы с такими текстовыми файлами, которые содержат только однобайтовые последовательности символов (например, ASCII, латиница-1), но не для файлов в формате Unicode!

```
user$ dos2unix файл.txt
```

5.4. Конвертер имен файлов (кодировка)

Команда convmv. Еще несколько лет назад во многих дистрибутивах Linux было принято записывать имена файлов в латинице-1. Однако со временем стандартной кодировкой стал считаться Unicode (UTF-8). При переводе имен файлов из одной кодировки в другую используется команда `convmv`, однако она редко имеется в дистрибутиве по умолчанию. В некоторых дистрибутивах ее можно без труда установить в виде одноименного пакета. Если же для конкретного дистрибутива не предусмотрен соответствующий пакет, то вам потребуется скачать сценарий на Perl с сайта http://j3e.de/linux/convmv/.

Чтобы преобразовать все файлы одного каталога из кодировки «латиница-1» в UTF-8, (в обратном порядке, с запросом о подтверждении преобразования каждого отдельно взятого файла), вызовите `convmv` следующим образом:

```
user$ convmv -r -i --notest -f iso-8859-1 -t utf8 название_каталога
```

Команда `convmv` изменяет только имена, а не содержимое файлов! При первых испытаниях рекомендую отказаться от параметра `--notest`: тогда `convmv` только отобразит запланированные изменения, но не будет их выполнять.

Команда `convmv` пытается сама опознавать имена файлов, которые уже работают с кодировкой UTF-8, и в этом случае не изменяет имени файла. Такой механизм защиты отключается с помощью параметра `--nosmart`.

5.5. Конвертер документов (PostScript, PDF, HTML, LATEX)

В этом разделе представлены команды, применяемые при обработке и преобразовании документов, созданных в форматах PostScript, PDF, HTML и т. д. В табл. 5.2. представлен их список.

Таблица 5.2. Конвертер документов

Формат	Команда
Text→PostScript	a2ps, enscript, mpage
HTML→Text, PostScript	html2text, html2ps
PostScript↔PDF	ps2pdf, epstopdf, pdf2ps, pdftops
PostScript, PDF → точечные рисунки, формат для вывода на печать	gs
PostScript→PostScript (извлечение страниц и т. д.)	psutils
PDF→PDF (извлечение страниц, изображений и т. д.)	pdftk, pdfnup, pdfjoin, pdfedit
PDF→Text	pdftotext
LATEX→DVI, PostScript, PDF	latex, dvips, dvipdf, dvipdfm

Text→PostScript

Если вы распечатываете текстовые файлы с помощью команды `lpr файл`, то система печати обычно автоматически форматирует текст. Если у вас есть свои соображения относительно того, как должен быть отформатирован полученный фрагмент текста, то рекомендуется отформатировать файл PostScript вручную и дополнительно его распечатать. Для выполнения этой задачи подходят, в частности, команды `a2ps`, `enscript` и `mpage`. Функционал трех этих команд одинаков, они отличаются только предлагаемым набором параметров форматирования.

Команда a2ps

Эта команда расшифровывается как Any to PostScript и может преобразовывать в PostScript, например, текстовые файлы справки. В следующих примерах мы огра-

ничимся обычными текстовыми файлами. Обратите внимание на то, что в текстовых файлах должна использоваться кодировка-латиница (а не Unicode)!

По умолчанию команда форматирует текст в виде двух столбцов с альбомной ориентацией.

```
user$ a2ps text.txt -o postscript.ps
```

Следующая команда обрабатывает несколько текстовых файлов и форматирует полученный результат как «четыре страницы на листе». Если a2ps опознает текст как программный код, она автоматически выделит синтаксис (ключевые слова — полужирным шрифтом, комментарии — курсивом и т. д.).

```
user$ a2ps файл1.c файл2.c файл3.h -4 -o postscript.ps
```

Описание важнейших параметров приводится на сайте http://www.inf.enst.fr/~demaille/a2ps/.

Команда enscript

Эта команда преобразует текстовые файлы в форматы PostScript, HTML и RTF. Она работает с текстовыми файлами в кодировке «латиница-1». Следующая команда создает страницы формата A4 с альбомной ориентацией и тремя столбцами текста:

```
user$ enscript -M A4 --landscape -3 text.txt -p postscript.ps
```

Команда mpage

Эта команда также преобразует текстовые файлы в формат PostScript, по умолчанию по четыре страницы на лист и в формате «письма». Следующая команда создает страницы A4 с альбомной ориентацией, с двумя страницами на листе:

```
user$ mpage -2 -bA4 text.txt > postscript.ps
```

Unicode

К сожалению, ни одна из указанных выше программ не может работать с кодировкой Unicode (UTF-8). **Если вы хотите распечатать документы, имеющие кодировку Unicode**, то лучше всего воспользоваться редактором, поддерживающим ее. Чтобы просто и быстро преобразовать текст в PostScript, можно также применить команду cnprint: http://www.neurophys.wisc.edu/~cai/software.

HTML→Text, PostScript

Команда html2text

Эта команда преобразует HTML-документы в обычные текстовые файлы. Это бывает полезно в тех случаях, когда HTML-файлы необходимо передать в виде, удобном для чтения из браузера.

```
user$ html2text файл.html > text.txt
```

Команда `html2text` выдает текст в кодировке «латиница-1» (не Unicode!). Форматирование текста управляется несколькими параметрами, а также `/etc/html2textrc` или `~/.html2textrc` (см. `man html2textrc`).

Для преобразования HTML в текст можно также пользоваться браузерами с текстовой ориентацией.

Команда html2ps

Для автоматического преобразования данных из HTML в PostScript подходит сценарий на языке Perl — `html2ps`. Работать с ним достаточно просто: `html2ps -D` *`имя`*`.html > `*`имя`*`.ps`. Благодаря параметру `-D` в файл PostScript вставляются DSC-совместимые[1] комментарии, что сильно облегчает дальнейшую работу с файлом.

PostScript↔PDF

Команда ps2pdf

Команда `ps2pdf` *`исходный.ps конечный.pdf`* создает из любого файла PostScript файл формата PDF. Функционально эта программа в принципе не отличается от коммерческого инструмента Adobe Distiller. Она основана на Ghostscript.

На настоящий момент команда создает файлы, совместимые с форматом PDF 1.2 для программы Acrobat Reader 3.*n*. Однако, судя по документации, в будущем установки, задаваемые по умолчанию, могут измениться. Если вы хотите гарантировать совместимость с конкретной версией PDF, используйте команды `ps2pdf12`, `ps2pdf13` и `ps2pdf14`.

Качество файлов в формате PDF во многом зависит от того, какие шрифты используются в документе PostScript. Если определенные шрифты не поддерживаются, то их символы необходимо заменять точечными рисунками, из-за чего качество шрифта серьезно снижается. Работа `ps2pdf` регулируется многочисленными параметрами. Полная документация находится по адресу http://www.cs.wisc.edu/~ghost/doc/cvs/Ps2pdf.htm.

Команда epstopdf

Если вы хотите преобразовать изображение EPS в файл PDF, воспользуйтесь командой `epstopdf`. EPS расшифровывается как Encapsulated PostScript (инкапсулированный PostScript) и включает в себя такие файлы, в которых с помощью так называемой *граничной рамки* задается точный размер рисунка. Файлы в формате EPS хорошо встраиваются в другие документы (например, LATEX или OpenOffice).

Команда `epstopdf` обычно входит в состав пакета `tetex`, содержащего TEX, LATEX и другие TEX-специфичные программы. В отличие от `ps2pdf`, команда `epstopdf` учитывает размер изображения. К сожалению, `epstopdf` не может переносить в PDF точечные рисунки, содержащиеся в файле EPS, в неизменном виде (а также файлы с параметром `--nocompress`).

---

[1] DSC — Соглашения о структурировании документа (Document Structuring Conventions).

Команды pdf2ps и pdftops

Команда `pdf2ps` *`исходный.pdf конечный.ps`* — это команда, обратная `ps2pdf`; `pdf2ps` также работает с командой `gs`.

Команда `pdftops` выполняет в принципе те же задачи, что и `pdf2ps`, но иначе встраивается в систему и содержит значительно больше параметров, влияющих на вид итоговых файлов PostScript. Например, можно указывать желаемый уровень PostScript, размер бумаги и т. д.

PostScript/PDF→ формат для вывода на печать/ точечная графика

Команда gs

Эта команда, также известная под названием Ghostscript, преобразует документы PostScript и PDF в различные форматы для вывода на печать и с использованием точечных рисунков. Ghostscript — это важнейший элемент системы печати в Linux (например, CUPS, см. раздел 20.9), так как он обеспечивает распечатку документов в формате PostScript на принтерах, не поддерживающих этот формат. Кроме того, эта программа применяется в различных инструментах для просмотра и преобразования PostScript.

Команда `gs` использует шрифты, установленные на компьютере, а также собственную коллекцию гарнитур, которая обычно находится в пакете `ghostscript-fonts`. Эти гарнитуры необходимы для того, чтобы преобразовать шрифты PostScript в точечную графику.

Существуют различные версии Ghostscript. В большинстве дистрибутивов Linux применяется GNU Ghostscript или его вариант ESP Ghostscript. Обе версии соблюдают лицензию GPL. Вариант ESP Ghostscript специально оптимизирован под совместную работу с CUPS. ESP — это название компании, означает Easy Software Products.

Кроме того, имеются коммерческие версии Ghostscript (AFPL Ghostscript, Artifex Ghostscript), продаваемые, например, производителями принтеров. Раньше в коммерческих версиях имелись дополнительные функции, которые появлялись и в бесплатных версиях по прошествии некоторого времени. С лета 2006 года новые функции предоставляются одновременно как в коммерческих, так и в бесплатных версиях Ghostscript. Более подробная информация по различным версиям имеется на следующих сайтах:

- http://www.ghostscript.com/;
- http://www.cs.wisc.edu/~ghost/;
- http://www.artifex.com/.

Внешние драйверы принтеров (Gutenprint)

В Ghostscript интегрировано множество драйверов для принтеров. Однако многие из них были разработаны вне проекта Ghostscript. Важнейший современный

проект по разработке драйверов называется Gutenprint. Он развился из Gimp-Print — коллекции драйверов печатающих устройств для Gimp. Теперь разрабатываемые драйверы не зависят от Gimp и также используются Ghostscript и CUPS, поэтому с лета 2006 года проект называется Gutenprint. Более подробная информация находится по адресу http://gutenprint.sourceforge.net/.

Здесь также стоит упомянуть о проекте HPLIP (HP Linux Imaging and Printing). В рамках этого проекта компания Hewlett-Packard предоставляет свободно распространяемые драйверы для различных принтеров и сканеров. Однако HPLIP не связан с Ghostscript и применяется в сочетании с CUPS (раздел 20.9).

Вызов вручную

Поскольку Ghostscript хорошо интегрирован в печатную систему и многие другие программы, вызывать `gs` вручную доводится нечасто. Чтобы `gs` работала правильно, необходимо задать как минимум два параметра: `-sOutputFile=` для указания файла, в который заносится результат и `-sDEVICE=`*имя* или `@`*имя*`.upp` для настройки формата вывода. Как правило, стоит также использовать параметр `-dNOPAUSE`. Если вы собираетесь печатать на бумаге DIN-A4, то укажите еще и `-sPAPERSIZE=a4`.

Следующая команда переводит `test.ps` в формат принтера HP-Laserjet 3. Результат записывается в файл `out.hp`:

```
user$ gs -sDEVICE=ljet3 -sOutputFile=out.hp -sPAPERSIZE=a4 -dNOPAUSE -dBATCH test.ps
```

Во втором примере PostScript преобразуется в формат PDF (`ps2pdf` — это просто сценарий, вызывающий `gs`).

```
user$ gs -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=out.pdf test.ps
```

И, наконец, вот команда, преобразующая файл EPS в формат PNG:

```
user$ gs -dNOPAUSE -dBATCH -sDEVICE=png16m -sOutputFile=out.png \
         -dEPSCrop -r100 bild.eps
```

Утилиты PostScript

Пакет psutils

При обработке файлов PostScript очень полезны команды из пакета `psutils`. Часть из них — самостоятельные программы, часть — сценарии bash или Perl. Подробно описать все эти команды здесь не позволяют размеры книги, но того списка команд, который приводится в табл. 5.3, вам должно хватить, чтобы иметь представление об этих инструментах. Более подробная информация есть на соответствующих страницах справки.

Таблица 5.3. Команды пакета psutils

Команда	Функция
epsffit	Регулирует размер EPS-файла
extractres	Анализирует файл и вставляет комментарии %%IncludeResource для всех необходимых шрифтов, файлов и т. д.
fixfmps	Приспосабливает файлы FrameMaker к правилам psutils

Команда	Функция
fixmacps	Приспосабливает файлы Macintosh к правилам psutils
fixscribeps	Приспосабливает файлы Scribe к правилам psutils
fixtpps	Приспосабливает файлы Troff/Tpscript к правилам psutils
fixwfwps	Приспосабливает файлы MS Word к правилам psutils
fixwpps	Приспосабливает файлы WordPerfect к правилам psutils
fixwwps	Приспосабливает файлы MS Write к правилам psutils
getafm	Создает файлы AFM для описания шрифтов
includeres	Вставляет комментарии, созданные extractres, в файл PostScript
psbook	Упорядочивает страницы с текстом так, что их можно печатать в виде тетрадей по 16 листов
psnup	Размещает на одном листе несколько уменьшенных страниц
psresize	Изменяет параметры страницы в документе. Эта команда позволяет решить проблему, регулярно возникающую при распечатке документов PostScript, созданных в формате «письма», используемом в США
psselect	Извлекает отдельные страницы из файла PostScript
pstops	Переупорядочивает страницы в документе

В следующем примере показано, как преобразовать файл PostScript с рукописью этой книги, созданный с применением LATEX и DVIPS в форму «64 страницы на листе». Таким образом, каждая страница уменьшится до размеров почтовой марки. При работе с программой для просмотра открываются возможности быстрого просмотра внешнего вида страниц (как в режиме Предварительный просмотр в Microsoft Word с максимальным уменьшением):

```
user$ psnup -b-0.4cm -64 -q < linux.ps > preview.ps
```

К сожалению, вышеуказанные команды функционируют лишь тогда, когда файлы PostScript содержат DSC-совместимые комментарии (как было указано выше, DSC означает «Соглашения о структурировании документа»). Комментарии не распечатываются, но содержат важную информацию о размере страницы, ее начале, конце и т. д. Файлы EPS — это одностраничные файлы PostScript, содержащие комментарии о том, как вставлять этот материал в другие документы (в частности, данные о граничной рамке, определяющей размер распечатки).

Объединение файлов PS

Чтобы объединить два и более файлов PostScript, лучше всего воспользоваться командой Ghostscript `gs`. Правда, достичь удовлетворительных результатов удастся лишь в том случае, если она найдет все файлы шрифтов.

```
user$ gs -sDEVICE=pswrite -sOutputFile=out.ps -dNOPAUSE -dBATCH in1.ps in2.ps ...
```

Утилиты PDF

Команда pdftk

Эта команда (инструментарий PDF) предлагает функции для обработки PDF, подобные тем, что `psutils` предлагает для файлов PostScript. С помощью этого

инструментария можно извлекать страницы, объединять несколько PDF-документов, создавать незашифрованную версию зашифрованного документа PDF (при условии, что вы знаете пароль), заполнять PDF-формы и т. д. Подробная информация по синтаксису инструментария приводится на сайте http://www.accesspdf.com/pdftk/.

Следующая команда считывает страницы 10–20 и 30–40 из файла `in.pdf` и записывает их в новый файл `out.pdf`:

```
user$ pdftk in.pdf cat 10-20 30-40 output out.pdf
```

Чтобы объединить несколько файлов в формате PDF, используйте команду `cat`:

```
user$ pdftk in1.pdf in2.pdf in3.pdf cat output out.pdf
```

В следующем примере для каждой отдельной страницы из файла `in.pdf` создается отдельный файл PDF с именем `pg_n`, где *n* — номер страницы:

```
user$ pdftk in.pdf burst
```

В следующем примере создается зашифрованный PDF-файл. Читать его можно и без пароля `xxx`, но распечатывать или вносить изменения нельзя. Если вы хотите защитить файл даже от чтения, используйте вместо `owner_pw` команду `user_pw`.

```
user$ pdftk in.pdf output encrpyted.pdf owner_pw xxx
```

Poppler

Это собрание команд, предназначенных для преобразования PDF-документов в другие форматы (Text, Bitmap, PostScript и т. д). Poppler в Linux используется во многих программах, предназначенных для просмотра PDF. Обычно эта программа находится в пакете `poppler-utils`.

Пакеты

Пакет xpdf-utils. Этот пакет содержит среди прочих команды `pdftops` (преобразует документы формата PDF в PostScript), `pdfinfo` (извлекает свойства документов PDF), `pdfimages` (извлекает изображения из файлов PDF) и `pdftotext` (извлекает текст из файлов PDF).

Пакет pdfedit. Он включает различные инструменты и пользовательский интерфейс, предназначенные для изменения файлов PDF.

Пакет pdfjam. Пакет содержит команды `pdfnup`, `pdfjoin` и `pdf90`. Они могут объединять и поворачивать документы PDF.

GUI

Если же вам не нравится работать с командами и их параметрами и недостает графического пользовательского интерфейса, то вам предлагается богатый выбор свободно распространяемых программ, часто основанных на Java. Кроме уже упомянутой программы PDFedit достаточно интересны прежде всего PDF-Shuffler, Bookbinder, JPDF Tweak, а также PDF Split и Merge (PDF Sam).

LATEX и компания

LATEX — это система для набора (верстки) научных текстов. В данном пораздeле будут кратко описаны важнейшие команды, позволяющие преобразовывать файлы LATEX (`*.tex`) в другие форматы, но вдаваться в подробности синтаксиса LATEX мы не будем.

Команда latex. Команда `latex` *`имя`*`.tex` создает из LATEX файл в формате DVI. Он содержит все команды, необходимые для верстки страницы на принтеро- или устройство-независимом языке.

Команда dvips. Когда файл DVI будет готов, его можно просмотреть с помощью программ `xdvi` или `kdvi`. Команда `dvips` позволяет преобразовать файл DVI в формат PostScript. На примере следующей команды понятен принцип построения синтаксиса в таких случаях.

```
user$ dvips [параметры] -o имя.dvi имя.ps
```

Команда dvipdf. Часто бывает необходимо преобразовать документы LATEX в PDF. Для этого имеется несколько возможностей.

- Сначала с помощью `dvips` создается файл PostScript, который затем преобразуется в PDF с помощью `ps2pdf` или Adobe Distiller (Adobe Distiller входит в состав коммерческого программного пакета Adobe Acrobat, однако, к сожалению, пока не существует версии этой программы для Linux).
- Можно преобразовать файл DVI в PDF с помощью `dvipdf` или `dvipdfm`. При этом `dvipdf` соответствует пункту, описанному выше, так как в качестве промежуточного этапа тоже создается файл PostScript.

В любом случае результатом является файл PDF, выглядящий так же, как и эквивалентный файл PostScript. Можно ли будет использовать дополнительные функции PDF (интерактивное оглавление, переходы по ссылкам), зависит от конкретного способа преобразования и дополнительных пакетов, использовавшихся при создании документа LATEX:

- `dvips/ps2pdf` или `dvipdf` — функции PDF могут использоваться с помощью пакета LATEX hyperref;
- `dvipdfm` — в данном случае следует вставить дополнительные команды `\special` в документ LATEX;
- `PDFTEx` — в этой программе имеется ряд дополнительных команд LATEX, предназначенных для управления функциями PDF.

6 Сетевые инструменты

В этой главе будут представлены команды, предназначенные для работы с простейшими сетевыми службами. Здесь мы рассмотрим, как с использованием `ssh` войти в сеть с другого компьютера, как передать файлы с помощью команды `wget` или `rsync` и т. д. Для работы с некоторыми из представленных здесь команд существуют удобные пользовательские интерфейсы. Однако, как и в предыдущих главах, мы рассмотрим в первую очередь текстовые команды.

Обратите внимание, что некоторые команды, показанные здесь, могут использоваться только администратором или с помощью `sudo`. Другие команды могут выполняться и обычными пользователями, однако они, как правило, не предоставляются по умолчанию (то есть в переменной окружения `PATH` отсутствует каталог, в котором находятся такие команды), поэтому необходимо точно указывать путь к этим командам. Обычно это `/sbin` или `/usr/sbin`.

Если раньше вы не очень много работали с сетевыми функциями, то вам может недоставать базовых знаний, а некоторые термины могут быть непонятны. Такая недостающая информация будет сообщена в главах 16–20. В них мы поговорим о конфигурации сетевых функций и о настройке сетевых серверов.

При работе в сети часто бывает нужно обратиться к файлам, которые находятся на другом компьютере. В UNIX и Linux обычно применяется путь NFS (сетевая файловая система). Как интегрировать каталог NFS с локальной файловой системой, рассказано в разделе 13.12. Доступ к сетевым каталогам Windows — независимо от того, выходите вы в локальную сеть с помощью программы Samba, написанной для Linux, или через систему Windows, — обычно осуществляется прямо через файловый менеджер. Кроме того, можно интегрировать каталоги Windows прямо в локальную файловую систему. Некоторые другие клиентские программы Samba будут описаны при обсуждении конфигурации сервера Samba (раздел 20.8).

6.1. Определение состояния сети

В этом разделе дается обзор команд, применяемых для тестирования основных функций сети. Более подробная информация по упоминаемым здесь командам

приводится в разделе 16.4, где мы рассмотрим, как конфигурировать доступ к сети вручную. Небольшой сетевой глоссарий, в котором объясняются, в частности, такие термины, как шлюз (Gateway), сервер имен (Nameserver), маскарадинг (Masquerading) и т. д., находится в разделе 16.3.

Определение сетевых интерфейсов

Команда `ifconfig` возвращает список всех известных сетевых интерфейсов. Обычно используются интерфейсы `eth`*n* (Ethernet) и `ppp`*n* (доступ к Интернету через модем, ADSL или VPN). Особую роль играет `lo`: этот интерфейс позволяет программам локальных компьютеров обмениваться информацией через сетевой протокол. Он работает даже тогда, когда соединение от компьютера к сети «наружу». Обычно результат выглядит примерно так:

```
root#   ifconfig
eth0    Link encap:Ethernet Hardware Address 00:11:25:32:4F:5D
        inet Address:192.168.0.15 Bcast:192.168.0.255 Mask:255.255.255.0
        inet6 Address: fe80::211:25ff:fe32:4f5d/64 Scope:Connection
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:11416 errors:0 dropped:0 overruns:0 frame:0
        TX packets:10415 errors:0 dropped:0 overruns:0 carrier:0
        Collisions:0 transmit queue:1000
        RX bytes:9600456 (9.1 MiB) TX bytes:2269502 (2.1 MiB)
        Base address:0x8000 memory:c0220000-c0240000
lo      Link encap:local loop
        inet Address:127.0.0.1 Mask:255.0.0.0
        inet6 Address: ::1/128 Scope:Machine
        ...
        RX bytes:6139586 (5.8 MiB) TX bytes:6139586 (5.8 MiB)
```

Если интерфейс отсутствует, это значит, что сетевая карта еще не активизирована. Вам поможет предусмотренный в дистрибутиве инструмент, предназначенный для конфигурирования сети. Вы также можете активизировать сетевой интерфейс вручную, например так: `ifconfig eth0 192.168.0.2`. Если при этом возникает ошибка `eth0: unknown interface: No such device`, это означает, что в ядре отсутствует модуль для управления сетевой картой (см. также раздел 16.4).

Тестирование доступности localhost

Программа `ping` ежесекундно посылает маленький сетевой пакет по указанному адресу. Если по этому адресу находится компьютер, он отсылает ответ (если брандмауэр не препятствует этому). Программа работает до тех пор, пока вы не завершите ее нажатием сочетания клавиш Ctrl+C. Локальный компьютер localhost проверяет наличие виртуального интерфейса для работы сетевых протоколов, который обеспечивает работу всех остальных функций компьютера.

```
user$ ping localhost
PING localhost (127.0.0.1): 56 data bytes
```

```
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=0.152 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=0.114 ms
...
```

Тестирование доступности локальной сети

Передав в программу `ping` не localhost, а IP-адрес другого компьютера, находящегося в локальной сети, вы сможете проверить, как работает сеть. Команда `-c 2` гарантирует, что `ping` не будет работать бесконечно, а отошлет всего два пакета.

```
user$ ping -c 2 192.168.0.99
PING 192.168.0.99 (192.168.0.99): 56 data bytes
64 bytes from 192.168.0.99: icmp_seq=0 ttl=255 time=0.274 ms
64 bytes from 192.168.0.99: icmp_seq=1 ttl=255 time=0.150 ms
...
```

Если в локальной сети находится сервер имен, присваивающий имя IP-адресу 192.168.0.99 (или если эту задачу выполняет файл `/etc/hosts`), то можно указать `ping` не IP-адрес, а имя компьютера.

```
user$ ping -c 2 mars
PING mars.sol (192.168.0.10) 56(84) bytes of data.
64 bytes from mars.sol (192.168.0.10): icmp_seq=1 ttl=64 time=0.281 ms
64 bytes from mars.sol (192.168.0.10): icmp_seq=2 ttl=64 time=0.287 ms
--- mars.sol ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.281/0.284/0.287/0.003 ms
```

Тестирование доступа к Интернету

Потом можно проверить, как работает соединение с Интернетом. Следующая команда тестирует сразу два пункта сетевой конфигурации: доступность сервера имен и работу шлюзов.

```
user$ ping -c 2 www.yahoo.com
PING www.yahoo-ht2.akadns.net (209.73.186.238) 56(84) bytes of data.
64 bytes from f1.www.vip.re3.yahoo.com (209.73.186.238): icmp_seq=1 time=122 ms
64 bytes from f1.www.vip.re3.yahoo.com (209.73.186.238): icmp_seq=2 time=123 ms
--- www.yahoo-ht2.akadns.net ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 122.731/123.256/123.782/0.631 ms
```

Если это не работает, причин может быть несколько.

- Возможно, сервер Yahoo! сейчас недоступен либо на сервере из соображений безопасности была деактивизирована функция ответа на `ping`. Попробуйте другой интернет-адрес.
- Если сервер имен не работает, вы получите сообщение об ошибке `ping: unknown host yahoo.com`. Проверьте, содержит ли файл `/etc/resolv.conf` адрес сервера имен.

- Если не работает шлюз, вы получите следующее сообщение об ошибке: `connect: Network is unreachable`. Выполните команду `route -n`. Последняя строка должна выглядеть примерно как в предыдущем примере, причем в столбце `Gateway` должен содержаться IP-адрес вашего шлюза:

```
root# route -n
Kernel IP routing table
Destination  Gateway       Genmask       Flags  Metric  Ref  Use  Iface
192.168.0.0  0.0.0.0       255.255.255.0 U      0       0    0    eth0
127.0.0.0    0.0.0.0       255.0.0.0     U      0       0    0    lo
0.0.0.0      192.168.0.10  0.0.0.0       UG     0       0    0    eth0
```

- Если функции шлюза выполняет один из компьютеров локальной сети, возможно, вы забыли о функции маскарадинга. В этом случае доступ к Интернету будет закрыт для всей сети. Более подробное руководство по конфигурации шлюза для доступа к Интернету дается в главе 17.

Отслеживание пути IP-пакетов

С помощью команды `traceroute` можно увидеть, какой путь проделал пакет с вашего компьютера на другой и сколько времени ушло на преодоление каждого отдельно взятого отрезка. По умолчанию команда предпринимает три попытки и возвращает три значения времени соответственно. Команда не работает, если на пути пакетов стоит брандмауэр, блокирующий порт UDP 33434, используемый `traceroute`. В таком случае команда возвращает для этого и всех последующих этапов только три звездочки.

Следующие строки показывают путь от моего рабочего компьютера к сайту google.at. Первая строка описывает интернет-шлюз (компьютер mars.sol), строка 2 — маршрутизатор ADSL, а строка 3 — шлюз интернет-провайдера. Начиная со строки 9, результат начинает дробиться между различными резервными компьютерами Google.

```
user$ traceroute google.at
traceroute to google.at (66.102.9.104), 30 hops max, 40 byte packets
 1 mars.sol.0.168.192.in-addr.arpa (192.168.0.1) 0.277 ms 0.194 ms 0.216 ms
 2 192.168.1.1 (192.168.1.1) 0.373 ms 0.367 ms 0.690 ms
 3 N704P030.adsl.highway.telekom.at (62.47.31.254) 8.598 ms 8.500 ms 11.593 ms
 4 172.19.90.193 (172.19.90.193) 11.864 ms 8.837 ms 7.986 ms
...
14 66.102.9.104 (66.102.9.104) 52.741 ms 53.306 ms 51.996 ms
```

Программа gnome-nettool

Если вы работаете с Gnome, то можете с удобством узнать большую часть вышеперечисленной информации с помощью программы `gnome-nettool` (рис. 6.1).

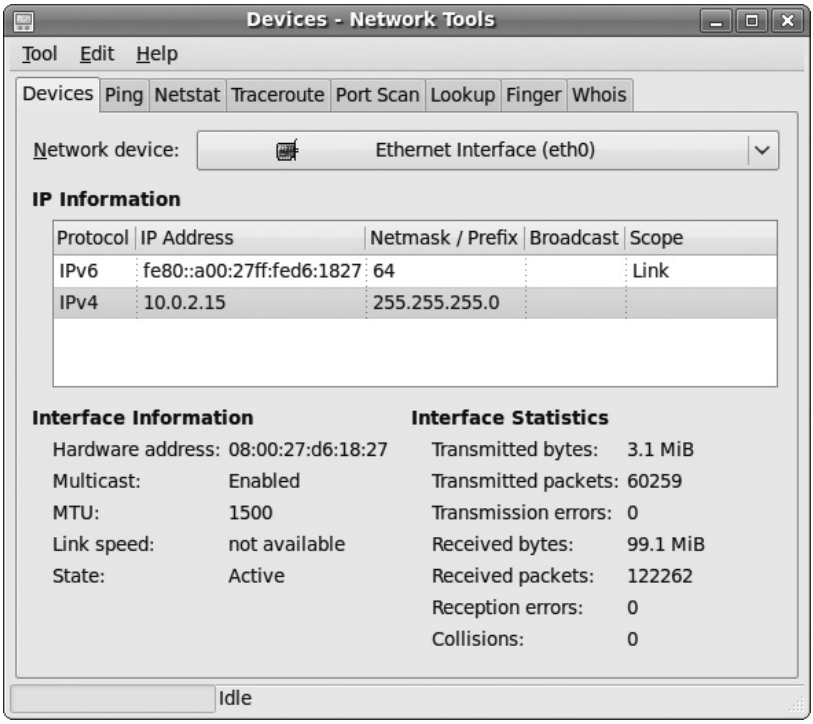

Рис. 6.1. Диагностика сети в Gnome

6.2. Работа на других компьютерах (SSH)

Программы `telnet`, `rlogin` и `ssh` позволяют работать на другом компьютере, словно он стоит прямо перед вами. Данный принцип работает и в командно-ориентированных программах, и в X-программах. В этом разделе я ограничусь описанием `ssh` *(безопасного командного процессора)*. Устаревающими программами `telnet` и `rlogin` я не рекомендую пользоваться по соображениям безопасности, так как в них учетные данные, в том числе пароль, передаются в незашифрованном виде.

Для использования `ssh` прежде всего необходим второй компьютер, на котором будет работать сервер SSH, то есть программа `sshd`. В некоторых дистрибутивах Linux она установлена по умолчанию, в других ее требуется установить (как минимум пакет `openssh-server`). Если на всех компьютерах включены брандмауэры, они не должны блокировать порт 22.

Информация об установке, конфигурировании и обеспечении безопасности SSH-сервера дается в разделе 19.1. Более подробные данные о клиентском применении данной техники сообщаются в `man ssh` и `man ssh_config`. Есть и отдельный сайт, посвященный `ssh`: http://www.openssh.org/.

Обычное shell-соединение

Если вы работаете на компьютере uranus и хотите запустить shell-соединение на компьютере mars, то выполните следующую команду, чтобы установить соединение:

```
user@uranus$ ssh mars
user@mars's password: xxx
```

Если вы впервые устанавливаете соединение с другим компьютером, то часто выводится предупреждение следующего вида:

```
The authenticity of host 'mars (192.168.0.10)' can't be established.
RSA1 key fingerprint is 1e:0e:15:ad:6f:64:88:60:ec:21:f1:4b:b7:68:f4:32.
Are you sure you want to continue connecting (yes/no)? {\bfs yes}
Warning: Permanently added 'mars,192.168.0.10' (RSA1) to the list
         of known hosts.
```

Это означает, что программа `ssh` не уверена, можно ли доверять компьютеру mars с IP-адресом 192.168.0.10. Возможно, чужой компьютер выдает себя за mars. Если ответить на запрос `YES`, то `ssh` сохранит имя, адрес и отличительную метку RSA (код для однозначной идентификации партнерского компьютера) в `~/.ssh/known_hosts`.

Если вы хотите работать на компьютере mars под другим логином, нежели на uranus (например, в качестве администратора), укажите имя с параметром `-l`:

```
user@uranus$ ssh -l root mars
root@mars's password: xxx
```

Выполнение команд

Можно не использовать `ssh` в интерактивном режиме, а просто выполнить на удаленном компьютере какую-либо команду. Команда и ее параметры в таком случае будут переданы `ssh` в качестве дополнительных параметров. После этого `ssh` завершит работу.

```
user@uranus$ ssh mars команда параметры
user@mars's password: xxx
```

Эта на первый взгляд тривиальная функция открывает вам широкие возможности: теперь можно, к примеру, запустить на удаленном компьютере `tar`, чтобы передать созданный архив стандартному выводу (ставим дефис после параметра `-f`), а затем с помощью | использовать стандартный вывод в качестве ввода для второй команды `tar`, работающей на локальном компьютере. Таким образом, можно безопасно копировать через SSH целые каталоги.

Следующая команда показывает, как я скопировал целое дерево каталогов `htdocs` с веб-сервера `kofler.cc` в локальный каталог `~/bak`:

```
user$ ssh -l username kofler.cc tar -cf - htdocs | tar -xC ~/bak/ -f -
username@kofler.cc's password: ******
```

SSH и X

В принципе соединение SSH позволяет выполнять и программу X. В таком случае программа работает на удаленном компьютере, но ее работа отображается на локальном компьютере, откуда можно вводить информацию с помощью мыши и клавиатуры. Поскольку теперь весь X-протокол работает через сеть, вам понадобится хорошее сетевое соединение, с которым будет удобно работать.

Для того чтобы выполнить X-программу, ее нужно запустить с параметром `-X` (если `/etc/ssh_config` содержит строку `ForwardX11 yes`, то параметром `-X` можно пренебречь). Программа `ssh` сама правильно настроит переменные `DISPLAY`.

Следующие команды запускают на компьютере mars редактор XEmacs. Однако окно редактора открывается на экране компьютера uranus, откуда с редактором теперь можно работать! Этот механизм функционирует даже тогда, когда на компьютере mars не работает X-сервер (уровень запуска 3). Однако должны быть установлены все X-библиотеки!

```
user@uranus$ ssh -X mars
user@mars's password: xxx
user@mars$ xemacs &
```

Безопасное копирование файлов с помощью scp

Для копирования файлов через SSH по сети применяется команда `scp`. Ее синтаксис таков:

```
user$ scp [[user1@]host1:]имя_файла1 [[user2@]host2:][имя_файла2]
user2@host2's password: ******
```

Таким образом, файл *имя_файла1* копируется с компьютера host1 на компьютер host2, где сохраняется в файле *имя_файла2*. Некоторые замечания по опциональным составляющим команды копирования:

- указывать host1 и host2 не нужно, когда имеется в виду локальный компьютер (то есть localhost);
- не нужно указывать user1, когда имеется в виду текущий пользователь;
- не нужно указывать user2, когда на компьютере host2 должно использоваться текущее имя пользователя с host1 или user1;
- *имя_файла1* также может быть каталогом; в таком случае необходимо указать параметр `-r`, чтобы можно было скопировать каталог и все его подкаталоги;
- *имя_файла2* не нужно указывать, когда имя файла должно остаться прежним; в таком случае файл будет скопирован в домашний каталог пользователя user2.

Вместо *имя_файла2* также можно указать целевой каталог, применив, как обычно, символ ~ для обозначения домашнего каталога user2.

В завершение приведу еще один пример: предположим, что пользователь gabi работает на компьютере uranus. Он хочет перенести файл `abc.txt` в каталог `~/efg` на компьютере mars. Команда `scp` будет выглядеть так:

```
gabi@uranus$ scp abc.txt mars:~/efg/
gabi@mars's password: ******
```

SFTP (Secure FTP) — это основанный на SSH безопасный вариант протокола FTP. В следующем разделе мы рассмотрим SFTP подробнее, коснувшись при этом вопросов передачи файлов через FTP и HTTP (раздел 6.3).

SSH-туннель

Опытные пользователи Linux могут применять SSH для создания туннелей. Конечно, машина или поезд по такому туннелю не пройдет, но зато по нему можно передавать любые IP-пакеты, направленные к определенному порту. Туннель SSH — это надежный путь, позволяющий переправлять IP-пакеты между компьютерами даже тогда, когда между ними стоит брандмауэр, блокирующий порт (вводная информация о мире IP-пакетов и рассказ о том, что такое порт, дается в разделе 18.4, посвященном брандмауэрам).

Если туннель создается с клиентского компьютера, применяется параметр `-L localport:remotehost:remoteport`. Например, следующая команда позволяет получить доступ к порту 3306 компьютера mars через порт 3307 локального компьютера. Эта же команда одновременно запускает SSH-соединение, но вы можете этому воспрепятствовать с помощью параметра `-N` (если вам нужен только туннель, а оболочка не нужна). Если требуется войти в систему на компьютере mars под другим именем, нужно указать имя для входа в систему как обычно: `-l name`.

```
user@uranus$ ssh -L 3307:mars:3306 mars
user@mars's password: ******
```

Туннель будет открыт до тех пор, пока вы не завершите SSH-соединение, нажав Ctrl+D. Если вы запустили `ssh` с параметром `-N`, выход из программы осуществляется нажатием Ctrl+C.

Как правило, MySQL используется через порт 3306. Теперь вы можете получить доступ с компьютера uranus к серверу MySQL, работающему на mars, через порт 3307 компьютера uranus. Вместе с командой `mysql` необходимо указать порт 3307 и имя хоста 127.0.0.1, чтобы SSH-туннель действительно использовался (по умолчанию MySQL устанавливает локальные соединения через сокет-файл).

```
user@uranus$ mysql -u mysqllogin -P 3307 -h 127.0.0.1 -p
Enter password: *******
```

Чтобы логин к MySQL работал, необходимо предварительно выполнить два условия. Во-первых, сервер MySQL должен принять IP-соединения, установленные на компьютере mars. (Из соображений безопасности MySQL-сервер может иметь такую конфигурацию, при которой соединения могут устанавливаться только через сокет-файлы. В таком случае туннель не поможет, так как им можно соединять только порты.) В качестве хост-имени применяется имя компьютера, к которому `ssh` построила туннель — в данном случае mars (или mars.sol, если использовать домен sol, как и в других примерах этой книги).

SSH-туннели могут применяться для решения гораздо более сложных и разнообразных задач. Например, туннель можно использовать для построения *виртуальной частной сети*. Подробная документация приводится на следующих сайтах:

- http://www.oreillynet.com/pub/a/wireless/2001/02/23/wep.html;
- http://www.tldp.org/HOWTO/VPN-HOWTO/.

Аутентификация с использованием ключей

Все примеры, которые приводятся в данном подразделе, предполагают, что после выполнения ssh, scp и т. д. вы войдете в систему с целевого компьютера так, как делаете это обычно. Однако это не единственный возможный способ. Для аутентификации можно также использовать файлы ключей. Для этого на клиенте с помощью программы ssh-keygen создается пара ключей. Данный ключ вы должны задать сами с помощью passphrase (это пароль, состоящий из нескольких слов). Кроме того, при аутентификации пароля скопируйте открытый ключ в файл key на сервере:

```
user@client$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa): <Return>
Enter passphrase (empty for no passphrase): ********
Enter same passphrase again: ********
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
user@client$ scp .ssh/id_rsa.pub user@server:key
user@server's password: *******
```

ПРИМЕЧАНИЕ

Если вы отвечаете на запрос пароля из нескольких слов, просто нажимая Enter или вводя слово empty, то команда ssh-keygen не производит шифрования. Это удобно, так как вы можете пользоваться SSH без запроса пароля. Однако такой подход представляет риск с точки зрения безопасности: если кто-то завладеет ключом от вашего клиентского компьютера, он сможет без труда входить в систему со всех компьютеров, на которых вы установили открытую часть ключа!

На этом сервере вам потребуется создать каталог /home/user/.ssh (если вы еще не сделали этого) и вставить открытый ключ вашего клиентского компьютера в конце файла .ssh/authorized_keys:

```
user@server$ mkdir -p .ssh
user@server$ cat key >> .ssh/authorized_keys
```

Теперь, создавая соединение с целевым компьютером, ssh обменивает данные ключей. В таком случае логин уже не нужен, при входе в систему вы указываете вместо него passphrase из закрытого файла ключа.

Программа ssh-agent

Применяя ключи, мы обеспечиваем в первую очередь безопасность, но не удобство работы. Одновременно надежным и достаточно удобным решением является программа ssh-agent. Она управляет всеми закрытыми ключами пользователя (обычно существует только один такой ключ). Программа запускается следующим образом:

```
user$ eval $(ssh-agent)
```

Таким образом, изменяются переменные окружения актуальной консоли. Программа ssh-agent продолжает работу в фоновом режиме, а ssh-add позволяет добавлять ваши закрытые ключи:

```
user$ ssh-add ~/.ssh/id_rsa
Enter passphrase for /home/user/.ssh/id_rsa: ******
```

Теперь ssh будет работать с файлами ключей, управляемыми ssh-agent. Это означает, что запросов о пароле для файла ключей вы больше получать не будете. Пароль нужно ввести только один раз. Серьезный недостаток ssh-agent заключается в том, что переменные окружения могут работать только с одной консолью.

Демон gnome-keyring

В Gnome паролями и ключами SSH обычно занимается демон gnome-keyring. При первом доступе к данным в зависимости от конфигурации требуется указать верный логин (авто-логин через GDM не подходит) или пароль уровня master. Для управления сохраненными ключами и паролями используется программа seahorse.

6.3. Передача файлов

Основы FTP

FTP означает File Transfer Protocol («Протокол передачи файлов»). Это достаточно старый способ передачи файлов по Сети. Своей популярностью FTP обязан методу Anonymous FTP (анонимный FTP-доступ): многие крупные серверы Интернета предоставляют своим пользователям доступ к так называемым FTP-архивам. Такой доступ не требует пароля (в отличие от обычного FTP).

Серьезный недостаток FTP заключается в том, что при входе в систему имя пользователя и пароль передаются в незашифрованном виде. Более надежная альтернатива — SFTP (Secure FTP), основанный на SSH. В качестве альтернативы для FTP также зачастую используется HTTP, то есть протокол для передачи сайтов.

В этой главе обсуждается только использование FTP, то есть проблема рассматривается с клиентской точки зрения. Чтобы FTP работал, на удаленной станции должен работать FTP-сервер. Его конфигурация описана в разделе 19.5.

Команда ftp

Предком всех FTP-клиентов является интерактивная текстовая команда ftp. Поскольку файлы обычно передаются из текущего каталога или в этот каталог, то перед запуском ftp нужно перейти в желаемый рабочий каталог командой cd. Чтобы начать FTP-сессию, вводится команда ftp *имя_пользователя*@*ftp-сервер* или просто ftp *ftp-сервер*. Если вы желаете работать с анонимным FTP, укажите в качестве имени пользователя слово anonymous.

Установив соединение и введя пароль, можно начинать работу: команды cd, pwd и ls, значения которых такие же, как в Linux, позволяют переходить между каталогами FTP-архива. Чтобы перенести файл из FTP-архива в текущий каталог вашего компьютера, выполните команду get *файл*. При этом имя файла не изменяется. И наоборот: с помощью put можно перенести файл из текущего каталога вашего компьютера в каталог FTP-архива. Разумеется, это работает лишь в том случае, когда у вас есть право на внесение изменений в этот каталог. При использовании анонимного FTP это обычно касается только каталогов с названием типа

/pub/incoming. FTP-соединение завершается командой quit или bye. Справка по важнейшим FTP-командам приводится в табл. 6.1.

Таблица 6.1. Команды для работы с FTP

Команда	Функция
?	Вывод списка всех команд FTP
!	Выполнение команд оболочки
ascii	Переход в текстовый режим
binary	Переход в двоичный режим
bye	Завершение работы FTP
cd dir	Переход в указанный каталог FTP
close	Завершение соединения с сервером FTP
get file	Передача файла из архива FTP в текущий каталог
help «команда»	Вывод краткой справочной информации по указанной команде
lcd dir	Смена текущего каталога на локальном компьютере
ls	Вывод списка всех файлов, находящихся на FTP-сервере
lls	Вывод списка всех файлов, находящихся на локальном компьютере
mget *.pattern	Передача всех файлов, отвечающих образцу, из FTP-архива в текущий каталог
open	Соединение со сторонним компьютером (если первая попытка не удалась)
prompt	Активизация/деактивизация автоматического запроса о подтверждении передачи любого файла с помощью mget
put «файл»	Передача файла из текущего каталога в FTP-архив (upload)
quit	Завершение работы FTP
reget «файл»	Продолжение передачи файла, который ранее был передан частично
user	Возможность ввести новый логин

ПРИМЕЧАНИЕ

Прежде чем передать файл, необходимо перейти в двоичный режим с помощью команды binary. При работе в текстовом режиме FTP интерпретирует файлы как текст и пытается преобразовать файл в формат конкретного компьютера. После такого преобразования работать с двоичными файлами уже нельзя (к счастью, конфигурация большинства FTP-серверов имеет установку binary по умолчанию).

В следующих строках показано, как загрузить код ядра Linux с FTP-сервера:

```
user$ cd ~/src
user$ ftp ftp.kernel.org
Connected to zeus-pub.kernel.org.
220 Welcome to ftp.kernel.org.
Name (ftp.kernel.org:kofler): anonymous
331 Please specify the password.
Password: name@mysite.de
Using binary mode to transfer files.
ftp> cd pub/linux/kernel/v2.6
250 Directory successfully changed.
ftp> ls
...
```

```
ftp> get linux-2.6.29.tar.bz2
local: linux-2.6.29.tar.bz2 remote: linux-2.6.29.tar.bz2
227 Entering Passive Mode (204,152,191,5,20,69)
150 Opening BINARY mode data connection for linux-2.6.29.tar.bz2 (56579370 bytes).
...
ftp> quit
```

Другие программы для работы с FTP

Работать с командой `ftp` достаточно неудобно. К счастью, альтернативных программ достаточно много.

- Браузеры, файловые менеджеры — все браузеры и файловые менеджеры, используемые в Linux, могут применяться и для закачки FTP. Некоторые программы даже обеспечивают удобную закачку (например, Nautilus и Konqueror).
- Графические FTP-клиенты — такие программы, как `gftp` (Gnome), специально оптимизированы под выполнение типичных FTP-задач. В них есть специальные функции, например управление закладками и паролями, обеспечивающие передачу нескольких файлов, синхронизацию каталогов и т. д.
- `ncftp` — хотя у этой альтернативной программы текстовый пользовательский интерфейс, работать с ней удобнее, чем с `ftp`.
- `sftp` — эта программа так же минималистична, как и `ftp`, но при этом она заметно надежнее. При ее использовании на удаленной станции обязательно должен работать SSH-сервер (а не FTP-сервер).
- `wget`, `curl`, `lftp`, `rsync`, `mirror` и `sitecopy` — эти команды полезны при автоматической передаче файлов или целых деревьев каталогов через FTP.

FTP-адрес с паролем

Если вы не хотите работать с протоколом FTP под именем `anonymous`, а желаете войти в систему под собственным именем и паролем, то в большинстве клиентов FTP будет использоваться следующий синтаксис:

`ftp://`*`имя_пользователя:пароль@имя_сервера`*

Пассивный режим. Некоторые FTP-клиенты могут работать неправильно, если между вашим компьютером и FTP-сервером находится брандмауэр или если вы работаете в локальной сети, которая соединена с Интернетом с использованием маскарадинга. В таких случаях почти всегда помогает перевод клиента в так называемый пассивный режим. К сожалению, единой команды для этого не существует, поэтому придется почитать документацию (большинство клиентов самостоятельно опознают такую ситуацию и автоматически переходят в пассивный режим).

SFTP

Команда `sftp` входит в состав пакета `openssh`. Внутри программа `sftp` использует принципиально иной протокол, нежели `ftp`, и может применяться в качестве `ssh`

только при условии, что на удаленной станции работает SSH-сервер. Анонимный FTP с `sftp` использовать нельзя. За исключением этого, работа с программой практически не отличается от работы с `ftp`. С помощью команды `sftp -b` *batch-файл* можно автоматизировать процесс SFTP-закачек.

Альтернативы для SFTP. Многие считают `sftp` слишком «спартанским» вариантом. В любом случае выбор SFTP-клиентов значительно уже, чем при работе с FTP. Кроме того, иногда требуется немного подождать, чтобы соединение установилось.

- `gftp` — программа предлагает многочисленные возможности конфигурирования SFTP (FTP ▸ Options ▸ SSH). Если возникнут сложности, проверьте, тот ли порт вы используете (для SSH — 22, а не 21, как для FTP). Часто может потребоваться установить флажок Use SSH2 SFTP Functions.
- В универсальном браузере Konqueror можно инициировать SFTP-соединение, введя в адресную строку `sftp://`*пользователь*`@`*имя_сервера*. После запроса пароля в Konqueror можно работать, как с обычными каталогами.

  Однако Konqueror поддерживает и SSH-протокол напрямую, а этот протокол работает также и в тех случаях, когда `sftp` в распоряжении нет. В таком случае адрес вводится в таком формате: `fish://`*пользователь*`@`*имя_сервера*.

WGET

Интерактивное применение команды не подходит для автоматизации загрузок, например, в виде одного сценария. Кроме того, `ftp` не может похвастаться гибкостью. Например, невозможно самостоятельно возобновить прерванную закачку. В данном случае будет полезна команда `wget`, специально разработанная для выполнения крупных загрузок или передачи целых деревьев каталогов. Эта команда в равной мере поддерживает протоколы FTP, HTTP и HTTPS.

Примеры. Программа `wget` скачивается на компьютер как самый обычный файл:

```
user$ wget ftp://myftpserver.de/имя.abc
```

Если закачка по каким-то причинам прерывается, ее можно без проблем возобновить с помощью параметра `-c`:

```
user$ wget -c ftp://myftpserver.de/имя.abc
```

На закачку очень больших файлов, например образов диска с дистрибутивами Linux, даже при хорошем соединении с Интернетом может уйти несколько часов, поэтому рекомендую ставить такие крупные загрузки на ночь. Следующая команда позволяет практически наверняка гарантировать, что наутро вы на самом деле найдете на компьютере файл, поставленный на закачку вечером. Благодаря параметру `-t 20` закачка будет 20 раз возобновляться, если вдруг прервется, то есть файл успешно закачается и после 20 обрывов связи. Благодаря `--retry-connrefused` даже при возникновении ошибки `connection refused` («в соединении отказано»)

будет предпринята новая попытка соединения. Это полезно в тех случаях, когда сервер загрузок наверняка ненадежен и определенно будет ненадолго недоступен время от времени.

```
user$ wget -t 20 --retry-connrefused http://mydownloadserver.de/имя.iso
```

Следующая команда скачивает все файлы, необходимые для того, чтобы можно было прочитать указанный сайт в неизменном виде, будучи отключенным от Интернета. Коротко о параметрах: `-p` скачивает все файлы таблиц стилей и изображения; `-k` преобразует ссылки, содержащиеся в скачанных файлах, так, чтобы они указывали на соответствующие файлы локального компьютера; `-E` добавляет к скачанным файлам сценариев (ASP, PHP) расширение HTML; `-H` обеспечивает переход по ссылкам на внешние сайты.

```
user$ wget -p -k -E -H http://mysite.com/site.html
```

Если вы хотите прочитать в режиме offline целый сайт, воспользуйтесь следующей рекурсивной командой для скачивания (параметр `-r`). С помощью параметра `-l 4` мы ограничиваем глубину рекурсии четырьмя уровнями.

```
user$ wget -r -l 4 -p -E -k http://mysite.com
```

Команда curl

Команда `curl` помогает при передаче файлов с сервера или на сервер FTP, HTTP или других типов. На соответствующей странице справки `man` приводится внушительный список протоколов, с которыми может работать `curl`. В этом разделе мы рассмотрим только закачки через FTP. При программировании сценариев особенно практичной является функция `curl`, помогающая обрабатывать данные стандартного ввода либо записывать информацию в стандартный вывод. Иначе говоря, вам не потребуется предварительно создавать файл `*.tar.gz`, а потом передавать его на FTP-сервер; обе операции можно будет выполнить одновременно, разделив в коде соответствующие записи вертикальной чертой.

Следующая команда передает указанный файл на FTP-сервер backupserver и сохраняет этот файл в каталоге dir:

```
user$ curl -T файл -u имя_пользователя:пароль ftp://backupserver/dir
```

Для обработки данных из стандартного канала ввода добавьте к `-T` в имени файла дефис. Следующая команда сохраняет результат, получаемый после выполнения команды `tar`, прямо в файл `name.tgz` на FTP-сервере:

```
user$ tar czf - dir/ | curl -T - -u usern:pw ftp://bserver/имя.tgz
```

Программа lftp

Программа `lftp` является хорошим интерактивным FTP-клиентом. Однако она хорошо подходит и для того, чтобы с удобством выполнять FTP-закачки и другие

команды в рамках одного сценария. При работе с lftp можно использовать параметр -c, вместе с которым вы передаете несколько FTP-команд, разделенных точками с запятой, либо параметр -f, после которого указывается файл, где построчно перечислены необходимые команды. При этом первой всегда будет идти команда вида user *имя_пользователя*,*пароль имя_сервера*, устанавливающая соединение с FTP-сервером. Следующая команда демонстрирует закачку файла:

```
root# lftp -c "open -u имя_пользователя,пароль backupserver; put www.tgz"
```

Если вы хотите дать новое имя файлу, находящемуся на FTP-сервере, дополнительно укажите параметр -o *новое_имя*. В ходе закачки lftp будет демонстрировать ход выполнения задачи.

Для переноса на резервный сервер целого каталога воспользуйтесь командой mirror -R. Как правило, команда mirror копирует каталоги с FTP-сервера на локальный компьютер. Параметр -R позволяет запустить обратный процесс. Вот пример:

```
root# lftp -c "open -u usern,passw bserver; mirror -R directory"
```

В отличие от других FTP-клиентов, lftp поддерживает команду du, с помощью которой можно определить, сколько места на диске уже занимают ваши резервные файлы. Это может быть важно, если дисковое пространство на сервере для резервного копирования строго ограничено. Следующая команда позволяет установить, сколько дискового пространства имеется в наличии, не прибегая к интерактивному вмешательству. Параметр -s указывает, что вас интересует только конечный результат. Параметр -m говорит о том, что в качестве единицы измерения будет применяться мегабайт.

```
user$ lftp -c "open -u имя_пользователя,пароль bserver; du -s -m"
2378  .
```

Если вы хотите использовать результат в дальнейших вычислениях, второй столбец (в котором находится точка, указывающая, что числовое значение относится к текущему каталогу) вам помешает. Добавьте к команде cut -f 1, чтобы извлечь первый столбец:

```
user$ lftp -c "open -u usern,passw bserver; du -s -m" | cut -f 1
2378
```

RSYNC

Мы уже познакомились с программой rsync в разделе 3.5. Она позволяет копировать или синхронизировать целые деревья каталогов и выгодно отличается от аналогов не только простой командой копирования cp, но и некоторыми другими достоинствами.

- rsync особенно хорошо оптимизирована для синхронизации каталогов по сети (возможно, с медленной скоростью соединения). Передаются только изменения, внесенные в информацию.
- Для обеспечения надежной передачи данных можно использовать rsync в комбинации с SSH.

- rsync располагает многочисленными параметрами управления, чьи возможности значительно превышают возможности cp. Например, можно удалить в целевом каталоге все файлы, которых больше нет в исходном каталоге.

Применение в сети

Для того чтобы rsync могла обеспечить синхронизацию между двумя компьютерами одной сети, программа должна быть установлена и на втором компьютере. Существует два варианта обмена данными между обеими копиями программы rsync (которые находятся на локальном и на удаленном компьютере).

- Экземпляры создают для обмена информацией специальную оболочку, обычно SSH (см. раздел 6.2). В таком случае данные передаются в зашифрованном виде. Чтобы этот принцип работал, на обоих компьютерах должна быть установлена SSH; кроме того, на удаленном компьютере необходимо конфигурировать SSH-сервер таким образом, чтобы можно было войти в систему через SSH.
- Программа rsync работает на удаленном компьютере как служебная программа (демон). Служебная программа rsync конфигурируется в файле /etc/rsyncd.conf (см. соответствующий раздел справки man) и запускается сценарием Init-V (как правило, /etc/init.d/rsyncd). Более подробно я не буду здесь описывать этот клиентско-серверный сценарий.

Примеры

Во всех примерах предполагается, что команды rsync обмениваются информацией через SSH. Поэтому, при вызове rsync необходимо использовать параметр -e ssh. Прежде чем экспериментировать с rsync, следует убедиться, что на удаленный компьютер можно войти с помощью обычного SSH-логина.

Если соединение в сети достаточно медленное, можно воспользоваться дополнительным параметром -z. Он обеспечивает архивирование данных, которыми обмениваются команды rsync. Правда, в таком случае увеличивается нагрузка на процессоры обоих компьютеров, а ожидаемое ускорение синхронизации достигается, к сожалению, не всегда.

При указании исходного и конечного каталога применяется только один из следующих способов записи: *хост-имя*:*каталог* или *имя_пользователя*@*хост-имя*:*каталог*, если не требуется использовать текущее имя пользователя.

Следующая команда позволяет синхронизировать каталог dir1 локального пользователя username на компьютере saturn.sol с каталогом dir2 на компьютере mars.sol. За ввод пароля отвечает SSH (кроме того, на компьютере mars.sol необходимо указать логин и пароль пользователя username).

```
username@saturn.sol$ rsync -e ssh -az dir1/ mars.sol:dir2/
username@mars.sol's password: ******
```

Программа rsync также может передавать файлы с удаленного компьютера на локальный. Следующая команда обеспечивает синхронизацию в обратном направлении:

```
username@saturn.sol$ rsync -e ssh -az mars.sol:dir2/ dir3/
username@mars.sol's password: ******
```

Если команда `rsync` должна вызываться автоматическим сценарием, предназначенным для резервного копирования, то интерактивный ввод пароля будет мешать. Можно решить проблему так: на локальном компьютере создается закрытый файл ключей, а на удаленном компьютере — подходящий к нему открытый ключ. Если при создании ключа не создавать пароля из нескольких слов (`Passphrase`), в систему можно будет войти через SSH без указания пароля. К сожалению, такой способ рискован (см. подраздел «Аутентификация с использованием ключей» раздела 6.2). Риск можно снизить, если создать такой файл ключей, который может использоваться только сценарием `rsync`. Этот метод описан на сайте http://www.jdmz.net/ssh/.

BitTorrent

BitTorrent — это протокол для эффективного скачивания крупных файлов, которые могут понадобиться сразу многим пользователям. Идея протокола проста: скачивание осуществляется не с центрального сервера, а со всех компьютеров, доступных в сети, на которых расположены фрагменты требуемого файла (имеются в виду так называемые *одноранговые сети*). Действует также обратный принцип: когда вы скачиваете большой файл через BitTorrent, в это время (а в идеальном случае — и после этого) данный файл могут скачивать и другие пользователи BitTorrent в этой сети.

При работе с Linux BitTorrent особенно интересен, так как образы дисков с некоторыми дистрибутивами лежат на торрентах. Когда новая версия выкладывается в Интернете, ее зачастую начинают одновременно скачивать тысячи пользователей, причем происходит это практически мгновенно после публикации. Это гораздо лучше, чем любой обычный FTP- или HTTP-сервер. Благодаря BitTorrent даже такие огромные файлы можно скачивать с приемлемой скоростью. Более подробная информация по основам и техникам работы с BitTorrent хорошо обобщена в следующей статье «Википедии»: http://de.wikipedia.org/wiki/BitTorrent.

Файлы .torrent

Загрузки BitTorrent стали известны благодаря torrent-файлам. Это относительно небольшие двоичные файлы, содержащие в числе прочего контрольные суммы для многочисленных фрагментов файлов. Таким образом, обеспечивается возможность скачивания файлов не в последовательном режиме (то есть не в исходном виде), а в случайном порядке (с последующей сборкой файла из фрагментов), причем параллельно с нескольких доступных в сети торрент-источников.

BitTorrent-клиенты

BitTorrent-клиенты — это программы, которые, во-первых, осуществляют закачку, а во-вторых, предлагают скачиваемые файлы другим торрент-клиентам. Из таких программ наиболее популярны BitTorrent, KTorrent (KDE), а также Transmission или Monsoon (оба Gnome), и все они имеют удобный интерфейс. Программа KTorrent для KDE показывает, какие части файла уже скачаны (рис. 6.2).

Рис. 6.2. Закачка с помощью KTorrent

Если вы хотите работать с закачками BitTorrent в интерактивном режиме на консоли или через сценарий, обратите внимание на варианты `bittorrent-curses` и `bittorent-console`, поставляемые одновременно в пакете `bittorrent`.

7 Vim

В этой главе мы поговорим о редакторе Vi и о его свободно распространяемом варианте Vim *(Vi improved)*. Научиться работать с этими редакторами относительно непросто. Учиться этому стоит лишь тогда, когда вы постоянно работаете с текстом, программным кодом, HTML-документами и т. д, то есть когда текстовый редактор — это ваш постоянный и непременный рабочий инструмент. И вот тогда Vi подарит вам практически бесконечное множество специальных функций.

Иными словами, если вас больше интересует всесторонняя конфигурируемость и программируемость, а удобный интерфейс — дело десятое, то эти редакторы для вас. Но если вы лишь время от времени изменяете какой-нибудь конфигурационный файл, а вообще работаете с OpenOffice, Firefox и другими прикладными программами, вам вполне подойдут маленькие текстовые редакторы вроде Kate (KDE) и Gedit (Gnome) для консоли (см. раздел 2.2).

И вот наступает момент истины: Vi или Emacs? Обе программы — это краеугольные камни в мире UNIX/Linux. Обе содержат многочисленные специальные функции, например автоматическое выделение синтаксиса для различных языков программирования и типов документов или для поиска и замен с использованием регулярных выражений. О том, какая из двух программ лучше, в Интернете ведутся долгие дискуссии. На этот вопрос, на самом деле, нельзя ответить объективно. Я написал все редакции этой книги в различных версиях редактора Emacs (кроме этой главы — она написана в Vi, такая уж тема), поэтому с Emacs я знаком гораздо лучше, чем с какими-либо версиями Vi.

Мне кажется, что Emacs построен более интуитивно и более легок для изучения. При работе с Vi некоторым пользователям ничего не стоит запутаться в различиях между работой в стандартном режиме и режиме вставки. Однако в пользу Vi и компании говорит то, что де-факто эта программа является стандартом в UNIX/Linux. Она потребляет значительно меньше ресурсов и имеется в самых маленьких системах-реаниматорах (где для Emacs места, как правило, не хватает). Истинные адепты UNIX/Linux, несомненно, должны овладеть основными функциями обоих редакторов (а в этой книге о многом рассказать не получится).

Изначально программа Vi была коммерческой и в Linux она не предоставлялась. Напротив, Vim — это свободно распространяемая программа, совместимая с Vi, а к тому же обогащенная многочисленными расширениями и изрядно оптимизированная. Эту программу можно запускать командой `vi` или `vim`.

Vim в графическом режиме. Как правило, Vim выполняется в текстовой консоли или в командном окне. Но если вам больше нравится настоящее меню с акку-

ратными полосами прокрутки, взгляните на `gvim` (рис. 7.1). Графический вариант Vim обычно нужно устанавливать отдельно, причем пакет обычно называется `vim-X11` или `vim-gnome`.

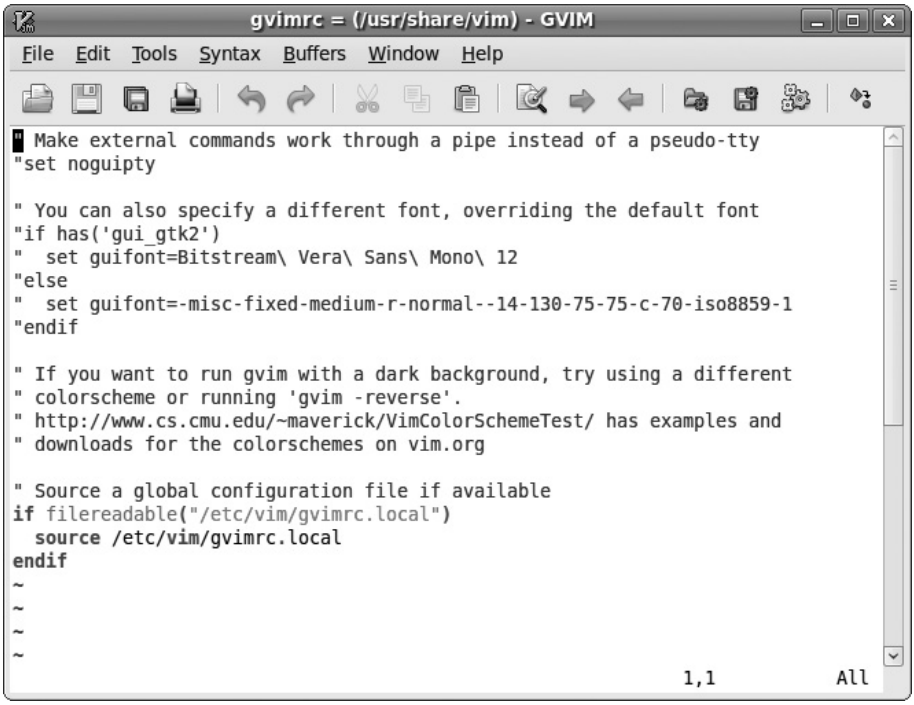

Рис. 7.1. Графический вариант редактора vim

Ссылки. Эта книга не безгранична, поэтому здесь я смогу сообщить лишь вводную информацию по Vim. Для начинающих очень полезно будет руководство, запускаемое командой `vimtutor` (при этом запускается Vim и загружается текст справки, в котором сообщается вводная информация и приводятся примеры). Далее приведены ссылки на сайты с более подробной информацией:

- http://www.vim.org/ — домашняя страница;
- http://www.eng.hawaii.edu/Tutor/vi.html — руководство;
- http://vimdoc.sourceforge.net/vimfaq.html — часто задаваемые вопросы;
- http://tnerual.eriogerg.free.fr/vim.html — обобщение важнейших сочетаний горячих клавиш;
- http://www.truth.sk/vim/vimbook-OPL.pdf — 500-страничная книга о Vim в формате PDF.

Vim — это благотворительная программа. Брэм Муленаар (Bram Moolenaar), главный разработчик Vim, подчеркивает, что это благотворительная программа: Vim можно приобрести бесплатно по лицензии, совместимой с GPL. Однако тех, кто пользуется Vim, регулярно просят вносить пожертвования, которые передаются

организации, оказывающей помощь детям Уганды. Более подробная информация сообщается, если нажать клавишу Esc, затем ввести `:help uganda` и нажать Enter.

7.1. Введение

Обычно Vim запускается командой `vim` *имя_файла* с текстовой консоли или из командного окна. Файл, в который требуется внести изменения, отображается прямо в консоли.

Режимы

Стандартный режим. Перед началом работы обязательно необходимо усвоить одно свойство программы: в ней существуют различные режимы работы. Стандартный режим предназначен не для ввода текста, а для выполнения команд. Если, например, нажать в стандартном режиме L, курсор переместится на один символ влево, нажатие D и W удалит слово, P вставит это слово там, где сейчас стоит курсор, и т. д.

Режим вставки. Чтобы вводить текст, необходимо перейти в режим вставки с помощью I (insert) или A (append). Теперь в первой строке слева Vim отобразит текст `--INSERT--`. В режиме вставки можно вводить текст, двигать курсор и удалять отдельные строки (Delete и Backspace). Разница между I и A заключается в том, что при использовании I ввод начинается с текущей позиции курсора, а при A — на символ перед курсором.

Прежде чем снова вводить команды, нужно перейти с помощью Esc обратно в обычный режим (он никак особо не обозначается, поэтому левая часть верхней строки остается пустой).

ПРИМЕЧАНИЕ

При переходе из режима вставки в обычный режим курсор перемещается на один символ влево (если только курсор не стоит в самом начале строки). Такое странное поведение, согласно разделу FAQ по Vim, введено намеренно, и отменить его нельзя. Чтобы выполнить только одну команду, не выходя из режима вставки и не меняя положения курсора, нажмите перед вводом команды сочетание Ctrl+O.

Удаление текста

В режиме вставки можно, как обычно, удалять символы с помощью Delete и Backspace. Если вы хотите удалить слова, строки или целые области, сначала перейдите с помощью Esc в обычный режим. При нажатии D, W удаляется целое слово, а при нажатии D, D – целая строка. Если поставить перед командой число, команда повторится такое количество раз. Так, например, нажатие 5, D, D удаляет пять строк. При введении точки (.) последняя выполненная команда повторяется.

Нажатие P (put) вставляет последний удаленный фрагмент текста перед текущей позицией курсора, а Shift+P — сразу после курсора. Нажатие U (undo) отменяет последние внесенные изменения, Ctrl+R (redo) восстанавливает внесенные

изменения (Vim 6 может отменить только последнее внесенное изменение, при повторном нажатии U изменение восстанавливается).

Сохранение и окончание работы

Чтобы сохранить измененный файл, перейдите с помощью Esc в обычный режим, а затем введите команду : w и нажмите Enter. Команда : q и нажатие Enter завершает работу редактора, если информация во всех открытых файлах была сохранена. С помощью команды :q! и нажатия клавиши Enter можно принудительно завершить работу программы, даже если в файлах есть несохраненные изменения. Команда : w и нажатие Enter комбинирует операцию сохранения и завершение программы.

В табл. 7.1 приводятся основные команды программы Vim.

Таблица 7.1. Основные команды

Клавиши	**Функции**
I	Активизирует режим вставки
A	Активизирует режим вставки, ввод текста начинается со следующего символа
Esc	Активизирует обычный режим либо прерывает ввод команды
Команды обычного режима	
D, W	Удаляет слово
D, D	Удаляет текущую строку
n D, D	Удаляет *n* строк
P	Вставляет последний удаленный фрагмент текста вслед за курсором
Shift+P	Вставляет последний удаленный фрагмент текста перед курсором
.	Повторяет последнюю команду
U	Отменяет последние изменения (в Vim 7 — до тысячи раз)
Shift+U	Отменяет все изменения, сделанные в текущей строке
Ctrl+R	Отменяет отмену изменений (начиная с Vim 7 — redo)
:w	Сохраняет файл
:q	Завершает работу Vim
:q!	Завершает работу Vim и в том случае, когда открыты несохраненные файлы
Команды в режиме вставки	
Ctrl+O *команда*	Выполняет команду, не покидая режима вставки

Справка

В Vim предоставляется обширная online-справка на английском языке. Чтобы запустить справку в любом режиме, нажмите F1. Можно также ввести в обычном режиме : help или : help *тема* и прочитать справочную информацию. Если вам нужно узнать, в каких справочных темах содержится слово abc, введите команду : help abc и нажмите Ctrl+D.

Окно справки. Текст справки выводится в специальной подобласти vim (она называется «окно справки», хотя и не является настоящим окном в том смысле, как этот термин понимается в графической системе Linux). Это окно закрывается командой `:q`. Однако вы можете оставить окно справки открытым и работать с первоначальным вариантом текста. Для этого перейдите в активное окно нажатием Ctrl+W+W. Более подробная информация по работе с окнами и буферами vim, а также по обработке нескольких файлов дается в разделе 7.5.

Навигация по справке. В тексте справки ссылки на другие справочные темы выделены цветом (в той версии, с которой работал я, ссылки светло-голубые). Чтобы перейти к соответствующей теме, наведите курсор на интересующее вас слово и нажмите Ctrl+]. Еще проще работать, когда активизирована мышь (см. раздел 7.7). В таком случае нужно дважды щелкнуть кнопкой мыши на теме, чтобы перейти к ней. Нажатие Ctrl+T вернет вас на исходную страницу.

7.2. Перемещение курсора

Как правило, клавиши для перемещения курсора работают и в обычном режиме, и в режиме вставки. Кроме того, для изменения положения курсора можно использовать и различные сочетания клавиш, если вы работаете в обычном режиме. Важнейшие сочетания обобщены в табл. 7.2. Поклонники Vi двигаются по тексту с помощью этих клавиш значительно быстрее, чем с помощью клавиш для перемещения курсора.

Таблица 7.2. Сочетания клавиш для перемещения курсора при работе в обычном режиме

Сочетание клавиш	Функция
Клавиши для перемещения курсора	Клавиши для перемещения курсора сохраняют обычное значение
H / L	Перемещает курсор влево-вправо
J / K	Перемещает курсор вверх-вниз
Shift+H / Shift+L	Перемещает курсор в начало или в конец текущей строки
Shift+M	Перемещает курсор на середину текущей строки
B / W	Перемещает курсор на одно слово влево или вправо
E	Перемещает курсор в конец слова
G, E	Перемещает курсор в начало слова
[,]	Перемещает курсор в начало текущего/следующего предложения
{ , }	Перемещает курсор в начало текущего/следующего абзаца
^ , $	Перемещает курсор в начало или в конец строки
Shift + G	Перемещает курсор в конец файла
G, G	Перемещает курсор в начало файла
n Shift + G	Перемещает курсор в строку *n*
n |	Перемещает курсор в столбец *n*
%	Перемещает курсор к соответствующей закрывающей скобке: (), [], {}

Одно из характерных свойств Vim заключается в том, что ← в начале строки не ставит курсор в конец предыдущей строки. Точно так же → в конце строки работает не так, как обычно. Чтобы заставить программу работать так же, как обычный текстовый редактор, выполните в обычном режиме `: set whichwrap= b,s,<,>,[,]` или выполните эту команду `set` в `~/.vimrc`.

Сохранение информации о положении курсора. Команда `M` *буква* сохраняет текущее положение курсора, и в следующий раз курсор начнет двигаться с этого места. Если еще раз нажать ' и ' то курсор вернется на последнюю позицию. Комбинации ' [или '] перемещают курсор соответственно в начало или в конец того раздела текста, в который вносились последние изменения.

Где я? В режиме вставки Vim показывает справа от статусной строки номер текущей строки и столбца, а также процентное значение, указывающее, в каком отрезке текста вы находитесь (например, если написано 92 %, значит, осталось менее 10 %). Если же вы потеряетесь, нажмите Ctrl+G. Vim отобразит в статусной строке имя файла, его состояние (например, изменен), общую длину строк и относительное положение в тексте (в процентах).

7.3. Обработка текста

Для многократной вставки текстовых символов укажите в обычном режиме число, нажмите A (append), введите желаемый символ и, наконец, нажмите Esc. Итак, чтобы 50 раз ввести символ =, введите `5 0 A =` и нажмите Esc. После выполнения этой команды вы снова окажетесь в обычном режиме.

Опечатки. Кроме того, Vim помогает исправлять типичные опечатки: символ ~ изменяет регистр конкретной буквы; нажатие X, P меняет местами следующие две буквы.

Удаление текста

В табл. 7.3 дается список важнейших команд, предназначенных для удаления текста. Если указать перед командой удаления число, она будет выполнена соответствующее количество раз. Как и в других случаях при работе с Vim, здесь действует правило: . повторяет последнюю команду, *n* . повторяет ее *n* раз.

Таблица 7.3. Команды для удаления текста

Сочетание клавиш	Функция
Команды в режиме вставки	
Delete, Backspace	Эти клавиши имеют обычное значение
Команды в обычном режиме	
X	Удаляет символ под курсором или выделенный текст
Shift+X	Удаляет символ перед курсором
D, D	Удаляет текущую строку
D *курсоркоманда*	Удаляет текст в соответствии с указанной командой перемещения курсора (см. табл. 7.2). Примеры: D, $ удаляет текст до конца строки. D, B удаляет предыдущее слово. D, W удаляет следующее слово

Обычно текст удаляется в буфер обмена. Последний удаленный фрагмент текста можно извлечь оттуда и вставить на месте курсора нажатием сочетания клавиш Shift+P либо за курсором — нажатием клавиши P.

Несколько своеобразный способ удаления текста и замены его новым текстом связан с клавишей C (change): например, C, W удаляет текущее слово и включает режим вставки. Теперь вы вводите новое слово и завершаете ввод нажатием Esc. Клавиша C аналогично работает и с другими командами курсоров.

Копирование текста

Можно также помещать текст в буфер обмена, не удаляя этот текст. В табл. 7.4 обобщены соответствующие команды (все они работают в обычном режиме).

Таблица 7.4. Копирование текста в буфер обмена

Сочетание клавиш	Функция
Y	Копирует выделенный текст в буфер обмена
Y, Y	Копирует текущую строку в буфер обмена
Y *курсоркоманда*	Копирует текст, захваченный движением курсора; пример: Y } копирует текст до конца абзаца

Выделение текста

Для выполнения некоторых команд (удаления) удаляемый фрагмент текста нужно выделить. В Vim для этого предусмотрено три различных режима выделения, которые активизируются и деактивизируются нажатием V, Shift+V или Ctrl+V в точке, с которой начинается выделение. Когда один из таких режимов активизирован, в самой нижней строке Vim виден текст `--VISUAL--`. Теперь перемещаем курсор к той точке, где выделенная область заканчивается, и увеличиваем область выделения специальными командами (табл. 7.5). Пока режим выделения активен, вам на выбор предлагаются различные команды для работы с выделенным текстом (табл. 7.6).

Таблица 7.5. Выделение текста

Сочетание клавиш	Функция
V	(Де)активизирует режим выделения символов
Shift+V	(Де)активизирует режим выделения строк
Ctrl+V	(Де)активизирует режим выделения блоков текста
A, W	Увеличивает область выделения на одно слово
A, S	Увеличивает область выделения на одно предложение
A, P	Увеличивает область выделения на один абзац
A, B	Увеличивает область выделения на один ()-уровень
A, Shift+B	Увеличивает область выделения на один {}-уровень
G, V	Еще раз выделяет последний выделенный фрагмент текста
O	Меняет положение курсора между началом и концом выделенного фрагмента

Таблица 7.6. Работа с выделенным текстом

Сочетание клавиш	Функция
X	Удаляет выделенный текст
Y	Копирует выделенный текст в буфер обмена
~	Изменяет регистр текста
J	Объединяет выделенные строки в одну длинную строку
G, Q	Делает разрыв строки (для стандартного текста)
>, <	Двигает текст на одну позицию табулятора вперед или назад
=	Заново выравнивает текст в соответствии с текущим режимом indent
!sort	Сортирует строки с помощью внешней команды sort

Строчный отступ

Правильный отступ строк особенно важен при редактировании текста. Vim всячески помогает вам в этом отношении. Простейшие команды — > > или < <. Они перемещают начало текущей строки на одну позицию табуляции вперед или назад. Если предварительно выделить несколько строк текста, то можно применять команды к целому блоку текста. При этом для обычного ввода используются команды > или <. Команда `: set shiftwidth=`*n* изменяет глубину отступа (обычно на восемь символов).

Кроме того, Vim может пытаться автоматически задавать отступ строки при вводе. Для этого необходимо активизировать режим отступа, например с помощью `:set cindent`. Далее кратко обобщим режимы отступа, используемые в Vim.

- `autoindent` — делает у следующей строки такой же отступ, как и у предыдущей.
- `smartindent` — работает как и `autoindent`, однако дополнительно учитывает уровни фигурных скобок. Чтобы Vim верно распознавал закрывающие скобки, их нужно ставить в начале новой строки. Степень отступа на каждом уровне фигурных скобок определяется параметром `shiftwidth`. Отступ в ранее выделенном тексте можно изменить командой =.
- `cindent` — работает как и `smartindent`, но учитывает и различные фрагменты кода из языков C и C++. Механизм отступа можно настраивать в соответствии с собственными предпочтениями, используя разнообразные параметры (см. `: help C-indenting`).

Обычный текст

Задаваемая по умолчанию конфигурация Vim такова, чтобы было максимально удобно писать код или изменять конфигурационные файлы. Именно поэтому в Vim не предусмотрены автоматические разрывы строк (то есть вы сами должны начинать новую строку нажатием клавиши Enter). Однако, разумеется, Vim можно использовать и для написания обычного текста (например, для электронных писем). В табл. 7.7 обобщены некоторые специальные команды и полезные параметры.

Таблица 7.7. Работа с обычным текстом

Сочетание клавиш	Функция
Shift+J	Соединяет текущую строку со следующей строкой. *n* Shift+J объединяет n строк в одну длинную строку
G, Q, A, P	Заново разбивает текущий абзац и ставит курсор в начале следующего абзаца
G, W, A, P	Как и в предыдущем случае, но курсор остается на прежнем месте
: set textwidth=*n*	Автоматический разрыв строки после максимума символов (как правило, 0 = деактивизирован)

Нажатие клавиши G приведет к тому, что будет автоматически учтен режим `autoindent`, а также настройки `textwidth`. Если `textwidth` содержит `0`, максимальная длина строки составляет 79 символов. Особенно много удобных возможностей для конфигурации и ввода обычного текста содержит параметр `formatoptions` (см. соответствующий раздел справки).

Дополнение слов

При записи длинных слов, названий функций и переменных вы постоянно рискуете допустить ошибку, а к тому же это еще и утомительно. Vim и здесь оказывает вам неоценимую помощь: вы просто указываете первые буквы, а потом нажимаете Ctrl+P. Если слово уже можно однозначно определить, то его недостающая часть автоматически будет дополнена. В ином случае можно выбирать с помощью Ctrl+P и клавиш управления курсором желаемое слово. При дополнении слов Vim учитывает все слова всех загруженных файлов, причем предпочтение отдается, во-первых, словам текущего файла, а во-вторых, словам, находящимся недалеко от курсора.

7.4. Поиск и замена

В обычном режиме команда / *искомый текст* и нажатие Enter перемещает курсор к искомому фрагменту текста. Нажатие сочетания Shift+N позволяет осуществить поиск в обратном направлении. Чтобы искать от самого начала, введите ? *искомый текст*. В табл. 7.8 обобщены важнейшие специальные символы, с помощью которых можно искать текст по заданному образцу.

Таблица 7.8. Специальные символы, используемые при поиске текста

Символ	Значение
.	Любой символ
^ $	Начало строки/конец строки
\> \>	Начало слова/конец слова
[a-e]	Символ между a и e
\s, \t	Пробел или знак табуляции
\ (\)	Объединяет шаблон поиска как группу
\=	Искомый текст должен встретиться не более одного раза
*	Искомый текст может встретиться любое количество раз (или ни разу)
\+	Искомый текст должен встретиться минимум один раз

Верхний и нижний регистр. В обычном режиме Vim различает поиск в верхнем и нижнем регистре. Если вы этого не хотите, поставьте перед поисковым шаблоном `/c` (действует только для данного конкретного поиска) или выполните команду `:set ignorecase` (будет действовать и при последующем поиске).

Инкрементный поиск. С помощью команды `:setincsearch` активизируется так называемый инкрементный поиск: уже при вводе искомого текста / *искомый текст* Vim перемещает курсор к первому подходящему месту. Нажатие клавиши Enter завершает поиск, а при нажатии Esc поиск прерывается. После окончания поиска все соответствия, найденные в тексте, остаются отмечены, пока вы не начнете новый поиск или не выполните команду `:nohlsearch`.

Поиск и замена. Чтобы заменить в тексте все фрагменты `abc` на `efg` без дополнительного запроса о подтверждении, выполните в обычном режиме команду `:%s/abc/efg/g`. Кроме того, команда `''` вернет вас к началу поиска. В табл. 7.9 обобщены некоторые варианты команд поиска и замены. При поиске и замене с запросом о подтверждении с помощью клавиши Y или N можно указать для каждого найденного фрагмента, следует ли заменить его новым фрагментом текста. Нажатие клавиши O прерывает процесс, клавиши A — заменяет все соответствия, найденные далее. При выражении замены можно указать на *n*-ю группу в найденных результатах, введя `\n`. Еще множество советов по поиску и замене, а также примеров приводится в online-справке (`:help substitute`).

Таблица 7.9. Команды для поиска и замены

Сочетание клавиш	Функция
: %s/abc/efg/g	Без запроса о подтверждении заменяет все экземпляры abc на efg
: %s/abc/efg/gc	С запросом о подтверждении заменяет все экземпляры abc на efg
: %s/abc/efg/gci	Производит замену без учета регистра

7.5. Одновременная обработка нескольких файлов

В обычном режиме команда `:e` *имя_файла* загружает новый файл. Новый файл заменяет тот, что обрабатывается в настоящий момент. Предварительно вам потребуется сохранить тот файл, с которым вы сейчас работаете, иначе Vim прекратит процесс. Правда, можно принудительно загрузить файл командой `:e!` *имя_файла*, но при этом изменения, внесенные в тот файл, что открыт сейчас, будут потеряны.

Буфер и окна

Разумеется, в Vim также можно одновременно обрабатывать несколько файлов. Правда, предварительно нужно понять следующую, не слишком логичную концепцию, описывающую способы отображения текстов в Vim и внутрисистемное управление текстами. Любой текст, отображаемый в окне, внутри системы помещается в так называемый буфер. Это касается как ваших файлов, так и текстов справки. Пока буфер только один, его обозначение присутствует на общем рабочем интерфейсе Vim. Для одновременного отображения нескольких буферов рабочий интер-

фейс разбивается на несколько «окон». Ту же ситуацию имеем при отображении текстов справки. В данном случае «окно» — это не элемент графической схемы Linux, а часть рабочего интерфейса.

Буферы изменяемых в данное время файлов всегда видны в окне. Буферы файлов, в которые с момента последнего сохранения не были внесены изменения, могут быть скрыты. При этом буферы остаются в оперативной памяти, но считаются неактивными. Будьте осторожны: если закрыть окно с еще не сохраненным файлом, все внесенные изменения будут потеряны! Буфер файла по-прежнему будет доступен, но файл, содержащийся в нем, будет соответствовать последнему сохраненному варианту.

Окна с вкладками

Начиная с версии 7, в окне Vim можно обрабатывать сразу несколько файлов, открывая их в отдельных вкладках (рис. 7.2). Вкладки — это наложенные друг на друга окна, открытые в строке выше окна, как это делается в Firefox и других браузерах. Окна с вкладками особенно удобны, если вы работаете с мышью (раздел 7.7): с ее помощью можно легко выбрать или закрыть активное окно (кнопка с крестиком справа вверху). Более подробная информация выводится по команде `:help tabpage`.

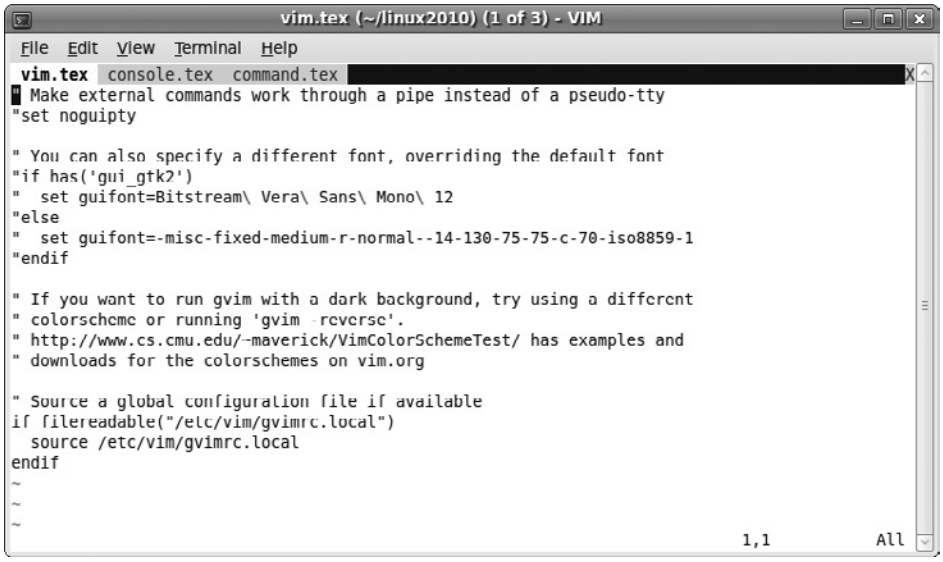

Рис. 7.2. Три файла в трех вкладках

Загрузка нового файла

В зависимости от того, с чем вы хотите работать — с окнами или вкладками, — новые файлы загружаются соответственно командами `:new` *имя_файла* или `:tabnew` *имя_файла*.

Разумеется, при запуске программы можно открывать сразу несколько файлов, например `vim файл1 файл2 файл3`. При этом по умолчанию загружается только один файл, а два оставшихся помещаются в невидимый буфер. Если вы хотите открыть все файлы в отдельных окнах или вкладках, дополнительно сообщите параметр `-o` или `-p`.

В табл. 7.10 обобщены важнейшие команды, предназначенные для загрузки и сохранения файлов, перехода между вкладками и т. д.

Таблица 7.10. Команды для работы с файлами, буфером и окнами

Сочетание клавиш	Функция
:e *имя_файла*	Загружает файл в текущий буфер
:w	Сохраняет текущий файл
:wall	Сохраняет все открытые файлы
:wq	Сохраняет и закрывает буфер
:q	Закрывает текущий буфер и завершает работу Vim, если больше не открыты никакие буферы
:q!	Закрывает буфер и в том случае, когда файл содержит несохраненные изменения
:qall	Закрывает все буферы и завершает работу Vim
:split	Делит окно надвое и показывает в обеих частях один и тот же текст
:new	Создает пустой буфер и показывает его в новом окне
:new *имя_файла*	Загружает файл в новый буфер
:only	Увеличивает текущее окно до максимального размера и закрывает остальные буферы
:all	Отображает все буферы, соответственно уменьшая их окна
:buffers	Возвращает список всех буферов
:buffer *n*	Отображает буфер *n* и опустошает текущий буфер
:buffer *имя_файла*	Отображает буфер с нужным файлом в текущем окне
:tabnew	Создает буфер и отображает его на вкладке
:tabnew *имя_файла*	Загружает файл и отображает его на вкладке
:tabnext	Переходит на следующую вкладку в окне
:tabprevious	Переходит на предыдущую вкладку в окне
Ctrl+Tab	Переходит на следующую/предыдущую вкладку
:tabclose	Закрывает данное окно с вкладками
:tabonly	Закрывает все остальные окна с вкладками

7.6. Что внутри?

Параметры

На предыдущих страницах мы видели, как конкретные функции в Vim активизируются или изменяются с помощью параметров. В этом разделе мы рассмотрим ряд других параметров. Обратите внимание, что некоторые из них действуют лишь локально, то есть для отдельно взятого файла или буфера. В таком случае команда `:set`

изменяет только локальную настройку. Для того чтобы глобально изменить параметр, используйте команду `:setglobal`. В табл. 7.11 обобщены важнейшие команды, предназначенные для работы с параметрами.

Таблица 7.11. Команды для работы с параметрами

Сочетание клавиш	Функция
:help *параметры*	Возвращает общую информацию, а также справку по параметрам
:help '*параметр*'	Возвращает информацию по заданному параметру
:set	Возвращает все параметры, которые находятся не в исходном виде
:set *булевый_параметр*	Активизирует булевый параметр
:set no*булевый_параметр*	Деактивизирует булевый параметр
:set inv*булевый_параметр*	Инвертирует актуальное состояние параметра
:set *параметр*?	Показывает настройку параметра
:set *параметр=значение*	Заново задает параметр
:set *параметр+=значение*	Изменяет параметр
:set *параметр-=значение*	Изменяет параметр
:set *параметр*&	Возвращает параметры в исходное состояние

Конфигурация

Все настройки параметров теряются после окончания работы Vim. Чтобы установленные параметры сохранялись, нужно изменить конфигурационный файл Vim `~/.vimrc`. Обратите внимание на комментарии: они вводятся символом "!

Далее приведен простой пример, показывающий, как может выглядеть файл `~/.vimrc`. Все параметры из этого примера были представлены в данной главе. Поищите в Интернете `~/.vimrc` — примеров будет предостаточно.

```
" Пример ~/.vimrc
" Закрепить положение курсора с помощью мыши
Set mouse=a
" <Курсор слева> если курсор находится в начале строки, передвигает его
" к концу предыдущей строки, <Курсор справа> если курсор находится в конце
" строки передвигает курсор к началу следующей строки
Set whichwrap=b,s,<,>,[,]
" Резервные копии (name~) создавать при сохранении
Set backup
" Активизировать инкрементный поиск
Set incsearch
" Везде заменить табуляцию пробелами
Set expandtab
```

Однако конфигурационные возможности Vim этим далеко не ограничиваются. Можно по-разному настраивать параметры для файлов различных типов, самостоятельно программировать новые функции и т. д. Однако не изобретайте велосипед! По адресу http://www.vim.org/scripts/ вы найдете множество готовых решений, работать с которыми совсем не сложно. Как правило, достаточно связать нужный файл с `~/.vimrc` с помощью команды `source` *имя_файла*.

Независимо от настроек резервного копирования при записи в файл Vim регулярно обновляет так называемый файл подкачки. Имя такого файла начинается с точки, далее идет имя файла, еще одна точка и расширение `.swp` (то есть, к примеру, `.mycode.c.swp`, если в настоящий момент вы работаете с mycode.c). В этом файле (в двоичном формате) содержатся все изменения, внесенные в документ со времени последнего сохранения. Он автоматически обновляется после ввода в документ более чем 200 символов или если в течение 4 секунд в файл не было внесено никакой информации. При обычном завершении работы Vim файл подкачки удаляется.

Если вдруг отключится электричество, можно восстановить незавершенную работу при следующем запуске Vim. При открытии документа Vim замечает, что в системе есть файл подкачки, и предлагает на выбор различные варианты. Как правило, нужно нажать клавишу W (восстановить) и сохранить спасенный таким образом файл. Затем нужно выйти из Vim и удалить файл подкачки самостоятельно (автоматически это не делается).

Кодировка

При запуске Vim определяет стандартную кодировку операционной системы (параметр `encoding`). При чтении нового файла Vim пытается распознать кодировку этого файла (параметр `fileencoding`). При этом программа по очереди пробует все кодировки, перечисленные в параметре `fileencodings`, пока не найдется кодировка, в которой текст будет отображаться без ошибок. Настройка по умолчанию для `fileencodings` — `utf-8, latin1`.

ПРИМЕЧАНИЕ

Если encoding и fileencoding не совпадают, то Vim при загрузке и сохранении автоматически выполняет преобразование. Если в encoding может быть представлено меньше символов, чем в fileencoding, то информация может быть потеряна. Во избежание этого старайтесь работать с Vim в окружении UTF-8. Такая настройка задана по умолчанию практически во всех дистрибутивах Linux.

Чтобы узнать, какая кодировка используется в файле, обрабатываемом в настоящий момент, выполните команду `:set fileencoding?`. С помощью команды `:set fileencoding=`*`новая_кодировка`* можно изменить кодировку. Если теперь сохранить файл, он будет сохранен в новой кодировке!

7.7. Советы и приемы: эффективный ввод команд

При вводе команд, начинающихся с `:`, используются некоторые подсказки для ввода: с помощью клавиш управления курсором можно листать последние использованные команды. Vim сохраняет команды в `~/.viminfo`, а также запоминает использованные команды после завершения программы. В дальнейшем вы сможете дополнять ключевые слова, с помощью клавиши табулятора (например, при вводе параметров). И, наконец, можно сокращать многие команды, начинающиеся с `:` (например, `:tabn` вместо `:tabnext`).

Отображение номеров строк. При вводе `:set number` сопровождает каждую строку номером. Команда `:set nonumber` вновь отменяет эту настройку.

Резервное копирование

По умолчанию Vim не создает резервной копии (копии исходного файла) при сохранении документа. Если хотите, можете выполнить в обычном режиме команду `:set backup`. Резервный файл получает имя вида `alternamе~`. Для того чтобы активизировать функцию резервного копирования для всех файлов, вставьте `set backup` в `~/.vimrc`.

Активизация мыши

Когда вы работаете с Vim в текстовой консоли или в командном окне, работа мыши в X ограничена ее основными функциями. С помощью мыши вы можете копировать текст и вставлять его на месте курсора. Однако мышью нельзя перемещать курсор.

Чтобы расширить функциональность мыши в Vim, можно либо использовать графический вариант gvim, либо выполнить в обычном режиме команду `:set mouse=a`. Теперь можно поместить курсор в новое место с помощью мыши, выбирать активное окно Vim, а также прокручивать текст колесиком мыши и т. д.

Такой режим работы с мышью в любом случае связан с определенным неудобством: теперь средняя кнопка мыши вставляет последний фрагмент текста, удаленный в Vim; мышь нельзя использовать для копирования текста из Vim в другие приложения. Но устранить эту проблему несложно: обычные функции мыши будут доступны, если дополнительно нажать клавишу Shift. Обратите внимание, что перед добавлением текста Vim должен работать в режиме вставки! В противном случае мышь интерпретирует введенный текст как команду и все пойдет не так.

Пробелы вместо табуляции

Для того чтобы Vim ставил в тексте не знаки табуляции, а обычные пробелы, выполните `:set expandtab` или запустите команду `~/.vimrc`. Чтобы заменить в имеющемся файле все знаки табуляции соответствующим количеством пробелов, дополнительно выполните `:retab`. Чтобы, наоборот, заменить пробелы знаками табуляции, выполните `:set unexpandtab` и потом `:retab`!

Макросы

Точка (`.`) повторяет последнюю команду — это вы уже знаете. Однако, если вы желаете многократно выполнить целую последовательность команд, можно написать макрос. Для этого перейдите из обычного режима в режим макросов, нажав клавишу O. Следующий символ задает имя макроса (если быть точным, то имя реестра, в котором сохраняется макрос). Все последующие команды сохраняются в макросе, пока вы не завершите ввод, вновь нажав O. Записанный таким образом

макрос выполняется командой `@имя_макроса`. Когда вы закрываете Vim, все сохраненные макросы теряются.

Пример: следующая последовательность клавиш записывает макрос, функция которого — ставить в начале и в конце слова кавычки.

```
0 A I " Esc F A " Esc 0
```

Теперь, если курсор расположен внутри слова и вы выполните `@A`, это слово будет заключено в кавычки. Команда `@@` повторяет последнюю команду реестра, избавляя вас от необходимости вспоминать название макроса или реестра.

Выполнение команд Linux

Если вы хотите выполнить в Vim команду Linux, не покидая самого Vim, выполните в обычном режиме `:! имя_команды` (например, `!ls`, чтобы получить список всех файлов, содержащихся в текущем каталоге). Vim отобразит результат выполнения команды. Нажав Enter, вы вновь вернетесь в редактор.

Для выполнения нескольких команд откройте с помощью `:sh` новую оболочку. Оттуда в редактор можно вернуться, нажав сочетание Ctrl+D.

Применение Vim в «простом режиме»

Если у вас не получается привыкнуть к различным режимам Vim, но и отказываться от этого редактора вы тоже не хотите, запустите программу с помощью `vim -y` или выполните `: setinsert-mode`. Таким образом, вы заставляете редактор работать только в режиме вставки. Теперь каждая команда будет вводиться нажатием сочетания Ctrl+O. Чтобы одновременно выполнять по несколько команд, можно также на долгое время остаться в обычном режиме, нажав для этого Ctrl+I. Покинуть обычный режим можно с помощью клавиши Esc.

По функциональности Vim еще более напоминает остальные редакторы, когда вы запускаете его с помощью программы `evim`: если она установлена, редактор будет запускаться с графическим пользовательским интерфейсом (`gvim`). Выделять текст можно с помощью Shift и клавиш управления курсором. Текст копируется с помощью Ctrl+C, удаляется с использованием Ctrl+X и вновь вставляется нажатием Ctrl+V.

В справке `man` простой режим называется «Vim для дураков» (а дураком быть как-то не хочется), и поэтому вывод следующий: переходя в простой режим, вы теряете так много основных функций Vim, что лучше сразу перейти к работе с другим редактором.

8 Bash (оболочка)

В этой главе речь пойдет об оболочке bash. Эта программа позволяет выполнять команды в командном окне или в текстовой консоли. Кроме того, оболочка является интерпретатором команд. В то же время bash располагает собственным языком программирования, применяемым для создания оболочковых программ (shell-сценариев). Работа с оболочкой будет интересна и для тех специалистов, кто хочет запускать команды и программы не только через меню KDE и Gnome, но и с клавиатуры.

На первый взгляд такой метод работы может показаться сложным, но с ним связаны некоторые дополнительные возможности, например комбинирование нескольких команд или сохранение результата выполнения команды в файле.

В этой главе мы поговорим о bash как об интерпретаторе команд и как об инструменте для программирования. Наиболее важными темами, рассматриваемыми в этой главе, являются введение в работу с bash, переадресация ввода и вывода, обмен информацией между несколькими процессами (программные каналы, подстановка команд) и управление переменными оболочки. Если вы интересуетесь bash-программированием, обратите внимание на подборку важнейших элементов этого языка и разнообразные примеры — эта информация приводится начиная с раздела 8.8. В конце главы приводится таблица, где перечислены специальные символы, используемые в bash.

8.1. Что такое оболочка?

Bash означает *bourne again shell*. Это англоязычная игра слов: bash построена на основе оболочки Борна, которая, наряду с оболочкой Корн и Си, считается одной из классических оболочек UNIX. В Linux можно работать и с двумя другими классическими оболочками, но по умолчанию обычно установлена оболочка bash.

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

В то же время оболочка является мощным языком программирования, с помощью которого можно автоматизировать рабочие процессы. Особые оболочковые команды позволяют использовать в рамках этой программы переменные, создавать

запросы и циклы и т. д. Получаемые в результате программы в зависимости от предпочтений автора, называются командными файлами, пакетными файлами, сценариями, процедурами оболочки и т. п. Независимо от названия, речь в данном случае идет о простых текстовых файлах, которые выполняются (интерпретируются) оболочкой. Более подробно об этом рассказано в разделе 8.8.

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

```
user$ echo $BASH_VERSION
4.0.16(1)-release
```

В справке `man` оболочке bash посвящен большой раздел. Кроме того, есть объемный справочный материал, который выводится по команде `info bash`. В некоторых дистрибутивах, кроме того, имеется файл `bashref.html`, в котором та же информация представлена в более удобном для чтения виде. Разумеется, этот файл есть в Интернете: http://www.gnu.org/software/bash/manual/bash.html.

Другие оболочки

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

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

Если вы не знаете, с какой оболочкой сейчас работаете, введите команду `echo $0`, которая есть во всех оболочках и везде имеет одинаковое значение. Эта команда выводит название оболочки.

```
user$ echo$0
-bash
```

Выбор другой оболочки, загружаемой по умолчанию

Для любого пользователя, вошедшего в Linux, система предусматривает стандартную оболочку. Она автоматически запускается в командном окне после входа в систему (логина). В X стандартная оболочка выполняется в каждом окне консоли.

Стандартная оболочка сохраняется в файле `/etc/passwd`. Название оболочки указывается в самом конце строки с учетной записью каждого пользователя. Чтобы

задать по умолчанию другую оболочку, выполните команду `chsh` (*change shell*). Программы оболочки сохраняются в каталоге `/bin`. Это значит, что требуется указать, например, `/bin/csh` в том случае, если вы собираетесь в дальнейшем работать с оболочкой C. Список доступных оболочек находится в `/etc/shells`.

/bin/sh

Большинство сценариев начинается с кода `#!/bin/sh`. Данная последовательность символов указывает, что сценарий должен выполняться оболочкой `/bin/sh` (раздел 8.8). Раньше в `/bin/sh` почти всегда ставилась ссылка на bash. Однако, поскольку bash (не в последнюю очередь благодаря множеству ее функций) считается относительно медленной и требует для работы много памяти, в некоторых дистрибутивах вместо нее используется оболочка, лучше подходящая для выполнения сценариев. В Ubuntu применяется, например, оболочка Debian-Almquist (dash). Она быстра, но не полностью совместима с bash. Если при программировании вы используете bash-специфичные функции, нужно указать в первой строке `#!/bin/bash`.

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

8.2. Базовая конфигурация

Функциональные клавиши в bash

Конфигурация клавиатуры в bash глобально настраивается в файле `/etc/inputrc` с помощью `~/.inputrc`. Если у вас не получается использовать специальные символы из национальных алфавитов либо если клавиши Delete, Pos1 и End работают неправильно, нужно настроить `inputrc` так, как это показано далее. Все распространенные дистрибутивы по умолчанию конфигурируются именно так, но существует и множество других настроек.

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

Этот файл управляет функцией `readline`, применяемой внутри bash для обработки ввода с клавиатуры. Первые три команды позволяют опознать при вводе восемь символов битов, а также гарантируют, что эти символы ни в коем случае не будут преобразованы в другие символы и, наконец, что эта информация действительно будет выведена. Последующие три строки управляют откликом на нажатие клавиш Delete, Pos1 и End.

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

Подсказки при вводе

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

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

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

В дополнение или в качестве альтернативы для PS1 можно также настроить переменную PROMPT_COMMAND. Она содержит команду, выполняемую всякий раз перед отображением PS1 (основные вопросы по настройке переменных окружения будут рассмотрены в подразделе «Важнейшие оболочковые переменные» раздела 8.6). О том, какие коды допускаются в PS1 (например, \w как сокращенное обозначение текущего кататлога), написано в документации по bash.

8.3. Ввод команд

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

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

Расширения названий команд и файлов

При автоматическом выставлении расширений для названий команд и файлов bash помогает значительно уменьшить долю рутинной печатной работы. Нужно всего лишь указать начальные буквы названия команды или файла и нажать клавишу Tab.

Если название команды или файла уже можно однозначно идентифицировать, это название дописывается полностью. Если имеется несколько названий, начинающихся одинаково, то название будет дополнено не до конца. Система выдаст в таком случае звуковой сигнал, означающий, что, возможно, имя записано не до конца.

Расширение имени файла лучше всего показать на примере. Ввод

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

автоматически дополняется (на моем компьютере) до

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

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

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

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

то bash приведет данный ввод в форму

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

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

```
user$ e Tab Tab
```

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

СОВЕТ

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

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

Определение пути к программе

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

- Команда `whereis` *имя* просматривает все стандартные каталоги.
- Команда `which` *имя* просматривает все каталоги, содержащиеся в переменной `PATH`, и определяет программу, которая выполнялась бы после ввода команды без указания пути. Команда `which` полезна в том случае, когда в системе есть

несколько версий одной программы и эти версии расположены в разных каталогах.

- Команда `type` *имя* работает подобно `which`, но учитывает и те команды, которые интегрированы в bash или заданы в форме псевдонимов (см. подраздел «Сокращения, связанные с псевдонимами» данного раздела).

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

Программно-специфичные расширения

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

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

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

ПРИМЕЧАНИЕ

Для того чтобы задать собственные правила дополнения имен, вам потребуется изучить изрядно запутанный синтаксис complete. Сжатое описание синтаксиса дается в файлах help complete и man bash (ищите Programmable Completion).

Другие советы по конфигурированию механизма дополнения имен предлагаются на следующих сайтах:

- http://www.caliban.org/bash/index.shtml;
- http://www.pl-berichte.de/t_system/bash-completion.html.

Важные сочетания клавиш

Таблица 8.1 построена исходя из предположения, что bash конфигурирована для работы с emacs. Именно так и обстоит дело почти во всех дистрибутивах. Если система реагирует на нажатие некоторых клавиш иначе, почитайте раздел 8.2 о конфигурировании. Если вы работаете под X, то и в том случае, когда bash конфигурирована правильно, управление клавиатурой в X может вызывать проблемы

(подробнее об этом рассказывается в разделе 12.5). Если вы работаете в Gnome, выполните в окне терминала Правка ▸ Комбинации клавиш и снимите флажок Деактивизировать все буквенные сокращения в меню.

Вставка последнего параметра

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

Поиск команд

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

Таблица 8.1. Сочетания клавиш в bash

Клавиши	Значение
↓, ↑	Прокрутить последние введенные команды
←, →	Передвинуть курсор назад или вперед
Pos1, End	Переместить курсор в начало или в конец строки
Ctrl+A, Ctrl+F	Как в предыдущем примере, когда Pos1 и End не работают
Alt+B, Alt+F	Переместить курсор на одно слово вперед или назад
Backspace, Delete	Удалить символы вперед или назад
Alt+D	Удалить слово
Ctrl+K	Удалить текст до конца строки
Ctrl+Y	Заново вставить последний удаленный текст
Ctrl+T	Поменять местами две предыдущих строки
Alt+T	Поменять местами два предыдущих слова
Tab	Дополнить названия команд и файлов
Ctrl+I	Погасить экран
Ctrl+R	Найти ранее введенные команды
Alt+.	Вставить последний примененный параметр
Ctrl+_	Отменить последнее изменение (Undo)

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

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

Сокращения, связанные с псевдонимами

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

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

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

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

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

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

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

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

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

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

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

8.4. Переадресация ввода и вывода

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

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

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

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

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

Сейчас в файле `content` находится список всех TEX-файлов, расположенных в текущем каталоге. Чаще всего применяется именно такой способ переадресации вывода. Однако есть еще два варианта: `2>` *файл* переадресовывает сообщения об ошибках в указанный файл; `>&` *файл* или `&>` *файл* переадресовывают в указанный файл и сообщения об ошибках, и программный вывод. Если использовать вместо > двойной символ >>, то весь ввод дописывается в конце уже имеющегося файла (см. также обзор синтаксиса в табл. 8.2).

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

ВНИМАНИЕ

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

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

Таблица 8.2. Переадресация ввода и вывода

Команда	Функция
Команда > файл	Переводит стандартный вывод в указанный файл
Команда < файл	Считывает ввод из указанного файла

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

Программные каналы

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

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

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

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

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

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

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

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

Только такие команды подходят для формулирования программного канала, из которого команды, предназначенные для ввода, будут считывать информацию. Если вы имеете дело с иной ситуацией, можно достичь сходных эффектов с помощью подстановки команд или команды `xargs` (см. раздел 8.6).

Размножение вывода командой tee

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

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

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

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

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

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

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

8.5. Выполнение команд

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

Фоновые процессы

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

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

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

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

Выполнение нескольких команд

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

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

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

```
user$ ls; date
```

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

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

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

Используя комбинации символов `&&` и `||`, можно выполнять команды относительно, то есть в зависимости от результата другой команды (табл. 8.3).

```
user$ команда1 && команда2
```

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

```
user$ команда1 || команда2
```

Другие возможности для создания условий и ветвления команд связаны с использованием оболочковой команды `if`, которая, однако, будет интересна лишь

тем, кто собирается заниматься программированием на языке оболочки (см. раздел 8.11).

Таблица 8.3. Выполнение команд

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

8.6. Механизмы подстановки

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

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

Механизм подстановки, применяемый в данном случае, называется подстановкой параметров. С помощью этого метода можно анализировать и изменять последовательности символов, сохраненные в переменных. Общий синтаксис таков: `${var__text}`, где `var` — это имя переменной, `_` — один или два специальных символа, а `text` — шаблон для поиска или установка по умолчанию. Этот механизм подстановки подробнее рассматривается в подразделе «Подстановка параметров» раздела 8.10.

Образование имен файлов с помощью * и ?

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

В качестве джокерных символов используются `?` (что означает любой символ) и `*` (что означает любое количество любых символов, в том числе ни одного). Последовательность символов `[a,b,e-h]*` означает файлы, имена которых начинаются с `a`, `b`, `e`, `f`, `g` или `h`. Если первый символ, указанный в квадратных скобках, — это `^` или `!`, то допустимы все символы, кроме указанных в скобках. Символ `~` можно использовать в качестве сокращенного названия домашнего каталога (см. также подраздел «Джокерные символы» раздела 3.1).

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

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

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

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

По умолчанию * не учитывает файлов и каталогов, имена которых начинаются с точки (иными словами, скрытых). Если вам требуется такой эффект, необходимо установить переменную окружения glob_dot_*-имена_файлов* (см. подраздел «Важнейшие оболочковые переменные» раздела 8.7).

Образование имен файлов с помощью **

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

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

Образование последовательностей символов с помощью {}

Из последовательностей символов, заключенных в фигурные скобки, bash образует любые мыслимые последовательности символов. Официальное название такого механизма подстановки — раскрытие скобок (brace expansion). Выражение part{1,2a,2b} означает part1 part2a part2b. Пользуясь раскрытием скобок, можно облегчить себе печатную работу, если вы обращаетесь ко многим файлам и каталогам

с похожими именами. По сравнению с джокерными символами * и ? есть определенное преимущество — можно образовывать имена еще не существующих файлов (это касается, например, mkdir).

```
user$ echo {a,b}{1,2,3}
a1 a2 a3 b1 b2 b3
user$ echo {ab,cd}{123,456,789}-{I,II}
ab123-I ab123-II ab456-I ab456-II ab789-I ab789-II
cd123-I cd123-II cd456-I cd456-II cd789-I cd789-II
```

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

```
user$ echo {1..5}
1 2 3 4 5
user$ echo {z..t}
z y x w v u t
```

Вычисление арифметических выражений с помощью []

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

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

В квадратных скобках можно использовать большинство операторов, известных из языка программирования C: +, -, *, / для четырех основных арифметических операций, % для вычислений по модулю, ==, !=, <, <=, > и >= для сравнений, << и >> для перемещений битов, !, && и || для логических операторов NO, AND и OR и т. д. Все вычисления выполняются с 32-битными целыми числами (диапазон чисел: ±2 147 483 648). Если следует извлечь отдельные значения из переменных, необходимо поставить символ $ (см. раздел 8.7 об управлении переменными).

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

Подстановка команд

Пользуясь подстановкой команд, можно заменить команду в командной строке результатом этой команды (табл. 8.4). Для этого команда должна быть заключена между двумя символами `. Альтернативная запись — $(*команда*). Второй метод предпочтителен, так как, во-первых, в таком случае вы избегаете путаницы с тремя разными кавычками (", ' и `), во-вторых, второй метод допускает вложения.

Обозначенная таким образом команда будет заменена результатом ее выполнения. Подобная запись обеспечивает вложенный вызов нескольких команд, причем

одна команда передает результат своего выполнения другой. Две представленные далее равнозначные команды проясняют принцип работы этого чрезвычайно мощного механизма:

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

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

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

Эта команда приводит к следующему результату:

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

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

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

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

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

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

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

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

Специальные символы в последовательностях

Поскольку в bash практически любой символ, за исключением букв и цифр, имеет специальное значение, представляется практически невозможным использовать такие знаки в последовательностях символов и именах файлов. Эта пробле-

ма решается двумя способами. Можно поставить перед специальным символом обратный слэш (\) либо заключить всю последовательность символов в апострофы или кавычки. Указав апострофы, вы можете, например, удалить файл с именем ab* $cd:

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

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

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

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

Команда возвращает в качестве результата последовательность символов This is the access path, за которой следует содержимое оболочковой переменной PATH. Если применить не кавычки, а обычные апострофы, то вся последовательность символов будет передана через echo в неизменном виде. Оболочковые переменные подробнее рассмотрены в следующем разделе. Справка по специальным символам, используемым в bash, предлагается в разделе 8.12.

Таблица 8.4. Механизмы подстановки

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

8.7. Оболочковые переменные

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Локальные и глобальные переменные (переменные окружения)

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

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

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

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

Если вы создаете переменные, которые должны управлять работой других команд Linux, эти переменные всегда должны быть глобальными! Для того чтобы

можно было, с одной стороны, использовать механизмы подстановки, действующие в оболочке, а с другой стороны, задавать глобальные переменные, эти переменные следует сначала присваивать с помощью `x=...`, а потом дополнительно определять их как глобальные с помощью `export x`.

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

Важнейшие оболочковые переменные

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

- `BASH` — содержит имя файла bash.
- `glob_dot_`*`имена_файлов`* — управляет дополнением джокерных символов в файлах: если определяется переменная (с любым значением), то символ * учитывает и те файлы, названия которых начинаются с точки. Чтобы снова выйти из этого состояния, необходимо написать `unset glob_dot_`*`имена_файлов`*.
- `HOME` — содержит путь к домашнему каталогу, например `/home/mk`.
- `LOGNAME` — хранит логин (имя пользователя).
- `HOSTNAME` — содержит хост-имя (имя компьютера).
- `MAIL` — включает путь к каталогу, в котором сохраняется входящая почта (только если у вас установлен локальный почтовый сервер).
- `OLDPWD` — содержит путь к последним использовавшимся каталогам.
- `PATH` — хранит список каталогов. Если bash должна выполнить команду, она просматривает все каталоги, перечисленные в `PATH`, в поисках команды. Каталоги отделяются друг от друга двоеточиями.

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

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

  Из соображений безопасности (чтобы избежать незапланированного выполнения программ в текущем каталоге), в `PATH` не указывается локальный каталог.

Если вы хотите выполнять программы, содержащиеся в текущем каталоге, не указывая перед ними `./`, нужно дополнить `PATH` точкой.

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

Кроме описанных здесь обычно используются и многие другие переменные окружения, управляющие функциями оболочки, а также многих других программ. Список всех заданных переменных выводится командой `printenv | sort`.

8.8. Программирование: введение и примеры

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

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

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

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

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

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

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

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

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

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

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

Пример 1: grepall

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

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

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

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

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

СОВЕТ

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

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

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

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

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

Пример 2: stripcomments

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

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

Коротко поясню: ^[[:blank:]]*\# находит строки, начинающиеся с #, причем между началом строки (^) и символом # может находиться сколько угодно знаков пробела и табуляции. Параметр -v инвертирует обычную функцию grep: вместо извлечения пустых строк команда возвращает все строки, не соответствующие образцу. Аналогичным образом вторая команда grep удаляет все строки, начинающиеся с ;. Наконец, третья команда grep удаляет все строки, состоящие только из символа начала строки и конца строки ($). Подробнее синтаксис grep описан на странице справки.

Пример 3: applysedfile

Два предыдущих примера позволяют понять, как можно облегчить себе работу по набору текста и умственную работу, однако нисколько не открывают широких возможностей сценарного программирования. Следующий пример уже гораздо лучше позволяет разобраться в этом вопросе. Предположим, у вас есть целый набор файлов и в каждом файле нужно провести несколько одинаковых процессов поиска и замены. Обычно это случается, когда требуется изменить названия переменных и процедур в программном коде, который содержится во многих файлах.

При решении таких задач вам поможет сценарная программа applysedfile. Она вызывается следующим образом:

```
user$ applysedfile *.tex
```

Теперь программа создаст из всех файлов *.tex резервные копии *.bak. Дополнительно используется команда UNIX sed, чтобы выполнить для каждого файла *.tex целую последовательность команд. Эти команды должны находиться в файле ./sedfile, который автоматически используется командой applysedfile. Код applysedfile выглядит так:

```
#! /bin/bash
# Пример sedfile
# Использование: applysedfile *.tex
# Применяется ./sedfile с перечнем передаваемых файлов
for i in $*
do
 echo "process $i"
 # сделать резервную копию старого файла
 cp $i ${i%.*}.bak
 # создать новый файл
 sed -f ./sedfile < ${i%.*}.bak > $i
done
```

Несколько коротких замечаний о том, как работает эта программа: три первых строки — это комментарии, вводимые символом #.

Ключевое слово for вводит цикл. При каждом выполнении цикла значением переменной i становится имя файла. Полный список имен файлов взят из $*. Данная комбинация является подстановочным символом для всех параметров и имен файлов, сообщаемых программе.

В теле цикла выводится имя каждого файла. С помощью cp создается резервная копия файла. При этом удаляются все символы, идущие после первой точки в имени файла, и имя файла дополняется расширением .bak. Наконец, для файла выполняется команда sed, при этом используется управляющий файл sedfile с локального компьютера.

Например, при замене буквосочетаний в немецкоязычном файле, чтобы текст соответствовал новым орфографическим правилам, первые строки рассмотренного файла будут выглядеть так:

```
s.da..dass.g
s.mu..muss.g
s.pa.t.passt.g
s.la.t.lasst.g
```

В каждой строке мы видим команду sed, с помощью которой первая последовательность символов заменяется второй (команда s). Последующая буква g означает, что данная команда должна выполняться в одной строке многократно (например, если в строке несколько раз встретятся слова «daß[1]» или «muß[2]»).

---

1 Что (нем.).

2 Должен (нем.).

Пример 4: сценарий резервного копирования

В следующем сценарии только две строки, но я уже не могу представить без него некоторой рутинной работы. Этот сценарий переносит все файлы *.tex из локального каталога в архив с именем backup.tgz.

Во второй строке этот файл копируется в каталог /backup. На моем компьютере /backup — это не просто каталог, а подключенный в систему сегмент второго жесткого диска. Таким образом, исходные файлы и их резервные копии оказываются на двух разных дисках. Весьма маловероятно, что оба диска одновременно выйдут из строя. В принципе резервные копии можно копировать и в NFS-каталог другого компьютера — так будет еще надежнее.

При копировании резервный файл сразу же переименовывается, при этом указывается год, дата и время (например, 20070406-2051.backup.tgz). Таким образом, вы гарантируете, что более ранние резервные копии не будут просто заменены более новыми. Кроме того, вы получаете доступ ко всем без исключения резервным копиям вне зависимости от того, когда они были созданы. Иначе говоря, если вы ошибочно удалите целый файл и заметите свою ошибку только через неделю, у вас сохранится более ранняя резервная копия того же файла.

Для генерации имен резервных файлов используется шаблон $(date "*формат*"). Таким образом, мы вставляем результат выполнения команды date в имя резервного файла:

```
#! /bin/sh
# Example mybackup
tar czfv backup.tgz *.tex
cp backup.tgz /backup/$(date "+%y%m%d-%H%M").backup.tgz
```

Пример 5: создание эскизов

Эскизами (thumbnails) называются миниатюрные версии файлов изображений. Следующий сценарий вызывается в форме makethumbs *.jpg. Он создает подкаталог 400 × 400 и сохраняет там уменьшенные копии исходных изображений. Максимальный размер новых изображений составляет 400 × 400 пикселов, причем пропорции оригинала остаются неизменными. Более мелкие изображения не увеличиваются.

Сценарий применяет команду convert из пакета Image Magick (см. раздел 5.1). Для уменьшения изображения используется параметр -resize. Параметр -size указывается только для ускорения обработки.

```
#!/bin/sh
# Usage: makethumbs *.jpg
if [ ! -d 400x400 ]; then # create subdirectory
mkdir 400x400
fi
for filename do # process all files
echo "processing $filename"
convert -size 400x400 -resize 400x400 $filename 400x400/$filename
done
```

8.9. Программирование: синтаксис

Все файлы оболочки должны начинаться со строки, состоящей из символов #! и имени желаемой оболочки. В таком случае для выполнения файла автоматически вызывается указанная оболочка. Для большинства сценариев оболочки выбирается #!/bin/sh. Только если вы пользуетесь bash-специфичными функциями, нужно специально указать #!/bin/bash.

Сценарии оболочки могут выполняться лишь тогда, когда в них установлены биты доступа, обеспечивающие чтение (r) и исполнение (x). Если сценарии расположены на внешних носителях или сегментах дисков, нужно убедиться, что к дереву каталогов подключен параметр exec.

ВНИМАНИЕ

В первой строке сценария нельзя использовать специальные символы национальных алфавитов, даже в комментариях. Оболочка bash отказывается исполнять файл и выводит сообщение cannot execute binary file (не удается выполнить двоичный файл).

В файлах сценариев нельзя разделять строки типичными для Windows сочетаниями команд carriage return (возврат каретки) и linefeed (переход на новую строку). Но бывает, что файл был создан в Windows и скопирован в Linux, и тогда такие сочетания будут попадаться. В таком случае bash вернет немногим более понятное сообщение об ошибке bad interpreter. При работе с файлами Unicode (UTF8) правильное разбиение на строки обеспечивается следующей командой:

```
recode u8/cr-lf..u8 < windowsfile > \ linuxfile
```

Если вы пишете коллекцию собственных сценарных программ для ежедневного использования, целесообразно сохранять эти сценарии в одном и том же месте. В качестве каталога подойдет ~/bin. Если дополнительно внести такие же изменения в файл .profile, эти сценарные программы могут выполняться и без указания полного пути. В некоторых дистрибутивах делать это не нужно, в них ~/bin всегда входит в состав PATH.

```
# Addition in ~/.profile or in ~/.bashrc
PATH=$PATH':~/bin'
```

8.10. Программирование: управление переменными

Вводная информация по работе с переменными уже давалась в разделе 8.7. Я объяснил, в частности, разницу между обычными переменными оболочки и переменными окружения. В этом разделе мы рассмотрим другие стороны работы с переменными, особенно важные для оболочкового программирования. Точнее, мы поговорим об их области определения и о некоторых переменных, заданных в bash (например, $* или $?), о механизме подстановки параметров для анализа и обработки последовательностей символов, содержащихся в переменных, и, наконец, рассмотрим вопросы ввода переменных в программах оболочки.

Область определения переменных

Чтобы понять тонкости применения переменных при выполнении программ оболочки, необходимы базовые знания о механизмах запуска команд и программ оболочки.

Для выполнения команды или программы bash создает новый процесс с собственным PID-номером (такой номер используется внутри системы Linux для идентификации процесса и управления им). Новому процессу сообщаются только те оболочковые переменные, которые были объявлены как переменные окружения (`export` или `declare -x`, см. раздел 8.7). Если команда запускается в приоритетном режиме, bash продолжает работать на фоне этой команды, ожидая, пока она будет выполнена. В противном случае обе программы (то есть bash и запущенная в фоновом режиме команда) выполняются параллельно.

Особый случай — запуск оболочковой программы. Программы оболочки выполняются отнюдь не в работающей оболочке, а в специально запускаемом для этого командном подпроцессоре. В таком случае одновременно работают два образца bash — один для интерпретации команд, а другой для выполнения программ оболочки. Отдельные командные подпроцессоры требуются оболочке для того, чтобы можно было параллельно выполнять несколько программ оболочки, чтобы они при этом не влияли друг на друга (даже если работают в фоновом режиме).

Использование командных подпроцессоров в особенности отражается на управлении переменными в тех случаях, когда каждый подпроцессор (или оболочка) обладает собственным набором переменных. При запуске любой другой программы ей сообщаются лишь те переменные интерактивной оболочки, которые были объявлены как переменные окружения. Переменные оболочки и командного подпроцессора работают полностью независимо друг от друга, то есть при внесении изменений в один набор переменных это никак не отражается на другом наборе.

Иногда может потребоваться объявить в работающей оболочковой программе новые переменные либо изменить имеющиеся переменные. Чтобы это получилось, можно выполнять оболочковые программы и в рамках уже запущенной оболочки bash, не используя командный подпроцессор. Для этого нужно поставить перед именем файла оболочковой программы точку или пробел. Именно так кратко записывается оболочковая команда `source`.

Вот пример: вы хотите написать оболочковую программу, которая бы дополняла переменную `PATH` путем к текущему каталогу. Нужная вам программа `addpwd` очень проста:

```
#! /bin/sh
# shell program addpwd adds the current directory to the path name
#
PATH=$PATH":"$(pwd)
```

Итак, теперь в переменной `PATH` сохраняется ее прежнее содержимое, двоеточие и результат выполнения команды `pwd` — последняя часть сохраняется благодаря подстановке команд. Следующий контрольный прогон показывает, что содержимое переменной `PATH` текущей оболочки изменится лишь тогда, когда вы поставите

перед запускаемой командой addpwd точку (в командном подпроцессоре, запускаемом при первом обращении к addpwd, PATH также изменяется, но эти изменения действуют, пока выполняется addpwd).

```
user$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
user$ addpwd
user$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
user$ . addpwd
user$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/user
```

Переменные, задаваемые оболочкой

Программы оболочки могут обращаться и к некоторым переменным, заранее заданным bash. Эти переменные нельзя изменять в ходе присваивания, их можно только считывать. Имя переменной состоит из различных специальных символов. В табл. 8.5 переменные даются уже с предшествующим символом $.

Таблица 8.5. $-переменные

Переменная	Значение
$?	Возвращаемое значение последней команды
$!	Номер PID последнего запущенного фонового процесса
$$	PID текущей оболочки
$0	Имя файла, содержащего только что выполненный файл оболочки (или имя символьной ссылки, указывающей на файл)
$#	Количество параметров, переданных программе оболочки
От $1 до $9	Параметр от 1 до 9
$* или $@	Совокупность всех переданных параметров

Еще несколько замечаний относительно работы с переменными: $0-$9, $# и $* служат для интерпретации параметров, передаваемых программе пакетной обработке. Эти переменные могут использоваться практически с любым из сценариев, показанных в этой главе.

В связи с интерпретацией параметров интересна команда bash под названием shift. Она как будто «перемещает» переданные параметры от переменной $0 до переменной $9. Если вы выполните shift 9, то первые девять параметров, переданных программе, будут потеряны, но вы легко сможете запросить следующие девять. Если указать shift без дополнительных параметров, то список сдвинется на один параметр.

Переменную $? можно использовать для определения условий, чтобы поставить дальнейшее выполнение программы в зависимость от результата последней команды. В принципе команду можно прямо задать как условие, указав ее таким образом в if. Достоинство переменной $? заключается в том, что она избавляет от работы с длинными и непонятными командами.

Переменная `$$` содержит **PID (идентификационный номер процесса)**. Это числовое значение применяется внутри системы Linux для управления процессами. PID является уникальным, то есть во всей системе гарантированно отсутствует второй процесс с таким же номером, поэтому такое значение отлично подходит для создания временного файла. Например, с помощью `ls > tmp.$$` можно сохранить список всех файлов в файле `tmp.`*nnn*. Даже если такой же пакетный файл одновременно работает на другом терминале, в двух оболочках процессы будут иметь разные идентификационные номера, что исключает возникновение конфликта на уровне имен.

Массивы

Кроме обычных переменных bash также различает массивы. Вплоть до версии 3 индекс должен был быть числом. Обратите внимание, что синтаксис для доступа к *n*-ному элементу отличается от синтаксиса, принятого в C.

```
x=()              # Определение пустого массива
x[0]='a'          # Присваивание элементов массива
x[1]='b'
x[2]='c'
x=('a' 'b' 'c')   # Краткий вариант записи четырех предыдущих строк
echo ${x[1]}      # Считывание элемента массива
echo ${x[@]}      # Считывание всех элементов массива
```

Для программистов исключительно важны массивы ассоциативных элементов, которые стали поддерживаться в bash, начиная с версии 4.0. Не забудьте, что массив сначала требуется специально объявить как ассоциативный с помощью `declare -A`. В противном случае система сочтет, что это обычный массив. Тогда последовательности символов, содержащиеся в индексе, будут интерпретированы как `0` и у вас получится обычный массив, состоящий из одного-единственного элемента (`Index 0`).

```
declare -A y        # Определение пустого массива ассоциативных элементов
y[abc]=123               # Присваивание элемента ассоциативного массива
y[efg]=xxx
y=( [abc]=123 [efg]=xxx )  # Краткий вариант записи двух предыдущих строк
echo ${y[abc]}             # Считывание одного элемента массива
```

Еще одно важное нововведение версии 4 заключается в том, что с помощью команды `mapfile` можно построчно преобразовать текстовый файл в элементы обычного массива:

```
mapfile z < текстовый_файл
```

Подстановка параметров

В bash **под подстановкой параметров понимается использование ряда команд, с помощью** которых можно обрабатывать последовательности символов, сохраненные в переменных. Обратите внимание, что перед именем переменной необходимо ста-

вить символ $. Опять же, если из переменной должен считываться эталонный образец, также необходимо использовать символ $.

- ${var:-default} — если переменная пуста, такое выражение возвращает в качестве результата установку, заданную по умолчанию. В противном случае возвращается содержимое переменной. Сама переменная не изменяется.
- ${var:=default} — аналогично предыдущему, однако если до операции переменная была пуста, ее содержимое изменяется.
- ${var:+new} — если переменная была пуста, то она остается пустой. Если же переменная уже определена, то ее предыдущее содержимое заменяется новым. Тогда выражение возвращает новое содержимое переменной.
- ${var:? Error_message} — если переменная пуста, то выводится имя переменной и сообщение об ошибке. Кроме того, программа оболочки завершает работу. В противном случае выражение возвращает содержимое переменной.
- ${#var} — в качестве результата возвращает количество символов, сохраненных в переменной (0, если переменная пуста). Переменная не изменяется.
- ${var#pattern} — сравнивает начало переменной с заданным шаблоном. Если удается опознать шаблон, то выражение возвращает содержимое переменной, за исключением самого краткого фрагмента текста, хранящего поисковый шаблон. Если же соответствий шаблону найти не удается, то возвращается содержимое переменной целиком. В поисковом шаблоне можно использовать подстановочные символы, с которыми мы познакомились, обсуждая образование имен файлов (* ? [abc]). Переменная в любом случае остается неизменной.

```
user$ dat=/home/mk/buch/buch.tar.gz
user$ echo ${dat#*/}
home/mk/buch/buch.tar.gz
user$ echo ${dat#*.}
tar.gz
```

- ${var##pattern} — аналогично предыдущему, но теперь удаляется самый крупный фрагмент текста, содержащий поисковый шаблон:

```
user$ dat=/home/mk/buch/buch.tar.gz
user$ echo ${dat##*/}
buch.tar.gz
user$ echo ${dat##*.}
gz
```

- ${var%pattern} — аналогично ${var#pattern}, но в данном случае сопоставление образцов производится в конце содержимого переменной. В конце переменной удаляется кратчайший фрагмент, содержащий поисковый шаблон. Сама переменная не изменяется.

```
user$ dat=/home/mk/buch/buch.tar.gz
user$ echo ${dat%/*}
/home/mk/buch
user$ echo ${dat%.*}
/home/mk/buch/buch.tar
```

- `${var%% pattern}` — аналогично предыдущему, но теперь удаляется самый крупный фрагмент текста, содержащий поисковый шаблон и находящийся в конце переменной.

```
user$ dat=/home/mk/buch/buch.tar.gz
user$ echo ${dat%%/*}
-- вывод отсутствует --
user$ echo ${dat%%.*}
/home/mk/buch/buch
```

- `${var/find/replace}` — замена первого совпадения с шаблоном `find` посредством `replace`:

```
user$ x='abcdeab12ab'
user$ echo echo ${x/ab/xy}
xycdeab12ab
```

- `${var//find/replace}` — замена всех совпадений с шаблоном `find` посредством `replace`:

```
user$ x='abcdeab12ab'
user$ echo echo ${x//ab/xy}
xycdexy12xy
```

- `${!var}` — возвращает содержимое переменной, имя которой в виде последовательности символов содержится в `var`:

```
user$ abc="123"
user$ efg=abc
user$ echo ${!efg}
123
```

Считывание переменных с помощью read

С помощью bash-команды `read` можно вводить данные в ходе работы программы оболочки. Как правило, для этого сначала с помощью `echo` задается краткий текст, сообщающий пользователю, какой ввод ожидается (например, `y/n`, числовое значение и т. д.). При этом целесообразно использовать параметр `-n`, позволяющий вводить данные сразу же после текста `echo`, а не со следующей строки. При выполнении последующей команды `read` bash ожидает от пользователя ввода строки и нажатия клавиши Enter.

Возьмем программу, в которой цикл `while` выполняется до тех пор, пока в переменной `a` не окажется последовательность символов, удовлетворяющая определенным условиям. В следующем примере мы видим, как работает эта программа:

```
user$ readvar
Введите число: a
Неверный ввод, повторите ввод, пожалуйста
Введите число: 12
12
```

После ввода информации с помощью read все содержимое переменной удаляется посредством подстановки параметров, если последовательность включает любые символы, кроме цифр, знака «минус» и пробелов. Правда, эта система контроля не идеальна (в соответствии с ней допускается как последовательность символов 12-34-5, так и 12 34), но все же весьма эффективна. О циклах while будет рассказано в разделе 8.11.

```
#! /bin/sh
# Пример readvar: считывание числового значения
a= # удаление содержимого переменной a
while [ -z "$a" ]; do
 echo -n "введите число: "
 read a
 a=${a##*[^0-9,' ',-]*}  # удаление последовательностей символов,
                         # содержащих любые символы, кроме 0-9,
                         # знака «минус» и пробела
 if [ -z "$a" ]; then
  echo "Неверный ввод, повторите ввод, пожалуйста"
 fi
done
echo $a
```

8.11. Программирование: условные переходы и циклы

Условные переходы в программах оболочки создаются с помощью команд if и case. В то время как if лучше подходит для обычных операций выбора, case предназначен для анализа последовательностей символов (сопоставления образцов).

If-условные переходы

В файле iftst с помощью if-запроса мы проверяем, были ли переданы два параметра. Если этого не произошло, выводится сообщение об ошибке. Программа завершается командой exit с возвращаемым значением, не равным нулю (индикатор ошибок). В противном случае содержимое обоих параметров выводится на экран.

```
#! /bin/sh
# Пример iftst
if test $# -ne 2; then
 echo "Команде должно быть передано ровно два параметра!"
 exit 1
else
 echo "Параметр 1: $1, Параметр 2: $2"
fi
```

Короткий тестовый запуск показывает, как работает программа:

```
user$ iftst a
Команде должно быть передано ровно два параметра!
user$ iftst a b
Параметр 1: a, Параметр 2: b
```

Критерием для условного перехода является возвращаемое значение последней команды перед then. Если эта команда возвращает значение 0, то условие выполняется. Если then указывается в той же строке, что и предыдущая команда (без перехода на следующую), то команда должна завершаться точкой с запятой.

ВНИМАНИЕ

Обратите внимание, что значение истинности (true) в bash равно 0, а ложности (false) — не равно 0. В большинстве других языков программирования значения прямо противоположны! Команды, завершаемые без ошибок, возвращают значение 0. Любое значение, не равное 0, указывает на ошибку. Некоторые команды возвращают различные значения в зависимости от типа ошибки.

В предыдущем примере условие было создано с помощью команды test оболочки bash. При этом оператор -ne означает «не равно» (not equal). Команда test применяется всякий раз, когда необходимо сравнить друг с другом две последовательности символов или два числа, когда необходимо проверить, существует ли файл, и т. д. Эта команда будет описана в следующем разделе.

Предыдущую программу можно было сформулировать и иначе: вместо test может использоваться краткий вариант в квадратных скобках. При этом перед названием команды [и после него] нужно ставить пробел.

Кроме того, из структуры if можно вычленить вторую команду echo, так как из-за наличия команд exit все строки кода, следующие за fi, будут выполняться только после того, как будет реализовано условие.

```
#! /bin/sh
# Пример iftst, 2. вариант
if [ $# -ne 2 ]; then
 echo " Команде должно быть передано ровно два параметра!"
 exit 1
fi
echo "Параметр 1: $1, Параметр 2: $2"
```

Формулирование условий с помощью test

В bash невозможно прямо задавать условия, например, для сравнения переменной и значения. Во-первых, вся архитектура bash основана на том, что все действия осуществляются с помощью команд, построенных по одинаковому образцу, во-вторых, специальные символы < и > уже закреплены для выполнения других функций. Поэтому для формулирования условий в циклах и точках перехода в bash необходимо использовать команду test. (Между прочим, test является самостоятельной командой и существует не только в bash. Она была интегрирована в bash, чтобы увеличить скорость обработки данных.)

Команда test возвращает значение 0 (истинно), если условие выполняется, или 1 (ложно), если не выполняется. Для сокращения печатной работы предусмотрен краткий вариант записи в квадратных скобках.

Команда test используется для решения задач двух классов: для сравнения двух чисел, для сравнения последовательностей символов и для того, чтобы узнать, существует ли файл и проявляет ли он определенные свойства. В следующих примерах показаны некоторые возможные варианты применения этой команды.

- test "$x" — эта команда проверяет, занят ли x (то есть, если в последовательности символов содержится 0 символов, имеем ложно; в противном случае истинно).
- test $x -gt 5 — проверяет, имеет ли переменная x числовое значение больше 5. Если x не содержит числа, выводится сообщение об ошибке. Вместо -gt (больше) могут также использоваться следующие операторы сравнения: -eq (равно), -ne (не равно), -lt (меньше), -le (меньше или равно) и -ge (больше или равно).
- test -f $x — проверяет, существует ли файл с именем, указанным в x.

Если необходимо интерактивно выполнять команду test в оболочке, после нее нужно считать значение переменной $? (возвращаемое значение последней команды) с помощью echo:

```
user$ a=20
user$ test $a -eq 20; echo $?
0
user$ test $a -gt 20; echo $?
1
```

Case-условные переходы

Конструкции case вводятся ключевым словом case, за которым следует параметр, предназначенный для анализа (обычно это переменная). После ключевого слова in можно указать несколько возможных шаблонов строки, с которыми будет сравниваться параметр. При этом могут использоваться те же джокерные символы, что и при работе с именами файлов. Шаблон завершается круглой скобкой, то есть, например, --*). Это делается для распознавания таких последовательностей, которые начинаются с двух знаков «минус». Несколько шаблонов можно отделять друг от друга символом |. В таком случае проверяются оба шаблона. Например, *.c|*.h) служит для распознавания файлов *.c и *.h в одном и том же ветвлении программы.

Команды, идущие вслед за скобками, должны завершаться двумя точками с запятой. Если потребуется переход else, то в качестве последнего шаблона необходимо указать * — такому шаблону будут соответствовать все последовательности символов. При обработке конструкции с case учитывается только первый переход, в котором параметр соответствует указанному шаблону.

В следующем примере casetst показано применение case для классификации переданных параметров на имена файлов. Цикл для переменной i выполняется со всеми параметрами, переданными файлу оболочки. В этом цикле каждый

отдельный параметр анализируется с помощью case. Если параметр начинается с дефиса (-), то он находится в конце переменной opt, в противном случае — в конце dat.

```
#! /bin/sh
# Пример casetst
opt=     # Удаление opt и dat
dat=
for i do # Цикл для всех переданных параметров
 case "$i" in
  -* ) opt="$opt $i";;
  * ) dat="$dat $i";;
 esac
done     # Конец цикла
echo "Options: $opt"
echo "Files: $dat"
```

Тестовый запуск файла оболочки на практике показывает, как работает этот простой оператор выбора. Параметры, переданные по порядку без сортировки, подразделяются на имена файлов и параметры:

```
user$ casetst -x -y dat1 dat2 -z dat3
Options: -x -y -z
Files: dat1 dat2 dat3
```

По тому же принципу условные переходы case могут использоваться для классификации определенных расширений файлов (путем указания поискового шаблона *.abc). Если вы хотите плотнее заняться case-анализом, посмотрите файл оболочки /usr/bin/gnroff. В нем дается синтаксис параметров, передаваемых nroff, в виде, понятном родственной команде groff.

For-циклы

Циклы в bash создаются с помощью трех команд. Команда for осуществляет цикл со всеми элементами указанного списка. Команда while осуществляет цикл до тех пор, пока указанное условие не перестанет выполняться. Команда until, наоборот, осуществляет цикл до тех пор, пока указанное условие не будет выполнено. Все три цикла можно досрочно завершить командой break. Команда continue пропускает оставшуюся часть тела цикла и запускает цикл заново.

В первом примере переменной i по очереди присваиваются последовательности символов a, b и c. В теле цикла между do и done выводится содержимое переменной. Обратите внимание, что в конце списка, а также в конце команды echo необходимо поставить точку с запятой. Отказаться от этих точек с запятой можно лишь тогда, когда ввод разделен на несколько строк (часто такое случается в сценарных файлах).

```
user$ for i in a b c; do echo $i; done
a
b
c
```

Эквивалентная многострочная формулировка вышеуказанной команды в сценарном файле выглядела бы так:

```
#! /bin/sh
for i in a b c; do
 echo $i
done
```

Список для `for` может быть построен с использованием джокерных символов для имен файлов или с использованием конструкций вида `{..}`, с помощью которых создаются последовательности символов (см. раздел 8.6). В следующем примере все файлы `*.tex` копируются в `*.tex~`. (В Linux/UNIX тильда (~) в конце имени файла обычно означает резервную копию. При работе с командой `cp` выражение `$file` всякий раз ставится в кавычках, чтобы имена файлов, содержащие пробелы, обрабатывались правильно.)

```
user$ for file in *.tex; do cp "$file" "$file~"; done
```

Если циклы `for` создаются без `in ...`, то переменные циклов получают по порядку все параметры, переданные при вызове (то есть это соответствует `in $*`). Пример такого цикла приводится при описании `case`.

Циклы while

В следующем примере переменной `i` присваивается значение `1`. Потом значение переменной, находящейся в теле цикла между `do` и `done`, при каждом выполнении цикла увеличивается на 1, пока не будет превышено значение `5`. Обратите внимание, что условия должны указываться в квадратных скобках, как это делалось с условными переходами `if`, с командой `test` или с ее сокращенным вариантом.

```
user$ i=1; while [ $i -le 5 ]; do echo $i; i=$[$i+1]; done
1
2
3
4
5
```

Следующий цикл обрабатывает все имена файлов, получаемые после выполнения команды `ls *.jpg`:

```
ls *.jpg | while read file
do
 echo "$file"
done
```

Циклы until

Единственное отличие между циклами `while` и `until` заключается в том, что условие формулируется с противоположной логикой. Следующая команда эквивалентна

вышеуказанному циклу `while`. При этом для формулировки условия `i>5` применяется оператор `-gt` (больше).

```
user$ i=1; until [ $i -gt 5 ]; do echo $i; i=$[$i+1]; done
1
2
3
4
5
```

8.12. Справка по важнейшим специальным символам bash

И при вводе команд, и при программировании оболочки для выполнения различных действий применяется множество специальных символов. В табл. 8.6 обобщены все специальные символы, рассмотренные в этой главе.

Таблица 8.6. Специальные символы, используемые в bash

Символ	Значение
;	Отделяет команды друг от друга
:	Команда оболочки, ничего не делает
.	Запуск оболочки без собственного командного подпроцессора (.file соответствует исходному файлу)
#	Ввод комментария
#!/bin/sh	Идентифицирует оболочку, в которой будет выполняться программа
&	Выполняет команду в фоновом режиме (com &)
&&	Выполнение одной команды в зависимости от результата другой (com1 && com2)
&>	Переадресация стандартного вывода и ошибок (соответствует >&)
|	Создание программных каналов (com1 | com2)
||	Выполнение одной команды в зависимости от результата другой (com1 || com2)
*	Джокерный символ для имен файлов (любое количество символов)
?	Джокерный символ для имен файлов (любой символ)
[abc]	Джокерный символ для имен файлов (любой символ из abc)
[expression]	Сокращенный вариант записи test expression
(...)	Выполнение команд в той же оболочке ((com1; com2))
{...}	Группирование команд
{ , , }	Объединение нескольких последовательностей символов (a{1,2,3} → a1 a2 a3)
{a..b}	Объединение нескольких последовательностей символов (b{4..6} → b4 b5 b6)
~	Сокращенное обозначение домашнего каталога
>	Переадресация вывода в файл (com > file)
>>	Переадресация вывода и добавление его в существующий файл
>&	Переадресация стандартного вывода и ошибок (соответствует &>)
2>	Переадресация стандартного вывода ошибок
<	Переадресация ввода из файла (com < file)
<< end	Переадресация ввода из активного файла до завершения

Символ	**Значение**
$	Обозначение переменных (echo $var)
$!	Номер PID последнего процесса, запущенного в фоновом режиме
$$	PID актуальной оболочки
$0	Имя выполняемого в данный момент сценарного файла оболочки
$1–$9	Первые девять параметров, переданных команде
$#	Количество параметров, переданных программе оболочки
$* или $@	Совокупность всех переданных параметров
$?	Возвращаемое значение последней команды (0=OK или номер ошибки)
$(...)	Подстановка команд (echo $(ls))
${...}	Различные специальные функции для обработки последовательностей символов
$[...]	Арифметические вычисления (echo $[2+3])
"..."	Предотвращение интерпретации большинства специальных символов
'...'	Предотвращение интерпретации всех специальных символов
`...`	Подстановка команд (echo `ls`)
символ	Отменяет действие указанного специального символа

9 Базовая конфигурация

Теперь обратимся к конфигурированию системы Linux — этой важной теме будут посвящены несколько следующих глав. Здесь я приведу некоторую вводную информацию и затрону такие темы:

- конфигурирование текстовых консолей;
- настройка даты и времени;
- управление пользователями;
- интернационализация, кодировка, Unicode;
- обзор конфигурации аппаратного обеспечения;
- файлы регистрации (журналирование).

В следующих главах будет рассмотрено управление пакетами, системными библиотеками, графическими системами (X), администрирование файловой системы, запуск системы (GRUB, Init-V, Upstart), работа с ядром и его модулями.

9.1. Введение

В этой и следующей главах мы заглянем «за кулисы» системы и увидим, как работает программа конфигурации Linux. Нам предстоит разобраться, что, где и как работает, как осуществляется управление системой и как задаются настройки по умолчанию. Кроме того, в этих главах я предоставлю вам много полезных фоновых сведений о работе системы в целом.

Как ни жаль, но каждый из множества дистрибутивов конфигурируется посвоему, не так, как остальные. И различия обычно заключаются в мелочах. В этой книге я попытался «привести все к общему знаменателю» и описать как можно больше систем. И все же не исключено, что именно в вашем дистрибутиве некоторые детали будут построены чуть по-другому. В таких случаях рекомендую обратиться к документации.

Вы можете не согласиться со мной, но я убежден, что при описании администрирования именно комплексный подход является оптимальным, то есть не стоит писать одну книгу об администрировании Red Hat, другую — о SUSE и т. д. В первую очередь потому, что даже отдельные дистрибутивы изменяются от версии к версии. Рано или поздно вам обязательно придется учиться, по крайней мере

читать и понимать англоязычные руководства, файлы справки и т. д. Эта книга не заменит оригинальных пособий или пошаговых руководств, однако поможет вам приобрести необходимые фоновые знания.

Где системный администратор?

Итак, до сих пор системный администратор был для вас таинственным незнакомцем, который снова и снова приходил к вам на помощь — часто нехотя, иногда очень усталый. Если вы не работаете на крупном предприятии, то администратор вообще становится для вас кем-то абстрактным, героем книг, в которых написано: «Если что-то не ладится — обратитесь к системному администратору».

Если вы сами установили Linux на своем компьютере, то картина меняется: теперь вы сами себе системный администратор! Не пугайтесь такого громкого титула: это просто специалист, который сам занимается конфигурацией своего компьютера. И если спектр задач ограничивается основными функциями Linux, то работа будет под силу каждому. Правда, вам постоянно придется заглядывать в книги по Linux — например, в эту или другие специальные источники.

Конфигурационные инструменты

В большинстве дистрибутивов имеются очень удобные конфигурационные программы, с которыми можно работать как в ходе установки, так и после нее: `drakconf` в Mandriva, различные программы `system-`*`xxx`* в Red Hat или Fedora, модули YaST в SUSE и т. д. При конфигурировании системы к этим инструментам необходимо прибегать в первую очередь! Они специально оптимизированы под конкретный дистрибутив и избавляют вас от лишней работы.

Наряду с многочисленными конфигурационными программами, специфичными для отдельных дистрибутивов, есть и такие, которые не привязаны к конкретному варианту системы. Из таких программ сегодня наиболее популярна Webmin (http://www.webmin.com/). Дополнительное преимущество программы заключается в том, что она может полностью обслуживаться через сетевой интерфейс и, соответственно, идеально подходит для удаленного обслуживания.

Кроме того, заслуживает упоминания многообещающая программа eBox (http://www.ebox-platform.com/), специально разработанная для администрирования сетевых функций серверов Ubuntu. С этими инструментами связаны и определенные проблемы — дело в том, что их поддержкой напрямую не занимаются фирмы или организации, создающие дистрибутивы. Пока конфигурационные инструменты, не связанные с дистрибутивами, оперативно обновляются вслед за внесением изменений в новые версии.

Администрирование сети

До сих пор предполагалось, что вы собираетесь конфигурировать только один (локальный) компьютер. Если же вам как администратору требуется управлять сотнями компьютеров с Linux, вышеуказанные конфигурационные инструменты

так или иначе будут неэффективны. Вместо этого вам потребуются программы, с помощью которых можно будет централизованно вносить конфигурационные изменения, касающиеся всех компьютеров или определенной группы машин. Например, вам понадобится установить на всех офисных компьютерах новую версию OpenOffice, задействовать на всех машинах, подсоединенных к Интернету, новые правила брандмауэра и т. д.

В следующем списке перечислены некоторые подобные инструменты. Эти программы сильно отличаются как функционально, так и по стоимости. Более подробно описать их здесь, к сожалению, невозможно — во-первых, не позволяют размеры книги, во-вторых, я не работал с этими программами.

- Red Hat, коммерческая (http://www.redhat.com/rhn/);
- Novell/SUSE, коммерческая (http://www.novell.com/products/zenworks/);
- свободное ПО, автоматически устанавливается для Debian (http://www.informatik.uni-koeln.de/fai/);
- администрирование и конфигурирование Debian/Ubuntu, свободное ПО (http://m23.sourceforge.net).

Конфигурационные файлы

Продуманные до мелочей конфигурационные инструменты с красивыми пользовательскими интерфейсами разработаны именно для того, чтобы облегчить вам работу при непосредственном изменении конфигурационных файлов Linux. Это исключительно удобно как раз для новичков в Linux. Однако есть множество причин, по которым я рекомендую вам вплотную заняться конфигурационными файлами, а значит, познакомиться с внутренней архитектурой Linux.

- Изменять конфигурационные файлы можно в любом текстовом редакторе, в том числе и тогда, когда вы работаете с текстовой консолью или по сети через SSH. А многие конфигурационные программы работают лишь в графическом режиме или только на локальном компьютере.
- Когда вы разберетесь, как конфигурируется та или иная функция Linux, вы сможете применить эти знания при работе с практически любым другим дистрибутивом Linux. А работа со многими конфигурационными инструментами зависит от дистрибутива.
- Только внося изменения непосредственно в конфигурационные файлы, вы сможете управлять всеми аспектами системных функций. А функциональность конфигурационных инструментов часто ограничена лишь немногими (самыми важными) явлениями.
- Конфигурационные файлы можно с легкостью копировать с одного компьютера на другой. Это помогает сэкономить массу времени, когда вы переходите к работе с новым дистрибутивом, заново устанавливаете Linux на другой компьютер и т. д.
- Чем лучше вы понимаете строение конфигурационных файлов и связанные с ними возможности управления, тем лучше вы понимаете саму Linux и тем реже компьютер оказывается «черным ящиком», в который невозможно заглянуть.

Практически все конфигурационные файлы Linux находятся в каталоге /etc.

Связанные друг с другом конфигурационные файлы крупных программ часто располагаются в отдельных подкаталогах. Например, все конфигурационные файлы графической системы X расположены в каталоге /etc/X11. Некоторые подкаталоги имеют особое значение:

- /etc/default — файлы, характерные для отдельных дистрибутивов (Debian, Ubuntu);
- /etc/init.d или /etc/rc.d — система Init-V (запуск системы, см. раздел 14.10);
- /etc/event.d — Upstart (запуск системы, см. раздел 14.11);
- /etc/sysconfig — файлы, специфичные для отдельных дистрибутивов (Fedora, Red Hat, SUSE).

Надо отметить, что рекомендуется создавать резервную копию всего каталога /etc. Тогда при внесении изменений вы в любой момент сможете оперативно узнать, как выглядел конфигурационный файл в исходном состоянии.

```
root# mkdir /etc-backup
root# cp -a /etc/* /etc-backup
```

СОВЕТ

При редактировании конфигурационных файлов следите за тем, чтобы последняя строка завершалась нажатием клавиши Enter. Если такое окончание отсутствует, некоторые программы Linux могут допускать ошибки при обработке файлов.

Поиск конфигурационных файлов. Если вы не можете найти в своем дистрибутиве какой-либо конфигурационный файл, на это может быть много причин: возможно, у вас не установлены базовые программные пакеты, или же конфигурационные файлы расположены в вашем дистрибутиве где-то в другом месте. При поиске пользуйтесь командами locate, find и grep. Следующая команда показывает, как выполнить поиск файлов, в названиях которых содержится последовательность символов abcde, в /etc и во всех его подкаталогах:

```
root# cd /etc
root# find -type f -exec grep -q abcde {} \; -print
```

Работая над книгой, мне часто приходилось искать, как в дистрибутиве *x* производится управление функцией *y* или как вызвать функцию *z*. Для этого вышеуказанную команду пришлось ввести не одну сотню раз. Чтобы сэкономить время и силы, я написал маленький сценарий grepall, выполняющий эту работу (см. раздел 8.8).

Как активизировать новую конфигурацию

В некоторых программах изменения, внесенные в конфигурацию, вступают в силу лишь после повторного запуска программы либо после того, как вы прямо скомандуете системе заново считать конфигурационные файлы. Обычно для этого требуется выполнить одну из двух следующих команд (подробности в разделе 5.5):

```
root# /etc/init.d/function name restart
root# /etc/init.d/function name reload
```

В отличие от Windows, в Linux практически никогда не требуется перезапускать компьютер. Исключение — необходимость внесения изменений в ядро, а также некоторые аппаратно-зависимые настройки, которые можно произвести лишь непосредственно при запуске системы.

9.2. Конфигурация текстовых консолей

В современных дистрибутивах Linux сразу запускается графическая система, и новички порой даже не знают, что в системе есть текстовая консоль. Разумеется, сплошь и рядом встречаются ошибки в конфигурации X либо работа с графической системой может быть не предусмотрена по каким-либо другим причинам. При некоторых вариантах установки сервера приходится даже намеренно отказываться от работы с графической системой. Именно в таких случаях вам пригодится умение пользоваться консолью (см. главу 2).

За простейшие настройки, такие как раскладка клавиатуры или вид шрифта, отвечает система `kbd` или более новый вариант `console` — в зависимости от дистрибутива. Однако в каждом дистрибутиве отдельные детали конфигурации отличаются.

Раскладка клавиатуры

Debian

В Debian за раскладку клавиатуры отвечает пакет `console-data`. Сценарий `/etc/rcS.d/S05keymap.sh` загружает при запуске системы заархивированную таблицу соответствий `/etc/console/bootime.kmap.gz` и обрабатывает ее с помощью команды `loadkeys`.

Чтобы создать этот файл, можно скопировать сюда файл клавиатуры из каталога `/usr/share/keymaps` с помощью команды `install-keymap`. Например, в немецкоязычном регионе используются файлы клавиатуры `i386/quertz/de-latin1.kmap.gz` или `de-latin1-nodeadkeys.kmap.gz`. Разница между двумя этими вариантами заключается в обращении со специальными символами `, ', ~ и ^: в стандартном варианте из них можно составлять символы иностранных языков. Если необходимо указать сам знак, дополнительно нажмите пробел. В варианте `nodeadkey` символ вводится сразу же.

Более удобно проводить конфигурацию с помощью команды `dpkg-reconfigure console-data`: она есть во многих диалоговых окнах и помогает выбрать нужную таблицу соответствий.

При необходимости `loadkeys` может выполняться и напрямую, для изменения действующей раскладки клавиатуры (только в консоли, а не под X!). Раскладка указывается в качестве параметра. Команда сама найдет нужный файл и разархивирует его.

```
root# loadkeys de-latin1
```

Fedora

В Fedora для настройки раскладки клавиатуры используется пакет kbd. Таблица соответствий устанавливается во время запуска системы сценарием Initrd. Раскладка клавиатуры определяется в конфигурационном файле /etc/sysconfig/keyboard. Изменения, внесенные в этот файл, вступают в силу лишь тогда, когда файлы Initrd создаются заново (см. подраздел «Создание файла Initrd» раздела 14.5). Кроме того, файл /etc/sysconfig/keyboard также интерпретируется программой fedora-setup-keyboard для конфигурации клавиатуры под X.

SUSE

В SUSE, как и в Fedora, используется пакет kbd и конфигурационный файл /etc/sysconfig/keyboard. На этом сходство заканчивается. Конфигурационный файл интерпретируется сценарием Init-V /etc/init.d/kbd. Настройки действуют только для консоли, а не для X.

Ubuntu

В Ubuntu за раскладку клавиатуры отвечают программы пакета console-setup. Конфигурационный файл /etc/default/console-setup определяет и настройки клавиатуры, и гарнитуру шрифта. При управлении раскладкой клавиатуры используются те же параметры, что и при конфигурации X (раздел 12.5):

```
# /etc/default/console-setup
...
# Шрифт
CHARMAP="UTF-8"
CODESET="Lat15"
FONTFACE="VGA"
FONTSIZE="16"
# Клавиатура
XKBMODEL="pc105"
XKBLAYOUT="de"
XKBVARIANT=""
XKBOPTIONS="lv3:ralt_switch"
```

Этот файл интерпретируется сценарием /bin/setupcon, который дважды выполняется при запуске системы: с /etc/init.d/keyboard-setup и /etc/init.d/console-setup.

Скажу несколько слов об обработке четырех клавиатурных переменных. Они интерпретируются сценарием ckbcomp, создающим из X-файлов клавиатуры таблицу соответствий, которая затем, как и в случае с kbd, активизируется командой loadkeys. Именно поэтому в Ubuntu также установлен пакет kbd.

Гарнитура шрифта

Как правило, консоли совместимы с Unicode. Однако в любом случае максимально возможное количество символов в шрифтах консолей очень невелико

(256 или 512), поэтому шрифты консолей включают лишь малую толику символов Unicode.

Debian

В Debian шрифт настраивается сценарием Init-V `/etc/rcS.d/S48console-screen.sh`. Конфигурационным файлом является `/etc/console-tools/config`.

Fedora

В Fedora шрифт настраивается с помощью команды `setfont`, сценарием файла `Initrd`, при этом интерпретируется конфигурационный файл `/etc/sysconfig/i18n`. Изменения, вносимые в этот файл, вступают в силу только при создании новых файлов `Initrd` (см. подраздел «Создание файла Initrd» раздела 14.5).

SUSE

В SUSE шрифт консоли настраивается с помощью `/etc/init.d/kbd`. Этот сценарий интерпретирует `/etc/sysconfig/console` и настраивает шрифт с помощью команды `setfont`. По умолчанию используется шрифт `lat9w-16.psfu`, содержащий, наряду с набором символов латиницы-1, еще и символ Евро.

Ubuntu

В Ubuntu гарнитура шрифта определяется в конфигурационном файле `/etc/default/console-setup`. Этот файл интерпретируется командой `setupcon` при запуске системы, выполняемой сценарием Init-V `/etc/init.d/console-setup`.

Gpm-конфигурация (мышь)

Мышь может применяться только в системе X Window System. Но программа `gpm` позволяет в ограниченном объеме использовать мышь и в текстовых консолях: в частности, можно выделять текст левой кнопкой мыши и вставлять его правой или средней кнопкой в том месте, где стоит курсор. Однако обратите внимание, что в большинстве программ для консоли невозможно изменять положение курсора мышью.

Как правило, `gpm` запускается процессом Init-V (раздел 14.10). Конфигурация, в зависимости от дистрибутива производится в файле `/etc/gpm.conf` или `/etc/sysconfig/mouse`, причем, как правило, не требуется изменять настройки, заданные по умолчанию. Чтобы запустить систему вручную, выполните следующую команду:

```
root# /etc/init.d/gpm start
```

9.3. Дата и время

Поскольку компьютерные сети раскинулись по всему миру, необходимо пользоваться единым международным временем. Таким временем является GMT (Greenwich Mean Time, или среднее время по Гринвичскому меридиану). На всех компьютерах

UNIX это время является «мерой всех вещей». Еще это время обозначается сокращением UTC (Universal Time Coordinated, или универсальное координированное время).

При сохранении файла компьютер не фиксирует текущее местное время, а пересчитывает его по этому международному стандарту. Если теперь просмотреть файл с помощью команды `ls -l`, то время будет пересчитано как местное, для того региона, где находится компьютер. Этот метод позволяет определить, какой файл актуальнее: тот, что был сохранен в Мюнхене в 18:00 по местному времени, или тот, что сохранили в 12:30 в Нью-Йорке — тоже по местному времени.

Настройка времени при запуске компьютера

На первом этапе работы процесса Init-V (см. подраздел «Создание файла Initrd» раздела 14.5) настраивается компьютерное время. Для этого команда `hwclock` считывает время с часов CMOS вашего компьютера. Часы CMOS могут содержать как местное время, так и GMT. Если ваш компьютер работает и с Windows, удобнее настроить часы CMOS на местное время.

В Red Hat, Fedora и SUSE в конфигурационном файле `/etc/sysconfig/clock` содержится информация о том, на какое время настроены часы CMOS — на местное или GMT и в каком часовом поясе находится ваш компьютер. В Debian и Ubuntu в файле `/etc/default/rcS` сообщается, на какое время настроены часы CMOS, а информация о часовом поясе содержится в файле `/etc/timezone`.

Можно и не полагаться на часы CMOS, которые знамениты своей неточностью, а узнать время на более точном сервере времени в Интернете или локальной сети (раздел 20.11). Для этого, конечно же, необходимо постоянное соединение с Интернетом или с локальным сервером времени. Лучше всего пользоваться этим методом при работе на компьютере, который постоянно включен.

Настройка часового пояса

Чтобы команды вроде `ls` или файловый менеджер KDE либо Gnome пересчитывали GMT в местное время и отображали его в таком виде, каждая программа должна «знать», в каком часовом поясе она работает.

Для этого почти во всех программах Linux используются функции библиотеки `glibc`. Эта библиотека интерпретирует файл `/etc/localtime`, являющийся копией файла часового пояса (time zone file) из каталога `/usr/share/zoneinfo`. Файл `localtime` также может быть символьной ссылкой на файл часового пояса.

Конфигурационные инструменты

В зависимости от дистрибутива для настройки часового пояса или для изменения времени и даты на часах компьютера можно применять различные конфигурационные программы:

- Gnome — `time-admin`;
- KDE — модуль центра управления Управление системой ▸ Дата/Время;

- Debian, Ubuntu — `tzconfig`, инструменты Gnome/KDE;
- Red Hat/Fedora — `system-config-date`;
- SUSE — YaST-модуль Система ▸ Часовой пояс.

9.4. Пользователи и группы, пароли

Когда мы говорим об управлении пользователями, то подразумеваем в первую очередь управление тем, кто имеет доступ к определенным файлам, какие программы имеет право выполнять, какими файлами устройств пользоваться и т. д. Управление пользователями и доступом требуется в тех случаях, когда на одном и том же компьютере могут работать несколько человек. Должны существовать правила, в соответствии с которыми один пользователь будет получать права читать и изменять файлы других пользователей.

В Linux для такого управления создается список пользователей. Кроме того, каждый может относиться минимум к одной, но, возможно, и сразу к нескольким группам. Группы позволяют нескольким пользователям получать доступ к общим файлам или программам.

Чтобы можно было наладить управление правами доступа, вместе с каждым файлом сохраняется и информация о его владельце, принадлежности владельца к группам и *биты доступа*. Поскольку программы — это тоже файлы, а доступ к аппаратным компонентам часто осуществляется через так называемые *файлы-устройства*, этот механизм практически универсален.

В этой главе мы поговорим только об управлении пользователями и группами. В главах 3 и 4 речь шла об управлении файлами и выполнении процессов, а также о связи этих проблем с управлением файлами. В разделе 3.7 давалась базовая информация о битах доступа, которые сохраняются вместе с каждым файлом. Советы о том, как можно выполнять системные программы, не владея при этом правами администратора, даются в разделе 4.2.

Конфигурационные программы

В принципе, будучи администратором, вы можете в значительной мере управлять пользователями вручную, непосредственно изменяя файлы, рассмотренные в этом разделе. Однако гораздо удобнее и надежнее было бы освоить инструменты для управления группами и пользователями, входящие в состав большинства дистрибутивов:

- Gnome — `users-admin` (входит в состав `gnome-system-tools`);
- KDE — модуль центра управления Управление системой ▸ Управление пользователями;
- Debian, Ubuntu — инструменты Gnome или KDE;
- Red Hat, Fedora — `system-config-users` (рис. 9.1);
- SUSE — модуль YaST Безопасность ▸ Пользователи и безопасность ▸ Группы.

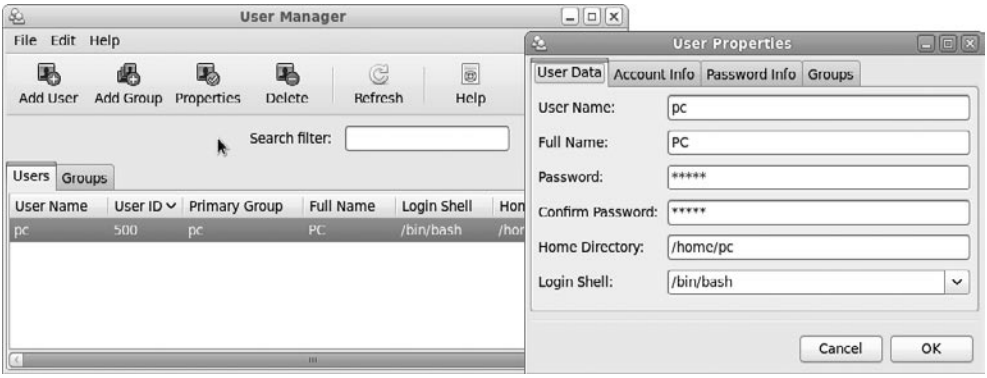

Рис. 9.1. Управление пользователями в Fedora

Команды

Если вы готовы отказаться от удобных пользовательских интерфейсов или хотите автоматизировать управление пользователями с помощью сценариев, можете применить командаы, представленные в табл. 9.1. В следующем примере показано, как создать учетную запись нового пользователя testuser и присвоить ему пароль.

```
root# useradd -m testuser
root# passwd testuser
New passwd: xxx
Re-enter new passwd: xxx
```

Таблица 9.1. Команды для управления пользователями и группами

Команда	Функция
adduser	Создает учетную запись нового пользователя (Debian)
addgroup	Создает новую группу (Debian)
chgrp	Изменяет групповую отнесенность файла
chmod	Изменяет биты доступа определенного файла
chown	Передает файл другому владельцу
chsh	Изменяет стандартную оболочку, применяемую при работе данным пользователем
delgroup	Удаляет группу (Debian)
deluser	Удаляет учетную запись пользователя (Debian)
groupadd	Создает новую группу
groupdel	Удаляет группу
groupmod	Изменяет свойства группы
groups	Показывает группы, к которым принадлежит данный пользователь
id	Показывает ID-номера текущей группы и текущего пользователя
newgrp	Изменяет активную группу определенного пользователя
newusers	Создает несколько новых пользовательских учетных записей
passwd	Изменяет пароль пользователя
useradd	Создает новую учетную запись пользователя
userdel	Удаляет учетную запись пользователя
usermod	Изменяет свойства учетной записи пользователя

Как вы, конечно же, заметили, для решения некоторых задач предусмотрены сразу по две команды (например, `adduser` и `useradd`). Команды `adduser`, `addgroup`, `deluser` и `delgroup` — это специфичные для Debian дополнения к общепринятым командам `useradd`, `groupadd` и т. д. В Debian, Ubuntu и других дистрибутивах, произошедших от Debian, эти команды учитывают правила, заданные в файлах `/etc/adduser.conf` и `/etc/deluser.conf`.

Как обычно, Red Hat и Fedora вносят в общую картину некоторую путаницу: в этих дистрибутивах также имеются команды `adduser`, `addgroup`, `deluser` и `delgroup`. Однако это совсем не те команды, что используются в Debian, а всего лишь ссылки на `useradd`, `groupadd`, `userdel` и `groupdel`. Именно поэтому синтаксис команд `adduser` и `useradd` в Fedora идентичен — и при этом отличается от синтаксиса `adduser` в Debian!

SUSE/Novell также не ищут легких путей: там нет команд `adduser/-group` и `deluser/-group`. Для выполнения функций `groupadd/-del/-mod` и `useradd/-del/-mod` применяется собственная разработка, которая совместима с командами других дистрибутивов по важнейшим, но не по всем параметрам.

Управление пользователями

В Linux и в других системах семейства UNIX существует три типа пользователей.

- **Суперпользователь, он же системный администратор, он же root:** обычно этот пользователь имеет в системе имя root. Если вы войдете в систему под именем root (разумеется, для этого необходимо знать соответствующий пароль), ваши права будут не ограничены: вы можете просматривать, изменять, удалять любые файлы, выполнять любые программы и т. д. Такие широкие полномочия требуются только для администрирования системы. Для выполнения любых других задач наделять пользователя правами администратора нельзя — из соображений безопасности.
- **Обычный пользователь:** таким пользователям система Linux нужна для работы. Они имеют неограниченный доступ к собственным файлам и ограниченный доступ к оставшейся части системы. По возможности их логин должен совпадать с именем (например, kathrin или hofer).
- **Системный пользователь для демонов и служебных программ:** наконец, существует несколько пользовательских учетных записей, для которых не предусмотрена интерактивная работа с компьютером, — такие «пользователи» нужны для выполнения определенных программ. Например, за работу веб-сервера Apache отвечает не системный администратор, а отдельный «пользователь», который в зависимости от дистрибутива может называться apache, или wwwrun, или httpd и т. д. Благодаря наличию таких пользователей гарантируется максимальная безопасность системы.

Файл /etc/passwd

Список всех пользователей сохраняется в файле `/etc/passwd`. Для каждого пользователя записывается следующая информация: логин, полное имя, идентификационные номера пользователя и группы (UID и GID), домашний каталог и используемая оболочка. Формат таков:

Логин:Пароль:UID:GID:Имя:Домашний каталог:Оболочка

Например, в следующих строках записана информация о нескольких пользователях, работающих в Ubuntu Linux (сохранено в каталоге /etc/passwd):

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
...
kofler:x:1000:1000:Michael Kofler,,,:/home/kofler:/bin/bash
huber:x:1001:1001:Herbert Huber,,,:/home/huber:/bin/bash
```

По названию passwd можно предположить, что в файле также сохраняются пароли. Раньше так и было, но со временем положение изменилось. Вместо паролей /etc/passwd теперь содержит только символ x. Информация о пароле (которая, разумеется, зашифрована) хранится в отдельном файле /etc/shadow (см. подраздел «Пароли» этого раздела).

Логин

Логин (имя для входа в систему) должен состоять только из букв нижнего регистра (буквы и цифры кодировки US-ASCII) и содержать не больше восьми символов. Хотя можно использовать пароли в других кодировках, а также пароли, содержащие более восьми символов, но лучше этого не делать, так как могут возникнуть проблемы совместимости с некоторыми программами. Если отдельно сохранять полные названия, эти ограничения не действуют.

UID и GID

Номер *UID (идентификатор пользователя)* служит для идентификации пользователя. Этот номер сохраняется отдельно в качестве дополнительной информации к каждому файлу, чтобы было известно, кому какой файл принадлежит.

Существуют правила распределения номеров UID: администратор всегда имеет UID = 0. Для серверных служб и демонов в большинстве дистрибутивов предусмотрены идентификационные имена в диапазоне от 1 до 999 (в Red Hat и Fedora, а также в некоторых других дистрибутивах все еще применяются старые правила с использованием диапазона значений от 1 до 499). Для обычных пользователей, соответственно, предусмотрены номера от 1000 (в Red Hat и Fedora — от 500).

Номер *GID (идентификатор группы)* указывает, к какой группе принадлежит пользователь. Подробнее идентификаторы групп будут рассмотрены в следующем разделе.

Домашний каталог

Это то место, где пользователь может хранить свои личные файлы. Для домашних каталогов обычных пользователей, как правило, применяется путь /home/login-name. В домашнем каталоге также сохраняются персональные конфигурационные настройки пользователя и различные программы. Например, в файле с расширением .emacs содержатся настройки конфигурации редактора Emacs. Поскольку названия таких конфигурационных файлов обычно начинаются с точки, эти файлы скрыты. Чтобы отобразить такие файлы, введите команду ls -la.

Чтобы учетные записи новых пользователей сразу создавались с удобными стандартными настройками, при их создании следует копировать все файлы из /etc/

skel в новый домашний каталог. Многие программы, создающие новые учетные записи, выполняют эту операцию автоматически. Таким образом, исходные настройки для каждого нового пользователя содержатся в каталоге /etc/skel.

Оболочка

Это интерпретатор, с помощью которого пользователь, вошедший в систему, может выполнять команды. Поскольку в Linux на выбор предлагается несколько оболочек, в файле passwd необходимо указать, какой оболочкой вы будете пользоваться. В Linux чаще других используется оболочка bash, подробно рассмотренная в главе 8 (в файле passwd необходимо сохранять полное название файла оболочки, то есть, например, /bin/bash).

Управление группами

Группы требуются для того, чтобы обеспечить нескольким пользователям общий доступ к определенным файлам. Для этого каждый пользователь причисляется к *первичной группе* (initial group). Кроме того, пользователь может относиться к разным *дополнительным группам* (supplementary groups), то есть состоять в нескольких группах сразу.

В файле /etc/group хранится полный список групп. Например, в следующих строках показаны определения некоторых групп в файле /etc/group. Соблюдается такой формат:

Название_группы:Пароль:Идентификатор_группы:Список_пользователей

Следующие строки взяты из файла group системы Ubuntu:

```
root:x:0:
daemon:x:1:
bin:x:2:
...
dialout:x:20:cupsys,kofler,huber
...
users:x:100:
admin:x:114:kofler
...
kofler:x:1000:
huber:x:1001:
...
```

Пользователь соотносится с группой двумя способами.

- Основная группа пользователя сохраняется в /etc/passwd. Основная группа пользователя kofler в нашем примере также называется kofler (идентификатор группы 1000 в файле /etc/passwd).
- Чтобы задать принадлежность пользователя к другим группам, можно указать его имя в последнем столбце в файле /etc/group. Так, kofler принадлежит к группам adm, admin и dialout. Благодаря этому пользователь kofler может выполнять в системах Ubuntu задачи администратора и устанавливать соединение с Интернетом.

При использовании идентификаторов групп номер 0 предусмотрен для администратора, номера с 1 по 99 — для системных служб. Номер GID = 100 обычно зарезервирован для группы users. Номера GID выше 100 можно распределять по собственному усмотрению.

Соотнесение пользователя с основной группой. Существуют две основные стратегии соотнесения пользователей с их основными группами.

- Устоявшийся метод, используемый в UNIX/Linux уже многие годы, — относить всех обычных пользователей к основной группе users. Этой очень простой стратегии следует, например, SUSE.
- Дистрибутивы, построенные на основе Debian, например Red Hat и Fedora, используют другой подход: каждый пользователь получает собственную основную группу. В таком случае оба пользователя — kofler и huber — будут отнесены к одноименным основным группам. Группа users в итоге больше не нужна.

В некоторых случаях второй подход предпочтительнее — например, когда несколько пользователей дополнительной группы создают общие файлы. Однако этими преимуществами можно воспользоваться лишь при умелом системном администрировании.

В очень многих образцах Linux, где отдельные группы пользователей вообще могут не иметь общих файлов либо необходимо управлять общими проектами, применяются специальные инструменты, например CVS (система контроля параллельных версий). При этом неважно, какие основные группы используются — общая или индивидуальные.

Пароли

Пароли Linux могут состоять только из символов кодировки ASCII (никаких международных специальных символов!). Из соображений безопасности рекомендуется создавать пароли, состоящие из букв как верхнего, так и нижнего регистра; кроме того, пароль должен содержать минимум одну цифру.

Пароли в Linux сохраняются в форме так называемых хэш-кодов, которые позволяют контролировать, но не реконструировать данные. В современных дистрибутивах сейчас используется надежный хэш-алгоритм SHA512 (см. переменную `ENCRYPT_METHOD` в `/etc/login.defs`). Данный алгоритм позволяет создавать пароли любой длины (в некоторых более старых дистрибутивах используются другие хэш-алгоритмы, учитывающие только восемь первых символов в пароле!).

Так называемые взломщики паролей (программы для расшифровки) действуют противоположным образом: они обычно примеряют к программе миллионы случайных паролей — как правило, в таком качестве взломщик использует слова из словаря в комбинации с числами. Поскольку многие пользователи применяют запоминающиеся пароли, а компьютеры становятся все быстрее, подобный метод подбора ужасающе часто приводит злоумышленника к цели.

Для того чтобы осложнить жизнь потенциальным агрессорам, в новых системах Linux зашифрованные коды-пароли сохраняются не прямо в `/etc/passwd`, а в отдельном файле `/etc/shadow`. Преимущество заключается в том, что этот файл может читать только администратор. (Файлы `/etc/passwd` и `/etc/group` должны быть

доступны для чтения всем пользователям системы, так как содержат элементарную информацию по управлению системой. А к файлу /etc/shadow могут иметь доступ только программы для верификации и изменения паролей. Поэтому потенциальный «агрессор» должен сначала получить доступ администратора, и только потом сможет прочитать зашифрованные коды-пароли.)

Формат файла shadow должен быть следующим:

```
Логин:Код-пароль:d1:d2:d3:d4:d5:d6:reserved
```

Далее приведен фрагмент файла:

```
root:$6$Ecrkix...:14391:0:99999:7:::
daemon:*:14391:0:99999:7:::
bin:*:14391:0:99999:7:::
...
kofler:$6$TZR7...:14391:0:99999:7:::
```

Поля d1-d6

В полях d1-d6 могут содержаться желаемые данные по времени:

- d1 указывает, когда пароль был в последний раз изменен (данные указываются в днях, истекших с 01.01.1970);
- d2 определяет, через сколько дней можно изменить пароль;
- d3 указывает, через сколько дней пароль обязательно необходимо изменить, прежде чем он станет недействителен (подробности об этих полях сообщаются в файле справки man 5 shadow);
- d4 определяет, за сколько дней до истечения срока действия пароля пользователь начнет получать об этом предупреждения;
- d5 указывает, через сколько дней просроченная учетная запись без действующего пароля будет полностью деактивизирована;
- d6 задает, когда именно учетная запись будет деактивизирована.

Как правило, для d1-d3 применяются стандартные значения, чтобы пароль можно было изменить в любое время и чтобы он оставался действителен в течение неограниченного срока. Но поля d1-d6 можно использовать и для того, чтобы ограничивать сроки действия паролей, автоматически временно деактивизировать учетные записи (например, для управления учетными записями студентов в вузе).

Что касается большинства системных пользователей (они рассмотрены выше, например daemon, bin, sys и т. д.), здесь вместо пароля сохраняется только звездочка или восклицательный знак. Это означает, что действующего пароля не существует и войти в систему от имени такого пользователя невозможно. И все же системными учетными записями можно пользоваться: программы, запускаемые с правами администратора, могут потом словно «поменять» своего владельца и продолжать работать уже от имени bin, daemon, lp и т. д. Именно так обстоит дело с большинством системных процессов: они автоматически запускаются от имени администратора при старте системы, а затем сразу же меняют пользователя из соображений безопасности.

Файл /etc/login.defs

Множество параметров, предназначенных для внутрисистемного управления паролями и учетными записями, находится в файле `/etc/login.defs`. Там можно установить время ожидания после указания неверных данных при входе в систему, количество допустимых попыток входа в систему, количество битов доступа, которые должны использоваться в новых создаваемых домашних каталогах и т. д. Здесь также можно разрешить `passwd` принимать откровенно «слабые» пароли.

Изменение паролей

Чтобы изменить собственный пароль, выполните команду `passwd`. Сначала у вас будет запрошен пароль, а затем вам придется дважды ввести новый пароль. Вводимая информация на экране не отображается. Только если обе последовательности символов для нового пароля совпадут, этот пароль будет принят. Теперь при входе в систему вы должны будете использовать новый пароль.

Длина пароля должна составлять от шести до восьми символов. В паролях могут содержаться буквы верхнего и нижнего регистра, цифры и знаки пунктуации. Команда `passwd` проводит несколько проверок и отказывается принимать откровенно «слабые» пароли. В хороших паролях кроме строчных букв содержится минимум одна прописная и одна цифра. Кроме того, в качестве паролей нельзя использовать слова (или их небольшие вариации), которые можно найти в словаре.

Обычные пользователи могут изменять лишь собственный пароль, а администратор может изменять и пароли других пользователей.

```
root# passwd hofer
New password: ******
Re-enter new password: ******
Password changed.
```

СОВЕТ

Что делать, если вы забыли пароль администратора? В этом случае нужно загрузить компьютер с установочного диска или восстановительной дискеты либо воспользоваться «живым диском» (например, Knoppix). С помощью mkdir создаем новый каталог на псевдодиске восстановительной или загрузочной системы и привязываем к нему тот сегмент диска, на котором расположена система Linux:

```
mount -t ext3 /dev/xxx /dir
```

Теперь у нас есть доступ к файлу dir/etc/shadow и мы можем удалить в нем пароль администратора (все символы между первым и последним двоеточием) либо заменить его известным нам зашифрованным паролем другого пользователя. Войти в систему с привилегиями администратора, не вводя при этом пароля, можно только из текстовой консоли — такие настройки заданы в файле /etc/pam.d/commonauth из соображений безопасности. Здесь, заново запустив систему, вы устанавливаете новый пароль администратора командой passwd, чтобы можно было работать под X в качестве root.

Если вы хотите не допустить ситуации, в которой любой пользователь мог бы изменить пароль вышеописанным способом, нужно отключить в BIOS компьютера все загрузочные устройства, кроме первого жесткого диска. Но тогда вам уже никак нельзя забывать пароль… Еще одна возможность — зашифровать целый сегмент диска.

Файл faillog

В зависимости от действующих настроек /etc/login.defs информация обо всех неудачных попытках входа в систему определенным образом заносится в /var/log/faillog. В отличие от большинства других файлов регистрации, в этом файле используется двоичный формат.

С помощью faillog -u *имя* вы определяете, сколько раз заданный пользователь может ошибиться при входе в систему. При превышении этого числа учетная запись блокируется, пока администратор вновь не разблокирует ее командой faillog -u *имя* -r (команда сбрасывает счетчик попыток).

Кроме того, можно задать единое для всей системы максимальное количество попыток входа в систему с помощью команды faillog -m max. Правда, в таком случае вам придется выполнить и faillog -u root -m 0, чтобы данная мера не распространялась на администратора. Иначе может случиться так, что вы сами как администратор не сможете войти в систему, после того как другой пользователь несколько раз безуспешно попытается войти в систему с правами администратора.

Групповые пароли

Можно определять пароли не только для отдельных пользователей, но и для целых групп (команда gpasswd). Однако, в то время как пользовательские пароли необходимы в любом случае, групповые пароли применяются редко. Их основной недостаток заключается в том, что пароль должны знать все члены группы, а это осложняет процесс администрирования.

Если групповые пароли и в самом деле необходимы, их обычно сохраняют в файле /etc/gshadow. Групповой пароль нужно вводить в тех случаях, когда пользователь с помощью команды newgrp переходит из одной активной группы в другую (активная группа определяет, к какой группе будут относиться новые файлы).

Взаимодействие конфигурационных файлов

Сейчас мы еще раз в обобщенном виде рассмотрим, как взаимодействуют три файла — passwd, groups и shadow. Файл passwd каждого пользователя содержит строку, построенную по следующему образцу:

```
# Строка в /etc/passwd
kofler:x:1000:1000:Michael Kofler:/home/kofler:/bin/bash
```

Здесь kofler — это логин; 1000 — идентификационный номер пользователя; 1000 — идентификационный номер основной группы; Michael Kofler — полное имя (для электронной переписки, новостей и т. д.); /home/kofler — домашний каталог пользователя, а /bin/bash — используемая им оболочка. Идентификатор группы — это уникальный номер, важный для управления правами доступа к файлам.

Соответствующая строка в /etc/shadow с информацией о пароле выглядит следующим образом:

```
# Строка в /etc/shadow
kofler:$6$9dkO$...:13479:0:99999:7:::
```

Последовательность символов после kofler: — это зашифрованный пароль. Если не задать эту последовательность, то в систему можно будет входить, не указывая пароль. Если определить вместо нее символ * или !, то вход в систему будет запрещен.

Номер GID из /etc/passwd должен соответствовать группе из /etc/group. Во многих дистрибутивах каждому обычному пользователю соответствует одноименная группа:

```
# Строка в /etc/group
kofler:x:1000:
```

Управление пользователями в сети

Если вы объединяете в сеть несколько компьютеров с Linux и хотите обеспечить двусторонний доступ к файлам через NFS, не забывайте, что номера UID и GID на всех компьютерах должны быть единообразными. Уже сама организация такого единообразия на множестве компьютеров в сети — задача не из легких. А если вы хотите, чтобы любой пользователь мог войти в систему под своим именем с любого компьютера (разумеется, всякий раз используя для этого одни и те же логин и пароль), то вам понадобится постоянно синхронизировать все файлы /etc/passwd. Такое администрирование потребует от вас титанических усилий.

Для того чтобы облегчить работу, администратор выделяет для управления пользователями специальный центральный сервер. Кроме того, существует множество альтернативных возможностей (протоколов) для аутентификации клиентов, описать которые не позволяют размеры этой книги:

- LDAP — облегченный протокол доступа к сетевым каталогам;
- NIS — сетевая информационная служба, протокол считается устаревшим;
- Kerberos;
- Samba, или управление пользователями для Windows.

Подключаемые модули аутентификации

Подключаемые модули аутентификации (PAM) — это библиотека, которая помогает проводить аутентификацию. Если вы входите в систему на компьютере Linux, производите аутентификацию иным образом (su, sudo, ssh и т. д.) или изменяете пароль (passwd), то соответствующие программы обращаются к библиотеке PAM. С PAM также работают Cron и PolicyKit. Исчерпывающая, но уже довольно старая документация по PAM имеется на следующем сайте: http://www.kernel.org/pub/linux/libs/pam/.

На обычном компьютере Linux в файле /etc/passwd содержатся имена пользователей, а в файле /etc/shadow — соответствующие им зашифрованные пароли. По умолчанию PAM конфигурируется так, что может интерпретировать эти данные. Если требуется дополнительно использовать еще один метод аутентификации (например, LDAP), то конфигурацию PAM необходимо соответствующим образом

изменить. Для этого в разных дистрибутивах предназначены различные инструменты:

- Fedora — `system-config-authentication`;
- SUSE — YaST-модуль Безопасность ▸ Управление пользователями и группами;
- Ubuntu — `pam-auth-update`.

Файл pam.conf

Конфигурационные файлы расположены в каталоге `/etc/pam.d/`. Кроме того, интерпретируется файл `/etc/pam.conf`. В конфигурационных файлах построчно перечисляются правила. Каждое правило состоит минимум из трех частей (столбцов):

Тип Реакция PAM-модуль [аргументы модуля]

Кроме того, каждой записи в `pam.conf` должно предшествовать имя службы (например, login или other для стандартных записей). В файлах каталога `/etc/pam.d` каждой службой управляет одноименный файл.

Тип правила (первый столбец). В PAM различается четыре типа правил (типа столбцов). В некоторых дистрибутивах (Debian, Ubuntu) в файлах `common-account`, `common-auth`, `common-password` и `common-session` содержатся стандартные правила для четырех этих типов:

- `account` — позволяет выставлять для служб ограничения, зависящие от времени суток, загруженности, способа входа в систему (например, через консоль) и т. д.;
- `auth` — управляет авторизацией (запросом и проверкой пароля) и дополнительным присвоением привилегий (например, принадлежности к группам);
- `password` — управляет механизмом для изменения пароля;
- `session` — позволяет выполнять действия перед запуском определенной службы или после него (например, журналирование, привязка/отсоединение файловых систем, отображение состояния почтового ящика и т. д.).

Реакция (второй столбец). Следующий столбец в конфигурационных файлах определяет, как должна реагировать библиотека PAM, если правило выполняется или не выполняется. Реакцию можно описать двумя способами: с помощью обычного ключевого слова (например, `required`, `requisite`) либо указав в квадратных скобках пару значение/результат (например, `[success=1 new_authtok_reqd=done default=ignore]`). Значение четырех важнейших ключевых слов простого синтаксиса объясняется в табл. 9.2.

Таблица 9.2. Реакция на нарушение правил PAM

Ключевое слово	Реакция
requisite	При нарушении правила функция PAM сразу же возвращает отрицательный результат и остальные правила уже не обрабатываются
required	При нарушении правила функция PAM возвращает отрицательный результат; дальнейшие правила обрабатываются, но результат обработки не учитывается

Ключевое слово	Реакция
sufficient	Если правило выполняется, то функция PAM сразу же возвращает положительный результат (если только уже не было нарушено предшествующее правило requisite); остальные правила не обрабатываются. При нарушении PAM продолжает работу со следующего правила
optional	Результат выполнения правила важен только тогда, когда речь идет о единственном правиле определенного типа и об определенной службе (например, su)

При использовании второго варианта синтаксиса вы указываете несколько пар значение/результат в форме [*значение1=результат1 значение2=результат2 ...*]. Для value существует множество заданных ключевых слов, выражающих результат выполнения правила; result может быть либо числом, указывающим, сколько следующих правил необходимо пропустить, либо ключевым словом, задающим желаемый результат выполнения PAM (ignore, bad, die, ok, done или reset). В следующем листинге показано, как выразить ключевые слова первого варианта синтаксиса вторым способом записи:

```
requisite  = [success=ok new_authtok_reqd=ok ignore=ignore default=die]
required   = [success=ok new_authtok_reqd=ok ignore=ignore default=bad]
sufficient = [success=done new_authtok_reqd=done default=ignore]
optional   = [success=ok new_authtok_reqd=ok default=ignore]
```

Модуль и параметры (третий столбец). В третьем столбце указывается номер модуля PAM, который будет применяться для интерпретации правила. На работу модуля могут влиять параметры. К сожалению, не существует общего документа о том, какие параметры допустимы, и об их значении.

В традиционной конфигурации модуль pam_unix.so отвечает за интерпретацию файлов /etc/passwd и /etc/shadow. Параметр nullok позволяет вводить пустые пароли; nullok_secure разрешает аутентификацию с пустым паролем, если вход в систему происходит через консоль, указанную в /etc/securetty (по умолчанию в этом файле перечисляются локальные консоли /dev/ttyn). С параметром obscure модуль проводит несколько простейших тестов, чтобы определить, не слишком ли тривиален пароль. Параметр md5 указывает, сколько паролей необходимо зашифровать.

Стандартная конфигурация

В следующем листинге обобщены стандартные настройки по всем четырем типам правил в конфигурации Fedora 11. Обратите внимание на то, что стандартные настройки от дистрибутива к дистрибутиву значительно отличаются и часто распределены между несколькими файлами (например, common-*xxx* в Debian и Ubuntu).

```
# Файл /etc/pam.d/password-auth
auth       required    pam_env.so
auth       sufficient  pam_unix.so nullok try_first_pass
auth       requisite   pam_succeed_if.so uid >= 500 quiet
auth       required    pam_deny.so
account    required    pam_unix.so
account    sufficient  pam_localuser.so
account    sufficient  pam_succeed_if.so uid < 500 quiet
```

```
account   required    pam_permit.so
password  requisite   pam_cracklib.so try_first_pass retry=3
password  sufficient  pam_unix.so sha512 shadow nullok try_first_pass use_authtok
password  required    pam_deny.so
session   optional    pam_keyinit.so revoke
session   required    pam_limits.so
session   [success=1 default=ignore] pam_succeed_if.so service in crond quiet use_uid
session   required    pam_unix.so
```

Диспетчер переключения имен (NSS)

При входе в систему, управлении правами доступа к файлам, доступе к сети и т. д. может потребоваться практически любая информация об именах пользователей и групп, UID и GID, хост-именах, портах сетевых служб и т. д. По умолчанию эти данные находятся в файлах `/etc/passwd`, `/etc/group`, `/etc/hosts`, `/etc/services` и т. д.

В номенклатуре UNIX или Linux доступ к таким службам обобщается термином *«службы имен»*. За выполнение этих задач отвечает диспетчер службы имен (NSS), сборник функций библиотеки `libc`. Как и при аутентификации, можно настроить источник данных для NSS, например, в том случае, если данные предоставляет сервер LDAP. Важнейшим конфигурационным файлом является `/etc/nsswitch.conf`. По умолчанию этот файл выглядит так:

```
# Файл /etc/nsswitch.conf
passwd:      compat
group:       compat
shadow:      compat
hosts:       files mdns4_minimal [NOTFOUND=return] dns mdns4
networks:    files
protocols:   db files
...
```

Первый столбец в `nsswitch.conf` содержит информацию о базе данных или файле. После двоеточия стоят ключевые слова, описывающие способ доступа к данным, а также иногда реакцию на полученные данные (в квадратных скобках). В следующем списке объяснены важнейшие ключевые слова, связанные с методами доступа. Если в одной строке указано несколько методов доступа, они применяются по порядку, пока один из них не подойдет. Более подробная информация по синтаксису содержится в `man nsswitch.conf`.

- `files` — обращается к традиционным конфигурационным файлам, например `/etc/passwd` или `/etc/group`.
- `compat` — аналогично `files`, причем данным о пользователях и группах могут предшествовать символы `+` и `-`. Так улучшается совместимость с NIS. Это слово нельзя комбинировать с другими ключевыми словами.
- `db` — считывает информацию из файла баз данных BDB (Berkeley Database).
- `dns` — связывается с сервером имен.
- `mdns` — использует многоадресную DNS (она же Zeroconf или Apple Bonjour/Rendezvous).

Для работы методов доступа необходимо установить надлежащую библиотеку, например `libnss_db` для `db`. Если библиотека не установлена, то соответствующее ключевое слово просто игнорируется (без сообщения об ошибке).

nscd (демон кэширования службы имен)

В некоторых дистрибутивах (например, Fedora, Red Hat и SUSE) по умолчанию устанавливается программа `nscd`, активизируемая при выполнении процесса Init-V. В Debian и Ubuntu ее также можно установить.

Аббревиатура nscd означает «демон кэширования службы имен». Эта программа отмечает логины, имена групп и хост-имена, применяемые при конфигурации системы, а также их IP-номера. В отличие от сервера имен (раздел 17.4), `nscd` предоставляет эту информацию только локальному компьютеру, но не остальным компьютерам, подключенным к сети.

Использовать `nscd` целесообразно лишь тогда, когда управление пользователями происходит через сетевую службу (например, LDAP). В таком случае `nscd` действует в качестве кэша для информации, исключая избыточные запросы LDAP, ответы на которые иногда приходят достаточно нескоро.

Файл /etc/nscd.conf. Конфигурация `nscd` осуществляется в `/etc/nscd.conf`. Обычно записи в этом файле подразделяются на три группы: `passwd` для логинов, `group` для групп и `host` для хост-имен. Настройки для каждой группы определяют, как долго должны храниться файлы, максимальное количество записей, которыми необходимо управлять, и т. д.

9.5. Языковые настройки, интернационализация, Unicode

В этом разделе будут рассмотрены два вопроса.

- **Локализация или языковые настройки** — определяет, на каком языке будут отображаться сообщения об ошибках, меню, диалоговые окна, тексты справки и т. д. В соответствии с этими настройками также изменяется формат даты, времени, валюты и т. п.
- **Кодировка** — определяет, какие коды используются для сохранения букв. В данном случае общепризнанные правила существуют только для 7-битной ASCII. Однако практически во всех кодировках букве A соответствует код 65. Существуют отклонения, касающиеся международных символов. Например, буква Ä может соответствовать различным кодам в зависимости от применяемой кодировки.

При работе с локализацией программ вы будете часто сталкиваться с загадочными сокращениями i18n и l10n. Они означают соответственно *интернационализацию* (internationalization = i + 18 букв + n) или *локализацию* (localization = l + 10 букв + n). Такие сокращения хороши при работе с поисковиками, если вам понадобится дополнительная информация по этим темам.

Основы кодировок

Кодировка (character set) описывает соответствия между числовыми кодами и буквами. Наиболее известны кодировки ASCII (7 бит), ISO-латиница-n (8 бит) и Unicode (16 бит).

- **ASCII** — включает только 127 символов, в том числе буквы от a до z (или от A до Z), цифры от 0 до 9 и различные знаки пунктуации.
- **ISO-8859, латиница** — кодировки ISO содержат кроме 127 символов ASCII до 128 дополнительных символов для различных языковых регионов. В следующем списке перечислены важнейшие кодировки ISO:
  - ISO-8859-1 = латиница-1 — Западная Европа;
  - ISO-8859-2 = латиница-2 — Центральная и Восточная Европа;
  - ISO-8859-3 = латиница-3 — Южная Европа;
  - ISO-8859-4 = латиница-4 — Северная Европа;
  - ISO-8859-9 = латиница-5 — турецкая;
  - ISO-8859-10 = латиница-6 — северная (саамская, эскимосская, исландская);
  - ISO-8859-13 = латиница-7 — балтийская;
  - ISO-8859-14 = латиница-8 — кельтская;
  - ISO-8859-15 = латиница-9 = латиница-0 — сходна с латиница-1, но с символом Евро.

  Латиница-0 пока имеет статус «проекта» и не получила официального названия, но используется достаточно часто. В Windows кодировки называются *кодовыми страницами* (code pages). Code Page 1252 очень похожа на латиницу-1, но есть и некоторые отличия.
- **Unicode** — чтобы разрешить путаницу различных 8-битных кодировок, была разработана 16-битная кодировка Unicode, она же ISO-10646. Ею можно закодировать не только все символы европейских языков, но и большинство азиатских символов (поскольку для каждого символа предусмотрено 16 бит, в этой кодировке найдется место для более чем 65 000 символов).

  Unicode определяет, какой код будет присвоен какому символу, но не задает метод сохранения кодов. На первый взгляд кажется, что проще всего было бы представить каждый символ в виде 2 байт (то есть 16 бит). Такой формат называется UTF-16 (Unicode Transfer Format — формат передачи Unicode). Но у него есть два недостатка: во-первых, удваивается потребность в памяти даже в тех случаях, когда требуется сохранять преимущественно европейские символы или даже только символы US-ASCII. Во-вторых, байтовый код 0 оказывается где угодно в последовательностях символов Unicode. Но многие программы C, почтовые серверы и т. д. настроены так, что байт 0 означает конец последовательности символов.

  Есть и другие возможности представления текстов в Unicode. Популярнейшей альтернативой UTF-16 является UTF-8. При этом символы US-ASCII (7 бит),

как и ранее, представлены одним байтом, наибольший разряд в котором равен 0. Все остальные символы представляются в виде последовательностей длиной от 2 до 4 байт. Очевидный недостаток этого формата заключается в том, что отсутствует непосредственная взаимосвязь между размером документа в байтах и количеством символов в документе. Однако, поскольку UTF-8 лучше совместима с существующими программами и обладает другими достоинствами, в UNIX/Linux она считается стандартом. В Microsoft Windows часто используется UTF-16. Итак, если мы говорим о Unicode и имеем в виду работу с Linux, то, как правило, речь идет об UTF-8. Практически во всех дистрибутивах UTF-8 применяется по умолчанию.

Влияние кодировки

Действующая кодировка определяет, как именно будут кодироваться символы в текстовых файлах или в именах файлов. Файловые системы Linux «понимают» любую кодировку. Именем файла считается всякая последовательность символов, которая заканчивается байт-кодом 0. Но, в зависимости от того, какая кодировка сейчас используется, последовательность и количество байт для такого имени файла, как `äöü.txt`, может быть совершенно разной! Если используемая кодировка — латиница-1, то это имя можно выразить 7 байтами (плюс один нулевой байт). Если же используется кодировка Unicode/UTF-8, то имя файла будет иметь длину 10 байт (так как для представления каждого из символов — ä, ö, ü — требуется по 3 байта).

Множество программ функционируют независимо от кодировки либо могут понимать сразу несколько кодировок: например, почтовые клиенты и браузеры могут отображать письма и сайты, где используется не та кодировка, которая сейчас задана в качестве активной. Современные программы для обработки текста обычно сохраняют текст в Unicode либо используют собственный код. Такие редакторы, как Emacs или XEmacs, обычно также могут обрабатывать и сохранять текстовые файлы в разных кодировках.

Проблемы с кодировками

Чаще всего проблемы возникают, если отправитель и получатель текстового файла пользуются разными кодировками. Например, пользователь, работающий с дистрибутивом Linux, применяет кодировку Unicode и создает в текстовом редакторе файл, содержащий международные специальные символы. И когда пользователю, работающему с другой оперативной системой, где установлена кодировка-латиница, придется продолжить обработку этого файла, он с удивлением обнаружит, что символы, не относящиеся к ASCII, отображаются неправильно. Обычно такие проблемы легко решаются с помощью команд `recode` или `iconf` (см. раздел 5.3).

Аналогичные проблемы касаются имен файлов, в частности при работе с NFS3: если вы создаете на компьютере, использующем кодировку UTF8, файл `äöü.txt`, а потом обращаетесь к этому файлу с другого компьютера с кодировкой-латиницей через NFS, то имя файла будет выглядеть примерно как `A¤A¶A..txt`. Эта проблема решается применением одинаковой кодировки на всех компьютерах в сети

или использованием NFS4. Если вы хотите изменить кодировку для имен уже имеющихся файлов (таких файлов может быть очень много), то вам поможет команда `convmv`, описанная в разделе 5.3.

Шрифт (гарнитура)

Не следует путать шрифт с кодировкой. Шрифт отвечает за то, как именно определенный символ будет отображаться на экране. Для этого существуют различные гарнитуры. Arial, Courier, Helvetica, Palatino — наиболее известные из них.

Разумеется, между шрифтами и кодировкой есть некоторое сходство: чтобы символ с кодом 234 мог правильно отображаться на экране, должно быть известно, какой набор символов использовался при кодировке. Некоторые старые X-шрифты были ограничены 256 символами и существовали в виде нескольких отдельных версий для различных кодировок. Новые гарнитуры (TrueType, PostScript) содержат гораздо больше символов и совместимы с несколькими кодировками.

Настройка локализации и кодировки

В зависимости от дистрибутива или операционной системы домашнего компьютера для конфигурирования языковых настроек используются различные инструменты. Практически в любых случаях применяется кодировка UTF-8. Лишь в немногих дистрибутивах сохранилась возможность использования 8-битной кодировки. Во всех дистрибутивах следует заново войти в систему, чтобы измененные языковые настройки вступили в силу. Gnome учитывает языковые настройки системы, но сам не содержит никаких конфигурационных инструментов.

- Debian — `dpkg-reconfigure locales`;
- Fedora, Red Hat — `system-config-language`;
- KDE — модуль центра управления Персональные данные ▸ Страна/Регион;
- SUSE — YaST-модуль Система ▸ Язык;
- Ubuntu — `gnome-language-selector`.

Кроме того, в диалоговых окнах входа в систему в KDE и Gnome (то есть в `kdm` и `gdm`) существует возможность выбрать для следующего соединения желаемый язык.

Конфигурационные файлы

Неудивительно, что в разных дистрибутивах конфигурационные настройки сохраняются в различных местах:

- Debian, Ubuntu — `/etc/default/locale`;
- Red Hat, Fedora — `/etc/sysconfig/i18n`;
- SUSE — `/etc/sysconfig/language`.

Кроме того, во многих дистрибутивах учитываются пользовательские настройки, сохраняемые в файле `~/.i18n`.

Переменные LC и LANG

Внутри системы локализация и кодировка управляются переменными окружения LC_CTYPE и LANG. Для интерпретации этих переменных применяется библиотека glibc, используемая почти во всех программах Linux.

Локализация может проводиться категориально. При правильной конфигурации можно, например, применять немецкий формат для указания даты и времени, а сообщения об ошибках отображать по-английски. В табл. 9.3 перечислены важнейшие переменные.

Таблица 9.3. Важнейшие локализационные переменные

Переменная	Значение
LANG	Определяет стандартное значение для всех ненастроенных LC-переменных
LC_CTYPE	Задает кодировку
LC_COLLATE	Указывает порядок сортировки
LC_MESSAGES	Определяет вид уведомлений, сообщений об ошибках и т. д.
LC_NUMERIC	Задает представление чисел
LC_TIME	Определяет представление даты и времени
LC_MONETARY	Задает представление денежных сумм
LC_PAPER	Определяет размер бумаги
LC_ALL	Имеет приоритет над всеми индивидуальными LC-настройками

Разумеется, не в каждой программе учитываются все эти категории (а во многих программах переменные LC_ полностью игнорируются). Если отдельные категории не настроены, то программы применяют в качестве стандартного значения C или POSIX. Это означает, что сообщения об ошибках выводятся по-английски, дата и время отображаются в американском формате и т. д.

Чтобы не настраивать по отдельности все перечисленные здесь переменные, можно просто задать переменную LANG. При этом для всех переменных, не определенных особо, будет применяться стандартное значение LANG. Только переменная LC_COLLATE сохранит базовую настройку POSIX. В большинстве дистрибутивов языковые настройки полностью устанавливаются через переменную LANG.

Переменная LC_ALL мощнее, чем LANG. Если установить LC_ALL, ее настройки будут действовать для всех категорий (независимо от того, как настроены другие переменные LC или LANG).

В большинстве программ сообщения об ошибках и другие тексты находятся в отдельных каталогах, например /usr/share/locale*/*язык*/LC_MESSAGES. Другая необходимая информация по темам локализации и интернационализации находится на странице man о команде locale и в руководстве по glibc: http://www.gnu.org/software/libc/manual/.

Тестирование локализации

Текущее состояние настроек локализации проще всего определить с помощью команды locale. Она также интерпретирует LANG и LC_ALL и узнает результирующие

параметры. В следующем примере показаны настройки, действующие на моем компьютере:

```
user$ locale
LANG=de_DE.UTF-8
LC_CTYPE="de_AT.UTF-8"
LC_NUMERIC="de_AT.UTF-8"
LC_TIME="de_AT.UTF-8"
...
LC_ALL=
```

Чтобы протестировать локализацию, можно также выполнить с ошибкой любую команду. Сообщение об ошибке должно быть выведено на том языке, который был настроен. Если настройкой LANG является de_DE, то сообщение об ошибке выполнения команды mount должно выглядеть как в следующем примере (у меня эти испытания успешно прошли только в SUSE; в других дистрибутивах сообщения об ошибках mount выводились лишь на английском языке).

```
user$ mount /xy
mount: Konnte /xy nicht in /etc/fstab oder /etc/mtab finden
```

Команда env

Если вы хотите выполнить отдельную команду с применением настроек другого языка, не изменяя всю конфигурацию, то лучше всего воспользоваться командой env. Она ожидает присвоения отдельных переменных и, наконец, самой команды, которая должна быть выполнена с учетом настроенных переменных:

```
user$ env LANG=C mount /xy
mount: can't find /xy in /etc/fstab or /etc/mtab
```

Если настройка LANG изменена, а сообщение об ошибке все еще выводится не по-английски, попробуйте также сбросить настройки LANGUAGE:

```
user$ env LANG=C LANGUAGE=C mount /xy
mount: can't find /xy in /etc/fstab or /etc/mtab
```

Чтобы настроить LANG для всей сессии целиком, выполните команду export LANG=C.

Допустимые настройки LC/LANG

Список всех возможных настроек для переменных можно узнать с помощью команды locale -a. Обычно используется запись вида *x_y*, причем *x*, записываемый двумя буквами, означает язык, а *y*, также записываемая двумя буквами, означает страну. В немецкоязычном регионе следует использовать настройки de_DE. Английская стандартная настройка может записываться кратко — C.

Новые версии glibc понимают и такие настройки, как deutsch или german. В файле /usr/share/locale/locale.alias содержится таблица, в которую занесены допустимые сокращенные записи и соответствующие им полные локализационные названия.

Пакеты локализации

Для того чтобы меню, диалоговые окна, сообщения об ошибках, тексты справки и т. д. отображались на нужном языке, следует установить необходимые пакеты локализации. Ради экономии места я опишу здесь только два языка — английский и немецкий. Если вы хотите настроить для своего дистрибутива, например, русский язык, вам следует установить соответствующие дополнительные пакеты для Gnome, KDE, OpenOffice, Firefox и т. д. При работе со SUSE и Ubuntu вам помогут конфигурационные инструменты, перечисленные в начале главы, а в других дистрибутивах установка выполняется вручную.

Не каждая программа Linux локализована для всех языков. Особенно недостает переводов электронной документации (страниц `man`, пособий, справочных систем). Если подходящих файлов локализации нет, то Linux отображает информацию по-английски.

Настройка кодировки. Вместе с локализацией настраивается и кодировка. Она указывается после кода страны, отделяется от этого кода точкой; например: `de_DE.ISO-8859-1` или `de_DE.utf8`.

9.6. Справка по аппаратным компонентам

В этой книге нет отдельной главы, посвященной аппаратному обеспечению. Однако правильная конфигурация оборудования рассматривается в тематически родственных главах. Иначе говоря, если у вас возникнут проблемы с сетевой картой, почитайте об этом в главе 16, где рассматривается конфигурация сети.

Итак, в этом разделе решаются две задачи: во-первых, систематизированная справка должна облегчить вам поиск дальнейшей информации об определенных аппаратных компонентах. Во-вторых, дается краткая информация об оборудовании, которое будет упоминаться в оставшейся части книги.

Разумеется, есть множество аппаратных компонентов, которые по различным причинам *не получится* описать в этой книге. Просто у меня нет возможности провести все те разнообразные тесты, которые может себе позволить, например, компьютерный журнал.

СОВЕТ

Перед покупкой оборудования узнайте, совместим ли приобретаемый вами компонент с Linux! Для этого недостаточно спросить продавца — он может просто не знать ответа на подобный вопрос или дать вам неверную информацию. Посмотрите сайты по аппаратному обеспечению в Linux. Попробуйте поискать в Интернете по запросу вида «linux название модели». Вы неизбежно столкнетесь с какой-нибудь статьей, в которой пользователи интересуются, как применить определенное аппаратное обеспечение в Linux. Эти вопросы, естественно, обсуждаются в специальных журналах, ориентированных на Linux, и информация в них актуальнее, чем в книгах.

Файлы-устройства. Основные аппаратные компоненты запрашиваются через так называемые файлы-устройства, например `/dev/sda` для жесткого диска SATA. При этом файлы-устройства динамически создаются системой udev. Список важнейших файлов-устройств Linux приводится в разделе 3.10.

Proc- и sys-файлы. Файловые системы дают подробную информацию по многим аппаратным компонентам в каталогах `/proc` и `/sys`. Обзор таких файлов представлен в разделе 15.3.

Модули ядра. Драйверы к многочисленным аппаратным компонентам находятся в модулях ядра. Часть этих модулей загружается при запуске системы, а оставшиеся — только при необходимости. Если автоматическая загрузка модулей не работает, посмотрите файлы `/etc/mmodprobe.conf` и `/etc/modprobe.conf.d/*`. Работа с модулями и функциональность этих файлов рассматриваются в главе 15.

Обзор аппаратного обеспечения. Чтобы получить обзор действующего аппаратного обеспечения, выполните команды `lshal`, `lspci` и `lsusb`. Кроме того, не помешает просмотреть сообщения ядра с помощью команды `dmesg`.

Процессор и память

Процессор. В этой книге рассматриваются только однопроцессорные компьютеры, совместимые с Intel-Pentium, то есть 32-битные и 64-битные процессоры Intel и AMD. Есть еще и версии Linux для процессоров со всевозможными иными вариантами архитектуры (например, для PowerPC).

О том, какие процессоры работают у вас на компьютере, вы можете узнать, просмотрев файл `/proc/cpuinfo`. Следующий сильно сокращенный вывод получен при проверке компьютера с процессором Intel-Dual-Core — это мой компьютер. Linux воспринимает оба ядра как самостоятельные процессоры. При этом в строке `model name` указывается максимальная тактовая частота, а в строке `cpu MHz` — текущая тактовая частота.

```
user$ cat /proc/cpuinfo
processor    : 0
model name   : Intel(R) Core(TM)2 CPU 6600 @ 2.40GHz
cpu MHz      : 1596.000
...
processor    : 1
model name   : Intel(R) Core(TM)2 CPU 6600 @ 2.40GHz
cpu MHz      : 1596.000
...
```

В процессорах с переменной тактовой частотой предусмотрены модули `cpufreq`, отвечающие за временное снижение частоты (в целях энергосбережения). Базовая информация по этой системе содержится в документации ядра по адресу http://www.kernel.org/doc/Documentation/cpu-freq/.

Информация о текущем состоянии системы и о возможностях управления содержится в файлах следующего каталога: `/sys/devices/system/cpu/cpun/cpufreq/`.

Оперативная память (RAM). Информация об имеющейся в распоряжении памяти выдается при введении команды `free`. Если вам кажется, что у компьютера возникли аппаратные проблемы с оперативной памятью (неисправные модули памяти), воспользуйтесь программой Memtest86 — это отличный инструмент для тестирования памяти. Если она у вас не заработает, скачайте на сайте http://www.memtest86.com/ ISO-образ, чтобы записать загрузочный диск.

Управление энергопотреблением

ACPI

Аббревиатура ACPI означает «улучшенный интерфейс для конфигурации и управления электропитанием» (Advanced Configuration and Power Interface). Он предназначен управления функциями энергопотребления в имеющихся на рынке ПК и ноутбуках и используется примерно с 1999 года. Ранее применялся стандарт APM (автоматическое управление питанием). ACPI поддерживается в Linux, начиная с версии ядра 2.6. Необходимые модули ядра загружаются автоматически — можете в этом убедиться, выполнив команду `dmesg | grep ACPI`. Одновременно запускается процесс ядра `kacpid` и демон ACPI acpid. Обе программы обрабатывают события ACPI. Демон `acpid` управляется файлами, находящимися в каталоге `/etc/acpi`.

Команда `acpi -V` и файлы каталога `/proc/acpi` содержат информацию о текущем состоянии системы ACPI (загруженность батареи, ее температура и т. д.).

Если ACPI вызывает проблемы при запуске, обратитесь к разделу 14.10, где описаны некоторые параметры ядра, позволяющие полностью или частично отключить ACPI. Прочая информация по ACPI и Linux содержится по адресу http://www.lesswatts.org/projects/acpi/.

Suspend

В ACPI поддерживается несколько разновидностей спящего режима, при которых компьютер потребляет мало энергии (режим stand-by) либо вообще не потребляет ее (сон, гибернация, ждущий режим). В большинстве дистрибутивов и локальных систем компьютер переводится в нужный спящий режим с помощью команд меню, например Система ▸ Выключение.

При работе с ноутбуком наиболее интересен режим «гибернация». При его включении содержимое оперативной памяти записывается в специальный своп-раздел на жестком диске и компьютер полностью выключается. Предполагается, что такой раздел должен быть достаточно велик!

При активизации данные оперативной памяти снова считываются с жесткого диска. Кроме того, заново инициализируются все аппаратные компоненты. Этот процесс очень сложен и требует безукоризненного взаимодействия ядра Linux, его модулей и системы ACPI.

ВНИМАНИЕ

Я предупреждаю, что мой опыт работы с различными спящими режимами был в основном горьким. Если при попытке активизации в системе возникала какая-нибудь ошибка, вывести компьютер из спящего режима уже не удавалось. Приходилось нажимать Reset либо вручную выключать и снова включать компьютер. Поэтому внимательно протестируйте все функции, связанные со спящим режимом. Перед этим сохраните все данные, выполните sync и отключите от дерева каталогов не нужные в данный момент файловые системы!

Меры по экономии энергии

В последние годы в сообществе разработчиков Linux прилагается масса усилий для того, чтобы минимизировать затраты энергии и увеличить длительность работы

ноутбука на аккумуляторе. Хороший обзор распространенных методов и соответствующих инструментов дается на следующем сайте: http://www.lesswatts.org/projects/.

Многие операции, описанные на этой странице, уже вошли в состав ядра, стали стандартными настройками современных дистрибутивов и включены в инструменты управления энергопотреблением KDE и Gnome.

Powertop

Команда powertop полезна при поиске программ, выводящих процессор из состояния бездействия. Одновременно программа сообщает, как можно минимизировать потребление энергии.

```
root#  powertop
PowerTOP version 1.11   (C) 2007 Intel Corporation
Cn                            Задержка       P-States   (Частоты)
C0 (Процессор работает)      ( 1,8%)       1,80 GHz       0,0%
циклический AbfraC1 остановка   0,0m       1,60 GHz       0,0%
C1 остановка          0,0ms  ( 0,0%)       1400 MHz       0,0%
C2                   17,7ms  (98,2%)       1200 MHz       0,0%
C3                    0,0ms  ( 0,0%)        600 MHz     100,0%
C4                    0,0ms  ( 0,0%)
Активизаций в секунду: 55,3     Интервал: 3,0s
Энергопотребление (Оценка ACPI): 20,1W (1,2 Std.)
Наиболее частые причины активизации:
23,0% (15,7)   <interrupt>       : ehci_hcd:usb1, uhci_hcd:usb2, uhci_hcd:usb3,
21,6% (14,7)  USB-устройство 3-2 : Optical USB Mouse (Logitech)
13,7% ( 9,3)         <interrupt> : extra timer interrupt
12,7% ( 8,7)       <Kernel Kern> : usb_hcd_poll_rh_status (rh_timer_func)
 9,8% ( 6,7)       <Kernel Kern> : hrtimer_start (tick_sched_timer)
 3,9% ( 2,7)      gnome-terminal : schedule_hrtimeout_range (hrtimer_wakeup)
Предложение: Активизируйте "USB autosuspend", нажав клавишу U,
либо задайте параметр загрузки "usbcore.autosuspend=1" в командной строке ядра,
либо укажите этот параметр в конфигурации GRUB.
```

Инструмент powertop — это то, что вам нужно, если вы специально хотите найти способы, которые помогли бы продлить срок службы вашего ноутбука.

Режим ноутбука

Это такая функция ядра, призванная свести к минимуму энергопотребление ноутбука, работающего от аккумуляторной батареи. Основная функция режима заключается в том, чтобы осуществлять запись данных на диск не сразу, а сохраняя в кэш. Синхронизация данных происходит лишь с интервалом несколько минут (если только не требуется одновременно сохранить очень много данных). В периоды между сохранениями информации жесткий диск можно перевести в энергосберегающий режим. Этот режим активизируется следующей командой:

```
root# echo 5 > /proc/sys/vm/laptop_mode
```

Обратите внимание, что если часто включать и выключать машину, срок службы жесткого диска может снизиться (см. также http://lwn.net/Articles/257426/). Не ис-

пользуйте режим ноутбука при работе с ПК, так как винчестеры ПК обычно рассчитаны на менее частое включение и выключение. Кроме того, необходимо сознавать, что если электричество отключится, вы потеряете все несохраненные данные.

Как правило, режим ноутбука управляется сценариями и конфигурационными файлами пакета `laptop-mode-tools`. Кроме того, этот пакет позволяет проводить и многие другие операции по сбережению энергии. Конфигурация осуществляется в каталоге `/etc/laptop-mode` (см. `laptop-mode.conf`).

В Ubuntu пакет `laptop-mode-tools` установлен по умолчанию, но не активизирован. Для активизации этого пакета измените строку в `/etc/default/acpi-support`:

```
# Изменение в /etc/default/acpi-support
ENABLE_LAPTOP_MODE=true
```

Дополнительную информацию по режиму ноутбука и инструментам `laptop-mode-tools` вы найдете по адресу http://samwel.tk/laptop_mode/.

Интерфейсы и системы шин

Последовательные и параллельные интерфейсы. В Linux последовательные и параллельные интерфейсы доступны через файлы-устройства `/dev/ttySn` или `/dev/lpn`. Скорее всего, вы сталкивались с такими устаревшими интерфейсами при конфигурации аналогового модема или старого принтера.

IDE, SATA, SCSI. Внутренние жесткие диски, приводы CD или DVD, а также многие другие носители данных, как правило, подсоединяются к компьютеру через системы шин IDE, SATA или SCSI (раздел 13.2). Современные версии Linux сообщаются с устройствами IDE, SATA или SCSI через SCSI-систему ядра. Лишь некоторые DIE-контроллеры, несовместимые с расширением libata системы SCSI, все еще используют старые драйверы IDE.

Информация о состоянии систем IDE и SCSI и всех связанных с ними устройств выводится командой `lsscsi`, а также содержится в следующих файлах:

- `/sys/bus/ide/*`;
- `/sys/bus/scsi/*`;
- `/proc/scsi/*`.

USB. *Универсальная последовательная шина* используется для связи компьютера с самыми разными внешними устройствами: от мыши до сканера. Необходимые для работы USB модули ядра загружаются автоматически. USB-носители (внешние винчестеры, флешки, внешние CD- и DVD-приводы) воспринимаются системой как SCSI-устройства.

Виртуальная файловая система usbfs сообщает информацию обо всех подключенных USB-устройствах в каталоге `/proc/usb`. Затем можно обратиться к данным, содержащимся в каталоге `/sys/bus/usb`. Подробный список всех USB-интерфейсов и устройств возвращает команда `lsusb -v` (пакет `usbutils`).

Firewire. Система шин Firewire — альтернатива USB. Firewire определяется стандартом IEEE 1394 и известна также под названием i.Link, используемым фирмой Sony. Firewire действует несколько быстрее, чем USB, и особенно популярна

для передачи данных с видеокамеры. Подробная информация о IEEE 1394 и связи этого стандарта с Linux находится на сайте http://www.linux1394.org/.

При подсоединении устройств Firewire автоматически загружаются требуемые модули, в частности ieee1394. Информация о подсоединенных устройствах и о состоянии системы Firewire содержится в файлах каталога /sys/bus/ieee1394.

PCI. Информацию о компонентах PCI, установленных на вашем компьютере, можно узнать, введя команду lspci. Файлы в каталогах /proc/bus/pci/ и /sys/bus/pci/ содержат ту же информацию, но они гораздо менее удобны для интерпретации. Следующий вывод ради экономии места сильно сокращен:

```
root# lspci
00:00.0 Host bridge: Intel Corporation 82P965/G965 Memory Controller Hub
00:01.0 PCI bridge: Intel Corporation 82P965/G965 PCI Express Root Port
00:1a.0 USB Controller: Intel Corporation 82801H (ICH8 Family) USB UHCI #4
00:1b.0 Audio device: Intel Corporation 82801H (ICH8 Family) HD Audio Controller
00:1f.0 ISA bridge: Intel Corporation 82801HB/HR (ICH8/R) LPC Interface Controller
00:1f.2 IDE interface: Intel Corporation 82801H (ICH8 Family) 4 port SATA IDE
        Controller
00:1f.3 SMBus: Intel Corporation 82801H (ICH8 Family) SMBus Controller
01:00.0 VGA compatible controller: nVidia Corporation G70 [GeForce 7600 GS]
02:00.0 Ethernet controller: Marvell Technology Group Ltd. 88E8056 PCI-E Gigabit
        Ethernet Controller (rev 12)
05:03.0 FireWire (IEEE 1394): Texas Instruments TSB43AB22/A IEEE-1394a-2000
Controller
```

PCMCIA. Мудреное сокращение PCMCIA означает «Международная ассоциация по разработке стандарта плат памяти персональных компьютеров». Карты PCMCIA размером не больше кредитки и представляют собой расширительные платы для ноутбуков, причем речь идет не только о картах памяти. Это значительно более широкий спектр устройств. Для большинства пользователей Linux в работе особенно важны платы WLAN-, UMTS- и ISDN.

Существует два типа плат PCMCIA — так называемые ПК-платы (старые) и карты CardBus (новые). Разница заключается в том, что передача данных между платой и ноутбуком в первом случае происходит через 16-битную шину, а во втором — через 32-битную шину. Linux поддерживает платы обоих типов.

Совместимые с Linux платы PCMCIA опознаются системой горячего подключения и автоматически инициализируются. Команда pccardctl возвращает информацию о подключенных платах PCMCIA. Поскольку платы, как правило, запрашиваются через шину PCI, программа часто ссылается на lspci. С помощью pccardctl можно управлять и некоторыми функциями PCMCIA, позволяющими, например, отключать платы PCMCIA от источника питания.

Сетевые интерфейсы. Подробная информация о конфигурации интерфейсов WLAN и LAN, а также о работе с модемами, сообщается в главе 16.

Bluetooth. Это технология обмена информацией с устройствами по беспроводной связи. Bluetooth не слишком распространен в качестве WLAN и используется преимущественно с сотовыми телефонами, КПК и другими мелкими электронными устройствами. Обычно Linux не составляет труда разобраться

с устройствами, использующими Bluetooth, и каких-либо сложных конфигурационных работ не требуется. Более подробная информация приводится на сайте http://www.bluez.org/.

Графика (X). Графические карты используются в Linux системой X Window. Их конфигурации посвящена глава 12.

Система горячего подключения

udev, D-Bus и HAL

К современным компьютерам можно подключать (а также извлекать из них) жесткие диски, флешки, платы расширения и другие устройства, не выключая при этом компьютер. Linux должна реагировать на изменение ситуации быстро и, по возможности, автоматически. Эту функцию выполняет *система горячего подключения*, компоненты которой в последние годы все время меняются. Примерно с 2006 года в большинстве дистрибутивов горячее подключение осуществляется таким образом.

1. Ядро обнаруживает изменения в аппаратном обеспечении, означающие, что пользователь, к примеру, вставил компакт-диск в привод или флешку в компьютер.
2. Ядро вызывает `/sbin/hotplug`, чтобы загрузить необходимые модули.
3. С помощью `udev` ядро создает новые файлы-устройства (см. раздел 3.10).
4. Программа `udev` указывает новые устройства на уровне аппаратных абстракций (HAL) (процесс ядра `hald`). Список всех аппаратных средств, известных HAL, возвращает команда `lshal`.
5. Программа `hotplug` также сообщает процессу `hald` о событиях, связанных с аппаратным обеспечением.
6. Процесс `hald` обновляет конфигурацию системы и через шинную систему обмена информацией D-Bus передает соответствующие сведения всем программам, которым это требуется.

   D-Bus — это система для обмена уведомлениями между программами. На основе библиотеки `libdbus` обмен информацией может происходить прямо между двумя программами. Если необходимо, чтобы программы обменивались уведомлениями, в качестве центральной точки коммутации применяется служебная программа `dbus-daemon`.
7. В KDE-4 для обработки уведомлений от HAL и D-Bus используется фреймворк устройств Solid. В Gnome версии 2.22 и выше внешними носителями данных занимается Nautilus вместе с PolicyKit (см. раздел 4.4); конфигурация производится в разделе Правка ▸ Настройки ▸ Медиа.

Для того чтобы конфигурация HAL, D-Bus и `udev` была правильной, очень важно точно настроить каталоги `/etc/hal`, `/etc/dbus-1`, `/etc/udev` и `/lib/udev/rules.d`. Поскольку это очень сложно, дистрибутив сам подскажет вам, как правильно сконфигурировать систему горячего подключения. Если возникнут проблемы,

изменить конфигурацию сможет только профессионал по Linux. Более подробная информация есть на следующих сайтах:

- http://webcvs.freedesktop.org/hal/hal/doc/spec/hal-linux26.png?view=co;
- http://www.freedesktop.org/wiki/Software/hal;
- http://www.reactivated.net/writing_udev_rules.html.

DeviceKit

Многим разработчикам Linux HAL не нравится, поэтому планируется заменить HAL на DeviceKit и libudev. DeviceKit уже встроен в современные дистрибутивы, но имеется и HAL. Пока неясно, когда появится первый дистрибутив, который бы полностью обходился без HAL.

DeviceKit, как и HAL, строится на `udev` и D-Bus и состоит из нескольких частей. Достаточно современными и хорошо продуманными уже являются инструменты DeviceKit-disks для управления жесткими дисками и их сегментами, а также DeviceKit-power для управления энергопотреблением. Поскольку DeviceKit не предназначен для конкретных пользователей, он не включает каких-либо «видимых» программ и руководств по конфигурации. Замена HAL на DeviceKit в значительно большей мере касается разработчиков пользовательских программ, ведь специалистам понадобится заново интегрировать функции для доступа к аппаратному обеспечению. Более подробная информация приводится на таких сайтах:

- https://wiki.ubuntu.com/Halsectomy;
- http://hal.freedesktop.org/docs/DeviceKit/;
- http://fedoraproject.org/wiki/Features/DeviceKit;
- http://lists.freedesktop.org/archives/hal/2008-May/011560.html.

Аудиосистема (ALSA)

Аббревиатура ALSA означает Advanced Linux Sound Architecture (продвинутая звуковая архитектура Linux). Она присутствует в ядре, начиная с версии 2.6, и отвечает за управление звуковыми картами на самом нижнем уровне. В более ранних версиях ядра для управления звуковыми картами применялась OSS — Open Sound System. На всякий случай ALSA содержит модули `snd-pcm-oss`, `sndseq-oss` и `snd-mixer-oss`, обеспечивающие совместимость с OSS.

Аудиоконтроллер, поддерживаемый ядром, автоматически загружает нужный модуль ALSA. Названия модулей ALSA начинаются с `snd`. При этом команда `lsmod grep snd` дает краткий обзор всех активных модулей ALSA. Доступ к различным звуковым функциям осуществляется через файлы, находящиеся в каталоге `/proc/asound`.

Конфигурация системы ALSA производится в файлах `/etc/alsa/*`, `/etc/asound.conf`, а также `~/.asoundrc`. Если вы собираетесь использовать аудиосистему как обычно, то менять эти файлы не нужно. Аппаратура должна опознаваться автоматически. Если у вас есть особые требования к аудио (скажем, вы музыкант), то вам нужно, чтобы система различала несколько звуковых карт. Если же у вас есть еще какие-то особые пожелания, обратитесь к следующему сайту и связанному с ним

источнику — здесь вы найдете подробную базовую информацию по ALSA и ее конфигурации: http://www.alsa-project.org/.

В случае остановки компьютера или при перезапуске сценарий Init-V (см. раздел 14.10) сохраняет или восстанавливает имеющиеся настройки громкости.

Инструментарий ALSA

Для непосредственного использования ALSA предусмотрено несколько команд (пакет alsa-utils), важнейшие из которых я коротко представлю: `alsactl` сохраняет или загружает все настройки ALSA (громкость звука и т. д.); `alsamixer` изменяет громкость или входной уровень сигнала различных аудиоканалов ALSA; `aplay` воспроизводит аудиофайл; `arecord` записывает аудиофайл.

СОВЕТ

Если колонки молчат, то причина может быть всего лишь в том, что регулятор громкости стоит на нуле. При обычной работе с компьютером важны три канала: ведущая громкость регулирует громкость сигнала в целом; PCM-громкость указывает, какой силы должен быть сигнал аудиофайлов, созданных аудио- и видеопроигрывателями (в данном случае PCM означает «импульсно-кодовая модуляция»); наконец, CD-громкость указывает, как будут вливаться в аудиопоток данные, считываемые непосредственно с CD, если CD-привод и звуковая карта соединены кабелем.

В современных дистрибутивах иногда отсутствуют графические пользовательские интерфейсы, позволяющие по отдельности настраивать сигнал на входах и выходах аудиоустройства. Что делать? Запустите alsamixer в текстовой консоли. Теперь с помощью клавиш для управления курсором можно выбирать каналы и настраивать уровни сигнала для них. M (mute) включает или выключает канал.

Аудиобиблиотеки

Многие аудиопрограммы работают с ALSA не напрямую, а обращаются к звуковым библиотекам, звуковым серверам и т. д. Этот промежуточный «слой» между низкоуровневой системой ALSA и самими аудиоприложениями упрощает программирование, приспосабливает эти приложения к работе в сети и гарантирует бесконфликтное выполнение работающих одновременно аудиопрограмм.

Загвоздка лишь в том, что пока не существует единой аудиоархитектуры «выше» системы ALSA. KDE и Gnome работают с системой каждый по-своему. Сложные аудиоприложения, для коротых недостаточно имеющихся библиотек, сами внедряют простейшие аудиофункции. Разработать аудиопрограмму, которая бы функционировала самостоятельно, независимо от локального компьютера, исключительно сложно. Специалисты фирмы Adobe сообщили, что разработка Flash-плагина для Linux сильно затянулась во многом из-за того, что имелись серьезные проблемы с аудио.

Далее я кратко представлю некоторые распространенные аудиосистемы.

- **aRts.** Это означает «аналоговый синтезатор реального времени» — основная аудиопрограмма в KDE 2.n и 3.n. aRts, состоит из нескольких модулей, создающих, объединяющих и фильтрующих аудиофайлы. Программы KDE запрашивают aRts с помощью демона artsd, который запускается вместе с KDE. Программы, несовместимые с aRts и поэтому обращающиеся напрямую к устройствам звуковых модулей ядра, автоматически получают переадресацию с помощью

artsdsp. Так или иначе техническая поддержка aRts уже прекращена и в версии KDE 4 этой программы уже не будет.

- **EsounD.** Название означает «Просветленный Звуковой Демон» — это была аналогичная aRts программа из Gnome. Совместимые с EsounD программы посылают аудиофайлы демону esd, запускаемому вместе с Gnome. Судьба EsounD, как и судьба aRts, сложилась незавидно: EsounD все еще устанавливается вместе с Gnome ради совместимости, но большинство аудиоприложений уже перешли к использованию GStreamer.
- **GStreamer.** Данная библиотека — широкий мультимедийный фреймворк (аудио и видео), используемый многими программами Gnome. Благодаря использованию плагинов архитектура программы получилась модульной и рассчитана на значительную расширяемость. В виде плагинов предоставляются и кодеки для обработки различных аудио- и видеоформатов. В отличие от aRts и EsounD, библиотека GStreamer обходится без звукового демона. В данном случае основную работу демона, то есть сведение нескольких звуковых сигналов, теперь выполняет непосредственно ALSA. Более подробная информация представлена на сайте http://www.gstreamer.net/.
- **Phonon.** Мультимедийный фундамент KDE 4 называется Phonon. Эта библиотека содержит унифицированный интерфейс программирования приложений (API) для работы с функциями аудио и видео, который обращается к имеющимся мультимедийным библиотекам (обычно это Xine, но в зависимости от того, какой внутренний интерфейс установлен, это могут быть и GStreamer или VLC). Phonon также применяется библиотекой Qt в качестве мультимедийного интерфейса. Более подробная информация сообщается на сайте Phonon: http://phonon.kde.org/.
- **PulseAudio.** Это звуковой сервер с поддержкой работы в сети, функционально схожий с программой `esd` и заменяющий ее в современных дистрибутивах. На первый взгляд это незаметно — все должно работать как раньше. Однако «за кулисами» выполняется масса работы: все аудиопотоки могут отдельно управляться программой `pavucontrol` и присваиваться разным звуковым картам и устройствам вывода. Кроме того, PulseAudio должен автоматически распознавать и активизировать дополнительные аудиоустройства, например внешние USB-дисководы. Воспроизведение аудио функционирует в целом хорошо, однако PulseAudio уже несколько лет производит впечатление «запущенного долгостроя», обрастая все новыми конфигурационными инструментами и проблемами. Более подробная информация приводится на следующих сайтах:
  - http://pulseaudio.org/;
  - http://pulseaudio.org/wiki/FAQ.

Наряду с этими аудиосистемами стоит упомянуть несколько программ, которые сами содержат большие аудио- и мультимедийные библиотеки и предоставляют эти библиотеки в пользование другим программам. Наиболее известны RealPlayer, или Helix Player на основе Helix DNA, а также видеопроигрыватель Xine на основе xinelib. Неудивительно, что многочисленные несовместимости между различны-

ми аудиопрограммами и библиотеками буквально предопределены. Другие причины, по которым аудиосистемы Linux оставляют желать лучшего, обобщены в статье *Audio: it's a mess* (по состоянию на сентябрь 2008 года): http://lwn.net/Articles/299211/.

Для музыкантов или людей, профессионально работающих с аудио, существуют дистрибутивы, специально разработанные для оптимального и безотказного использования аудиопрограмм:

- http://64studio.com/;
- http://ubuntustudio.org/.

9.7. Журналирование

Ядро, различные инструменты администрирования (PAM, APT, `dpkg`) и большинство сетевых служб протоколируют любые мыслимые события в многочисленных файлах, расположенных в каталоге `/var/log`. Эти файлы регистрации событий исключительно удобны при запуске в эксплуатацию новой службы, так как помогают обнаружить ошибки конфигурации.

Программы sysklogd и klogd

Для того чтобы каждой программе не пришлось изобретать велосипед, ядро и некоторые инструменты администрирования обращаются к центральным функциям журналирования. Во многих дистрибутивах для этого применяются программы `sysklogd` и `klogd`, которые объединяются термином Syslog.

- Программа `syslogd` из пакета `sysklogd` предоставляет функции журналирования для работы с внешними программами.
- Программа `klogd` отвечает за уведомления ядра, но для сохранения информации обращается к `syslogd`.

Обратите внимание — отнюдь не все сетевые службы пользуются Syslog! В частности, «крупные» серверные службы, например Apache, CUPS и Samba, применяют собственные функции журналирования, интегрированные в программу. Таким образом, эти программы не поддаются глобальной конфигурации, настраиваемой в `syslog.conf`. Параметры журналирования настраиваются в соответствующих конфигурационных файлах конкретной программы. Если вы хотите составить общее представление о файлах регистрации, управляемых Syslog, выполните следующие команды:

```
root# cd /var/log
root# ls $(find -user syslog)
```

Как правило, `syslogd` входит в состав пакета `sysklogd`. Это связано с тем, что `sysklogd` является новой версией демона BSD-Syslog. Дополнительная буква «К» указывает на тесную интеграцию с демоном журнала ядра `klogd`.

Однако `sysklogd` ни в коем случае не единственный демон журналирования во всей Linux. Кроме него используются `dsyslog`, `rsyslog` (последний применяется

в Fedora) и syslog-ng (применяется в SUSE). Но ради экономии места здесь будет рассмотрен только sysklogd.

Конфигурация

Файл /etc/syslog.conf определяет, какие уведомления куда заносятся. Он содержит правила, состоящие из двух частей: первая часть (селектор) указывает, какую информацию следует зарегистрировать, а вторая часть (действие) определяет, что должно произойти с сообщением. Правила можно разбивать на несколько строк с помощью символа \. Возможно, что к одному сообщению будут применяться несколько правил. В таком случае сообщение будет зарегистрировано или передано несколько раз.

Каждый селектор состоит из двух частей, разделенных точкой: *служба.уровень_приоритета*. Можно указывать сразу несколько селекторов, разделяя их точками с запятой. Кроме того, можно объединять *в одном селекторе* несколько служб, разделяя их запятыми. Все программы Linux, использующие Syslog, должны присваивать своим сообщениям службу и приоритет.

Syslog известны следующие службы: uth, authpriv, cron, daemon, ftp, kern, lpr, mail, news, syslog, user, uucp, а также службы от local0 до local7. Символ * объединяет все службы.

Кроме того, Syslog известны следующие степени приоритета (по нарастанию важности): debug, info, notice, warning = warn, error = err, crit, alert и emerg = panic. Ключевые слова warn, error и panic считаются устаревшими — используйте вместо них warning, err и emerg. Символ * объединяет все степени приоритета. Ключевое слово none используется с уведомлениями, которым не присвоен никакой приоритет.

При указании степени приоритета учитываются также все степени, находящиеся в иерархии выше. Иными словами, селектор mail.err охватывает и все сообщения почтовой системы уровней crit, alert и emerg. Если вам требуются уведомления только с определенной степенью приоритета, поставьте перед этой степенью символ = (то есть, например, mail.=err).

В качестве *действия* чаще всего указывается имя одного из файлов регистрации. Обычно файлы регистрации синхронизируются после каждого вывода. Если перед именем файла стоит символ -, Syslog не производит синхронизации. Такой метод работы более эффективен, но при аварийном завершении работы системы будут потеряны сообщения, еще не сохраненные физически.

Syslog также может переадресовывать уведомления к файлам FIFO (первым пришел — первым обслужен) или к программным каналам. В таком случае перед именем файла ставится символ |. Файл /dev/xconsole, показанный в следующем листинге, — это особый файл типа «FIFO», предназначенный для передачи уведомлений графической системе X.

Символ * означает, что уведомление переадресовывается на все консоли или всем пользователям, вошедшим в систему через SSH. Поскольку это очень неудобно, по умолчанию он используется только с критически важными сообщениями.

Более подробная информация по синтаксису syslog.conf содержится на одноименной странице справки man. В следующих строках приводится слегка сокращен-

ная стандартная конфигурация Syslog в Ubuntu, немного отформатированная для удобства:

```
# Файл /etc/syslog.conf в Ubuntu
# Селектор               Действие
auth,authpriv.*          /var/log/auth.log
*.*;auth,authpriv.none  -/var/log/syslog
daemon.*                -/var/log/daemon.log
kern.*                  -/var/log/kern.log
lpr.*                   -/var/log/lpr.log
user.*                  -/var/log/user.log
*.=debug;\
auth,authpriv.none;\
news.none;\
mail.none               -/var/log/debug
*.=info;\
*.=notice;\
*.=warn;\
auth,authpriv.none;\
cron,daemon.none;\
mail,news.none          -/var/log/messages
*.emerg *
daemon.*;\
mail.*;\
news.err;\
*.=debug;\
*.=info;\
*.=notice;\
*.=warn                 |/dev/xconsole
```

Рассмотрим, что означает этот код.

- `/var/log/auth` содержит аутентификационные сообщения со всеми степенями приоритета. Сюда относятся как удачные, так и неудачные попытки входа в систему (в том числе по SSH), сообщения PAM, команды `sudo` и т. д. Поскольку единственным файлом регистрации является `auth`, он сразу же синхронизируется при каждом сообщении.
- `/var/log/syslog` содержит все сообщения, зарегистрированные Syslog (в том числе аутентификационные, которым не присвоен приоритет). Такой всеобъемлющий подход и хорош, и плох одновременно. С одной стороны, вы сможете получить из единственного файла любую информацию, которая вам понадобится. С другой стороны, в такой «куче» исключительно сложно найти нужные записи.
- `/var/log/daemon` специально предназначен для работы с уведомлениями, поступающими от демонов (сетевых служб). Все сетевые службы, использующие Syslog, регистрируют в этом каталоге свои сообщения об ошибках. К ним относятся, например, `dhcpd`, `kadmind`, `krb5*`, `mountd`, `named`, `pptpd`, `squid` и т. д. Если при установке новой службы возникают проблемы, то проверить нужно в первую очередь соответствующий демон.

- /var/log/kern.log содержит все сообщения ядра (см. также следующий раздел).
- /var/log/lpr предназначен для работы с сообщениями печатающих устройств. Правда, этот файл обычно пуст, так как система печати CUPS использует собственные функции журналирования и регистрирует информацию в файлах каталога /var/log/cups/.
- /var/log/user предназначен для общих пользовательских уведомлений. На сервере в этом каталоге обычно находится всего несколько малозначимых сообщений.
- /var/log/debug содержит уведомления, связанные с отладкой системы, а также сообщения для служб auth, authpriv, news и mail без указания приоритета.
- /var/log/messages регистрирует общие указания и предупреждения (но не сообщения об ошибках!).
- Критические системные сообщения (например, о близящемся «обвале» системы или об ошибке ядра) переадресовываются на все консоли.
- Различные предупреждения и сообщения об ошибках передаются в систему X. Чтобы отслеживать такие сообщения в X, запустите программу xconsole. Она выглядит как маленькое окно терминала, но вводить в нее информацию нельзя.

Журналирование ядра

Сообщения ядра записываются в кольцевой буфер размером 16 Кбайт, расположенный в оперативной памяти. Когда этот буфер заполнен, старые сообщения удаляются, освобождая место для новых. Содержимое этого буфера можно просмотреть с помощью команды dmesg. Если указать при этом параметр -c, кольцевой буфер будет автоматически опустошаться.

Кроме того, все уведомления ядра записываются в виртуальный файл /proc/kmsg. Он служит для переадресации сообщений ядра в Syslog. Детали этого процесса зависят от конкретного дистрибутива. Ubuntu запускает команду dd, передающую сообщения из /proc/kmsg в файл-конвейер /var/run/klogd/kmsg. Программа klogd считывает оттуда сообщения ядра и передает их далее, к syslogd.

Часто сообщениям ядра (dmesg) предшествует указание времени в форме [nnn.nnnnnn]. Число после запятой указывает количество секунд, истекших с запуска системы, а следующие шесть знаков конкретизирует время вплоть до миллионной доли секунды. При сохранении уведомлений ядра в файле регистрации такой показатель времени обычно дополняется указанием абсолютного времени. Следующие строки взяты из файла /var/log/kern.log дистрибутива Ubuntu:

```
root# less /var/log/kern.log
...
Jul 24 08:42:04 uranus kernel: [2.313612]  brd: module loaded
Jul 24 08:42:04 uranus kernel: [2.313849]  loop: module loaded
Jul 24 08:42:04 uranus kernel: [2.313903]  Fixed MDIO Bus: probed
Jul 24 08:42:04 uranus kernel: [2.313907]  PPP generic driver version 2.4.2
Jul 24 08:42:04 uranus kernel: [2.313950]  input: Macintosh mouse button emulation
                                           as /devices/virtual/input/input2
Jul 24 08:42:04 uranus kernel: [2.313970]  Driver 'sd' needs updating -
```

```
                                        please use bus_type methods
Jul 24 08:42:04 uranus kernel: [2.313976]  Driver 'sr' needs updating -
                                        please use bus_type methods
Jul 24 08:42:04 uranus kernel: [2.314009]  ahci 0000:03:00.0: version 3.0
...
```

Сообщения Init-V

Сообщения процесса Init-V (см. раздел 14.10) регистрируются лишь в некоторых дистрибутивах: в Fedora — в файле `/var/log/boot.log`, в SUSE — в файле `/var/log/boot.msg`.

Программа logrotate

Как правило, рано или поздно накапливается множество файлов регистрации, поэтому без регулярной архивации и удаления старых файлов регистрации каталог `/var/log` станет самым крупным на вашем сервере.

Если хотите держать размер файлов регистрации под контролем, пользуйтесь программой `logrotate`. Она ежедневно вызывается сценарием Cron `/etc/cron.daily/logrotate`. Затем сценарий обрабатывает все файлы регистрации, описанные в конфигурационных файлах каталога `/etc/logrotate.d`. Подробности обработки файлов регистрации программой `logrotate` зависят от конкретной конфигурации программы. Однако принципиально этот процесс протекает одинаково.

1. Программа `logrotate` переименовывает имеющийся архив файлов регистрации. Из *имя*`.4.gz` получается *имя*`.5.gz`, из *имя*`.3.gz` — *имя*`.2.gz` и т. д. Этот процесс называется *ротацией* и присваивает пакету собственное имя.
2. Если количество пакетов с файлами регистрации превышает максимальную заданную величину, то более старые архивные файлы удаляются.
3. Программа `logrotate` переименовывает актуальный файл регистрации. Этот файл теперь называется *имя*`.0`.
4. Программа создает новый пустой файл регистрации «*имя*».
5. При работе с большинством серверных служб `logrotate` с помощью команды `/etc/init.d/`*имя* `reload` требует от демона заново считать конфигурацию. Демон понимает, что появился новый пустой файл регистрации, и работает именно с ним.
6. Программа `logrotate` архивирует *имя*`.0` или *имя*`.1` (параметр `delaycompress`). Указанный параметр позволяет избежать конфликтов с демоном, который, возможно, еще записывает информацию в *имя*`.0`, и командой архивации (обычно это `gz`).

Конфигурация. В `/etc/logrotate.conf` содержатся некоторые установки, действующие для `logrotate` по умолчанию. Эти настройки действительны, если программно-специфичные файлы конфигурации не содержат иных данных.

В `/etc/logrotate.d` содержатся стандартные настройки для различных программ, создающих файлы регистрации. Эти файлы относятся не к пакету Logrotate, а к пакетам соответствующих программ. Так, например, пакет samba расположен в каталоге `/etc/logrotate.d/samba`. Это позволяет гарантировать, что файлы устанавливаемой в настоящий момент программной версии подходят и `logrotate`

информирует каждую службу о переименовании файлов регистрации либо запускает ее заново.

В следующих строках показан пример конфигурационного файла Apache: программа `logrotate` обрабатывает файлы регистрации раз в неделю. Файлы регистрации переименовываются и архивируются. Размер архива ограничен 52 файлами, то есть при необходимости вы можете просмотреть файлы регистрации за целый год.

```
# Файл /etc/logrotate.d/apache
/var/log/apache2/*.log {
  weekly
  missingok
  rotate 52
  compress
  delaycompress
  notifempty
  create 640 root adm
  sharedscripts
  postrotate
    if [ -f "`. /etc/apache2/envvars ; \
      echo ${APACHE_PID_FILE:-/var/run/apache2.pid}`" ]; then
      /etc/init.d/apache2 reload > /dev/null
    fi
  endscript
}
```

Пока Apache работает, команда `reload` сообщает ему, что появились новые файлы регистрации.

10 Управление программами и пакетами

В Windows программы обычно устанавливаются с помощью исполняемого файла `setup.exe`. В Linux применяется совершенно другой подход: система управления пакетами руководит базой данных, в которой содержится информация обо всех уже установленных программных пакетах. Новые программы устанавливаются с помощью команд системы управления пакетами.

Такая концепция имеет много преимуществ: можно учитывать взаимозависимости и конфликты между программными пакетами. Если, к примеру, для работы программы *A* требуется библиотека *B*, то система управления пакетами разрешит установить программу *A* только после того, как вы установите библиотеку *B*. В любое время можно определить, к какому пакету относится тот или иной файл, находится ли еще этот файл в исходном состоянии и т. д.

Форматы пакетов. На рынке Linux доминируют две системы управления пакетами: Red Hat, Fedora, Mandriva, SUSE. Другие многочисленные дистрибутивы используют формат пакетов RPM, разработанный Red Hat (см. раздел 10.1). Debian и другие дистрибутивы, созданные на его основе, применяют, напротив, формат DEB (см. раздел 10.4). Команды для инсталляции, деинсталляции и обновления этих пакетов (`rpm`, `dpkg` и т. д.) в любом случае достаточно примитивны. Например, они позволяют определить недостающие зависимости, но не позволяют решить связанные с этим проблемы.

Системы управления пакетами. На основе `rpm` или `dpkg` были разработаны другие программы с различными дополнительными функциями. К таким функциям относится, в частности, автоматическая установка зависимых пакетов, возможность обновлений для всей системы и учет источников пакетов, находящихся на CD/DVD или в Интернете. Кроме того, система управления пакетами позволяет создать унифицированную систему обновления для *всех* установленных программ. К подобным системам управления пакетами относятся, в частности, Yum и ZYpp — в формате RPM (см. разделы 10.2 и 10.3), а также APT и Aptitude — в формате DEB (см. раздел 10.5). Графические пользовательские пакеты также имеются в изобилии. Популярны программы YumExtender, PackageKit и Synaptic.

Инструменты, специфичные для отдельных дистрибутивов. В качестве дополнения к этим стандартным программам в некоторых дистрибутивах имеются собственные программы для управления пакетами и выполнения обновлений:

- Debian, Ubuntu — `update-manager`;
- Fedora версии 9 и выше, RHEL 6 — `packagekit`;
- Fedora версии 8 и ниже, RHEL 5 — `pirut` и `pup`;
- SUSE — YaST-модули из группы Программное обеспечение;
- Ubuntu — `gnome-app-install`, `gnome-language-selector`.

ВНИМАНИЕ

Пакеты каждого отдельного дистрибутива Linux согласованы друг с другом. Это означает, что они используют одинаковые библиотеки, собираются одним и тем же компилятором и т. д. Поэтому рекомендую всем начинающим работу с Linux устанавливать только те пакеты, которые предназначены специально для вашего дистрибутива.

Установка пакетов, разработанных для других дистрибутивов, может представлять проблему. Так, например, может не получиться установить пакет для Red Hat в SUSE (или наоборот) из-за того, что отсутствуют определенные библиотеки или не выполняются необходимые зависимости между пакетами. Для устранения этих проблем обычно требуются глубокие специальные знания.

Безопасность. В большинстве систем управления пакетами предпринимаются серьезные меры обеспечения безопасности. В частности, пакеты сами по себе и описание источников пакетов снабжаются криптографическими подписями, исключающими манипуляции с ними. Поэтому при создании новых источников пакетов зачастую требуется установить новый ключ. Подробный анализ аспектов безопасности систем управления пакетами различных дистрибутивов делается в следующей статье: http://lwn.net/Articles/326817/.

Централизованное администрирование нескольких компьютеров. Если вы отвечаете за 50, 100 или 1000 компьютеров с Linux, то администрирование и управление пакетами станет для вас мукой, несмотря на все разнообразие инструментов, описанных в этой главе. Вам понадобится инструмент, который позволил бы проводить обновления на всех компьютерах сети или некоторых предварительно выбранных машинах, разрешил установить новую программу или изменить конфигурацию. В зависимости от дистрибутива для этих целей применяются такие программы, как Red Hat Network, ZENworks (Novell/SUSE) или m23 (Debian, Ubuntu):

- http://www.redhat.com/rhn/;
- http://www.novell.com/de-de/products/zenworks/configurationmanagement/;
- http://m23.sourceforge.net/.

Управление пакетами с помощью tar. Команда `tar` позволяет собрать множество файлов в одном архиве либо распаковать такой архив. На заре Linux, когда еще не было пакетов в форматах RPM и DEB, в большинстве дистрибутивов вместо пакетов использовались TAR-архивы. Сегодня осталось совсем мало дистрибутивов, в которых используются TAR-архивы, а не пакеты (например, Slackware).

И все же TAR-архивы еще играют важнейшую роль в повседневном труде опытных пользователей Linux. Многие разработчики программ, не желающие создавать пакеты RPM или DEB, просто предоставляют вместо этого обычные TAR-архивы

со всеми необходимыми файлами. Установить такие файлы не сложнее, чем обычный пакет (`tar czf имя.tgz`), но при этом программа ускользает от системы управления пакетами конкретного дистрибутива, ее сложно обновлять, нельзя деинсталлировать и к тому же она может вызывать конфликты. Рекомендую устанавливать TAR-пакеты лишь в тех случаях, когда вы знаете, что делаете, и когда желаемая программа недоступна ни в какой другой форме.

Другие форматы пакетов. Кроме рассматриваемых в этой книге пакетов RPM и DEB существуют некоторые другие форматы, которые, правда, пока не приобрели существенного значения. К важнейшим целям, которых разработчики пытаются достичь с помощью новых пакетов, относится обеспечение более тесной связи с дистрибутивом, а также возможность установки пакетов или программ в локальные пользовательские каталоги, не требующая при этом наличия прав администратора.

Более подробную информацию можно найти на следующих сайтах:

- http://labix.org/smart;
- http://autopackage.org/;
- http://0install.net/.

10.1. Управление пакетами RPM

Команда `rpm` позволяет установить RPM-пакеты и управлять ими. Она помогает, в частности:

- в рамках установки автоматически вносить изменения в уже имеющиеся файлы (например, в сценарные файлы);
- заменить версию программы более новой версией (причем измененные файлы автоматически обновляются);
- снова удалить все файлы определенной программы;
- проверить, был ли изменен файл с момента установки пакета;
- определить, к какому пакету относится определенный файл.

Информация, необходимая при управлении, содержится в каждом RPM-пакете. При установке такая информация заносится в базу данных (файлы каталога `/var/lib/rpm`).

Происхождение и обслуживание. Формат RPM первоначально разрабатывался для Red Hat. После того как в течение многих лет код поддерживался без энтузиазма, эта разработка раскололась на два самостоятельных проекта. Используемая в Fedora, Red Hat, SUSE и т. д. версия RPM 4.n теперь поддерживается и развивается под руководством Fedora и в кооперации с другими дистрибутивами в рамках проекта rpm.org (http://rpm.org/). Параллельно другая группа разработчиков занимается RPM 5.n (http://rpm5.org/). Однако эта версия в настоящее время применяется лишь в немногих относительно небольших дистрибутивах.

Основы

Большинство RPM-пакетов предоставляются в двух вариантах: в виде двоичного пакета и в виде пакета с исходным кодом. Двоичный пакет содержит файлы,

необходимые для исполнения программы. Пакет с исходным кодом интересен для разработчиков. Он хранит тот код, который необходим для создания двоичного пакета.

Уже в названии пакета содержится немало информации: например, abc-2.0.7-1.i686.rpm означает пакет abc версии № 2.0.7 выпуска 1 (если при составлении пакета возникла ошибка, была добавлена дополнительная онлайн-документация либо были внесены другие изменения, то номер выпуска больше 1 на число, равное номеру конкретной версии, то есть номер версии касается самой программы, а номер выпуска — сборки rpm).

Обозначение i686 указывает на то, что пакет содержит двоичные файлы для процессоров, совместимых с Pentium-II (разумеется, существуют версии Linux и для других процессоров). Если пакет `abc` содержит сценарные или текстовые файлы, не зависящие от архитектуры процессора, то вместо обозначения процессора используется сокращение noarch. Если пакет содержит исходный код, то здесь применяется сокращение src.

Кратко объясню различные сокращения x86:

- i386 = 386 и совместимые с ним процессоры;
- i486 = 486 и совместимые с ним процессоры;
- i586 = Pentium и совместимые с ним процессоры;
- i686 = Pentium II и совместимые с ним процессоры.

Пакет i386 будет работать со всеми вышеуказанными процессорами. Напротив, пакет i686 использует различные расширения Pentium II, поэтому код может выполняться немного быстрее. В любом случае программа не будет работать с другими процессорами.

Метаданные

Кроме файлов, предназначенных для установки, файл пакета содержит подробную информацию по администрированию: краткое описание пакета, опять же, информацию о номерах версий, о месте в групповой иерархии, информацию о взаимозависимости с другими пакетами и т. д. Взаимозависимости возникают, если для работы пакета требуется определенный язык программирования (например, Perl) или определенная библиотека. В таком случае сначала потребуется установить необходимые пакеты.

Программа `rpm` управляет базой данных, содержащей информацию обо всех установленных двоичных пакетах. Эта база данных сохраняется в виде отдельных файлов в каталоге `/var/lib/rpm`. В базе данных содержится информация только относительно двоичных пакетов; если у вас установлены пакеты с двоичным кодом, они не учитываются в базе данных.

Файлы базы данных RPM ни в коем случае нельзя изменять непосредственно! Для того чтобы база данных RPM содержательно совпадала с действительной конфигурацией установки, пакеты необходимо удалять не простым удалением файлов, а с помощью процедуры деинсталляции (`rpm -e`).

Пакеты Delta-RPM

Чтобы обновить пакет RPM, следует целиком загрузить новый пакет. И именно при проведении обновлений для системы безопасности, когда зачастую требуется

внести незначительнейшие изменения всего в несколько файлов, такой метод неэффективен. Поэтому в SUSE были разработаны так называемые *пакеты Delta-RPM*, в которых хранятся изменения, видимые лишь в сравнении с определенной версией пакета. В Mandriva и Fedora (начиная с версии 11) эта идея также прижилась и теперь используется при обновлениях.

В принципе применять пакеты Delta-RPM очень просто: сначала команда `applydeltarpm` создает из пакета Delta-RPM и оригинального пакета (либо относящихся к нему установленных файлов) новый, обновленный RPM-пакет. Он устанавливается как обычно (`rpm -U`). Команда `applydeltarpm` входит в состав пакета `deltarpm`.

Однако у Delta-RPM есть и недостатки: `applydeltarpm` потребляет очень много ресурсов процессора, и для ее работы необходимо, чтобы была установлена совершенно конкретная версия пакета. Если это не так либо если файлы этой версии после установки были изменены, потребуется обновить оригинальный файл RPM.

Проблемы

Система RPM — совершенно великолепная вещь, но и она имеет свои слабые стороны. Перечислю самые распространенные проблемы.

- Хотя в пакетах RPM и сохраняются межпакетные взаимосвязи, команда `rpm` не в состоянии автоматически их отменять. Для этого используются в первую очередь такие программы, как Yum.
- Управление взаимозависимостями пакетов нарушается, если смешиваются пакеты из разных дистрибутивов, если вы устанавливаете отдельные программы с применением `tar` или если вы сами компилируете программы. Причина все та же: либо информация о том, какие программы установлены на компьютере, отсутствует в принципе, либо данные не стыкуются (так как в каждом дистрибутиве действуют собственные принципы формулирования межпакетных взаимосвязей).
- Новые версии RPM хотя и обладают обратной совместимостью, но не совместимы «снизу вверх». Это означает, что, например, команда `rpm` версии 4.4 не может работать с файлами RPM, созданными с помощью версии 4.7. Что делать? Установите новейшую версию `rpm`.
- Дисковое пространство, необходимое для базы данных RPM (то есть для файлов каталога `/var/lib/rpm`) достаточно велико. При установке программ общим объемом 5 Гбайт файлы базы данных RPM займут около 50 Мбайт.
- Иногда база данных RPM содержит противоречивую информацию. В результате становится невозможно пользоваться командой `rpm` либо выдаются сообщения об ошибках вроде «невозможно открыть пакеты базы данных». В таких случаях обычно помогает команда `rm -f/var/lib/rpm/__db*`, а потом — `rpm --rebuilddb`. Тогда база данных RPM создается заново. Правда, на это требуется некоторое время.

Примеры

Подробная справка по синтаксису `rpm` приводится в `rpm --help` и `man rpm`. В следующих примерах показаны типичные ситуации, в которых используется эта команда.

- Установка программы `abc`:

```
root# rpm -i abc-2.0.7-1.i386.rpm
```

- Загрузка файла ключа GPG с http://myserver.com и приведение его в форму файла ключа для RPM:

```
root# rpm --import http://myserver.com/RPM-myserver-GPG-KEY
```

- Обновление программы abc, причем файл пакета скачивается прямо с указанного сервера:

```
root# rpm -U http://myserver.com/mypath/abc-2.1.0-2.i386.rpm
```

- Вывод списка всех файлов, входящих в abc, которые были изменены с момента установки:

```
root# rpm -V abc
```

- Удаление программы abc:

```
root# rpm -e abc
```

- Составление списка всех установленных пакетов:

```
root# rpm -qa
```

- Составление списка всех установленных пакетов, причем пакеты сортируются по дате установки (чем позже был установлен пакет, тем выше в списке он расположен):

```
root# rpm -qa --last
```

- Составляет список всех установленных пакетов, в названии которых содержится последовательность символов mysql (в любом регистре):

```
root# rpm -qa | grep -i mysql
```

- Выдает информацию о пакете perl (если он установлен):

```
root# rpm -qi perl
```

- Перечисляет все файлы из пакета perl:

```
root# rpm -ql perl
```

- Перечисляет все файлы документации из пакета perl:

```
root# rpm -qd perl
```

- Перечисляет все конфигурационные файлы пакета cups:

```
root# rpm -qc cups
```

- Выдает информацию об еще не установленном пакете. Для этого на локальном компьютере должен быть доступен RPM-файл:

```
root# rpm -qip abc-2.0.7-1.i386.rpm
```

- Указывает, из какого пакета взят файл /usr/lib/libz.so (результат: kdelibs):

```
root# rpm -qf rpm -qf /usr/lib/libkdnssd.so
```

- Указывает, какие атрибуты предоставляет пакет php-mysql. Результат в Fedora 11: ysql.so, mysqli.so, pdo_mysql.so, php-mysqli, php_database и php-mysql=5.2.9-2.

Атрибуты применяются для отмены межпакетных взаимосвязей. Обычно названия атрибутов соответствуют названиям программ или библиотек, предоставляющих пакет. Так или иначе от администратора пакетов любого дистрибутива зависит, каким образом будут определяться атрибуты. Атрибут также может содержать номер версии.

```
root# rpm -q --provides php-mysql
```

- Указывает, какие условия должны выполняться для установки пакета `php-mysql`. В Fedora конечный список атрибутов получается очень длинным, поэтому здесь я приведу только выдержки из него: `libc.so.6`, `libm.so.6`, `libmysqlclient.so.16`, `php-common=5.2.9-2` и `php-pdo`.

```
root# rpm -q --requires php-mysql
```

- Команды `rpm -q --provides` или `rpm -q -provides` также можно комбинировать с параметром `-p`, чтобы узнавать списки атрибутов для еще не установленных пакетов (в предыдущем примере — `alien`):

```
root# rpm -q --requires -p alien-8.56-2.i586.rpm
```

- Указывает, какой из уже установленных пакетов предоставляет атрибут `mysqli.so` (результат: пакет `php-mysql`):

```
root# rpm -q --whatprovides mysqli.so
```

- Выдает практически бесконечный список всех установленных пакетов, зависящих от атрибута `libpthread.so.0` (кстати, одноименная библиотека предоставляется пакетом `glibc`):

```
root# rpm -q --whatrequires libpthread.so.0
```

Проблемы 32/64 бит

В 64-битных дистрибутивах может случиться так, что команда `rpm -qi name` выдаст не название конкретного пакета, а информацию по *двум* пакетам. Это не ошибка — по-видимому, речь идет о двух одноименных пакетах с файлами для 32-битного и 64-битного вариантов программы или библиотеки.

SUSE старается обходиться без применения одноименных пакетов с разным содержимым, помечая 32-битные варианты пакетов окончанием `32bit`. В этом дистрибутиве команда `rpm -qa grep 32bit` возвратит удивительно длинный список всех 32-битных пакетов — такие пакеты нужны по причинам, связанным с совместимостью (к сожалению, пока невозможно компилировать все программы в 64-битном формате).

Взаимозависимости пакетов

Вероятно, вы впервые столкнетесь со взаимозависимостями пакетов, когда при попытке установить пакет получите сообщение об ошибке, которое будет гласить: failed dependencies: attributname is needed by paketname (неверные зависимости: для работы *название_атрибута* требуется *название_пакета*).

Для того чтобы решить такие проблемы, при установке лучше использовать не RPM, а Yum, Zypper или другой инструмент управления пакетами, предусмотренный

в вашем дистрибутиве. Если это невозможно, придется поискать пакет, предоставляющий *название_атрибута*. Для этого можно воспользоваться поисковиком http://rpmfind.net, позволяющим искать файлы и атрибуты, содержащиеся в пакете RPM.

10.2. Yum

Программа Yum облегчает управление пакетами. Она основана на RPM и предоставляет множество дополнительных функций.

- В качестве источников данных (репозиториев) служат архивы Yum, расположенные в Интернете. Yum-архив — это собрание пакетов RPM, дополнительно снабженных метаданными (каталог repodata). В метаданных находится информация о содержимом и зависимостях всех пакетов. Благодаря разделению пакетов и метаданных обработка информации ускоряется (так как не нужно считывать все пакеты, чтобы извлечь данные). Yum способна автоматически переходить между несколькими зеркалами, на которых расположен источник пакетов.
- Yum автоматически отменяет все взаимозависимости пакетов, загружает все необходимые пакеты и устанавливает их. При этом учитываются все источники пакетов. Если вы, например, устанавливаете пакет из репозитория *A*, может быть, что Yum предварительно загрузит определенные связанные с ним пакеты из источников *B* и *C*.
- Программа Yum может обновить все установленные пакеты одной-единственной командой. Для этого она проверяет, имеются ли в доступных источниках обновленные версии установленных пакетов. Если это так, то необходимые пакеты скачиваются и устанавливаются. Разумеется, при этом снимаются и все взаимозависимости пакетов.

Yum была разработана на языке программирования Python. Программа по умолчанию применяется в том числе в Fedora и в Red Hat. В этом разделе будет рассмотрено применение Yum в Fedora 11. Более подробная информация по Yum дается по следующему адресу: http://www.linux.duke.edu/projects/yum/.

Конфликты с блокировкой

Нельзя одновременно запускать несколько образцов Yum. Если уже работает программа или команда `Yum`, то при повторном запуске выводится сообщение об ошибке another copy is running (работает другой экземпляр).

Суть проблемы заключается в работе системы автоматических обновлений Yum. Этот процесс несложно обнаружить: файл `/var/run/yum.pid` содержит ID-номер программы, вызывающей конфликт. С помощью команды `ps grep id` можно узнать имя программы. Если необходимо, можно временно остановить систему обновлений с помощью команды `/etc/init.d/yum-updatesd stop`, а затем вновь запустить ее командой `start`.

Конфигурация

Основная конфигурация Yum выполняется в файле `/etc/yum.conf`. В следующих строках показаны выдержки из конфигурации в системе Fedora 12:

```
# Файл /etc/yum.conf
[main]
cachedir=/var/cache/yum/$basearch/$releasever
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=3
```

Кратко опишу важнейшие настройки. Благодаря `keepcache=0` скачанные пакеты не архивируются после установки. Как правило, эта установка полезна, так как дисковое пространство, необходимое для пакетов, со временем сильно вырастет, и, скорее всего, не будет причин устанавливать пакеты повторно. В любом случае Yum может обнаружить в ходе установки проблему и прервать установку. Если есть возможность устранить проблему и повторно произвести обновления, все пакеты потребуется скачать заново. Чтобы избежать этого, укажите `keepcache=1`. Для того чтобы специально удалить пакеты, находящиеся в `/var/cache/yum`, выполните команду `yum clean packages`.

При наличии `exaktarch=1` Yum учитывает только такие обновления, архитектура которых соответствует архитектуре уже установленного пакета. Иначе говоря, заменить пакеты i386 новыми пакетами x86_64 не получится.

При использовании `gpgcheck=1` Yum проверяет ключ аутентификации пакета. Переменная `gpgcheck` также можно настроить иначе, чем указано в `yum.conf`, — отдельно для каждого репозитория. Переменная `plugins` определяет, будет ли Yum учитывать плагины.

Некоторые пакеты Yum должна установить, но не должна обновлять. К ним относятся, в частности, пакеты ядра: при обновлении ядра дополнительно устанавливается новый пакет ядра, не затрагивая при этом старый пакет. Переменная `installonlypkgs` позволяет указывать названия таких пакетов. По умолчанию она имеет настройки `kernel`, `kernel-smp`, `kernel-bigmem`, `kernel-enterprise`, `kernel-debug`, `kernel-unsupported`. Наконец, переменная `installonly_limit`, содержащаяся в `yum.conf`, указывает, сколько версий подобных пакетов может быть установлено параллельно. При стандартной настройке одновременно остаются установлены только три новейшие версии пакета. Более старые пакеты ядра удаляются.

Создание репозиториев. Каждый репозиторий определяется в собственном REPO-файле в каталоге `/etc/yum.repos.d`. В следующем фрагменте показаны репозитории для получения основных пакетов Fedora 12:

```
# Файл /etc/yum.repos.d/fedora.repo
[fedora]
```

```
name=Fedora $releasever - $basearch
failovermethod=priority
mirrorlist=
  https://mirrors.fedoraproject.org/metalink?repo=fedora-$releasever&arch=$basearch
enabled=1
metadata_expire=7d
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-$basearch
```

Адрес репозитория можно по выбору указать в абсолютной форме с помощью `baseurl=...` или с использованием `mirrorlist=...` в виде файла-зеркала. В этом файле расположен список зеркальных серверов. Yum самостоятельно выбирает одно из зеркал. В конфигурационном файле Yum заменяет переменные `$releasever`, `$arch` и `$basearch` номерами версии дистрибутива Linux с указанием ее архитектуры. Кратко рассмотрим происхождение этих трех переменных.

- `$arch` определяется с помощью команды `uname` (если быть точным — с помощью одноименной функции Python, основанной на `uname`) и возвращает, к примеру, `x86_64`, если мы имеем дело с процессором 64-Bit-Intel/AMD.
- `$basearch` — это базовая архитектура, лежащая в основе `$arch` (например, i386).
- `$releasever` получается из номера версии пакета `edhat-release` или `fedora-release` (также можно указать в `yum.conf` название другого пакета с помощью ключевого слова `distroverpkg`; номер версии этого пакета считается номером версии дистрибутива).

Переменная `metadata_expires` указывает, в течение какого срока остаются действительны скачанные метаданные. **Yum сохраняет метаданные в кэше и не скачивает** их повторно, если прежние метаданные еще не устарели. Так экономится время и объем закачек (некоторые файлы с метаданными просто огромны), но в этом случае Yum может проигнорировать изменения, недавно внесенные в репозиторий (при необходимости можно удалить локальные метаданные командой `yum clean metadata`, тогда при следующем запуске Yum придется заново считать метаданные всех пакетов).

Оптимальная настройка для `metadata_expires` отличается в зависимости от репозитория: если его содержимое обновляется редко или вообще не обновляется, лучше задать длительный промежуток. Напротив, при работе с пакетами обновлений желательно задать короткий период или вообще отказаться от этой настройки.

Запрет пакетов. Если вы хотите, чтобы **Yum не контактировала с определенными** пакетами и при появлении новой версии они также не обновлялись, вставьте в `yum.conf` или в файл `*.repo` репозитория строку вида `exclude` *`имя1 имя2 имя3`*. В названиях пакетов можно использовать джокерные символы, то есть допускается и такая запись: `exclude xemacs*`.

Плагины. Yum можно расширять с помощью плагинов — тогда функционал программы становится еще богаче. Плагины можно настроить в файлах каталога `/etc/yum/pluginconf.d`.

Presto. Один из интереснейших плагинов Yum называется Presto: он позволяет использовать для обновлений пакеты Delta-RPM. Эти пакеты доступны только начиная с версии Fedora 11. В любом случае для установки Presto необходимо прямо указать `yum install yum-presto`, чтобы можно было использовать пакеты

Delta-RPM. В результате объем загрузок радикально снижается (на 60–80 %!), но нагрузка на процессор при проведении обновлений выше.

Дистрибутивные обновления. В принципе с помощью Yum можно проводить и обновление дистрибутивов. Для этого сначала требуется вручную установить пакеты `fedora-release-n.noarch.rpm` и `fedora-release-n.noarch.rpm`, а затем выполнить команду `yum upgrade`. Надо отметить, что я не сторонник дистрибутивных обновлений. При этом обязательно что-то идет не так, независимо от того, насколько интеллектуальна система управления пакетами. Когда это только возможно, старайтесь проводить новую установку.

Примеры

Управление пакетами целиком осуществляется с помощью команды `yum`. Синтаксис этой команды понятен из следующего примера. Приведенные далее команды показывают, как работает программа (вывод сокращен ради экономии места).

```
root# yum check-update
...
anacron.i586              2.3-75.fc11       updates
brasero.i586              2.26.2-1.fc11     updates
brasero-libs.i586         2.26.2-1.fc11     updates
brasero-nautilus.i586     2.26.2-1.fc11     updates
cheese.i586               2.26.2-1.fc11     updates
cronie.i586               1.3-1.fc11        updates
...
root# yum update
...
Transaction Summary
Install   1 Package(s)
Update   42 Package(s)
Remove    0 Package(s)
Total download size: 52 M
Is this ok [y/N]: y
...
root# yum install emacs
...
Installing:
  emacs                           i586    1:22.3-12.fc11  updates   2.0 M
Installing for dependencies:
  emacs-common                    i586    1:22.3-12.fc11  updates    19 M
  xorg-x11-fonts-ISO8859-1-100dpi noarch  7.2-8.fc11      fedora    1.1 M
Transaction Summary
Install   3 Package(s)
Update    0 Package(s)
Remove    0 Package(s)
Total download size: 22 M
Is this ok [y/N]: y
...
```

Если вы впервые выполняете команду `yum`, система скачивает метаинформацию по всем созданным репозиториям. В дальнейшем эти файлы регулярно обновляются.

Группы пакетов

Программа Yum может распределять пакеты по группам, чтобы проще было установить все пакеты, необходимые для решения определенной задачи. Список доступных групп пакетов выводится командой `yum grouplist`, а `yum groupinfo name` подсказывает, какие пакеты относятся к одной и той же группе. Команда `yum groupinfo` подразделяет все пакеты на три категории: `mandatory` (обязательные), `default` (по умолчанию) и `optional` (необязательные). Выполнив `yum groupinstall` *имя*, вы установите все пакеты групп mandatory и default. Команда `yum` не содержит параметров, которые позволили бы автоматически установить и необязательные пакеты. Если вам нужна такая возможность, внесите в файл `yum.conf` следующие изменения:

```
# Дополнение в /etc/yum.conf
group_package_types = mandatory default optional
```

Если хотите обновить или удалить группу пакетов, используйте `yum groupupdate` или `yum groupremove` соответственно.

Пакеты с исходным кодом

Сама по себе команда `yum` не может устанавливать пакеты с исходным кодом. Эту задачу выполняет команда `yumdownloader` из пакета `yum-utils`. Следующая команда загружает пакет с исходным кодом редактора gedit в локальный каталог. При этом активизируются источники `source`, находящиеся в файлах `*.repo` и по умолчанию неактивные. Когда я проводил испытания, загрузка исходного кода, словно по волшебству, работала только тогда, когда для источников `source` в файлах `*.repo` было установлено `enabled=0`.

```
user$ yumdownloader --source gedit
```

Автоматические загрузки и обновления

Если установлен пакет `yum-updatesd`, то при запуске компьютера процесс Init-V активизирует одноименную программу `yum-updatesd` (как управлять автоматическим запуском сценария Init-V, объясняется в разделе 4.5).

Команда `yum-updatesd` позволяет регулярно проверять, есть ли доступные обновления. В зависимости от того, какие настройки сделаны в `/etc/yum/yum-updatesd.conf`, пакеты обновлений также загружаются и даже устанавливаются автоматически. В следующем листинге показана стандартная конфигурация Fedora 11, при которой автоматические обновления *не* производятся. Если вы хотите производить автоматические обновления, в трех случаях измените `no` на `yes`:

```
# /etc/yum/yum-updatesd.conf
[main]
# Проверять один раз в час, доступны ли новые обновления
run_interval = 3600
# Связываться с сервером обновлений не реже, чем каждые 10 минут
updaterefresh = 600
# Осуществлять локальное оповещение об уведомлениях через dbus
emit_via = dbus
# Автоматическая загрузка обновлений
do_download = no
```

```
# Загрузить для обновления зависимые пакеты
do_download_deps = no
# Автоматически установить обновления
do_update = no
```

К сожалению, если программа обновлений Yum будет работать постоянно, это иногда может приводить к проблемам, связанным с блокировкой (см. подраздел «Конфигурация» этого раздела). Временно остановите `yum-updatesd`, прежде чем приступить к управлению пакетами. Не забудьте потом снова запустить `yum-updatesd`!

```
root# /etc/init.d/yum-updatesd stop
root# ... вручную установить или удалить пакеты ...
root# /etc/init.d/yum-updatesd start
```

Yum Extender (Yumex)

Yumex — это простой и удобный графический пользовательский интерфейс для Yum. При запуске Yum обновляет локальные метаданные по всем репозиториям. Кроме того, вы можете найти пакеты, которые необходимо установить, отметить их для установки и, наконец, установить (кнопка Обработать очередь) (рис. 10.1). Теперь Yumex интерпретирует все взаимозависимости пакетов и выведет обобщающее диалоговое окно. Только после вашего подтверждения пакеты действительно будут скачаны и установлены.

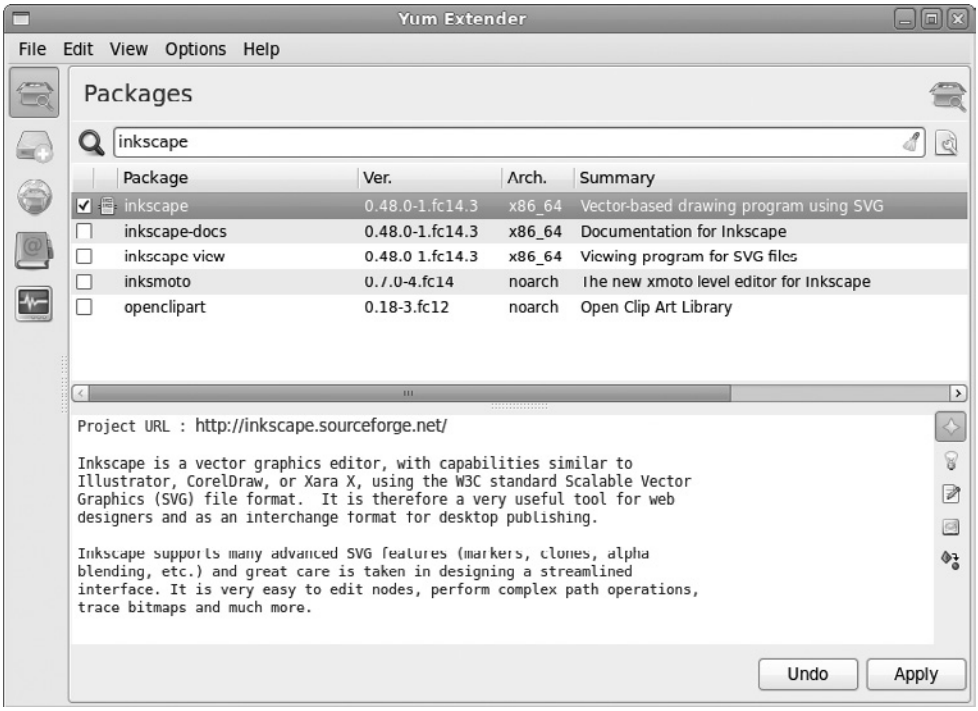

Рис. 10.1. Управление пакетами с помощью Yum Extender

10.3. ZYpp

В SUSE, как и в Fedora и Red Hat, используются RPM-пакеты. ZYpp, система управления пакетами, основанная на Yum, была разработана именно для Novell/SUSE. ZYpp означает «ZENworks, YaST, пакеты и патчи», причем использование Zenworks не является обязательным и предусмотрено только в корпоративных дистрибутивах Novell/SUSE. Подробная информация по ZYpp имеется на сайте http://en.opensuse.org/ZYpp.

Библиотека libzypp

За кулисами программы работает библиотека `libzypp`, предоставляющая основные функции ZYpp. Эта библиотека разбирается с репозиториями как YaST, так и YUM. Все файлы конфигурации, базы данных и кэша находятся в каталоге `/var/lib/zypp`. YaST и PackageKit при работе в SUSE обращаются к библиотеке `libzypp`.

ПРИМЕЧАНИЕ

Zypper различает обновления и патчи. Обновления — это пакеты, для которых доступна более новая версия, чем та, что установлена у вас. Патчи, в свою очередь, — это дополнительные пакеты (Delta-RPM).

SUSE использует для обновления пакетов патчи в форме Delta-RPM. Внешние репозитории, например Packman, предлагают новые версии пакетов в форме обновлений.

Репозитории

Репозитории сохраняются в текстовых файлах каталога `/etc/zypp/repos.d`. Если вы изменяете эти файлы в текстовом редакторе, не забудьте дополнительно удалить и все резервные копии! В противном случае в вашем списке репозиториев появятся «двойники». В следующих строках дается определение свободно распространяемого репозитория для openSUSE 11.2:

```
# Файл /etc/zypp/repos.d/repo.oss.repo
[repo-oss]
name=openSUSE-11.2-Oss
enabled=1
autorefresh=1
baseurl=http://download.opensuse.org/distribution/11.2/repo/oss/
path=/
type=yast2
keeppackages=0
```

Интерфейс zypper

Это командный интерфейс для `libzypp`. Таким образом, `zypper` — аналог SUSE для `aptget` или `yum`. С его помощью можно искать, устанавливать, обновлять и удалять пакеты, а также управлять репозиториями. Команда `zypper` должна выполняться администратором. Более подробная информация приводится на странице справки `man zypper` и на сайте http://de.opensuse.org/Zypper/Anleitung.

На следующих примерах демонстрируется применение zypper. Первая команда перечисляет репозитории, вторая обновляет источники, третья устанавливает редактор nano, а четвертая определяет, какие есть обновления.

```
root# zypper repos
# | Псевдоним       | Имя                   | Включено   | Обновить
--+-----------------------+-----------------------------------+---------+--------
1 | openSUSE 11.2-0 | openSUSE 11.2-0       | Yes        | No
2 | repo-non-oss    | openSUSE-11.2-Non-Oss | Yes        | Yes
3 | repo-oss        | openSUSE-11.2-Oss     | Yes        | Yes
...
root# zypper refresh
Все репозитории были обновлены.
root# zypper install nano
Считывание установленных пакетов…
Будет установлен следующий НОВЫЙ пакет: nano
Общий объем загрузок: 335.0 K. После операции будет использовано еще 1,2 Мбайт
Продолжить? [YES/no]: yes
Поиск пакета nano-2.1.5-1.38.x86_64 (1/1), 335.0 K (1,2 Мбайт распаковано)
Поиск: nano-2.1.5-1.38.x86_64.rpm [done]
Установка: nano-2.1.5-1.38 [done]
root# zypper list-updates
...
Репозиторий         | Имя             | Версия  | Категория    | Статус
Haupt-Update Repos  | MozillaFirefox  | 4572-0  | security     | Needed
Haupt-Update Repos  | NetworkManager  | 4548-0  | recommended  | Needed
...
```

Группы пакетов. Чтобы установить пакеты, необходимые для решения определенной задачи, например для применения компьютера в качестве файлового сервера, ZYpp использует так называемый *шаблон* (pattern). Команда `zypper search -t` *шаблон* выдает список всех таких групп пакетов. Команда `zypper info -t` *шаблон имя* показывает, какие пакеты относятся к группе *имя*. С помощью `zypper install -t` *шаблон имя* устанавливаются все пакеты группы *имя*.

History. В файле `/var/log/zypp/history` содержится исключительно практичная информация о том, когда какой пакет был установлен или удален из какого репозитория и какие конфигурационные работы были при этом проведены.

Обновления дистрибутивов. Начиная с версии openSUSE 11.1, вы можете, как в Debian или Ubuntu, обновлять дистрибутив, не останавливая его работу:

```
root# zypper updates          (Обновление имеющейся версии)
root# ...                     (Замена версии репозиториев)
root# zypper dup              (Замена старых пакетов новыми)
root# reboot                  (Новый запуск)
```

10.4. Управление пакетами Debian (команда dpkg)

Управление пакетами Debian осуществляется на двух уровнях: в этом разделе описана команда dpkg, которая отвечает за установку пакетов и управление ими на

нижнем уровне. Эту команду можно сравнить с rpm. Команда может устанавливать отдельные пакеты, обновлять их, удалять и при этом тестировать, выполняются ли все взаимозависимости пакетов. Но эта команда не может самостоятельно отменять невыполненные взаимозависимости пакетов либо загружать информацию из репозиториев.

Выполнением этих задач занимается APT (усовершенствованное средство управления программными пакетами) (см. раздел 10.5). Он основан на dpkg и функционально напоминает описанные выше системы Yum и ZYpp. Собственно для управления пакетами предлагаются две команды: apt-get обычно применяется в Ubuntu, а aptitude считается основным инструментом для установки пакетов в Debian 5.0 и выше.

Если рассмотреть этот и следующий абзац с точки зрения установки пакетов в Debian, то будут выполняться все правила, действующие для других дистрибутивов Linux, использующих этот формат пакетов. Кроме Debian это, например, дистрибутивы семейства Ubuntu — Mepis, Mint и Knoppix. Если вы переходите с дистрибутива на основе RPM на дистрибутив с пакетами Debian, рекомендую вам посмотреть следующий сайт — там дан хороший обзор команд rpm, а также эквивалентных им команд dpkg и apt: https://help.ubuntu.com/community/SwitchingToUbuntu/FromLinux/RedHatEnterpriseLinuxAndFedora.

Примеры

Полная справка по синтаксису dpkg приводится на страницах dpkg --help и man dpkg. В следующих примерах разъясняется использование команды в стандартных ситуациях. Кроме этого случая, dpkg вряд ли пригодится вам на практике. Администрирование пакетов в большинстве дистрибутивов, использующих вариант пакетов для Debian, осуществляется непосредственно с помощью APT или основанного на нем инструмента.

```
root# dpkg --install test.deb
...
Распаковка теста ...
Создание теста ...
root# dpkg --search /etc/mediaprm
fdutils: /etc/mediaprm
root# dpkg --listfiles fdutils
/.
/usr
/usr/bin
/usr/bin/diskd
...
```

Статус пакета

Команда dpkg --list возвращает список всех установленных пакетов. При этом отображается код состояния, состоящий из одной-трех букв. Первая буква указывает желаемое состояние (u = unknown, i = install, r = remove, p = purge, h = hold), вторая буква — фактическое состояние (n = not, i = installed, c = config files,

u = unpacked, f = failed config, h = half installed), третья — код ошибки (h = hold, r = reinstall required, x = hold + reinstall required).

```
root# dpkg --list | grep hplip
ii hplip       3.9.2-3ubuntu4  HP Linux Printing and Imaging System (HPLIP)
ii hplip-data  3.9.2-3ubuntu4  HP Linux Printing and Imaging - data files
rc hplip-gui   3.9.2-3ubuntu4  HP Linux Printing and Imaging - GUI utilities
```

Чаще всего встречаются статусные коды ii (установленный пакет) и rc (пакет удален, но данные о его конфигурации еще доступны). Чтобы полностью удалить пакет rc, выполните команду dpkg--purge *имя*.

Статус hold означает, что пакет не должен изменяться, когда скачиваются обновления к нему. Две следующие команды показывают, как перевести пакет в состояние hold или снова отменить этот статус:

```
root# echo "имя_пакета hold"    | dpkg --set-selections
root# echo "имя_пакета install | dpkg --set-selections
```

Более подробные сведения по статусам пакетов и устранению связанных с этим проблем дает справка man dpkg.

В некоторых пакетах кроме автоматических конфигурационных сценариев имеются установочные инструменты, позволяющие сконфигурировать каждую отдельную программу. Если позже вы захотите изменить конфигурацию, выполните команду dpkg-reconfigure *имя_пакета*.

Метаданные

Команда dpkg располагает подробной метаинформацией по всем пакетам (описание пакета, список всех файлов пакета, данные о взаимозависимости и т. д.). Эти файлы предлагаются в формате DCTRL (Debian Control). В пакете dctrl-tools содержатся различные команды, позволяющие производить запросы данных DCTRL. В справке man grep-dctrl приводится подробное описание этой команды и множество конкретных примеров применения.

10.5. APT

APT для пакетов Debian так же важен, как и Yum для пакетов RPM: это высокоуровневая система управления пакетами, самостоятельно скачивающая пакеты из репозиториев и автоматически отменяющая межпакетные взаимозависимости. Комбинация из пакетов Debian и инструмента APT на настоящее время (2009 год) представляется наиболее проработанной системой управления пакетами в Linux. Среди прочего, она применяется в Ubuntu и Debian в качестве стандартной системы управления пакетами.

Для управления пакетами предлагаются две различные команды: apt-get и aptitude. Обе команды очень похожи друг на друга и при всей схожести обслуживания даже имеют практически одинаковый синтаксис. Команды apt-get install *имя_пакета* или aptitude install *имя_пакета* загружают указанный пакет и все пакеты, зависящие от него, и устанавливают их.

В настоящее время apt-get по умолчанию применяется в Ubuntu, а в Debian версии 5 и выше рекомендуется использовать aptitude. Неважно, работаете вы с Debian или с Ubuntu, — *обе* команды устанавливаются по умолчанию.

Кроме того, существуют версии APT для пакетов RPM, которые, однако, распространены не так широко, как Yum. В этом разделе будет рассмотрен только APT для пакетов Debian. Информация по APT находится на следующем сайте: http://apt-rpm.org/.

Конфигурация

Конфигурация APT производится в двух файлах — apt.conf.d/* и sources.list, находящихся в каталоге /etc/apt. Более подробная информация по репозиториям может храниться в каталоге sources.list.d.

apt.conf.d/*, как правило, содержит лишь немногие базовые настройки, которые обычно нужно оставить в том виде, в котором они заданы по умолчанию (также см. man apt.conf). Файл sources.list (справка man sources.list) уже интереснее. Этот файл содержит построчное описание репозиториев APT. Синтаксис каждой строки таков:

```
pakettyp uri distribution [компонент1] [компонент2] [компонент3] ...
```

Тип обычных пакетов Debian называется deb, а для пакетов с исходным кодом — deb-src. Наряду с каталогами HTTP и FTP APT поддерживает обычные каталоги, RSH и SSH-серверы, а также CD и DVD.

Правда, случай с репозиториями CD и DVD особый: такие пакеты создаются командой apt-cdrom, описанной чуть ниже. Просто добавить строку deb cdrom недостаточно.

В третьем столбце стоит название дистрибутива (ведь на сервере могут находиться пакеты для нескольких дистрибутивов или их версий). Во всех остальных столбцах указываются компоненты дистрибутива, которые можно учитывать. Названия компонентов зависят от конкретного дистрибутива и пакета! Например, в Ubuntu различаются пакеты main, restricted, universe и multiverse, а в Debian — main, contrib, non-free и т. д.

Версии пакетов, названные раньше, считаются предпочтительными: то есть если пакет доступен для скачивания в нескольких источниках, APT скачивает его по первой ссылке. В следующих строках показан синтаксис. Это строки из немецкой версии Ubuntu 9.10 с использованием немецкого зеркального сервера.

```
# Файл /etc/apt/sources.list
deb http://de.archive.ubuntu.com/ubuntu/ karmic          main restricted universe
multiverse
deb http://de.archive.ubuntu.com/ubuntu/ karmic-updates  main restricted universe
multiverse
deb http://security.ubuntu.com/ubuntu    karmic-security main restricted universe
multiverse
```

Изменения в sources.list лучше всего вносить в текстовом редакторе. Если не хотите пользоваться редактором, попробуйте графический пользовательский интерфейс, например Synaptic.

Декларирование CD в качестве репозитория

В качестве источников данных для APT также можно использовать CD. В таком случае для каждого диска необходимо выполнить команду `apt-cdrom add`. Она считывает метаданные APT компакт-диска и заносит доступные пакеты в файл кэша. Кроме того, обновляется `sources.lst`. Если `apt-cdrom` жалуется на то, что не может найти CD/DVD, укажите каталог с параметром `-d`.

```
root# apt-cdrom -d /media/dvd add
```

Установка APT-ключа

В большинстве APT-источников в Интернете метафайлы для описания репозиториев зашифрованы криптографическими ключами. Кроме того, файлы-оглавления APT содержат контрольные суммы для всех пакетов. Такие механизмы контроля позволяют обеспечить постепенное изменение пакета. Однако этот механизм контроля действует лишь в том случае, если APT известна общедоступная часть ключа, позволяющая определить происхождение пакета. Чтобы настроить ключ для APT, пользуйтесь командой `apt-key`:

```
root# apt-key add кодовый_файл.gpg
```

Команда apt-get

Само управление пакетами производится командой `apt-get` или `aptitude`. Синтаксис `apt-get` понятен на следующих примерах. Показанный в них вывод сокращен ради экономии места.

Обновление информации по APT. Перед установкой пакетов выполните `apt-get update`, загрузив таким образом новейшую информацию из репозиториев. Пакеты при этом ни устанавливаются, ни обновляются. В данном случае речь идет только об описаниях пакетов!

Установка пакетов. Дополнительно выполните `apt-get install` — при этом необходимо указать правильное название пакета. Если команда обнаружит, что определенные взаимозависимости пакетов не соблюдаются, она также предложит установить недостающие пакеты. Когда вы примете это предложение, `apt-get` скачает файлы пакетов и установит их. В следующем примере `apt-get` рекомендует установить несколько дополнительных пакетов.

```
root# apt-get update
Получение:1 http://security.ubuntu.com dapper-security Release.gpg [189B]
...
root# apt-get install apache2
...
Устанавливаются следующие дополнительные пакеты:
apache2-mpm-worker apache2-utils apache2.2-common libapr1 libaprutil1
Предлагаемые пакеты:
apache2-doc apache2-suexec apache2-suexec-custom
Устанавливаются следующие НОВЫЕ пакеты:
apache2 apache2-mpm-worker apache2-utils apache2.2-common libapr1 libaprutil1
0 обновлено, 6 новых установлено, 0 к удалению и 11 не обновлено.
```

```
Необходимо скачать архивы общим размером 1472kB.
После этой операции на диске будет занято еще 5452kB места.
Хотите продолжить? [Y/n]?
...
```

Удаление пакетов. Команда `apt-get remove paketname` удаляет указанный пакет. Пакеты, которые первоначально были загружены с удаляемым пакетом и связаны с ним зависимостями, остаются при этом нетронутыми. В данном случае используйте `apt-get autoremove`. Эта команда удаляет все пакеты, которые более не нужны.

Обновление пакетов. Команда `apt-get dist-upgrade` обновляет все установленные пакеты. Если изменились межпакетные взаимосвязи и требуется установить новые пакеты или удалить старые, эти операции также выполняются.

```
root# apt-get upgrade
...
Следующие пакеты будут обновлены:
libmozjs1d xulrunner-1.9 xulrunner-1.9-gnome-support
3 обновлено, 0 новых установлено, 0 к удалению и 0 не обновлено.
Необходимо скачать архивы общим размером 8208kB.
После этой операции на диске будет занято еще 8192kB места.
Хотите продолжить? [Y/n]? Y
...
```

У команды `apt-get dist-upgrade` есть еще вариант `upgrade`: он также обновляет все пакеты. Разница заключается в том, что при этом не устанавливаются никакие новые или имеющиеся пакеты.

Установка исходного кода. Команда `apt-get source` *`имя_пакета`* устанавливает исходный код нужного пакета в текущий каталог. Более подробно эта операция описана в разделе 11.2.

Программа aptitude

Текстовая программа `aptitude` также построена на основе APT. Если вы используете эту программу в командном режиме (`aptitude install` *`имя_пакета`*), ее синтаксис во многом совместим с `apt-get`. Кроме того, эту программу можно использовать в консоли — с текстовым пользовательским интерфейсом (рис. 10.2). Для этого просто запустите программу без дополнительных параметров. Для перехода в меню нажмите клавишу F10.

По сравнению с `apt-get` программа `aptitude` имеет принципиальное преимущество: она отмечает, какие зависимые пакеты были установлены, и автоматически удаляет их при деинсталляции. Если, например, вы устанавливаете программу *xyz*, для работы которой требуется пять дополнительных пакетов (`lib-abc`, `lib-efg` и т. д.), эти пакеты удаляются (если от них не зависит какой-нибудь другой пакет). Если же вы удаляете *xyz* с помощью `aptget` или Synaptic, зависимые пакеты `lib-abc`, `lib-efg` и т. д. остаются в системе. Пройдет немного времени, и никто уже не вспомнит, зачем были установлены эти пакеты.

В Debian 5 для управления пакетами специально рекомендуется использовать `aptitude` вместо `apt-get`. Конечно же, вы можете не следовать этому совету. Однако

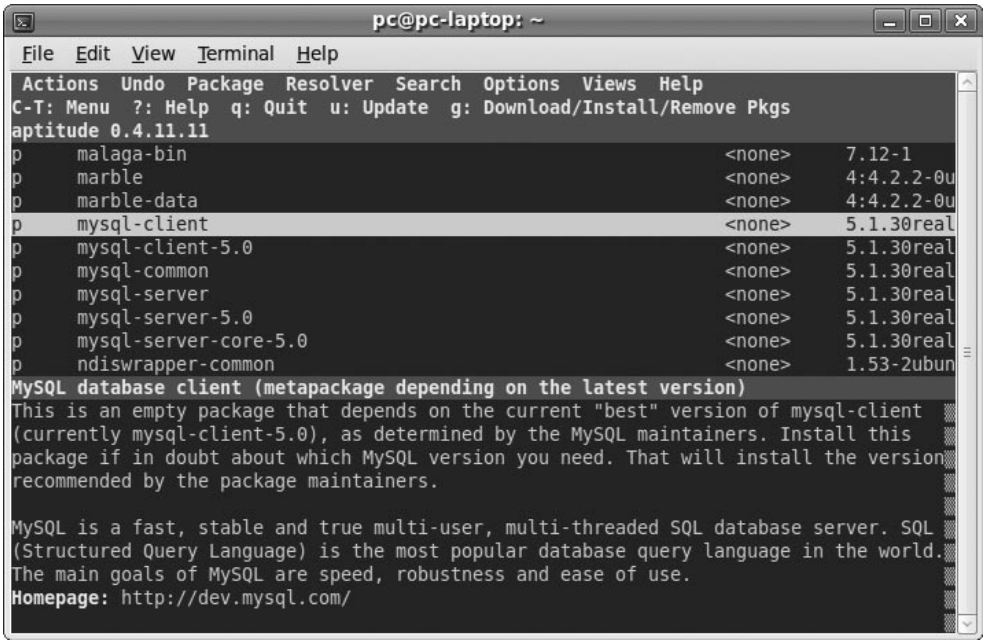

Рис. 10.2. Управление пакетами с помощью программы aptitude

очень нежелательно смешивать эти команды! Обе управляют собственными базами данных с информацией об уже установленных пакетах и их взаимосвязях и т. д. Если параллельно использовать `apt-get` или `aptitude`, эти базы данных будут несовместимы.

Пример. Две следующие команды сначала устанавливают пакет `html2ps`, а потом снова его удаляют. Отмечу, что большинство дополнительных пакетов, установленных вместе с `html2ps`, снова удаляются. Большинство, но не все — команда `aptitude` несовершенна.

```
root# aptitude install html2ps
Считываются списки пакетов… готово
...
Дополнительно устанавливаются следующие пакеты:
gs-gpl html2ps libcompress-zlib-perl libfont-afm-perl libhtml-format-perl
libhtml-parser-perl libhtml-tagset-perl libhtml-tree-perl libmailtools-perl
...
Хотите продолжить? [Y/n/?] Y
...
root# aptitude remove html2ps
...
Следующие пакеты не используются и будут УДАЛЕНЫ:
libfont-afm-perl libhtml-format-perl libhtml-parser-perl libhtml-tagset-perl
libhtml-tree-perl libmailtools-perl libtimedate-perl liburi-perl libwww-perl
perlmagick
...
```

Команда tasksel

Как правило, для установки групп пакетов в Debian и Ubuntu используются метапакеты: это пустые пакеты, определяющие только отдельные межпакетные взаимосвязи. Например, вместе с метапакетом `build-essential` устанавливается несколько пакетов с важными инструментами разработки (`Compiler`, `make` и т. д.).

Есть еще один способ определения групп пакетов — на основе команды `tasksel`. Этот механизм предназначен в первую очередь для того, чтобы можно было с легкостью выбирать пакеты при установке дистрибутива. Разумеется, `tasksel` можно использовать и позже, при эксплуатации компьютера. Список всех имеющихся групп пакетов выводится командой `tasksel --list-task`. Для установки групп пакетов используется команда `tasksel install groupname`. Если `tasksel` выполняется без дополнительных параметров, то система выводит диалоговое окно, в котором можно выбрать требуемые группы пакетов.

Команда apt-cache

Команда `apt-cache` узнает различные данные относительно доступных или уже установленных пакетов:

```
root# apt-cache show apache2
Package: apache2
Priority: optional
Section: web
...
Description: передовой, масштабируемый, расширяемый веб-сервер
  Apache v2 — представитель следующего поколения вездесущего веб-сервера
  Apache. Данная версия — полностью переработанная — содержит множество
  нововведений, в частности функцию обработки потоков, новый интерфейс API,
  поддержку IPv6, фильтрацию запросов и откликов и многое другое.
...
```

Автоматизация обновлений

Как правило, после установки APT требуется откорректировать некоторые конфигурационные файлы, чтобы система могла автоматически скачивать обновления, а при необходимости и устанавливать их. Нужные конфигурационные файлы входят в состав пакета `unattended-upgrades`, который по умолчанию устанавливается в Debian и Ubuntu.

Для автоматизации скачивания и установки обновлений была разработана программа Cron, которая ежедневно выполняет сценарий `/etc/cron.daily/apt`. Этот сценарий интерпретирует конфигурационный файл `/etc/apt/apt.conf.d/50unattended-upgrades` и при необходимости выполняет команду обновления `unattended-upgrade`. Чтобы активизировать автоматические обновления, нужно вручную внести в файл `50unattended-upgrades` следующие изменения:

```
// Файл /etc/apt/apt.conf.d/50unattended-upgrades
// Активизировать ежедневные обновления
APT::Periodic::Unattended-Upgrade "1";
// Проводить стандартные обновления и обновления системы безопасности
Unattended-Upgrade::Allowed-Origins {
"Ubuntu karmic-security";
"Ubuntu karmic-updates";
};
// Не обновлять следующие пакеты
Unattended-Upgrade::Package-Blacklist {
// "vim";
// ...
};
// Прислать электронное сообщение о статусе обновления
// Unattended-Upgrade::Mail "root@localhost";
```

В первой строке должно быть указано, с каким интервалом (в днях) система должна пытаться скачивать обновления. С помощью `Allowed-Origins` укажите, какие репозитории должны учитываться при проведении обновлений. С использованием `Package-Blacklist` можно автоматически исключить определенные пакеты при проведении обновлений. Наконец, команда `mail` позволяет указать адрес, на который команда `unattended-upgrade` посылает краткое статусное уведомление после того, как обновление будет успешно завершено. Для рассылки электронной почты необходимо настроить на компьютере почтовый сервер (MTA) и дополнительно установить пакет `mailx`.

В заключение скажу еще несколько слов о процессе обновления: в каталоге `/etc/cron.daily/apt` содержится команда `sleep`, задерживающая выполнение сценария на случайное количество секунд (до 1800). Такая принудительная пауза позволяет избежать ситуации, в которой тысячи компьютеров, использующих cron, одновременно обращались бы к одним и тем же репозиториям.

Сценарий `/usr/bin/unattended-upgrade` протоколирует все обновления или попытки обновлений в файлах регистрации в каталоге `/var/log/unattended-upgrade`. Кроме того, `/etc/cron.daily/apt` закладывает в каталоге `/var/lib/apt/periodic/` файлы с отметками времени, показывающими, когда именно в последний раз проводилась определенная операция.

Команда `unattended-upgrade` не обновляет те пакеты, в которых имеется так называемая инструкция conffile prompt, то есть конфигурационные файлы которых были обновлены вручную. К сожалению, эти данные можно узнать только из файлов регистрации, но не из статусного сообщения, пришедшего по электронной почте. Поскольку на практике такие изменения проводятся довольно часто, несмотря на автоматические обновления, вам придется регулярно проверять, появились ли обновления, которые необходимо установить вручную.

Обновления ядра вступают в силу только после перезагрузки компьютера. Команда `unattended-upgrade` этим не занимается, то есть вы сами должны перезапустить компьютер командой `reboot`.

СОВЕТ

Автоматические обновления хороши тем, что снижают риск проникновения злоумышленника в случайную брешь в системе безопасности вашего компьютера. Если из-за такого неправильного обновления некоторые функции сервера перестанут работать, то последствия могут быть катастрофическими. В прошлом Ubuntu не раз отключала обновления, которые не прошли проверку.

Альтернатива автоматических обновлений — сценарий Cron, ежедневно проверяющий, доступны ли свежие обновления (например, с помощью команды apt-get dist-upgrade --simulate), и пересылающий результат администратору по электронной почте.

Обновления версий или дистрибутивов

С помощью команды `apt-get dist-upgrade` можно обновить целый дистрибутив до новейшей версии. Для этого потребуется соответствующим образом заново настроить репозитории в `/etc/apt/sources.list`.

В Ubuntu обновление дистрибутива можно произвести с помощью команды `do-release-update -m desktop` (для ПК) или `-m server` (для сервера). В версиях Ubuntu LTS (рассчитанных на долговременную поддержку) новые релизы предусмотрены лишь для следующих версий, то есть, например, 8.04 или 10.04. Если вы хотите обновить версию, которая не рассчитана на долговременную поддержку, вам сначала нужно в каталоге `/etc/updatemanager/release-upgrades` установить для переменной `Prompt` значение `normal` вместо `lts`.

Несмотря на то что в Debian хорошо проработана система управления пакетами, обновление версий этой системы — дело очень деликатное. Если после обновления все программы и серверные службы будут работать как надо, то это счастливое исключение, а не правило.

Программа apt-cacher (буфер обмена пакетов)

Если в вашей сети работают десятки компьютеров с Debian и Ubuntu и каждый обращается за обновлениями к внешнему репозиторию, то общий ежемесячный трафик достигает нескольких гигабайт. Даже при том, что тарифы на пользование Интернетом в основном безлимитные, обновления засоряют сеть и чрезмерно замедляют передачу данных по LAN. В данном случае кажется очевидным, что не помешает организовать центральный буфер обмена (прокси), из которого все компьютеры будут брать пакеты для обновления. Именно эту задачу и выполняет программа `apt-cacher`.

Серверная конфигурация

На кэш-сервере (то есть на компьютере, управляющем буфером обмена) необходимо установить пакет `apt-cacher`. (Из многочисленных руководств по `apt-cacher`, имеющихся в Интернете, можно сделать вывод, что вам также потребуется установить Apache. Это неверно — `apt-cacher` работает сам! Взаимодействие с Apache оправданно только в тех случаях, когда обмен информацией с `apt-cacher` должен производиться через HTTP-порт 80.)

```
root# apt-get install apt-cacher
```

Чтобы в дальнейшем пакет apt-cacher запускался автоматически, как демон, внесите небольшие изменения в файл /etc/default/apt-cacher:

```
# Файл /etc/default/apt-cacher
AUTOSTART=1
...
```

Остальные параметры конфигурации содержатся в файле /etc/apt-cacher/apt-cacher.conf. Большинство установок можно оставить как есть, тогда APT-прокси будет доступен всем компьютерам через порт 3142. Файлы пакетов будут сохраняться в каталоге /var/cache/apt-cacher. По соображениям безопасности рекомендуется внести в конфигурационный файл и следующие изменения:

```
# Файл /etc/apt-cacher/apt-cacher.conf
...
daemon_addr=192.168.0.1
allowed_hosts=192.168.0.0/24
...
```

Демон связывается с определенным адресом (в данном случае 192.168.0.1, этот показатель важен для компьютеров с несколькими интерфейсами) и позволяет пользоваться прокси только клиентам, относящимся к адресному пространству 192.168.0.*. После внесения в конфигурацию таких изменений мы в первый раз запускаем apt-cacher:

```
root# /etc/init.d/apt-cacher start
```

Пакет apt-cacher заносит все случаи доступа, а также возможные ошибки в файлы регистрации, находящиеся в каталоге /var/log/apt-cacher.

Импорт имеющихся пакетов

Как правило, в каталоге /var/cache/apt/archive на сервере уже имеется множество пакетов, загружаемых локальной системой управления пакетами для установки либо обновления. Вы можете импортировать эти пакеты в кэш apt-cacher. Это целесообразно делать в тех случаях, когда ожидается, что данные пакеты потребуются другим компьютерам, работающим в локальной сети:

```
root# cd /usr/share/apt-cacher
root# ./apt-cacher-import.pl /var/cache/apt/archives
```

Клиентская конфигурация

На клиентской машине необходимо включить браузер и убедиться, что apt-cacher доступен через HTTP. Для этого укажите следующие адреса (конечно же, при этом необходимо изменить мой apt-cacher на хост-имя вашего компьютера или прокси-сервера):

- http://mein-apt-cacher:3142;
- http://mein-apt-cacher:3142/report.

На первой странице обобщается конфигурация, а на второй сообщается информация об эффективности прокси, которая тем выше, чем дольше работает прокси и чем больше клиентов им пользуются.

Когда `apt-cacher` уже будет работать, вам останется внести еще одно маленькое изменение в конфигурацию APT, чтобы прокси-сервером могла пользоваться и команда `apt-get`. Для этого потребуется создать следующий новый файл и изменить `mein-apt-cacher` на хост-имя компьютера или адрес прокси-сервера:

```
// Файл /apt/apt.conf.d/01proxy
Acquire::http::Proxy "http://mein-apt-cacher:3142/";
```

Теперь все команды APT будут пользоваться новым прокси. Файлы, которых на нем еще нет, разумеется, нужно, как и раньше, скачивать из Интернета. Но если другой компьютер в сети также решит обновиться или установить пакет, который был незадолго до этого установлен на ином компьютере, необходимые пакеты уже будут в распоряжении (не забудьте прокомментировать первую строку, поставив символы //, — возможно, вы захотите поработать с ноутбуком в дороге и вам при этом понадобится установить обновления и пакеты, не имея доступа к прокси-серверу)!

Synaptic

Для APT существует множество графических пользовательских интерфейсов, и лучшая из таких программ на настоящий момент — Synaptic (рис. 10.3).

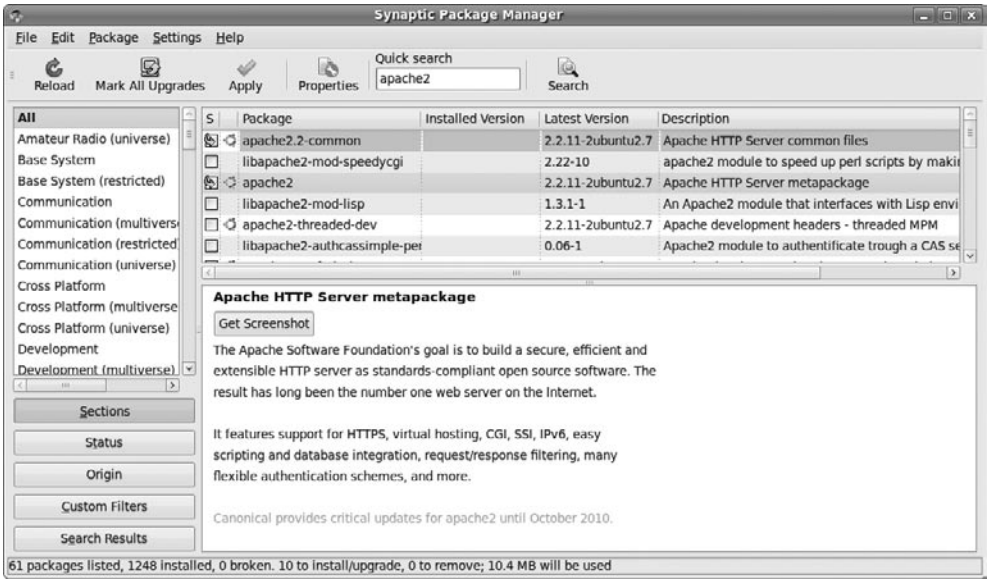

Рис. 10.3. Окно программы Synaptic

Поиск пакетов. Synaptic обладает сразу двумя поисковыми функциями: быстрым поиском по названиям и описаниям пакетов и обычным поиском, учитывающим другие критерии. Обе поисковые функции можно комбинировать друг с другом: тогда быстрый поиск будет производиться по результатам обычного поиска. Однако часто такая связь не нужна: удалите поисковый запрос в поле для быстро-

го поиска либо щелкните в столбце Поиск результатов на записи Все, чтобы вернуться к обычной функции поиска.

Установка. Если хотите установить определенный пакет, перейдите к инсталляции двойным щелчком кнопкой мыши. Если пакет зависит от другого пакета, то появляется диалоговое окно, где перечислены пакеты, которые также необходимо установить. Сама установка начинается после нажатия кнопки Принять, после чего вы должны «одобрить» сумму планируемых действий.

Если сначала нажать кнопку Настроить фильтр, а затем выбрать пункт Сохранить изменения в открывшемся меню, то программа выведет список всех пакетов, подготовленных для установки. Вы всегда можете отследить процесс, выбрав меню Файл ▸ История.

Управление репозиториями. Управлять репозиториями можно в диалоговом окне Настройки ▸ Репозитории. Здесь отображаются все известные репозитории. Установив флажки напротив определенных пакетов, вы можете быстро активизировать или деактивизировать нужные вам пакеты. С помощью EDIT можно изменять свойства имеющихся репозиториев, а с помощью ADD — добавлять новые репозитории.

Блокировка. Одновременно может работать лишь одна программа управления пакетами. Если вы попытаетесь запустить сразу две такие программы, то система выдаст предупреждение: unable to get exclusive lock (невозможно обеспечить исключающую блокировку). Это означает, что программа не может обратиться к внутренним системным файлам, отвечающим за управление пакетами. Необходимо завершить обе конфликтующие программы.

Иногда предупреждение о блокировке выводится и тогда, когда на первый взгляд работает только одна программа управления пакетами. Причина обычно заключается в том, что при завершении предыдущей программы не был правильно удален lock-файл. При необходимости просто удалите его сами:

```
root# rm /var/lib/dpkg/lock
```

Дефектные пакеты. Иногда бывает так, что при инсталляции или деинсталляции пакета возникает проблема и процесс не удается правильно завершить. В результате пакет помечается как дефектный. Synaptic и другие инструменты для управления пакетами отказываются работать, пока эта проблема не будет решена.

Для устранения проблемы нажмите в Synaptic в списке страниц кнопку Пользовательская настройка, а затем выберите пункт Дефектный. Тогда Synaptic отобразит список всех дефектных пакетов. Отметьте все пакеты, нажав сочетание Ctrl+A, щелкните в списке правой кнопкой мыши и выберите пункт контекстного меню Запретить для новой установки. Далее проведите установку заново, нажав кнопку Принять. Если опять возникнут проблемы, отметьте дефектные пакеты и удалите их.

10.6. PackageKit

PackageKit — это еще один пользовательский интерфейс для установки пакетов и управления ими. Важнейшая особенность этой программы заключается в том, что она совместима со многими системами управления пакетами, в том числе с APT, Yum и Zypper. PackageKit (либо его вариант для KDE — KPackageKit) применяется среди прочего в Fedora, Kubuntu и openSUSE, правда, в последней только для

выполнения обновлений в Gnome. Для получения прав администратора PackageKit обращается к PolicyKit (см. раздел 4.4).

При всей универсальности PackageKit нельзя не упомянуть фундаментальные недостатки этой программы: поисковая функция сведена к абсолютному минимуму, в ходе обновления или установки пакетов статусная информация или вообще не отображается, или ее очень мало. До тех пор пока программа сможет конкурировать с Synaptic или с модулем YaST Установка программного обеспечения, вам предстоит поработать. Не лишена недостатков и документация, в которой пока очень много пробелов.

Содержание и конфигурация системы

PackageKit состоит из нескольких частей, которые обычно содержатся в отдельных пакетах: к важнейшим компонентам относятся базовые функции или команды (пакет `packagekit`), интерфейс для работы с внутренней системой управления пакетами (например, `packagekitbackend-apt`) и графический пользовательский интерфейс (пакеты `gnome-packagekit` или `kpackagekit`). Названия пакетов отличаются в зависимости от дистрибутива.

PackageKit конфигурируется с помощью файлов из каталога `/etc/PackageKit/*`. Важнейшая настройка — это `DefaultBackend` в файле `PackageKit.conf`: она указывает, к какому *внутреннему интерфейсу* (backend) должна обращаться программа.

Для координирования операций PackageKit требуется демон `packagekitd`. При необходимости программа автоматически запускается командами PackageKit либо с соответствующих интерфейсов.

Используя команду `pkcon`, можно выполнять операции с пакетами с консоли либо автоматизировать их с помощью сценария. К сожалению, по этой команде отсутствует какая-либо документация (есть лишь для команды `pkcon --help`). Обратите внимание, что `pkcon` нельзя выполнять от лица администратора! Ее нужно запускать, будучи обычным пользователем. Если потом вам нужно получить права администратора, можно запустить программу PolicyKit (это правило действует и для графических пользовательских интерфейсов). Если вы хотите отслеживать с консоли, чем занята PackageKit, выполните команду `pkmon`.

Установка пакетов

Далее будет описан графический интерфейс PackageKit для работы с Gnome — я пользовался версией для Fedora 11. Для установки дополнительных пакетов либо для удаления имеющихся запустите программу `gpk-application`. Теперь можно выполнить поиск по имени пакета (рис. 10.4) или пролистать разные группы взаимозависимых пакетов (Наборы пакетов).

В меню Фильтр можно отфильтровать списки результатов, так как иногда они просто огромны. Например, можно отдельно отобразить уже установленные пакеты, пакеты для разработки и пакеты с графическим пользовательским интерфейсом. Нажатие кнопки Применить запускает инсталляцию или деинсталляцию выбранного пакета.

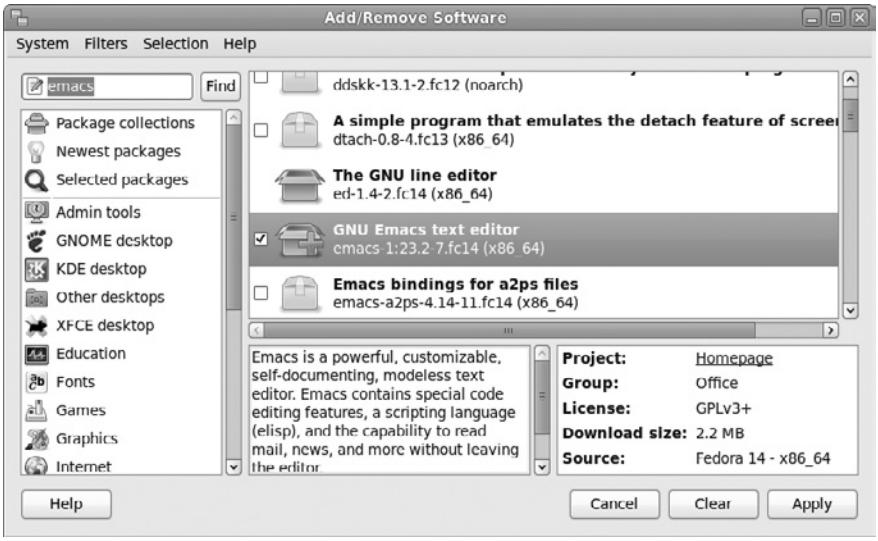

Рис. 10.4. Окно программы PackageKit, используемой для установки пакетов

Обновления

Программа `gpk-update-icon`, которая по умолчанию работает в Gnome, в тестовом режиме регулярно проверяет, доступны ли новые обновления. Если доступны, об этом сообщит значок на панели. Затем щелчком кнопкой мыши запускается программа `gpk-update-viewer`, выдающая подробную информацию о доступных обновлениях. После запуска процесса диалоговое окно состояния информирует вас о том, как протекает обновление.

10.7. TAR

Опытным пользователям Linux часто приходится устанавливать такое программное обеспечение, которое по форме не является пакетом для определенного дистрибутива. На различных серверах Linux в Интернете лежат гигабайты программ для Linux в архивах TAR.

Архивы, запакованные с помощью программмы `gzip`, обычно имеют расширения `*.tgz` или `*.tar.gz`. Архив устанавливается на компьютере благодаря программе `tar`.

```
root# tar -tzf archiv.tar.gz   (Отображение содержимого архива)
root# tar -xzf archiv.tar.gz   (Распаковка файлов относительно текущего каталога)
root# tar -xzf archiv.tar.gz "*.tex"  (Распаковка только файлов *.tex)
root# tar -xzf archiv.tar.gz -C dir   (Распаковка в один каталог)
```

Все чаще для создания архивов применяется новая, более мощная программа `bzip2`. Она распознает такие архивы по расширению `*.tar.bz2`. Чтобы отобразить или распаковать подобный архив, нужно использовать вместо `-z` параметр `tar -j`, то есть, например, `tar -tjf archiv.tar.bz2`.

Зачастую программы доступны только в виде исходного кода, и перед применением их еще надо скомпилировать (см. также раздел 11.2). Предполагается, что у вас должны быть инструменты разработки (gcc, make и т. д.), а также все необходимые библиотеки (devel-пакеты).

ВНИМАНИЕ

Если вы устанавливаете программные пакеты, распакованные из tar, они не учитываются ни одной из систем управления пакетами. Базы данных RPM «не знают» о программах, которые вы установили. По этой, да и по другим причинам, всегда старайтесь устанавливать пакеты, созданные специально для вашего дистрибутива.

10.8. Преобразование одних форматов пакетов в другие

Что делать, если вам необходим пакет в формате RPM, но работать с ним предполагается в Debian или Ubuntu? Что делать, если вы хотите просмотреть всего один файл из пакета RPM? В таких случаях используйте команду alien. Она преобразует пакеты из одних форматов в другие (RPM, DEB, архив TAR и Stampede SLP).

К сожалению, alien безошибочно работает только с простыми пакетами. Если же в пакете используются установочные сценарии либо конкретный формат пакетов имеет другие специфические свойства, установка преобразованных пакетов часто завершается неудачно. В принципе alien — это инструмент для профессионалов от Linux.

Желаемый формат пакета задается с помощью параметров --to-deb (Debian), --to-rpm (RPM) или --to-tgz (архив TAR). Команда alien должна выполняться администратором, чтобы можно было правильно настроить пользователя и права доступа к новому пакету. Следующая команда преобразует Debian в RPM:

```
root# alien --to-rpm paket.deb
```

Приведенные ниже команды показывают, как извлечь из RPM-пакета конкретный файл. Для этого пакет сначала преобразуется в архив TAR, затем с помощью команды tar из него извлекается файл, и благодаря команде less этот файл отображается. Разумеется, вместо tar можно воспользоваться файловым менеджером Konqueror или архивными программами, например ark или file-roller. Они отображают содержимое архива в удобном виде.

```
root# alien --to-tgz paket.rpm
root# tar -xzf paket.tgz ./usr/share/doc/packages/paket/TODO
root# less ./usr/share/doc/packages/paket/TODO
```

10.9. Управление параллельными установками

В Linux часто можно выбирать из нескольких равнозначных программ, выполняющих одни и те же задачи, а иногда даже использующих одни и те же команды. Это печатные

системы, редакторы, команды FTP, окружения Java и др. В некоторых ситуациях бывает целесообразно параллельно установить несколько вариантов или даже несколько версий одной и той же программы. Если при этом каждая программа будет находиться в собственном каталоге, то конфликты при установке не возникнут. Однако какая именно программа будет применяться для выполнения той или иной команды?

Для решения этого вопроса многие распространенные дистрибутивы используют концепцию, впервые появившуюся в Debian и основанную на символьных ссылках, находящихся в каталоге `/etc/alternatives`. В следующем списке указано, в каком пакете в разных дистрибутивах содержится каталог alternatives и соответствующая управляющая команда `update-alternatives`.

- Debian, Ubuntu — пакет `dpkg`;
- Red Hat, Fedora — пакет `chkconfig`;
- SUSE — пакет `update-alternatives`.

Эту концепцию лучше объяснить на примере. Предположим, что на одном компьютере установлены две версии Java. Программы Java выполняются с помощью класса `java`. В таком случае `/usr/bin/java` строится в виде ссылки на `/etc/alternatives/java`, а это, в свою очередь, еще одна ссылка, указывающая на нужную версию Java.

```
user$ ls -l /usr/bin/java
... /usr/bin/java -> /etc/alternatives/java
user$ ls -l /etc/alternatives/java
... /etc/alternatives/java -> /usr/lib/jvm/jre-1.5.0-sun/bin/java
```

Управление ссылками обычно происходит автоматически и осуществляется сценариями при установке пакета. При этом используется команда `update-alternatives`. В Red Hat и Fedora эта команда также доступна под названием `alternatives`.

Перечисление альтернатив

С помощью команды `update-alternatives --display` можно определить, какие версии определенной программы доступны и какая версия используется по умолчанию. В следующих строках показан результат для Java в Ubuntu 9.10 при установке двух версий Java. Строки `slave` (ведомые) относятся к командам, подчиненным данной программе, а также к страницам справки `man`. При изменении ссылки на команду программа `update-alternatives` автоматически обновляет все ведомые ссылки.

```
root# update-alternatives --display java
update-alternatives --display java
java - Auto-Modus
Сейчас ссылка указывает на /usr/lib/jvm/java-6-openjdk/jre/bin/java
/usr/lib/jvm/java-6-openjdk/jre/bin/java – приоритет 1061
Slave java.1.gz: /usr/lib/jvm/java-6-openjdk/jre/man/man1/java.1.gz
/usr/lib/jvm/java-6-sun/jre/bin/java – приоритет 63
Slave java.1.gz: /usr/lib/jvm/java-6-sun/jre/man/man1/java.1.gz
```

«Лучшей» версией, существующей в настоящее время, является `/usr/lib/jvm/java-6-openjdk/jre/bin/java`.

Обычно управление ссылками происходит автоматически: каждый установленный пакет получает указатель приоритета. Команда `update-alternative` при каждой (де)инсталляции активизирует альтернативу с наивысшим приоритетом.

Выбор других альтернатив

Команда `update-alternatives -config` определяет вариант, который станет активным. Она выдает список альтернатив, предлагаемых на выбор, и из них нужно выделить один. Теперь `update-alternatives` обновит ссылки. При необходимости можно снова перейти в автоматический режим с помощью команды `update-alternatives -auto`.

```
root# update-alternatives --config java
Для альтернативы java существует два варианта на выбор
(они находятся в /usr/bin/java).
Вариант  Путь                                        Приоритет  Статус
------------------------------------------------------------
* 0      /usr/lib/jvm/java-6-openjdk/jre/bin/java  1061       Автоматически
  1      /usr/lib/jvm/java-6-openjdk/jre/bin/java  1061       Вручную
  2      /usr/lib/jvm/java-6-sun/jre/bin/java      63         Вручную
Нажмите Enter, чтобы подтвердить сделанный выбор [*],
либо укажите номер варианта: 2
update-alternatives: использую /usr/lib/jvm/java-6-sun/jre/bin/java,
чтобы перевести /usr/bin/java (java) на управление вручную.
```

Внутрисистемная информация об управлении ссылками сохраняется в каталоге `/var/lib/alternatives` или `/var/lib/rpm/alternatives` в зависимости от дистрибутива.

11 Библиотеки, Java и Mono

В этой главе мы поговорим о библиотеках, которые необходимы для выполнения программ. Большинство программ для Linux существуют в скомпилированной форме и обращаются к различным библиотекам, которые динамически загружаются по мере необходимости. В разделе 11.1 рассказывается, как в Linux организовано управление библиотеками.

Если вы работаете с одним из распространенных дистрибутивов, то обычно вам приходится устанавливать скомпилированные программы (так называемые двоичные пакеты). Если же вы хотите поработать с совсем новой версией программы или с редко используемым приложением, может получиться так, что вы не найдете заранее скомпилированной версии программы, которую можно просто загрузить. В таких случаях вам придется загрузить исходный код (как правило, он пишется на языках C или C++) и самостоятельно скомпилировать программу. В разделе 11.2 даются некоторые вводные советы о том, как это сделать (не вдаваясь в детали необъятной темы «Разработка программ для Linux»).

В этой главе также рассказано, как выполнять в Linux программы, написанные для Java или .NET. Для этого потребуется установить среду выполнения Java или Mono. Во многих дистрибутивах это делается по умолчанию.

Сценарии, выполняемые интерпретатором, тематически выходят за рамки этой главы. В Linux используются различные сценарные языки, в том числе Perl, Python, PHP (для сайтов) и оболочка bash. В этом издании подробно описана оболочка bash (см. главу 8). Вкратце рассматривается и PHP: в главе 19 мы поговорим об установке сервера LAMP (Linux + Apache + MySQL + PHP).

11.1. Библиотеки

Практически все программы Linux используют одни и те же стандартные функции, например для обращения к файлам, вывода изображения на экран, поддержки X и т. д. Было бы нецелесообразно записывать все эти функции прямо в коде не самой большой программы — тогда файлы программ стали бы гигантскими. Вместо этого большинство программ Linux обращается к так называемым *разделяемым библиотекам*: при выполнении программы автоматически загружаются и требуемые библиотеки. В чем заключается преимущество? Если несколько программ

используют функции одной и той же библиотеки, то эту библиотеку необходимо загрузить лишь один раз.

Библиотеки играют ключевую роль, когда определяется, какие программы можно будет выполнять на компьютере. Если не хватает одной-единственной библиотеки (или в наличии имеется только старая версия), то прямо при запуске программы выводится сообщение об ошибке. Чтобы в таких случаях вы не оставались на произвол судьбы в недрах Linux, в этом разделе я предоставлю базовую информацию по библиотекам.

Динамическая связь программ с помощью ссылок. Большинство программ Linux при работе обращаются к разделяемым библиотекам. Так экономится место на диске (ведь двоичные файлы программ компактны) и меньше загружается оперативная память (поскольку один и тот же код не требуется грузить многократно). Замечание для программистов, работающих с Windows: разделяемые библиотеки сравнимы с DLL — динамически подключаемыми библиотеками.

Программы, связанные статическими ссылками. При компилировании программ библиотеки можно связывать и статическими ссылками. Это значит, что библиотечные функции интегрируются прямо в программный код. При этом программный файл увеличивается в размерах, зато не зависит от каких-либо библиотек. Это практично, если программу требуется передавать с компьютера на компьютер, — тогда она будет работать «с ходу», независимо от того, какие библиотеки уже установлены на компьютере. Иногда статическими ссылками связываются и некоторые простейшие команды для администрирования, чтобы такие команды оставались доступны и тогда, когда разделяемые библиотеки невозможно использовать из-за ошибок в конфигурации.

Форматы и версии библиотек

На протяжении истории Linux в библиотеки не раз вносились изменения, которые были столь же фундаментальны, сколь и несовместимы друг с другом. К числу таких изменений относится, например, замена формата A.OUT на ELF или замена библиотеки `libc` 5 версии на `glibc` версии 2.*n*, причем к последней можно обращаться и как `libc` 6 (в настоящее время актуальна версия `glibc` 2.10).

В обоих случаях замены библиотек были технически оправданны. Новые форматы или версии позволяют с большей легкостью управлять библиотеками и функциями, обеспечивают более полную совместимость различных платформ Linux (Intel, Sun-Sparc, DEC-Alpha) и пр.

Однако при замене возникают проблемы, связанные с тем, что скомпилированные программы могут выполняться только тогда, когда в системе установлены нужные библиотеки и система может их найти. Если вы попытаетесь выполнить программу для `glibc` в старом дистрибутиве, в котором `glibc` не поддерживается, то получите загадочное сообщение об ошибке следующего содержания:

```
root# programmxy
bash: /usr/local/bin/programmxy: No such file or directory
```

ПРИМЕЧАНИЕ

Из-за того, что в настоящее время имеются проблемы с поддержкой библиотеки glibc, готовящаяся к выходу версия Debian Squeeze предположительно будет использовать не оригинальную библиотеку glibc, а полностью совместимую библиотеку eglibc. Эта замена не причинит неудобств ни конечным пользователям, ни разработчикам. Причины, по которым была предпринята такая замена, описаны на следующей странице: http://lwn.net/Articles/332000/.

Автоматическая загрузка библиотек

Если вы работаете с Linux только как пользователь, а не как программист, то вы столкнетесь с библиотеками лишь в тот момент, когда какой-то из них будет недоставать. Обычно такие проблемы возникают, если вы постепенно устанавливаете новую программу. При попытке запустить ее выводится сообщение об ошибке, указывающее на отсутствие определенной библиотеки. Часто актуальные версии программ ссылаются на новейшие версии соответствующих библиотек, которые у вас, возможно, еще не установлены. Со старыми программами вероятен прямо противоположный случай. Возможно, они еще ссылаются на устаревшие библиотеки, которые уже не поддерживаются в вашем дистрибутиве.

Определение списка библиотек

Команда `ldd` передается в качестве параметра, который добавляется к полному имени программы. В ответ `ldd` перечисляет все библиотеки, которые нужны для выполнения программы. Кроме того, указывается, где находится подходящая библиотека и какие библиотеки доступны только в устаревшей версии.

```
user$ ldd /bin/cp
      linux-vdso.so.1 => (0x00007ffff83fe000)
      libselinux.so.1 => /lib/libselinux.so.1 (0x00007fd0eff6f000)
      libacl.so.1     => /lib/libacl.so.1 (0x00007fd0efd67000)
      libc.so.6       => /lib/libc.so.6 (0x00007fd0ef9f5000)
      libdl.so.2      => /lib/libdl.so.2 (0x00007fd0ef7f1000)
      libattr.so.1    => /lib/libattr.so.1 (0x00007fd0ef5ec000)
      /lib64/ld-linux-x86-64.so.2 (0x00007fd0f018b000)
```

Что касается программ X-, KDE- и Gnome, здесь список библиотек гораздо обширнее. Именно по этой причине эти программы запускаются достаточно долго.

Если `ldd` возвратит результат `not a dynamic executable`, то вы имеете дело с программой, которая уже содержит все необходимые библиотеки, то есть это программа со статическими ссылками.

Названия библиотек

Краткая информация о наименованиях библиотек: окончание .so указывает, что мы имеем дело с *разделяемой библиотекой*, окончание .a определяет *статическую библиотеку*. Следующая цифра указывает номер основной версии. Например, `ls` требует библиотеку `libc` версии 6.

Каталоги, в которых обычно располагаются библиотеки (например, /lib, /usr/lib, /usr/local/lib, /usr/X11R6/lib и /opt/lib), часто содержат ссылки, связывающие основную версию библиотеки с той, что установлена на вашем компьютере. Так, для cp (см. выше) требуется библиотека ld-linux-x68-64.so.2. Но на самом деле на компьютере установлена версия ld-2.9.so, совместимая «снизу вверх».

```
user$ ls -l /lib/ld*
... /lib/ld-2.9.so
... /lib/ld-linux.so.2 -> /lib32/ld-linux.so.2
... /lib/ld-linux-x86-64.so.2 -> ld-2.9.so
```

Запуск программ

При запуске программ нужно найти и загрузить все библиотеки — за это отвечает так называемый *компоновщик времени выполнения*. При этом учитываются все каталоги, указанные в переменной окружения LD_LIBRARY_PATH. Эти каталоги разделяются двоеточиями.

Кроме того, компоновщик интерпретирует файл /etc/ld.so.cache. Это двоичный файл, содержащий всю важную информацию о библиотеке (номера версий, пути доступа и т. д.). Он нужен только для того, чтобы сэкономить время, которое компоновщик в противном случае потратил бы на поиск библиотек.

Файл /etc/ld.so.cache создается программой ldconfig, которая, в свою очередь, интерпретирует /etc/ld.so.conf. В этом файле обычно содержится список всех библиотечных каталогов или список ссылок на другие файлы с каталогами (каталоги /lib и /usr/lib учитываются в любом случае и поэтому отсутствуют в ld.so.conf или других конфигурационных файлах. Если кроме /lib и /usr/lib не придется учитывать никаких каталогов, то ld.so.conf можно вообще опустить).

В некоторых дистрибутивах команда ldconfig выполняется при каждом запуске компьютера, чтобы гарантировать максимально обновленное состояние файла кэша. Ее всегда нужно выполнять в тех случаях, когда вы вручную устанавливаете новую библиотеку, иначе система «не увидит» библиотек. Если библиотеки находятся в новом каталоге, нужно соответствующим образом дополнить файл /etc/ld.so.conf. При установке пакетов с библиотеками эти задачи обычно выполняет менеджер пакетов.

32- и 64-битные библиотеки

Большинство распространенных дистрибутивов в настоящее время существуют в как минимум двух вариантах сборки: для 32-битных процессоров, совместимых с Intel/AMD, и для 64-битных процессоров, совместимых с Intel/AMD. Разумеется, для 32-битных процессоров предусмотрены только 32-битные библиотеки. Однако, как ни жаль, того же нельзя сказать о 64-битных дистрибутивах: были и остаются программы, которые не компилируются для 64-битных систем. Наиболее известная программа такого рода — Acrobat Reader компании Adobe.

Для выполнения 32-битных программ в 64-битных дистрибутивах вам потребуются 32-битные библиотеки. Чтобы можно было избежать конфликтов, библиотеки устанавливаются в различные каталоги. Среди профессионалов в Linux этот

метод называется *мультиархитектура*, или *би-архитектура*, так как параллельно поддерживается несколько (или две) архитектуры процессоров. В большинстве дистрибутивов встречаются каталоги /lib32 или /lib64, позволяющие не смешивать библиотеки с различной разрядностью. Такая двойственность, разумеется, связана с определенными недостатками: при установке многочисленных библиотек в двух экземплярах тратится больше дискового пространства, кроме того, при этом осложняется техническая поддержка.

Предварительное связывание

При запуске программы, которая обращается к динамическим библиотекам, нужно установить соединение между программой и библиотеками. Этот процесс именуется *связыванием*. При работе со сложными программами на связывание тратится немало времени.

Программа prelink может заранее выяснить информацию, необходимую для связывания. В первый раз этот процесс длится очень долго. При этом требуется просмотреть все исполняемые программы. Файл /etc/prelink.conf определяет, какие каталоги с программами и библиотеками учитывает prelink. Другие функции можно установить в /etc/sysconfig/prelink или /etc/default/prelink (Debian, Ubuntu).

В дальнейшем каждая подготовленная таким образом программа будет обращаться к своим библиотекам гораздо быстрее, а значит, и быстрее запускаться. Такое ускорение особенно заметно в OpenOffice или в программах KDE, на запуск которых теперь потребуется вполовину меньше времени, чем раньше. В любом случае данные предварительного связывания необходимо обновлять при каждом обновлении библиотеки.

Предварительное связывание имеет еще один недостаток — эта операция изменяет исполняемые файлы всех программ и библиотек. Кроме того, вы уже не сможете контролировать целостность таких файлов (то есть не сможете убедиться, что после установки файлы остались прежними). Когда угодно можно отменить любые изменения, внесенные в ходе предварительного связывания, с помощью команды prelink -ua. Базовая информация по предварительному связыванию сообщается в справке man, а также по следующему адресу: http://www.gentoo.org/doc/en/prelink-howto.xml.

Debian, Ubuntu. Чтобы пользоваться предварительным связыванием, необходимо установить пакет prelink и задать в файле /etc/default/prelink настройку PRELINKING=yes. Предварительное связывание будет ежедневно выполняться в качестве одной из задач Cron.

Red Hat, Fedora. Функция предварительного связывания установлена по умолчанию. Данные регулярно обновляются (задача Cron /etc/cron.daily/prelink, конфигурационный файл /etc/sysconfig/prelink).

SUSE. Для того чтобы использовать предварительное связывание, необходимо установить пакет prelink и указать в файле /etc/sys-config/prelink настройку PRELINKING=yes. Затем prelink будет выполняться модулем YaST после установки любой новой программы или библиотеки (сценарий /sbin/conf.d/SuSEconfig.prelink).

11.2. Как самостоятельно компилировать программы

Существует как минимум две причины, по которым вам, возможно, придется компилировать программы для Linux самостоятельно: либо вы не найдете для вашего дистрибутива двоичного пакета с готовой скомпилированной нужной программой, либо вам может понадобиться скомпилировать программу с конфигурацией, отличающейся от стандартной.

Условия. Прежде чем приступить к делу, потребуется выполнить несколько предварительных условий.

- Установить *набор компиляторов проекта GNU* (`gcc` и `gcc-c++`). В них содержатся компиляторы для C и C++.
- Инсталлировать вспомогательные инструменты, в частности `make`, `automake`, `autoconf` и т. д. Эти программы нужны для конфигурации и выполнения компилирования.
- Установить версии различных библиотек, предназначенных для разработки. Названия соответствующих пакетов обычно заканчиваются на `-devel` (Red Hat, SUSE) или `-dev` (Debian, Ubuntu). Например, в `glibc-devel` или `libc6-dev` содержатся инструментальные файлы для базовой библиотеки `glibc`. Понадобятся ли вам еще какие-нибудь инструментальные пакеты, зависит от конкретной программы, которую вы хотите скомпилировать. Если компилятор или компоновщик будет выдавать сообщения об ошибках, гласящие, что для работы не хватает библиотек, это однозначно указывает на то, что вы пропустили какой-то важный инструментальный пакет.

Debian, Ubuntu. Метапакет определяет взаимозависимости между важнейшими инструментальными пакетами. Поэтому при установке `build-essential` автоматически инсталлируются многие другие пакеты, которые вместе образуют базовую конфигурацию для разработки программ на C/C++.

Fedora. Для выполнения основных предварительных условий лучше всего выполнить команду `yum groupinstall development-tools`. Чтобы разрабатывать программы для KDE и Gnome, также существуют специальные группы пакетов: `kde-software-development` и `gnome-software-development`.

SUSE. Здесь в модуле YaST устанавливаются все пакеты схемы «Базовая среда разработки». Если вы собираетесь писать программы для KDE или Gnome, установите подборки KDE- или GNOME-DEVELOPMENT. Если предпочитаете `zypper`, выполните команду `zypper install -t` *`шаблон`* `devel_basis` или `devel_kde bzw. devel_gnome`.

Распаковка кода

Исходный код обычно лежит в Интернете в виде TAR-архивов (расширение `*.tar.gz`, `*.tgz` или `*.tar.bz2`). После того как скачаете код, распакуйте его в локальный каталог:

```
user$ tar xzf имя.tar.gz       (Для .gz или .tgz)
user$ tar xjf имя.tar.bz2      (Для .bz2)
user$ cd имя
```

Пакеты SRMP

Кроме TAR-архивов существуют и другие пакеты, содержащие именно тот исходный код, из которого была скомпилирована определенная программа вашего дистрибутива. Такие пакеты с исходным кодом, как правило, находятся на FTP-серверах вашего дистрибутива. Файлы с исходным кодом для тех дистрибутивов, что работают на основе RPM-пакетов, находятся в пакетах SRMP с расширением *.src.rpm. Для установки выполните, как обычно, rpm -i:

```
root# rpm -i имя.src.rpm
```

Фактическое место расположения этого кода зависит от конкретного дистрибутива: в Fedora и Red Hat — это /usr/src/redhat/, а в SUSE — это /usr/src/packages/.

- SOURCES/*имя*.tar.*xxx* содержит сам исходный код. Распаковка TAR-архива производится так, как это описано выше.
- SOURCES/*имя-xxx*.patch (Red Hat) или SOURCES/*имя*.dif (SUSE) содержит дистрибутивно-специфичные изменения исходного кода. Если вы хотите соответствующим образом изменить (пропатчить) файлы кода, выполните следующую команду:

  ```
  user$ cd имя-исходный_каталог
  user$ patch < имя.dif/patch
  ```

  В зависимости от того, какой каталог в данный момент текущий и какая информация о каталоге записана в патч-файле, нужно дополнительно указать параметр -p1 (см. справку man patch).
- SPECS/*имя*.spec содержит описание пакета, также служащее для создания RPM-пакетов. (Если вы хотите сделать RPM-пакет из самостоятельно скомпилированной программы, используйте для этого команду rpmbuild, на которой мы более подробно останавливаться не будем. Прочтите man rpmbuild!)

Пакеты с исходным кодом для Debian

В дистрибутивах, построенных на основе Debian, исходный код находится в нескольких файлах, которые нужно установить в текущий каталог, лучше всего с помощью команды apt-get source.

```
user$ apg-get исходное имя_пакета
```

Теперь в текущем каталоге вы найдете три новых файла и один каталог:

- *имя_пакета*.dsc содержит краткое описание пакета;
- *имя_пакета*.orig.tar.gz содержит TAR-архив с первоначальным исходным кодом, написанным разработчиком программы;
- *имя_пакета*.diff.gz включает в себя все изменения оригинального исходного кода, характерные для Debian или Ubuntu;
- новый каталог *имя_пакета*/ содержит, наконец, информацию, уже извлеченную из *имя_пакета*.diff.gz, причем все изменения из DIFF-файла уже выполнены.

Компилирование программы

Для компилирования и установки программ нужны три команды, которые иногда называются «три в одном»: `./configure`, `make` и `make install` (далее они будут описаны более подробно). При этом должен быть открыт каталог с исходным кодом.

Сценарий configure. Это сценарий, который проверяет, доступны ли все необходимые программы и библиотеки. Поскольку сценарий находится в локальной папке, его нужно выполнять в виде `./configure`. Этот сценарий адаптирует файл `Makefile`, содержащий все команды, для компилирования и компоновки различных файлов кода. В некоторых (обычно небольших) программах `configure` может отсутствовать. В таком случае сразу выполняйте `make`.

```
user$ ./configure
```

Команда make. Она инициирует обработку команд компилирования и компоновки. Теперь вы увидите (иногда практически бесконечные) уведомления и предупреждения о различных процессах компилирования, переполняющие окно консоли. Если не происходит ошибок, можете просто игнорировать эти сообщения. В результате в каталоге с исходным кодом должен появиться исполняемый файл *имя*.

```
user$ make
```

Во многих случаях на этом этапе уже можно запускать программу (команда `./`*имя*) и тестировать ее. Однако обратите внимание, что некоторые службы, в частности сетевые, требуют специальной конфигурации и обычно должны запускаться с помощью сценариев Init-V!

Команда make install. На заключительном этапе мы должны обеспечить всем пользователям доступ к программе. Нам потребуется скопировать файлы программы и, возможно, файлы библиотек, в общедоступные каталоги. Для этого необходимы права администратора. Перед выполнением `make install` следует убедиться, что нужная программа еще не установлена! В ином случае ее предварительно потребуется деинсталлировать.

```
root# make install
```

Возможные проблемы

При компилировании программ могут возникать разнообразные проблемы. Чаще всего они связаны с отсутствием каких-либо вспомогательных компилирующих инструментов или библиотек. Как правило, такие проблемы идентифицирует уже `configure`, и устранить их совсем не сложно — нужно просто установить недостающий пакет.

Ситуация осложняется, когда `configure` требует библиотеку, которая недоступна в вашем дистрибутиве, либо у вас нет необходимой версии этой библиотеки. Тогда вам придется искать в Интернете нужную библиотеку или, возможно, сначала скомпилировать библиотеку. Что касается сложных программ, например Apache или mplayer, в Интернете имеются точные руководства по компилированию, где пошагово описано, что и в какой последовательности необходимо установить и скомпилировать.

Еще хуже, если в процессе компилирования возникает синтаксическая ошибка и процесс компилирования обрывается сообщением об ошибке. Часто причиной этому служит не программная ошибка, а несовместимость вашего компилятора и кода. Некоторые программы можно скомпилировать только определенной версией `gcc` (часто *не* новейшей!), то есть проблема решается установкой нужной версии компилятора. На этот случай в Интернете или в файлах `README`, сопровождающих исходный код, часто можно найти точные указания.

Управление пакетами. Самостоятельно скомпилированные программы и библиотеки вносят путаницу в управление пакетами. Проблема заключается в том, что хотя программа *abc*, скомпилированная вами, уже установлена в системе, база данных RPM или DEB ничего об этом «не знает». Если теперь вы попытаетесь установить пакет *xyz*, зависящий от *abc*, то получите сообщение об ошибке, так как, по мнению системы, не выполняются межпакетные взаимозависимости. Однако установить пакет с помощью `rpm` можно благодаря параметрам `nodeps` и `-force`.

Самое красивое решение — не устанавливать программу с помощью `make install`, а сначала подготовить и установить пакет. Для этого вам нужно познакомиться с командами, отвечающими за упаковку пакетов. В целом этот метод очень неудобный, в особенности если программа должна не один раз тестироваться и компилироваться заново.

Примеры

HelloWorld на языке C. Размеры этой книги не позволяют дать даже вводную информацию о программировании на C и C++. Однако, поскольку в прошлом мне постоянно приходилось отвечать на связанные с этим вопросы, я кратко расскажу, как написать и скомпилировать классическую программу Hello World на C и C++. Для C-версии запишем в редакторе следующие строки в файл `hello.c`:

```
// hello.c
#include <stdio.h>
int main(void)
{
  printf("Hello World!\n");
}
```

С помощью следующих команд программа компилируется и выполняется:

```
user$ gcc -o hello hello.c
user$ ./hello
Hello World!
```

HelloWorld на языке C++. Аналогичный код на C++ выглядит так:

```
// hello.cpp
#include <iostream>
int main()
{
  std::cout << "Hello World!\n";
  return 0;
}
```

Теперь для компилирования применяется не gcc, а g++:

```
user$ g++ -o hello hello.cpp
user$ ./hello
Hello World!
```

СОВЕТ

Если вам нужна удобная среда разработки для программирования на C или C++ под Linux, попробуйте KDevelop (KDE) или Anjuta (Gnome). Возможная альтернатива — среда разработки Eclipse для Java, которая, однако, первоначально предназначалась не для C или C++.

Настоящие приверженцы UNIX/Linux отличной средой разработки считают редакторы Vi или Emacs.

11.3. Java

На персональных компьютерах Java важен в первую очередь как плагин для браузера, а также для выполнения различных дополнительных функций OpenOffice. Кроме того, значение Java повышается в силу постоянно растущего количества программ Java, не зависящих от конкретной платформы. Гораздо более важную роль Java играет на серверах Linux, предлагающих сайты или веб-службы, написанные на Java (Tomcat, Jakarta и т. д.). Java интересен и для многих школьников и студентов, которые часто учатся программированию на этом языке и, в частности в среде разработки Eclipse, выполняют проектные работы и т. д.

Стандартная версия Java (Java SE), Java-компилятор javac, части инструментария для разработки на Java (JDK), виртуальная машина Java (JVM) и библиотека классов JDK с 2006 или 2007 года предоставляются в виде открытого кода по лицензии GPL (версия 2).

Варианты Java

Кроме официальных пакетов Java, выпущенных фирмой Sun, которые предоставляются бесплатно и даже входят в состав некоторых дистрибутивов в форме двоичных пакетов, существуют варианты, состоящие только из открытого кода: они основаны на GPL-коде от Sun, но включают и элементы из других проектов с открытым кодом. Это IcedTea и OpenJDK.

Java на основе OpenJDK, по умолчанию устанавливаемый во все большем количестве дистрибутивов, демонстрирует практически 99 %-ную совместимость с Java от Sun. Оставшийся процент следует списать на те компоненты Java, которые невозможно предоставить в виде открытого кода из-за проблем с лицензией и для которых еще не написаны свободно распространяемые замены.

В начале 2009 года компания Oracle объявила о намерении купить Sun. Если этой покупке никто не помешает (по антимонопольным мотивам), Oracle станет владельцем «официального» языка Java. Как это отразится на будущем Java, пока совершенно неясно.

Версии и номенклатура Java

Если вам кажется, что нумерация, принятая в ядре Linux, слишком непонятна, это значит, что вы еще не работали с номенклатурой и номерами версий Sun-Java.

Только представьте себе: JRE 1.5, Java 2 Standard Edition 1.5 и Java 5 — это одна и та же версия Java. В табл. 11.1 и 11.2 обобщены важнейшие сокращения и номера версий, используемые в Java. По-видимому, никто (даже компания Sun на своем официальном сайте) не приведет в систему те названия и номера версий, которые действительно являются официальными.

Таблица 11.1. Сокращения Java

Сокращение	Значение
JVM	Виртуальная машина Java (выполняет программы Java)
JRE	Среда исполнения Java (выполняет программы Java; содержит JVM, а также множество библиотек Java)
JDK	Инструментарий для разработки на Java (для разработки Java-программ)
Java SE = JSE	Стандартная версия Java (для применения на локальном компьютере)
Java EE = JEE	Версия Java для создания распределенных приложений масштаба предприятия (для применения на сервере)
Java ME = JME	Микроверсия Java (применяется на КПК и сотовых телефонах)
JavaFX	Версия Java для веб-приложений (сравнима с Adobe Flash и Microsoft Silverlight)

Таблица 11.2. Номера версий Java SE

Год	Официальное название	Версия JRE/JDK	Внутренний номер версии
1996	Java 1	1.0	1.0
1997	Java 1.1	1.1	1.1
1998	J2SE 1.2 (Java 2)	1.2	1.2
2000	J2SE 1.3 (Java 3)	1.3	1.3
2002	J2SE 1.4 (Java 4)	1.4	1.4
2004	JSE 5 (Java 5)	5	1.5
2006	Java 6 SE	6	1.6
Планируется в 2010	Java SE 7	7	1.7

Как определить, какая версия Java установлена на компьютере

Чтобы определить, какая версия Java установлена на вашем компьютере, выполните команду, приведенную ниже. В следующих строках показан результат для Fedora 11. Если команда `java` недоступна, это значит, что Java вообще не установлен.

```
user$ java -version
java version "1.6.0_0"
OpenJDK Runtime Environment (IcedTea6 1.5) (fedora-22.b16.fc11-i386)
OpenJDK Client VM (build 14.0-b15, mixed mode)
```

11.4. Mono

.NET-Framework — это огромная библиотека классов, во многом сходная с библиотеками для Java. С языком C#, разработанным для программирования в .NET, специалисты по Java также осваиваются без проблем. .NET-Framework и C# вместе

образуют объектно-ориентированный фундамент, применяемый в программировании для Windows и веб-программирования, и обойтись без этого фундамента (по крайней мере в мире Microsoft) уже невозможно.

Какое отношение все это имеет к Linux? Хотя сообщество Linux, откровенно говоря, мало связано с Microsoft, вышеописанная концепция пришлась по душе некоторым специалистам, разрабатывающим свободное ПО. Тогда появилось название Mono, объединившее разработки с открытым кодом, использующие язык C# и в значительной мере .NET-Framework. Разработку проекта Mono начала компания Ximian. Позже эту фирму приобрела Novell, и теперь Novell — важнейший спонсор проекта Mono. Кроме того, существует партнерское соглашение между Microsoft и Novell, что позволяет исключить потенциальные проблемы, связанные с патентами (разумеется, только для клиентов Novell).

Тем временем проект Mono хорошо зарекомендовал себя на практике и по умолчанию устанавливается во многих дистрибутивах. Некоторые проекты, связанные с Gnome, в частности Beagle (поиск на локальном компьютере, см. подраздел «Поисковики для персональных компьютеров (Beagle, Tracker, Strigi)» раздела 3.4), F-Spot (работа с изображениями) и Tomboy (электронная записная книжка), также основаны на Mono.

Совместимость между Mono и .NET-Framework так или иначе не совсем полная, то есть нельзя гарантировать, что вы сможете свободно работать в Linux с любой программой, скомпилированной для Windows. Исчерпывающая информация по проекту Mono и его текущему состоянию (то есть о степени совместимости с C# и .NET-Framework) предлагается на следующем сайте: http://www.mono-project.com/.

Проблемы, связанные с патентами, и их решение

Разумеется, проект Mono имеет свои недостатки. Противники указывают на серьезную зависимость проекта от компании Microsoft, которая могла бы попытаться одолеть его в сфере патентования программного обеспечения. Однако в последнее время стало казаться, что сама Microsoft очень оптимистично настроена по отношению к Mono. Платформа .NET-Framework отличается определенной независимостью и усиливает свои позиции относительно Java. Кроме того, язык C# и .NET-Framework описаны в стандартах Европейской Ассоциации производителей компьютеров — как минимум те элементы, что описаны в этих стандартах, должны иметь надежную правовую почву.

Наиболее активное сопротивление Mono долгое время оказывал Red Hat. Ключевое значение имела коллекция патентов, которыми владела компания Open Invention Network, ведь именно с ее учетом было принято решение интегрировать Mono в Fedora. Организация Open Invention Network поддерживается различными фирмами, разрабатывающими свободное ПО, владеет множеством патентов, которые предназначены для защиты от патентных исков, направленных против проектов Linux (в том числе Mono). Даже Debian намерен интегрировать Mono в свою новую версию Squeeze.

В то же время Red Hat/Fedora снова отказываются от использования Mono. В Fedora 11 пакеты Mono еще содержатся, но не устанавливаются по умолчанию, а Fedora 12 вообще может выйти без поддержки Mono. Мне так и не удалось найти общедоступного обоснования такого шага.

Внутренняя организация Mono

Обычно устанавливается в виде пакета `mono-`*`xxx`*, важнейшая часть которого называется `mono-core`. В ней содержится в том числе компилятор C# mcs, *Виртуальная машина Mono*, собрание .NET-совместимых библиотек (файлы `*.dll` в каталоге `/usr/lib/mono/gac`), а также некоторые конфигурационные файлы Mono (каталог `/etc/mono`).

Программы Mono имеют расширение EXE и предоставляются, подобно программам Java, в виде байт-кода. Чтобы выполнить программу Mono, вы передаете имя EXE-файла команде `mono`. Поскольку на практике этот процесс достаточно сложен, для запуска пользовательских программ Mono существуют маленькие сценарии (посмотрите, например, файл `/usr/bin/f-spot`).

Для Mono-разработчиков создан графический пользовательский интерфейс MonoDevelop, который сначала был построен по образу программы Windows SharpDevelop, но теперь уже не зависит от нее.

12 Система X

Система X Window (коротко — X) представляет собой собрание функций и протоколов, с помощью которых на монитор выводится графическая информация и происходит управление мышью и клавиатурой. Эти функции доступны и при работе в сети. Первая свободная реализация системы X Window появилась в рамках проекта XFree86. После того как у разработчиков XFree86 возникли разногласия, а также после изменений в версии 4.4., сделавших систему несовместимой с лицензией GPL, возник альтернативный проект X.org.

В настоящее время все распространенные дистрибутивы Linux используют сервер X.org. В этой главе будут рассмотрены версии X.org 7.4 и 7.5, а также версия сервера X.org 1.6. Здесь будут описаны различные аспекты конфигурации X-сервера, в том числе интеграция двоичных драйверов для ATI/AMD и NVIDIA. Другие важные вопросы, рассматриваемые в этой главе, — это функции трехмерного интерфейса рабочего стола и управление шрифтами. Более подробная информация по X или X.org находится на сайте http://xorg.freedesktop.org/wiki/.

12.1. Основы

Система X Window. Система включает, по существу, лишь базовые функции для рисования точек, прямоугольников и т. д. X также содержит сетевой протокол, позволяющий выполнять программу для X на компьютере *A*, а результаты передавать по сети и отображать на компьютере *B*. Версия X.org 7.5, новейшая по состоянию на 2009 год, основана на X11R7.5. Разработка системы X Window была начата в Массачусетском технологическом институте (Massachusetts Institute of Technology — MIT).

На основе X построен графический пользовательский интерфейс для Linux. Однако сама X не предоставляет никакого пользовательского интерфейса! «Внешний вид» программ для X и способы работы с ними зависят от того, какие библиотеки использовались при программировании (например, GTK у программ Gnome, QT — у программ KDE), а также от того, какой диспетчер окон активен.

X-сервер. X-сервер представляет собой интерфейс между системой X Window и оборудованием (графической картой, мышью). Сервер имеет модульную организацию: это означает, что к серверу добавляются модули, обеспечивающие выполнение специфических функций для каждой графической карты.

Дополнительные модули X. Стандартный функционал X-сервера можно расширять с помощью различных дополнительных модулей, которые отвечают, например, за трехмерную графику, отображение видео и т. д.

Диспетчер окон. Это программа для X, отвечающая за управление окнами. С помощью диспетчера окон вы можете запускать новые программы, переходить между окнами, перемещать и закрывать окна и т. д., то есть выполнять достаточно тривиальные задачи. И все же не забывайте, что эти задачи выполняет именно диспетчер окон, а не сама система X. В KDE и Gnome используются собственные диспетчеры окон.

Дилемма с драйверами

Прежде чем перейти к следующим разделам, где будет подробно рассказано о конфигурации и эксплуатации системы X, хотелось бы затронуть основную проблему X: недостаток свободно распространяемых драйверов для современных графических карт (глава была написана в августе 2009 года; ситуация уже, возможно, изменилась на тот момент, когда вы читаете эту книгу).

Подавляющее большинство всех современных ПК и ноутбуков использует графические чипы трех следующих фирм (по алфавиту): ATI/AMD, Intel и NVIDIA, причем графические чипы Intel доступны только в виде полноценных чипсетов (но не в виде отдельных графических карт). Следует также упомянуть о все более широком применении графических решений VIA — прежде всего на недорогих материнских платах в составе набора микросхем.

Сначала хорошая новость: свободные графические драйверы, содержащиеся в X, совместимы с большинством распространенных графических карт. А теперь плохая новость: скорость работы с такими драйверами пока достаточно низкая, к тому же нельзя пользоваться многими дополнительными функциями графических карт. Сюда относятся, в частности, функции 3D, целенаправленное управление несколькими выходами и т. д.

Решить проблему помогут двоичные драйверы, бесплатно предоставляемые ATI/AMD или NVIDIA. Правда, эти драйверы построены не на открытом коде, из-за чего возникает множество проблем, которые я опишу далее. Ниже мы рассмотрим, как в настоящее время обстоят дела с драйверами, разделив все вопросы на несколько частей, в соответствии с производителями графических карт.

ATI/AMD

Для графических карт ATI/AMD существуют как свободные драйверы (radeon, radeonhd), так и некоторые двоичные драйверы ATI/AMD (fglrx). Хорошего во всех отношениях драйвера пока, к сожалению, нет: в свободно распространяемых драйверах для современных графических микросхем отсутствует поддержка 3D (в частности, для чипсетов R600 и R700). А двоичные драйверы, в свою очередь, поддерживают *только* современные модели (но не рассчитаны на работу с сериями R*xxx*–R500, включающими марки Radeon 9000, X1 и X2); кроме того, версии X.org на несколько месяцев опережают актуальные версии двоичных драйверов, то есть в некоторых ситуациях вам не удастся воспользоваться новейшей версией X.org.

Однако есть на что надеяться: с осени 2007 года компания ATI/AMD сотрудничает с разработчиками свободного ПО и опубликовала значительную часть спецификации микросхем. Пусть свободные драйверы разрабатываются несколько медленнее, чем ожидалось, но появление хорошего свободно распространяемого драйвера, который мог бы работать с современными графическими картами ATI/ADM, — всего лишь вопрос времени.

Intel

Графические чипы Intel в настоящее время характеризуются, пожалуй, наилучшей поддержкой со стороны свободного ПО. Причина этого состоит в том, что Intel прекрасно наладила сотрудничество с сообществом разработчиков свободных программ. Поэтому драйверы intel, разработанные Intel, стали неотъемлемой частью семейства драйверов X.org.

Так или иначе в 2009 году драйвер intel был значительно изменен. В процессе перехода со старой версии на новую временно возникли серьезные сложности со стабильностью и скоростью работы, сильно затронувшие, например, блестяще проработанные драйверы для Ubuntu 9.04. К тому моменту, как вы будете читать эту книгу, все проблемы, вероятно, уже будут устранены.

Обратите внимание, что чипсеты GN40 и GMA500, также известные под названием Poulsbo и иногда применяемые в нетбуках, в настоящее время поддерживаются в ограниченном объеме или *не* поддерживаются. Компания Intel применила в этих чипсетах стороннюю, купленную технологию, осложняющую работу с драйверами. Советы о том, как при необходимости все же запустить чипсет GMA500, даются на следующих сайтах:

- http://www.pro-linux.de/news/2009/14763.html;
- http://mok0.wordpress.com/2009/05/25/ubuntu-on-the-dell-mini-10-2/;
- http://www.phoronix.com/scan.php?page=news_item&px=NzI2OA.

NVIDIA

Компания NVIDIA до сих пор придерживается мнения, что лицензионные соглашения с другими компаниями и патенты исключают возможность разработки свободно распространяемых драйверов и не позволяют опубликовать техническую документацию о внутренних интерфейсах. По этой причине она предоставляет бесплатный двоичный драйвер nvidia. Еще недавно этот драйвер был получше качеством, чем аналоги ATI/AMD, но и для него характерны те же недостатки, что и для других драйверов, не распространяемых свободно. Имеющиеся свободно распространяемые драйверы для карт NVIDIA поддерживают функции 2D (nv, nouveau), причем на экспериментальном уровне драйвер nouveau поддерживает и некоторые функции 3D.

VIA

Совсем иначе обстоит ситуация с VIA: свободно распространяемый драйвер, интегрированный в X, к сожалению, можно использовать только со старыми моделями VIA. Что касается более новых моделей, вам придется выбирать из трех

(плохих) вариантов: медленный драйвер VESA, свободно распространяемый драйвер фирмы VIA, репутация которого, однако, не на высшем уровне, почему он и не интегрирован в X, и еще один свободно распространяемый драйвер проекта openChrome, который еще немного не доработан. Подробно рассматривать чипсет VIA в этой главе мы не будем. Более детальная информация о проекте openChrome находится здесь: http://www.openchrome.org/.

Что купить

Если вы хотите купить компьютер или графическую карту с хорошей поддержкой функций 3D, обеспечиваемой свободным ПО, вам придется выбирать между компьютером с материнской платой, содержащей графический чипсет от Intel и устаревшей картой ATI.

Для того чтобы пользоваться 3D-функциями, имея современную карту ATI или NVIDIA, придется обязательно установить драйвер от производителя. Если вас это устраивает, то лучше остановиться на карте NVIDIA. Двоичные драйверы этой компании поддерживаются более качественно. В долгосрочной перспективе могут взять верх разработки ATI/AMD: свободно распространяемый драйвер для графических карт ATI постоянно улучшается, хотя разработка идет не слишком быстро.

Проблемы, связанные с «несвободными» драйверами

Возможно, вы считаете, что различие между «настоящими» свободно распространяемыми драйверами и бесплатными драйверами от производителя (двоичные драйверы, также нередко именуемые *проприетарными драйверами*) — из области буквоедства, главное, чтобы все работало. Однако есть аргументы в пользу свободных драйверов и не в пользу двоичных.

- В недавнем прошлом стабильность драйверов от производителя оставляла желать лучшего. Хотя в самое последнее время ситуация, к счастью, и улучшилась, нет гарантии, что в будущем она не усугубится.
- Графические драйверы должны подходить к версии X. Пользователи Fedora, в дистрибутиве которых часто содержится новейшая версия X, любят приговаривать, что раньше приходилось по нескольку месяцев ждать, пока производитель сможет предоставить совместимый с ней драйвер.
- Для того чтобы графические драйверы работали эффективно, необходимо тесное сцепление с ядром Linux. Для этого между самим драйвером (закрытый код) и ядром (GPL) должен находиться маленький модуль ядра, функционирующий только в качестве интерфейса — код этого модуля должен быть доступен.

  Многие разработчики Linux сомневаются в том, что такой метод соответствует лицензии GPL, и терпят его лишь скрепя сердце. Разработчики ядра считают ядро *«запятнанным»* (дословно tainted), если в нем содержится драйвер, не соответствующий GPL, и отказываются от какой-либо поддержки, если возникают проблемы со стабильностью такой системы.

  Сцепление с ядром таит еще один недостаток: после обновления ядра также требуется обновить связующий модуль между ядром и графическим драйвером,

чтобы драйвер был совместим с новым ядром. Сложность этого процесса зависит от конкретного дистрибутива. В идеальном случае новый драйвер автоматически скачивается и устанавливается системой управления пакетами. В наихудшем случае графическая система перестает работать после обновления ядра и нужно в консоли компилировать новый модуль для сцепления ядра и графической системы.

- Из-за вышеупомянутых конфликтов с GPL передавать двоичные драйверы сложно — возникают проблемы с лицензией. Во многих дистрибутивах нужно дополнительно скачивать и инсталлировать драйверы после установки дистрибутива. Сложность этого процесса варьируется от дистрибутива к дистрибутиву: в некоторых из них выше ценится удобство для пользователей, в других — идеалы «свободного ПО».
- Поскольку код графических драйверов не разглашается за пределами конкретных фирм, любой контроль их надежности «снаружи» представляется невозможным. Если с драйвером возникает проблема, связанная с безопасностью (в прошлом такие случаи уже бывали), то пользователям Linux остается только надеяться, что фирма-производитель в кратчайшие сроки выпустит исправное обновление (если обнаружится ошибка в открытом коде, то сообщество разработчиков исправит ее собственными силами — как правило, это происходит гораздо быстрее).
- Не имея кода, невозможно адаптировать драйвер для работы с другой операционной системой или для процессора с иной архитектурой. Фирмы-производители сами решают, какие системы будут поддерживаться (например, долгое время не существовало драйверов для 64-битных процессоров или для систем BSD).
- Поддержка графики в Linux зависит от предпочтений всего нескольких фирм. Устаревшие версии графических карт обычно не поддерживаются, поэтому приходится применять новые версии X или покупать новейшее оборудование.

Еще недавно «дилемма драйверов» решалась следующим суждением: для поддержки функций 2D хватает и свободно распространяемых драйверов, а 3D-функции в Linux так или иначе не очень важны. Однако этот аргумент уже неактуален: во-первых, трехмерный рабочий стол становится стандартом. Во-вторых, некоторые новые графические чипы уже не поддерживают и 2D-функций без фирменных драйверов.

В долгосрочной перспективе Linux может остаться системой, использующей только свободное ПО, лишь в том случае, если все важнейшие компоненты будут доступны в виде открытого кода. Графические драйверы же, несомненно, относятся к числу таких компонентов. При выборе нового компьютера или новой графической карты обязательно проверяйте, есть ли к нему (к ней) свободные драйверы!

Личная рекомендация

Если вам предстоит купить компьютер и вы не предъявляете никаких экстраординарных требований к 3D-системе, вам подойдет ноутбук или материнская плата

с интегрированным графическим чипсетом Intel (но не GN40 или GMA500!). Так вы сэкономите массу времени и сил, которые пришлось бы потратить на установку и конфигурацию графических драйверов.

Если вам нужна «настоящая» графическая карта, то я рекомендую вам NVIDIA. Очень досадно, что NVIDIA — одна из последних крупных компьютерных фирм, которая делает разработку свободных драйверов практически невозможной. При этом необходимо отметить, что техническая поддержка двоичных драйверов NVIDIA в течение многих лет характеризуется значительно лучшим качеством, нежели поддержка аналогичных драйверов ATI/AMD, и что при конфигурации и эксплуатации оборудования вас ожидает гораздо меньше проблем, чем при работе с ATI/AMD (об этом свидетельствует мой горький опыт).

Глоссарий

Мир X полон сокращений и непонятных терминов. В этом разделе я приведу ориентировочную информацию, которая пригодится вам на первом этапе (термины даются в алфавитном порядке). К сожалению, X пока напоминает огромную стройплощадку — в каждой последующей версии появляются новые компоненты, а старые компоненты (в зависимости от драйвера) постепенно упраздняются. Даже профессионалам Linux часто бывает сложно держать ситуацию под контролем.

AIGLX. Accelerated Indirect GL X, коротко — AIGLX, позволяет использовать функции GLX на уровне X-сервера. AIGLX необходим для обеспечения трехмерных эффектов, предоставляемых на локальном компьютере программой Compiz или современными диспетчерами окон.

DRI и DRM. Интерфейс непосредственного вывода (DRI) позволяет использовать 3D-функции графической карты, если у вас есть подходящий DRI-драйвер для данной карты. В настоящее время актуальна версия DRI2, представляющая собой усовершенствованный вариант первоначального функционала DRI. Кроме того, двоичные драйверы ATI/AMD приспособлены к совместной работе с DRI-модулем системы X. Напротив, в драйвере NVIDIA функции DRI реализованы по-своему.

Часть DRI необходимо подключать в ядре (а не с компонентами графической карты). Эта часть называется DRM (диспетчер непосредственного вывода).

EXA. Это библиотека, предназначенная для ускорения 2D-операций (в частности, перетаскивания элементов рабочего стола) средствами графического аппаратного обеспечения. EXA и ее вариант UXA пришли на смену XAA, но и сами, предположительно, являются всего лишь переходными этапами: в долгосрочной перспективе для комплексного ускорения графических операций будет использоваться Open GL, этот же механизм позволит работать с 3D-функциями графической карты. Понятие EXA хорошо определено в глоссарии Xorg — это архитектура для акселерации, сокращенное название которой плохо расшифровывается.

GEM. Графический менеджер GEM (Graphics Execution Manager) — это библиотека, находящаяся в ядре Linux версий 2.6.28 и выше и предназначенная для управления памятью для графических драйверов. В настоящее время GEM используется только актуальными графическими драйверами Intel. Предполагается,

что GEM — промежуточный этап, она будет использоваться, пока не будет готова GEM-совместимая версия библиотеки TTM.

GLX и libGL. В системе X функции Open GL используются с применением библиотеки GLX. Эта библиотека помогает установить соединение между системой X Window и Open GL. Например, GLX гарантирует, что вывод информации Open GL производится в видимой части окна и не происходит конфликтов с другими окнами. GLX интегрируется в X с помощью специального модуля.

Библиотека libGL обеспечивает связь между модулем GLX и программой OpenGL. Библиотека должна подходить к используемому вами варианту Open GL (DRI или NVIDIA). Поэтому ссылки из каталога `/usr/lib/libGL.*` указывают на файлы библиотек.

KMS. При использовании технологии поддержки переключения видеорежимов на уровне ядра (Kernel Mode Setting) графический режим настраивается ядром Linux, а не системой X. KMS уже поддерживается драйверами Intel и Radeon, другие драйверы должны последовать их примеру. KMS позволяет настраивать нужную разрешающую способность графического устройства сразу же после запуска компьютера. В таком случае должно исчезнуть «дрожание», с которым в настоящее время запускается система X. Fedora, использующая в процессе загрузки интегрированную программу Plymouth, доказала, что такой метод работает как минимум с некоторыми графическими картами Intel и ATI.

Open GL. Open GL (коротко — просто GL) — это разработанная SGI библиотека, предназначенная для отображения трехмерной графики. Такая библиотека имеется практически на всех компьютерах UNIX/Linux. Таким образом, Open GL можно считать аналогом библиотеки Microsoft DirectX.

Поскольку первоначально кода Open GL не было в свободном доступе (сейчас ситуация изменилась), появилась совместимая с ней свободная библиотека Mesa. Сначала Mesa была только программным решением, но потом стала использовать 3D-функции графической карты, реализуемые через модуль DRI.

RandR. Расширение RandR (Resize and Rotate) позволяет изменять некоторые настройки X прямо в ходе эксплуатации системы. Среди этих настроек можно назвать разрешение, кадровую частоту и вращение изображения. С помощью RandR также можно активизировать второй монитор.

TTM. Карты таблиц преобразования (TTM) — это библиотека управления памятью для графических драйверов, подобная GEM. TTM не так сильно завязана на Intel, как GEM, и с ней проще организовать взаимодействие с другими драйверами (например, от карт AMD/ATI). Возможно, разработчики решат заменить библиотеку GEM новой версией TTM, способной выполнять многие функции GEM, и тогда новая библиотека сможет войти в состав официального ядра Linux.

UXA. Архитектура акселерации UMA (UXA) — это характерный для Intel вариант библиотеки EXA, то есть она занимается обработкой 2D-графики. Важное отличие от EXA заключается в том, что UXA использует функции управления памятью из GEM (например, для сохранения растровой графики в памяти графической карты).

XAA. Архитектура акселерации X (XAA) ускоряет обработку 2D-графики. Такой способ аппаратного ускорения появился раньше, чем 3D-функции, и уже давно по умолчанию поддерживается в системе X. К сожалению, взаимодействие

между XAA и Open GL, а также с 3D-функциями вообще пока проблематично. По этой причине многие графические драйверы уже перешли от использования XAA к EXA и UXA.

Xgl. Это устаревший вариант AIGLX, удаленный из системы X в середине 2008 года. Xgl был разработан Novell и в течение нескольких лет применялся в дистрибутивах SUSE для реализации 3D-эффектов. При этом сначала запускался обычный X-сервер, который использовался для отображения всего одного окна без обрамления. Это окно требовалось для работы с 3D-функциями библиотеки Open GL. За содержимое этого окна отвечает сервер Xgl, который показывает в нем рабочий стол.

X Render. Расширение рендеринга для X (коротко — X Render) — это библиотека для создания эффектов прозрачности и наложения (альфа-смешивание). Она используется и для отображения текста. Для ускорения работы X Render реализует 3D-функции с помощью аппаратного обеспечения.

Ссылки. На перечисленных далее сайтах вы найдете полезную информацию о системе X.

- Официальная, правда, не совсем понятная и не всегда актуальная документация по X.org содержится в следующем вики-источнике: http://www.x.org/wiki.
- Наилучшее освещение всех современных тенденций в разработке X дается на следующем сайте: http://www.phoronix.com/.
- Взгляд на ситуацию с точки зрения Intel (по состоянию на середину 2009 года) с хорошим толкованием всех терминов представлен здесь: http://keithp.com/blogs/Sharpening_the_Intel_Driver_Focus/.

Кроме того, статьи по практически по всем упомянутым выше понятиям имеются в «Википедии» (иногда только в английской версии).

12.2. Запуск и завершение работы X

В этом разделе я кратко расскажу о запуске и завершении работы системы X. Вам стоит беспокоиться лишь в тех случаях, когда система X не включается при запуске (и не выключается при остановке) компьютера автоматически либо если требуется вручную внести изменения в процесс.

Запуск и завершение работы X вручную

Экранный менеджер. Обычно сначала запускается не сама система X, а так называемый *экранный менеджер*. Эта программа запускает систему X, отображает экран для входа в систему, а после введения учетных данных загружает систему непосредственного взаимодействия (рабочий стол, например Gnome или KDE) либо менеджер окон. Последний целесообразно запускать лишь в том случае, если на компьютере установлено несколько рабочих столов.

В зависимости от того, с каким рабочим столом вы работаете, используется один из двух вариантов экранного менеджера — `kdm` для KDE или `gdm` для Gnome. В тех дистрибутивах, где не предусмотрен рабочий стол, в качестве экранного менеджера

применяется совсем минималистичная программа xdm. Экранный менеджер был разработан в трех вариантах с учетом визуальных преимуществ, чтобы можно было оптимально приспособить эту программу к используемому варианту рабочего стола. Но в принципе каждый экранный менеджер способен запустить любой рабочий стол. Иначе говоря, можно запустить KDE с помощью gdm или Gnome с использованием kdm!

Процесс Init-V. В большинстве дистрибутивов экран активизируется процессом Init-V (см. раздел 14.10) при запуске компьютера. В Debian и Ubuntu процесс Init-V выполняет сценарий /etc/init.d/gdm или /etc/init.d/kdm на уровнях запуска 2–5. В SUSE процесс Init-V выполняет сценарий /etc/init.d/xdm на уровне запуска 5.

Upstart. Программа Upstart используется для запуска X в Fedora, начиная с версии 11 и в Ubuntu — с версии 9.10. Необходимые для этого правила формулируются в /etc/event.d/prefdm или /etc/init/gdm.conf.

Перезапуск X. Если в конфигурацию X были внесены изменения, они вступят в силу только после перезапуска системы. В большинстве дистрибутивов для этого необходимо выйти из системы X, перейти в текстовую консоль и выполнить там приведенную ниже команду. В зависимости от дистрибутива может потребоваться изменить kdm на gdm или xdm.

```
root# /etc/init.d/gdm restart
```

В Fedora также необходимо выйти из X и перейти в текстовую консоль; там следует выполнить две следующие команды Upstart:

```
root# stop prefdm
root# start prefdm
```

Завершение работы X. Как правило, X выключается только при остановке компьютера, если ее работу не потребуется завершить раньше. Пользователь может входить в систему с определенного компьютера и снова выходить из нее любое количество раз по мере надобности. Чтобы действительно завершить работу X, выйдите из действующей системы рабочего стола, перейдите в консоль и выполните init 3. Этот метод работает во всех дистрибутивах, где уровень запуска 3 отводится для работы сетевого окружения, а не для X:

```
root# init 3                              (Fedora, Red Hat, SUSE)
```

В дистрибутивах, построенных на основе Debian, вам потребуется остановить работу экранного менеджера gdm, kdm или xdm из текстовой консоли:

```
root# /etc/init.d/gdm|kdm|xdm stop     (Debian, Ubuntu)
```

Отключение автоматического запуска. Иногда бывает полезно отключить автоматический запуск X — например, если компьютер работает в качестве сервера. В большинстве дистрибутивов (в том числе Fedora, Red Hat, SUSE) установите в качестве стандартного уровня запуска Init-V не 5, а 3. Для этого необходимо изменить строку initdefault в /etc/inittab:

```
# в /etc/inittab (Fedora, Red Hat, SUSE)
...
# стандартный уровень запуска 3 (многопользовательская система без X)
id:3:initdefault:
```

Это изменение вступит в силу после перезагрузки компьютера. Чтобы вновь активизировать автоматический запуск X, установите для стандартного уровня запуска значение 5.

Совершенно иначе обстоит ситуация в дистрибутивах, построенных на основе Debian. Здесь уровень запуска не изменяется, но вы отменяете автоматический запуск экранного менеджера. Опять же, может быть так, что у вас будет применяться не gdm, а его варианты kdm или xdm.

```
root# update-rc.d -f gdm remove     (Debian, Ubuntu до 9.04)
root# update-rc.d gdm stop 1 0 1 2 3 4 5 6 .
```

Для того чтобы в будущем вновь запускать X автоматически, нужно использовать две следующие команды. Здесь значения от 1 до 30 означают порядок приоритетов запуска или остановки. В зависимости от дистрибутива они могут изменяться. Указанные ниже значения задаются по умолчанию в Debian 5.

```
root# update-rc.d -f gdm remove     (Debian, Ubuntu bis 9.04)
root# update-rc.d gdm defaults 30 1
```

В Ubuntu 9.10 и выше X запускается сценарием Upstart. Чтобы отменить автоматический запуск, перед командой start on (filesystem and started hal) в файле /etc/init/gdm.conf добавьте символ комментария.

Запуск X вручную. Когда система X не работает, ее можно запустить вручную двумя способами. Пока конфигурация не завершена, используйте первый вариант.

- startx — команда запускает X напрямую. Окно для входа в систему не выводится. Пользователь, выполнивший команду startx, входит в систему X.
- Запуск с помощью экранного менеджера — в большинстве дистрибутивов для запуска экранного менеджера нужно перейти на уровень запуска 5:

  ```
  root# init 5                        (Fedora, Red Hat, SUSE)
  ```

  В дистрибутивах, построенных на основе Debian, уровень запуска не изменяется. В таком случае экранный менеджер необходимо специально перезапустить:

  ```
  root# /etc/init.d/gdm|kdm|xdm start  (Debian, Ubuntu до 9.04)
  ```

  В Ubuntu 9.10 и выше необходимо выполнить следующую команду:

  ```
  root# start gdm                     (Ubuntu 9.10 и выше)
  ```

Xsession. При запуске X выполняется сценарный файл /etc/X11/Xsession, а также сценарии из каталога /etc/X11/Xsession.d. В этом каталоге очень удобно при запуске X изменять те или иные настройки или вносить в конфигурацию другие изменения.

Параметр DontZap. Раньше можно было завершать работу X, нажав клавиши Ctrl+Alt+Backspace. В результате сразу закрывались все программы, работающие под X. Чтобы уберечься от неожиданного завершения работы системы, чреватого потерей всех несохраненных данных, это сочетание отключено в современных дистрибутивах X. Большинство дистрибутивов Linux восприняло эти изменения,

правда, разными способами. Чтобы снова активизировать это сочетание клавиш, в современных дистрибутивах применяются собственные (различные) методы.

В Fedora и в Ubuntu 9.10 и вышеуказанное сочетание клавиш отключено. Воспользуйтесь программой gnome-keyboard-properties.

В SUSE это сочетание будет функционировать, если дважды нажать его в течение двух секунд. Если вы хотите срочно завершить работу X, удалите в файле /etc/X11/xorg.conf строку "ZapWarning" "on".

В Ubuntu 9.04 для активизации рассматриваемого сочетания клавиш нужно добавить в файл /etc/X11/xorg.conf три следующие строки:

```
# /etc/X11/xorg.conf (Ubuntu)
...
Section "ServerFlags"
  Option "DontZap" "false"
EndSection
```

Конфигурация экранного менеджера

Gdm. Это экранный менеджер для рабочего стола Gnome. Файлы конфигурации находятся в /etc/gdm или /etc/X11/gdm. Кроме того, вы можете указать, какие программы, команды и сценарии должны использоваться для реализации различных функций экранного менеджера. С помощью программы gdmsetup можно определить, должны ли пользователи автоматически регистрироваться в системе при запуске X.

Kdm. Это аналог gdm из KDE. Конфигурация kdm осуществляется в файле kdmrc, который очень похож на описанный выше файл конфигурации gdm.conf. Место сохранения kdmrc зависит от дистрибутива. Возможные места:

- /etc/kde*n*/kdm/kdmrc;
- /opt/kde*n*/share/config/kdm/kdmrc.

Многие настройки kdmrc удобно вносить в модуле Управление системой ▸ Управление логинами центра управления. К этим настройкам относятся: визуальное оформление диалогового окна для входа в систему, оформление фона, представление пользователей с помощью значков, авторегистрация пользователей и т. д.

Файлы .desktop. При входе в gdm или kdm можно выбрать в меню, какой рабочий стол или диспетчер окон следует запустить. Данные этого меню в большинстве дистрибутивов располагаются в файлах с расширением .desktop в каталоге /usr/share/xsession (ключевое слово SessionDesktopDir в конфигурации gdm). Например, в DESKTOP-файле для запуска Gnome содержатся следующие строки:

```
[Desktop Entry]
Encoding=UTF-8
Name=GNOME
Comment=This session logs you into GNOME
Exec=/usr/bin/gnome-session
...
```

В системах, работающих на локальных компьютерах, часто бывает нужно, чтобы основной пользователь данного компьютера при запуске системы регистриро-

вался в ней автоматически. В этом, конечно, есть доля риска (что делать, если вы потеряете ноутбук?), но все же такой метод очень удобен. Если вы пользуетесь экранным менеджером gdm, вставьте следующие строки в раздел [daemon] файла custom.conf:

```
# Файл /etc/gdm/custom.conf
...
[daemon]
    AutomaticLoginEnable=true
    AutomaticLogin=loginname
```

В некоторых дистрибутивах, чтобы задать в gdm обязательную авторегистрацию, можно воспользоваться графическим пользовательским интерфейсом gdmsetup:

Авторегистрация в kdm. В kdm вставьте в kdmrc следующую строку:

```
# Файл /etc/kde4/kdm/kdmrc
...
AutoLoginUser=loginname
```

В SUSE предусмотрены собственные конфигурационные файлы авторегистрации в KDE и Gnome. Старайтесь не изменять конфигурационные файлы kdm или gdm напрямую! При следующем использовании YaST или SUSEconfig ваши изменения будут удалены.

Файл протоколов X

При запуске X в файле /var/log/Xorg.0.log сохраняются различные сообщения, предупреждения и сообщения об ошибках (если таковые имеются). В этом стартовом протоколе содержится подробная информация о том, какой конфигурационный файл использовался, какие модули были загружены, какие проблемы при этом возникли, какие графические режимы были отвергнуты и почему и т. д. Записи в файле регистрации помечаются следующими кодами:

- (**) — настройка из конфигурационного файла;
- (++) — настройка из командной строки;
- (==) — стандартная настройка X;
- (--) — настройка, полученная из распознанного оборудования;
- (!!) — указание;
- (II) — указание;
- (WW) — предупреждение;
- (EE) — ошибка.

Если в /var/log/ находится несколько файлов регистрации X, найдите самый актуальный файл. К сожалению, поскольку в Xorg.0.log содержится масса информации, поиск данных, которые действительно важны, уподобляется поиску иголки в стоге сена. По возможности отошлите весь файл регистрации специалисту, который лучше в этом разбирается, либо вывесьте файл на форум, посвященный технической поддержке.

Определение версии X

Если вы хотите узнать, какая версия X-сервера используется на вашем компьютере, выполните следующую команду. На моем воображаемом компьютере работает сервер X.org версии 1.6.

```
user$ X -showconfig
...
X.Org X Server 1.6.0
Release Date: 2009-2-25
X Protocol Version 11, Revision 0
...
```

Еще один способ — использовать команду `xdpyinfo`:

```
user$ xdpyinfo | grep release
vendor release number: 10600000
```

12.3. Базовая конфигурация

Для конфигурации X используется файл `/etc/X11/xorg.conf`. Раньше он играл более важную роль: при его отсутствии запустить систему X было невозможно. Однако со временем ситуация коренным образом изменилась: актуальные версии X обходятся вообще без `xorg.conf`: система опознает при запуске имеющееся аппаратное обеспечение (графическую карту, монитор, мышь, клавиатуру) и автоматически загружает необходимые драйверы и модули. Если при конфигурации не нужно выполнять каких-либо необычных условий, X работает вообще без `xorg.conf`!

Проводить конфигурацию вручную нужно лишь тогда, когда автоматическую конфигурацию выполнить не удается. В этом разделе я сообщу вводную информацию о синтаксисе `xorg.conf` и дам некоторые советы о конфигурации системы. Не забывайте, что изменения, внесенные в X, вступят в силу только после перезагрузки (см. раздел 12.2). Если в `xorg.conf` возникнут ошибки, то вы, возможно, вообще не сможете запустить X. В таком случае следует перейти в консоль и исправить эти ошибки. Научитесь с ней работать, прежде чем всерьез заниматься `xorg.conf` (см. главу 2)!

Инструменты конфигурации

Обычно X-сервер конфигурируется уже при установке. В зависимости от того, с каким дистрибутивом вы работаете и какие двоичные драйверы установили — ATI/AMD или NVIDIA, — вы можете воспользоваться следующими инструментами конфигурации:

- ATI/AMD — `amdccle` (Catalyst Control Center);
- Debian, Ubuntu — `dpkg-reconfigure xserver-xorg`;
- NVIDIA — `nvidia-settings`;
- SUSE — модуль YaST Аппаратное обеспечение ▸ Графическая карта или sax2.

Кроме того, в KDE и Gnome содержатся конфигурационные инструменты, с помощью которых можно настраивать разрешение экрана и кадровую частоту, а в несложных ситуациях задавать конфигурацию для одновременной работы с двумя мониторами. Так или иначе эти программы не вмешиваются в `xorg.conf`, а динамически изменяют конфигурацию X с помощью механизма RandR. Результат будет сохранен в пользовательском конфигурационном файле и будет действителен только для активного пользователя, а не для всей системы локального компьютера. Более подробно о RandR рассказывается в разделе 12.6.

Структура конфигурационного файла xorg.conf

Файл `/etc/X11/xorg.conf` разбит на несколько разделов, которые начинаются с `Section "`*`имя`*`"` и заканчиваются `EndSection` (табл. 12.1).

Таблица 12.1. Разделы xorg.conf

Раздел	Значение	Подробности в книге
Monitor	Информация о мониторе	Подраздел «Раздел Monitor»
Device	Конфигурация графической карты	Подраздел «Раздел Device (графическая карта)»
Screen	Разрешение экрана	Подраздел «Раздел Screen»
Files	Имена файлов (например, каталоги со шрифтами)	Подраздел «Раздел Files»
Module	Дополнительные модули (например, freetype, dri)	Подраздел «Раздел Module»
ServerFlags	Различные серверные параметры	Подраздел «Раздел ServerFlags»
InputDevice	Конфигурация мыши и клавиатуры	Раздел 12.5

Пример

В современных дистрибутивах вообще нет файла `xorg.conf`, а если он и есть, то содержит всего несколько разделов. Однако подробный `xorg.conf` старого образца все еще встречается в openSUSE 11.1, где такие файлы создаются конфигурационной программой SaX2:

```
# /etc/X11/xorg.conf (openSUSE 11.1)
Section "Files"
  FontPath     "/usr/share/fonts/Type1"
  ...
  InputDevices "/dev/gpmdata"
  InputDevices "/dev/input/mice"
EndSection
Section "ServerFlags"
  Option       "AIGLX"               "on"
  Option       "AllowMouseOpenFail" "on"
  Option       "IgnoreABI"           "on"
  Option       "ZapWarning"          "on"
EndSection
Section "Module"
```

```
  Load         "dri"
  Load         "dbe"
  ...
EndSection
Section "InputDevice"
  Driver       "kbd"
  Identifier   "Keyboard[0]"
  Option       "Protocol"    "Standard"
  Option       "XkbLayout"   "de"
  Option       "XkbModel"    "microsoftpro"
  Option       "XkbRules"    "xfree86"
  Option       "XkbVariant"  "nodeadkeys"
EndSection
Section "InputDevice"
  Driver       "mouse"
  Identifier   "Mouse[1]"
  Option       "Buttons"       "9"
  Option       "Device"        "/dev/input/mice"
  Option       "Protocol"      "explorerps/2"
  Option       "Vendor"        "Sysp"
  Option       "ZAxisMapping"  "4 5"
EndSection
Section "Monitor"
  Identifier   "Monitor[0]"
  DisplaySize  519 324
  HorizSync    30-94
  VertRefresh  43-85
  Option       "DPMS"
  Option       "PreferredMode"   "1680x1050"
  Option       "CalcAlgorithm"   "XServerPool"
EndSection
Section "Screen"
  DefaultDepth 24
  SubSection "Display"
    Depth      24
    Modes      "1680x1050" "1600x1024" "1600x1000" "1400x1050" ... "640x480"
  EndSubSection
  Device       "Device[0]"
  Identifier   "Screen[0]"
  Monitor      "Monitor[0]"
EndSection
Section "Device"
  Identifier   "Device[0]"
  Driver       "fglrx"
  Option       "XAANoOffscreenPixmaps"   "true"
  Option       "Capabilities"            "0x00000000"
  Option       "OpenGLOverlay"           "off"
  Option       "VideoOverlay"            "on"
EndSection
Section "ServerLayout"
```

```
  Identifier   "Layout[all]"
  InputDevice  "Keyboard[0]"            "CoreKeyboard"
  InputDevice  "Mouse[1]"               "CorePointer"
  Option       "Clone"                  "off"
  Option       "Xinerama"               "off"
  Screen "Screen[0]"
EndSection
Section "DRI"
  Group        "video"
  Mode         0660
EndSection
```

Идентификатор

В строке `Identifier` указывается название раздела; кроме того, здесь делаются перекрестные ссылки между разделами. Например, раздел `Screen` указывает на `Device0`. В некоторых файлах конфигурации `xorg.conf` во многих разделах вы встретите строки `Board`, `Vendor` и `ModelName`. Эта дополнительная информация помогает лучше ориентироваться в конфигурационном файле. Она не интерпретируется X и не влияет на работу системы.

Важнейшие ключевые слова будут описаны в следующих разделах. Полная справка содержится в `man xorg.conf`.

Раздел Monitor

Как правило, раздел `Monitor` является излишним, так как современные мониторы передают свои контрольные показатели графической карте. Если этот механизм не будет работать (это возможно только при использовании очень старых мониторов), то в этом разделе можно указать допустимый диапазон горизонтальной частоты строчной развертки (в КГц) и кадровой частоты (в Гц). Следующие данные взяты с монитора с разрешением 1600×1200 пикселов с максимальной кадровой частотой 75 Гц:

```
Section "Monitor"
  ...
  HorizSync   30-95   # Частота развертки  30-95 КГц (строк/сек)
  VertRefresh 58-78   # Кадровая частота   58-78 Гц (кадров/сек)
EndSection
```

Параметр ModeLine

С помощью параметра `ModeLine` вы можете точно указать, в каком графическом режиме должен использоваться монитор. Для определения графического режима необходимо задать имя режима и девять числовых значений. Например:

```
ModeLine  "640x480"   25.175   640 664 760 800   480 491 493 525
```

Эта строка описывает графический режим с разрешением 640×480 пикселов. Строка `"640x480"` является именем этого режима. Число `25.175` указывает частоту элементов графических изображений (ширину полосы видеочастот) в МГц.

Следующие четыре значения (в пикселах) описывают горизонтальную выдержку (Timing): одна экранная строка с 640 *видимыми* пикселами на самом деле будет состоять из 800 *виртуальных* пикселов. Первые 640 пикселов действительно будут отображаться. На оставшихся 160 пикселах поток электронов будет перебрасываться импульсом HSync к началу следующей строки. В такой момент передачи интенсивность потока электронов равна нулю. Таким образом, четыре значения передают следующую информацию:

- `640` — отобразить на экране 640 пикселов;
- `664` — оставить еще 24 пиксела темными;
- `760` — создать импульс HSync продолжительностью 96 пикселов;
- `800` — затемнить еще 40 пикселов, в итоге имеем 800 виртуальных точек.

Аналогично горизонтальной выдержке указываются и параметры вертикальной (в экранных строках):

- `480` — отобразить 480 строк;
- `491` — затемнить 11 строк;
- `493` — создать импульс VSync продолжительностью 2 строки;
- `525` — затемнить еще 32 строки, в итоге имеем всего 525 виртуальных строк.

Последние значения этих четверок и частота элементов графических изображений дают горизонтальную частоту развертки и вертикальную кадровую частоту: отношение 25,175 МГц к 800 пикселам в строке дает частоту развертки, равную 31,469 КГц. Отношение частоты развертки к 525 строкам на экране дает кадровую частоту 60 Гц.

Раньше я работал с очень старым ЖКИ-монитором, имевшим разрешение 1600 × 1200 точек с максимальной частотой сигнала 160 МГц (современные модели с таким разрешением выдают сигнал с гораздо большей частотой). При подключении монитора к компьютеру DVI-кабелем изображение не появлялось. Внимательно изучив файл `/var/log/Xorg.0.log`, я обнаружил, что X превышает максимальную сигнальную частоту монитора. Оказалось, что нужно установить следующий графический режим с сигнальной частотой всего лишь около 130 МГц (1728 * 1250 * 60):

```
ModeLine   "1600x1200"   129.60   1600 1664 1696 1728   1200 1201 1204 1250
```

Параметр DisplaySize

Наконец, с помощью параметра `DisplaySize` мы задаем ширину и высоту монитора (в миллиметрах). Система X интерпретирует эту информацию, чтобы определить значение DPI (см. подраздел «Настройка DPI» раздела 12.10).

```
DisplaySize 336 252
```

Раздел Device (графическая карта)

Самое важное ключевое слово этого раздела — `Driver`. Здесь определяется, какой драйвер необходимо загрузить. Имеющиеся в распоряжении графические драйве-

ры находятся в каталоге /usr/lib[64]/xorg/modules/drivers. Как правило, система X сама распознает нужный драйвер. Специально задавать его приходится только при работе с самыми новыми графическими картами или при использовании двоичного драйвера от производителя.

Если в компьютере установлено несколько PCI-графических карт, то с помощью BusID вы можете точно указать одну из них. Три цифры задают шину PCI, номер устройства и функцию. Чтобы узнать правильные значения, выполните в текстовой консоли X команду -scanpci (в это время система X не должна работать).

```
Section "Device"
  Driver "radeon"
  BusID "1:0:0"
EndSection
```

Какой драйвер к какой графической карте подходит?

Если вы не знаете, какая графическая карта установлена на вашем компьютере, то можете выполнить команду lspci с привилегиями администратора:

```
root# lspci

...
01:00.0 VGA compatible controller: ATI Technologies Inc M10 NT
                                   [FireGL Mobility T2] (rev 80)
```

К сожалению, из результата выполнения этой команды не всегда понятно, какой драйвер нужен. При выборе драйвера полезно изучать релизы новых версий X.org (ищите раздел «Видеодрайверы») и указанные в них страницы справки man (http://xorg.freedesktop.org/wiki/XorgReleases). Например, в man radeon подробно рассказано о драйвере X.org-Radeon, в man nv — о драйвере X.org-NVIDIA.

Если вам не повезет, то ваша новая графическая карта не будет поддерживаться X.org либо будет поддерживаться частично. Иногда графическую карту просто не удается распознать — скажем, вы указали в разделе Device правильный модуль, но X не распознает графическую карту. В таком случае можно попытаться вставить в раздел Device ID-номер совместимой карты с помощью ключевого слова ChipId (например, ChipId "0x1234"). Список действующих ID-номеров находится в файле pci.ids. Место расположения этого файла может отличаться в зависимости от дистрибутива; сначала поищите его в каталоге /usr/share/misc.

Если вам не удастся запустить свою графическую карту, почитайте в подразделе «Драйверы VESA, Framebuffer и VGA» раздела 12.4 советы по работе с драйверами vga, vesa или fbdev. Эти драйверы работают практически с любой графической картой, правда, скорость работы при этом невысока, кроме того, не поддерживаются 3D-функции.

Параметры драйверов

При работе практически с любым драйвером предусмотрены параметры для управления специальными настройками, решения проблем и активизации особенных функций. Описать их здесь я не смогу из-за ограниченных размеров книги. Подробная информация дается на соответствующих страницах справки man (то есть, допустим, man radeon). В следующих строках, например, показано, как работать

с DisplayPriority. Этот параметр требовался для работы со старыми ноутбуками, в которых использовался стыковочный узел, позволяющий получать стабильное изображение на внешнем ЖК-мониторе (через кабель DVI).

```
Driver  "radeon"
Option  "DisplayPriority"  "HIGH"
```

Раздел Screen

Раздел Screen обеспечивает связь между монитором и графической картой и указывает, с каким разрешением и количеством цветов должна использоваться карта. Ключевые слова Device и Monitor указывают на определенные выше графическую карту и монитор, DefaultDepth задает количество цветов, используемых по умолчанию. Параметр дается в битах на пиксел. При показателе 24 бита на каждый из основных цветов выделяется по 8 бит, то есть по 256 оттенков красного, зеленого и голубого, всего 2^{24} цветов. При показателе 16 бит на каждый оттенок выделяется всего 5 бит — один бит обычно остается неиспользуемым.

В разделе Screen можно задать несколько подразделов Display, по одному для каждой конфигурации цветов (ключевое слово Depth). В следующем примере определен только один режим — 24 бита на пиксел:

```
Section "Screen"
  Identifier "Screen0"
  Device "Videocard0"
  DefaultDepth 24
  SubSection "Display"
    Depth 24
    Modes "1280x1024"
  EndSubSection
EndSection
```

В необязательной строке Modes можно указать желаемое разрешение. Если опустить эту строку, то X автоматически установит наилучшее разрешение, возможное для данной графической карты и конкретного монитора.

Кроме того, в каждом разделе Display можно настраивать размер виртуального экрана. Например, Virtual 1600 1200 означает, что вы работаете с виртуальным монитором с разрешением 1600×1200 точек, независимо от того, монитор с каким разрешением используется на самом деле.

Раздел Files

В разделе Files указываются участки различных каталогов, из которых X-сервер загружает файлы. Эти данные необходимо указывать, если они отличаются от определенных по умолчанию.

```
Section "Files"
  FontPath "/etc/X11/fonts/Type1"
  ...
EndSection
```

Раздел Module

В разделе Module с помощью ключевого слова Load указывается, какие модули *расширения (extensions)* должен использовать X-сервер:

```
Section "Module"
  Load "modulname"
  ...
EndSection
```

Раздел Module — необязательный, как правило, все необходимые модули загружаются автоматически. Файлы модулей находятся в подкаталогах /usr/lib/xorg/modules/. Чтобы узнать, какие модули загружены, действуйте так:

```
root# grep LoadModule /var/log/Xorg.0.log
(II) LoadModule: "extmod"
(II) LoadModule: "dbe"
(II) LoadModule: "glx"
...
```

Раздел ServerFlags

В разделе ServerFlags можно указать параметры, влияющие на работу X-сервера:

```
Section "ServerFlags"
  Option "DontZap" "false"
EndSection
```

Далее мы рассмотрим важнейшие из этих параметров. Полный перечень всех параметров находится на странице справки man xorg.conf.

- AllowMouseOpenFail (по умолчанию off) — при настройке on X-сервер сам запускается, если не удается инициализировать или распознать мышь.
- DefaultServerLayout — параметр говорит о том, какая конфигурация сервера (ServerLayout) должна применяться. Он необходим, если в xorg.conf содержится несколько разделов о конфигурации сервера.
- DontZap (базовая настройка true) — настройка false активизирует сочетание клавиш Ctrl+Alt+Backspace для немедленного завершения работы X-сервера. В некоторых дистрибутивах (например, в Fedora) нужно, кроме того, активизировать это сочетание в gnome-keyboard-properties.

12.4. Графические драйверы (ATI/AMD, NVIDIA)

В этом разделе даются советы по установке и конфигурации графических драйверов для карт или графических чипов ATI/ADM, Intel и NVIDIA. В конце этого раздела описаны драйверы VESA, Framebuffer и VGA, которые подойдут в качестве компромиссного решения, если нет лучшего драйвера (например, для некоторых графических чипов VIA).

Графическая карта ATI/AMD

Для современных графических карт ATI/AMD на выбор предлагается три драйвера.

- radeon — это интегрированный в X.org драйвер с открытым кодом, предназначенный для современных графических карт ATI/AMD с графическим чипом Radeon. В принципе этот драйвер совместим со всеми доступными в настоящее время чипами radeon. Правда, функции 3D и XVideo предоставляются только для чипов семейства R100–R500 (подробности — на странице справки `man radeon`). Поскольку компания ATI/AMD предоставила спецификацию своего чипа сообществу свободных разработчиков с осени 2007 года, есть надежда, что в скором времени появится поддержка 3D для новейших моделей.
- radeonhd — это еще один драйвер, разработанный (в основном силами Novell) параллельно с драйвером radeon. В современных версиях SUSE этот драйвер используется по умолчанию, но в остальных дистрибутивах он практически не применяется (в Fedora он предоставляется в специальном пакете, но по умолчанию не устанавливается).

  Свойства radeonhd и radeon в настоящее время во многом сходны. Дополнительную информацию вы можете почерпнуть на следующих сайтах:

  - http://www.x.org/wiki/radeon;
  - http://www.x.org/wiki/radeonhd;
  - http://www.phoronix.com/scan.php?page=article&item=radeon_vs_radeonhd&num=1.
- fglrx — это двоичный драйвер фирмы ATI/AMD для моделей Radeon R600 и выше. Как и следовало ожидать, этот драйвер поддерживает все функции современных графических карт. Правда, он несовместим с более старыми моделями, вышедшими до серии R500. Раньше приходилось ждать по несколько месяцев, пока появлялись версии fglrx, совместимые с актуальными версиями ядра или X.org.

В самых графических картах с графическими чипами Mach8/32/64 или ATI-Rage-128 еще были драйверы mach64 и r128, которые мы здесь рассматривать не будем. Драйвер ati — это просто «оболочка», активизирующая один из следующих драйверов: radeon, r128 или mach64. Если вы не знаете, какой чип стоит в вашей графической карте, вам поможет следующая страница «Википедии»: http://en.wikipedia.org/wiki/Comparison_of_ATI_Graphics_Processing_Units.

Драйвер radeon

Драйвер radeon входит в состав системы X.org, отдельно устанавливать его не требуется. В следующих строках показана минимальная конфигурация `xorg.conf`:

```
Section "Device"
  Identifier  "Device0"
  Driver      "radeon"
EndSection
```

Различные специальные функции графической карты управляются различными параметрами (см. `man radeon`). Далее показан пример, описывающий некоторые

возможности карты и связанные с ними сложности. Такая конфигурация потребовалась, чтобы заставить работать вместе старенький ноутбук IBM и «несговорчивый» мультимедийный проектор. Кратко опишу использованные параметры.

- `MonitorLayout` указывает, что для вывода информации должен использоваться и монитор ноутбука, и внешний аналоговый выход (CRT).
- `MergedFB` означает, что оба сигнальных выхода обращаются к совместно используемой области видеопамяти.
- Оба параметра `CRT2` указывают допустимые частоты сигнала на аналоговом выходе (для проектора).
- Параметр `IgnoreEDID` позволяет игнорировать данные EDID, предоставляемые проектором. EDID означает «расширенная идентификация данных дисплея» (Extended Display Identification Data), это часть информации, передаваемой от монитора к графической карте по каналу DDC (канал отображения данных). Проектор, имевшийся у меня в распоряжении, выдавал очевидно неверные данные.
- `MetaModes` указывает, какие разрешения должны использоваться на обоих сигнальных выходах, — в данном случае по 1024 × 768 пикселов:

```
Section "Device"
  Identifier  "device0"
  Driver      "radeon"
  Option      "MonitorLayout"  "LVDS,CRT"
  Option      "MergedFB"       "yes"
  Option      "CRT2HSync"      "30-120"
  Option      "CRT2VRefresh"   "58-65"
  Option      "IgnoreEDID"     "yes"
  Option      "MetaModes"      "1024x768-1024x768"
EndSection
```

Драйвер fglrx

fglrx — это двоичный драйвер фирмы AMD для графических карт ATI (для краткости его называют просто «графический драйвер ATI/AMD»). Поскольку этот драйвер не является программой с открытым кодом, он не содержится в большинстве дистрибутивов и его нужно устанавливать дополнительно. Одно из немногих исключений — Ubuntu.

Для большинства других дистрибутивов существуют неофициальные, но хорошо поддерживаемые репозитории, значительно упрощающие установку драйвера.

Команда aticonfig. В отдельных дистрибутивах вам может понадобиться выполнить команду `aticonfig --initial` с привилегиями администратора, чтобы установить этот драйвер. Команда изменяет `xorg.conf` так, что драйвер fglrx начинает использоваться вместо того драйвера, что применялся ранее. В простейшем случае в разделе `Device` будет всего две строки:

```
Section "Device"
  Identifier  "Device0"
  Driver      "fglrx"
EndSection
```

Команда aticonfig также бывает полезна при настройке сложных конфигураций, например для одновременного использования нескольких мониторов (aticonfig --initial=dual). Если ввести команду aticonfig без дополнительных параметров, то система выдаст обзор ее синтаксиса и несколько примеров. Правда, систематической документации по многочисленным параметрам этой команды, как кажется, нет. Новый файл xorg.conf будет задействован, как обычно, только после перезапуска X.

Часто fglrx применяется для работы с 3D-функциями графических карт (см. также раздел 12.8). Для этого требуется, чтобы был загружен модуль ядра fglrx, — система загружает его при запуске. Чтобы узнать, активен ли этот модуль ядра на самом деле, лучше всего выполнить команду lsmod grep fglrx. Если модуль не загрузился, он, скорее всего, не установлен либо несовместим с текущей версией ядра. Попробуйте обновить соответствующий пакет или заново скомпилировать модуль (в Debian: m-a a-i fglrx). Этот модуль ядра нужен лишь для реализации 3D-функций. Драйвер fglrx работает и без модуля ядра, но в таком случае 3D-функции приходится эмулировать программными средствами, а этот процесс очень медленный.

Для постепенного изменения конфигурации пользуйтесь либо командой aticonfig, либо графическим интерфейсом fireglcontrol. Программа должна выполняться с привилегиями администратора, она изменяет файл xorg.conf. Как вы понимаете, изменения вступают в силу только после перезапуска X.

Установка драйверов вручную. Если для вашего дистрибутива не предусмотрены готовые пакеты с драйверами либо если пакет устарел (довольно частая проблема!), вам придется сделать все вручную. В таком случае сначала установите все инструменты разработки, необходимые для компилирования модулей ядра (раздел 15.1). Кроме того, скачайте с сайта http://support.amd.com/us/gpudownload/Pages/index.aspx установочную программу, подходящую для вашей графической карты и архитектуры вашего дистрибутива (32 или 64 бит). Программа имеет размер около 90 Мбайт.

С помощью sh выполняется программа установки:

```
root# sh ./ati-driver-installer-<n.n>-x86.x86_64.run
```

Теперь рассмотрим диалоговые окна графической установочной программы. Подробности о процессе установки сообщаются в файле протокола /usr/share/ati/fglrx-install.log. Если не возникает никаких проблем, программа установки наконец загружает модуль ядра fglrx, в чем вы можете убедиться, выполнив команду lsmod grep fglrx.

Существует и другая возможность — необходимо сначала создать пакет драйверов для вашего дистрибутива, а потом установить данный пакет. Такой метод несколько сложнее, но позволяет упростить техническую поддержку системы, а также удалить пакет любым инструментом для управления пакетами. Команда atidriver-installer --listpkg возвращает список поддерживаемых форматов пакетов и дистрибутивов, а ati-driver-installer --buildpkg distribution/version создает соответствующий пакет.

Если можно загрузить модуль ядра flgrx, вам еще потребуется выполнить команду `aticonfig`, чтобы соответствующим образом изменить `xorg.conf`, приспособив его к использованию драйвера fglrx.

```
root# aticonfig --initial
```

Обратите внимание, что программу установки необходимо заново выполнять после каждого обновления ядра. При этом модуль ядра снова компилируется, чтобы быть совместимым с текущей версией ядра!

Для деинсталляции драйвера выполните следующую команду:

```
root# /usr/share/ati/fglrx-uninstall.sh
```

Существует несколько сайтов, управляемых SUSE и содержащих советы по работе с этим дистрибутивом:

- http://en.opensuse.org/Howto/ATI_Driver;
- http://www.suse.de/~sndirsch/ati-installer-HOWTO.html.

Для дальнейшей конфигурации драйвера flgrx я советую вам использовать Catalyst Control Center (команда `amdccle`). Хотя пользовательский интерфейс и излучает очарование прошлого тысячелетия, работать с этой программой гораздо удобнее, нежели с `aticonfig`. Программа доступна, если у вас есть права обычного пользователя. Она не касается `xorg.conf`, а сохраняет настройки в файлах каталога `/etc/ati`.

Драйвер Intel

Драйвер Intel с открытым кодом совместим со всеми распространенными чипсетами Intel, за исключением GN40 и GMA500. Два последних чипсета, используемые в некоторых нетбуках, в настоящее время не поддерживаются. Общая информация по этому драйверу содержится в `man intel`, а также на следующем сайте: http://intellinuxgraphics.org/.

Поскольку этот драйвер официально входит в состав X, вы избавлены от утомительной установки дополнительных пакетов. Драйвер поддерживает 3D-графику (непосредственное отображение). Правда, скорость работы 3D-функций существенно уступает таковой в современных графических картах ATI и NVIDIA (рис. 12.1), но имеющейся скорости вполне достаточно для работы с трехмерными эффектами локальных программ или с Картами Google.

Драйвер Intel самостоятельно распознает оборудование, специально устанавливать параметры, как правило, не требуется. Когда я проводил испытания, возникали проблемы при одновременном использовании драйвера Intel и чипсета GM45-Express в конфигурации «Dual-Head»[1]; эти проблемы описаны в разделе 12.7.

Работа с X без прав администратора. Одна из особенностей драйвера Intel заключается в том, что он позволяет работать с X-сервером, если у вас нет прав администратора. Это удобно с точки зрения безопасности. Поскольку остальные

[1] Способ работы с двумя мониторами; информация, отображаемая на мониторах, разная.

графические драйверы такой функцией не обладают, во многих дистрибутивах система X по-прежнему требует от пользователя административных привилегий. Исключение представляет оптимизированный под оборудование Intel дистрибутив Moblin. Он разработан для нетбуков и содержит только драйвер Intel.

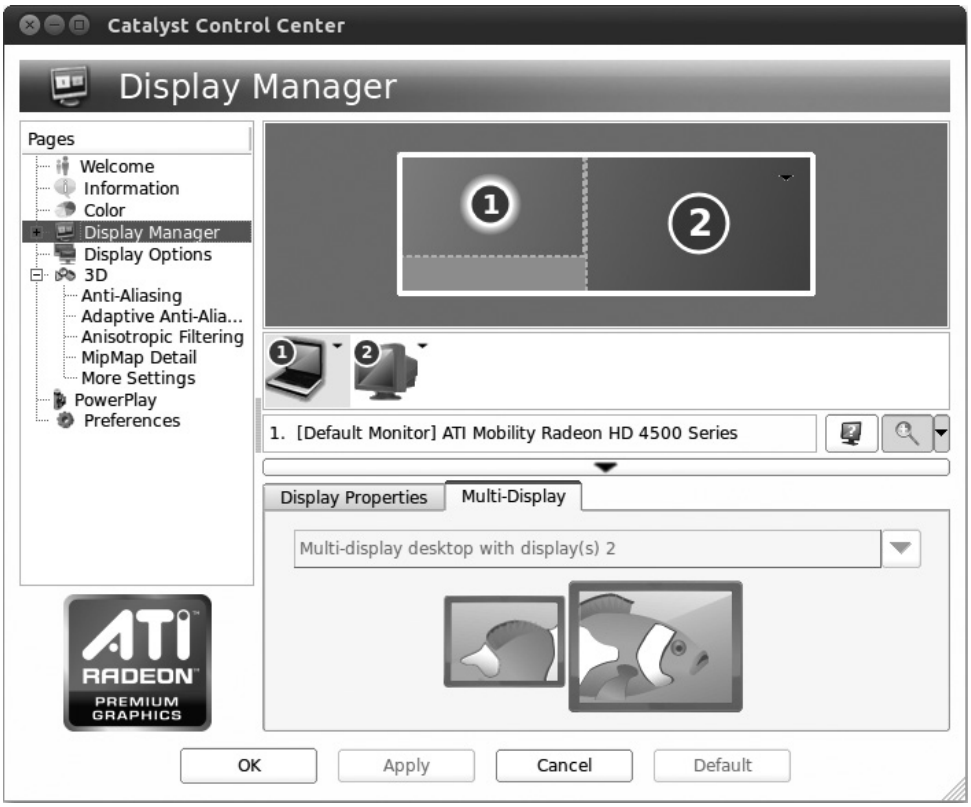

Рис. 12.1. Конфигурация драйвера ATI/AMD

Графическая карта NVIDIA

Если вы работаете с графической картой NVIDIA, то можете выбрать из следующих драйверов.

- nv — драйвер с открытым кодом, интегрированный в X.org еще несколько лет назад. Он работает со всеми распространенными моделями, но не поддерживает 3D-функций и имеет некоторые другие ограничения, в частности при работе с двумя мониторами, при использовании TV-выхода и т. д.
- nouveau — относительно новый драйвер с открытым кодом, который все больше составляет конкуренцию nv. 2D-функции этого драйвера могут соперничать с соответствующими функциями драйвера nv, а в новейших разрабатываемых версиях уже функционирует и поддержка *KMS (Переключение видеорежимов*

на уровне ядра). В будущем этот драйвер должен поддерживать и 3D-функции.

- nvidia — двоичный драйвер фирмы NVIDIA. Он поддерживает практически все функции современных графических карт. Этот драйвер доступен в двух вариантах: официальная версия поддерживает только современные модели графических карт, а версия Legacy — и устаревшие.

Драйвер nv

Драйвер nv входит в состав системы X.org, поэтому устанавливать его отдельно не требуется. В `man nv` перечислены некоторые параметры, позволяющие пользоваться специальными функциями этого драйвера. Минимальная конфигурация выглядит так:

```
Section "Device"
  Identifier  "Device0"
  Driver      "nv"
EndSection
```

Драйвер nouveau

Fedora 11 — первый дистрибутив, в котором драйвер nouveau установлен по умолчанию, но без экспериментальных 3D-функций. В других дистрибутивах драйвер nouveau предоставляется в виде пакета, но по умолчанию не устанавливается. Если вы хотите испробовать этот драйвер, то минимальная необходимая конфигурация выглядит так:

```
Section "Device"
  Identifier  "Device0"
  Driver      "nouveau"
EndSection
```

В `man nouveau` описано сравнительно немного параметров. Более подробная информация имеется здесь: http://nouveau.freedesktop.org/wiki/.

Драйвер nvidia

Лишь в немногих дистрибутивах имеется драйвер nvidia и одноименный модуль ядра — в виде пакета или в уже установленном виде. Однако часто встречаются репозитории, в которых содержатся пакеты драйверов, совместимые с текущей версией ядра. Тогда установка максимально упрощается: после создания пакета драйвер устанавливается с помощью обычных команд управления пакетами.

Драйвер nvidia не обращается к элементу DRI системы X.org, а содержит собственные DRI-драйверы. Поэтому при установке драйвера инсталлируются и специальные версии библиотек `libGL`. Одновременно изменяются ссылки `/usr/lib/libGL*`. При деинсталляции драйвера исходные варианты ссылок восстанавливаются командой `nvidia-installer -- uninstall`.

xorg.conf. Чтобы активизировать драйвер, вы должны, имея привилегии администратора, выполнить команду `nvidia-xconfig`. Она изменит `xorg.conf` так, что вместо драйвера, использовавшегося ранее, будет применяться драйвер nvidia. Если у вас есть особые пожелания к конфигурации, передайте `nvidia-xconfig` дополнительные

параметры, описанные на странице справки man. В простейшем случае в разделе Device будет всего две строки:

```
Section "Device"
  Identifier  "Device0"
  Driver      "nvidia"
EndSection
```

Драйвер nvidia несовместим с DRI-расширением X.org и сам выполняет функции этого расширения. Поэтому в xorg.conf не должно быть строки Load "dri".

```
Section "Module"
    # Load "dri" # удалить строку или прокомментировать
    ...
EndSection
```

Настройки nvidia. Когда драйвер NVIDIA заработает, его дальнейшая конфигурация будет выполняться программой nvidia-settings (рис. 12.2). Она позволяет непосредственно изменять различные параметры, то есть не требует перезапуска X. Если результат вас устраивает, можно сохранить изменения в xorg.conf. Правда, чтобы работать с программой таким образом, необходимы права администратора.

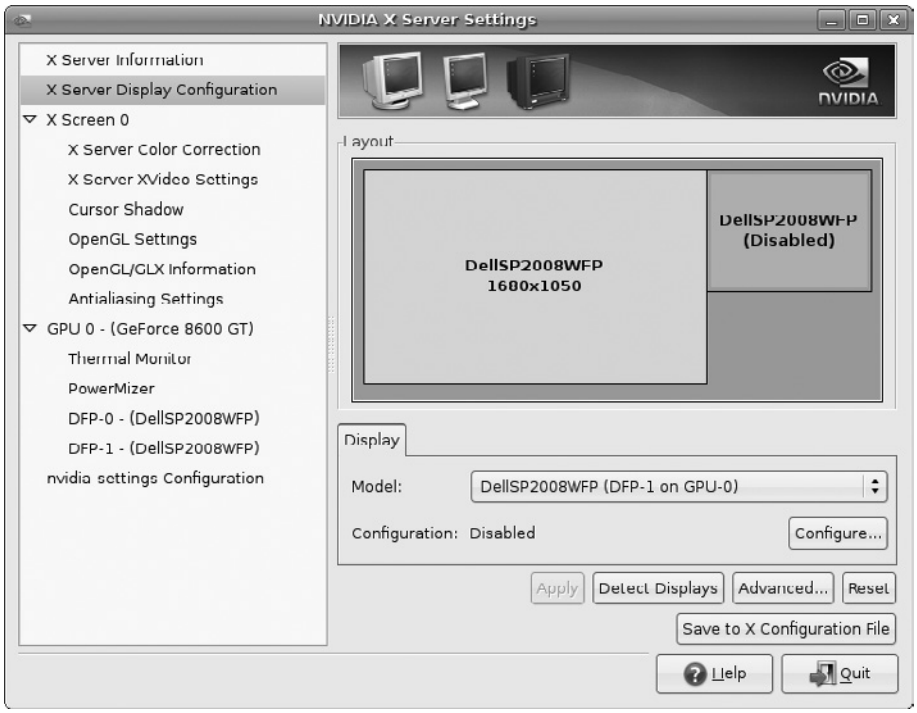

Рис. 12.2. Конфигурация драйвера NVIDIA

Когда я проводил испытания, у меня неоднократно возникали проблемы с сохранением актуальных настроек в xorg.conf в тех случаях, когда этот файл еще не

существовал или был пуст. Убедитесь, что в `xorg.conf` есть хотя бы короткий раздел `Device` с настройкой `Driver "nivdia"`!

Модуль ядра NVIDIA. В отличие от ATI/AMD, для работы драйвера NVIDIA в ядре обязательно должен содержаться одноименный модуль. Если этот модуль отсутствует либо скомпилирован для неподходящей версии ядра, то драйвер не будет работать и запустить X будет невозможно!

TwinView. Для работы с двумя мониторами драйвер NVIDIA предусматривает режим TwinView — интересную альтернативу Xinerama (который также поддерживается). В режиме TwinView драйвер NVIDIA управляет сплошной областью экрана, которая распределяется между обоими мониторами. Преимущества по сравнению с Xinerama заключаются в том, что все 3D-функции можно использовать на обоих мониторах и конфигурация очень проста. Конечно, есть и недостатки: на некоторых рабочих столах и в диспетчерах окон возникают проблемы с размещением окон, так как границы мониторов не учитываются или учитываются лишь частично (согласно NVIDIA это ошибки соответствующих программ, а не драйвера NVIDIA).

В следующих строках показаны важнейшие разделы простой конфигурации TwinView. При этом ЖК-монитор с разрешением 1920 × 1200 пикселов подключен к DVI-выходу графической карты, а второй ЖК-монитор с разрешением 1600 × 1200 пикселов — к выходу CRT. TwinView создает виртуальное разрешение 3520 × 1200 пикселов.

Хотя в такой системе два монитора, область экрана является единой. Основные характеристики обоих мониторов определяются автоматически. Благодаря параметру `"TwinViewXineramaInfoOrder" "DFP"` DVI-выход графической карты считается основным. Этот параметр выводит на экран, например окно для входа в систему X, а также панели KDE и Gnome (если в графической карте есть только один выход, и этот выход — CRT, то драйвер NVIDIA считает данный выход основным, что уже не отвечает требованиям времени).

```
Section "Device"
  Identifier  "Device0"
  Driver      "nvidia"
  VendorName  "NVIDIA Corporation"
  BoardName   "GeForce 7600 GS"
EndSection
Section "Screen"
  Identifier  "Screen0"
  Device      "Device0"
  DefaultDepth 24
  Option      "TwinView" "1"
  Option      "TwinViewXineramaInfoOrder" "DFP"
  Option      "metamodes" "DFP: nvidia-auto-select +0+0, \
                           CRT: nvidia-auto-select +1920+0"
  SubSection  "Display"
    Depth      24
  EndSubSection
EndSection
```

Установка вручную. Если для вашего дистрибутива не предусмотрены пакеты с нужными драйверами или такие пакеты устарели, вам потребуется самостоятельно установить драйвер, в частности скомпилировать модуль ядра nvidia. Для этого сначала установите все элементы, необходимые для компилирования модуля ядра (`C-Compiler`, `make`, файлы заголовков ядра и т. д. — см. также раздел 15.1). Вам также понадобятся пакеты `xorg-x11-server-sdk` или `xserver-xorg-dev`, а также `pkgconfig` или `pgk-config`.

Кроме того, скачайте с сайта http://www.nvidia.com/Download/index.aspx программу установки, подходящую для вашей графической карты и архитектуры вашего дистрибутива (32 или 64 бит). Программа имеет размер около 20 Мбайт.

С помощью метода `init 3` или `/etc/init.d/gdm stop` (подробности — в разделе 12.2) вы завершаете работу X, а затем, с правами администратора, запускаете программу установки:

```
root# sh NVIDIA-Linux-версия.run
```

Программа установки действует в текстовом режиме, но управляется через диалоговые окна и удобна в обслуживании. Сначала она проверяет, есть ли на сайте NVIDIA готовый скомпилированный модуль, который предназначается для вашей версии ядра. Если его нет, то программа компилирует подходящий модуль. Если все необходимые условия будут выполнены, эта операция занимает несколько секунд. Программа дополнительно устанавливает NVIDIA-специфичные библиотеки `libGL` и после запроса о подтверждении выбранных действий выполняет упомянутую выше команду `nvidia-xconfig`.

Подробный протокол всех совершенных действий находится в файле `/var/log/nvidiainstaller.log`. Если никаких ошибок не возникло, то можно перезапускать X с новым драйвером NVIDIA.

Обратите внимание — после каждого обновления ядра программу установки необходимо выполнять заново. При этом модуль ядра nvidia компилируется заново, чтобы он был совместим с текущей версией ядра.

Если хотите деинсталлировать драйвер, выполните приведенную далее команду. Она восстанавливает прежний файл `xorg.conf`, удаляет модуль ядра nvidia и создает ссылки на оригинальные библиотеки `libGL`.

```
root# nvidia-installer --uninstall
```

По драйверу NVIDIA также имеется подробная и полезная документация. Просто перейдите на сайт загрузки NVIDIA по ссылке README. В SUSE к тому же был создан дополнительный сайт с советами, специфичными для данного дистрибутива: http://www.suse.de/~sndirsch/nvidia-installer-HOWTO.html.

Драйверы VESA, Framebuffer и VGA

Если вы используете графическую карту, к которой (пока) нет драйверов, то в качестве временного решения вам подойдут следующие три драйвера. Хотя изображение формируетсся сравнительно медленно, и, разумеется, не поддерживаются никакие 3D-функции, эти драйверы как минимум позволяют работать с графической системой.

Драйвер VESA

С помощью драйвера VESA вы можете использовать все VESA-режимы своей графической карты. Некоторые фоновые сведения: Ассоциация Видеоэлектронных Стандартов (Video Electronics Standard Association) разработала нормы некоторых графических режимов для стандартных разрешений экрана. Каждый режим определяется следующими ключевыми показателями: разрешение (например, 1280 × 1024 пикселов), глубина цвета и кадровая частота. Почти все графические карты поддерживают не только собственный графический режим, но и несколько режимов VESA.

Как показано в следующих строках, применять VESA-драйвер совсем не сложно. Если в остальном конфигурационный файл не содержит ошибок, то учитываются все режимы VESA, которые поддерживаются графической картой и которые монитор может отобразить.

```
# in /etc/X11/xorg.conf
...
Section "Device"
  Identifier  "myDevice"
  Driver      "vesa"
EndSection
```

Драйвер Framebuffer

Драйвер fbdev обращается прямо к памяти (кадровому буферу) графической карты. Таким образом, он работает еще на уровень глубже, чем драйвер VESA. Он должен работать практически с любыми графическими картами, если ядро Linux было скомпилировано с учетом поддержки кадрового буфера. Если такая поддержка предусмотрена, то в системе должен быть файл `/proc/fb`.

Если вы хотите иметь возможность работать с этим драйвером, важно, чтобы уже при загрузке компьютера был выбран правильный режим VGA. До перезапуска компьютера вы можете использовать X только в заданном графическом режиме. Чтобы выбрать режим, вставьте в конфигурационный файл GRUB или LILO параметр ядра `vga=`*n*. Правильные (десятичные) значения для *n* приведены в табл. 12.2. Кроме того, в SUSE с помощью команды `hwinfo -framebuffer` предусмотрена возможность получить список режимов кадрового буфера, поддерживаемых вашей графической картой.

Таблица 12.2. Десятичные значения для параметра ядра

	640 × × 400	**640 × × 480**	**800 × × 600**	**1024 × × 640**	**1024 × × 768**	**1152 × × 720**	**1280 × × 1024**	**1440 × × 900**	**1600 × × 1200**
8 bpp	768	769	771	874	773	869	775	864	796
16 bpp (5:5:5)	797	784	787	875	790	870	793	865	797
16 bpp (5:5:5)	798	785	788	876	791	871	794	866	798
24 bpp	799	786	789	877	79	872	795	867	799
32 bpp	834	809	814	878	824	873	829	868	834

В xorg.conf требуется только указать соответствующую драйверу последовательность символов:

```
# in /etc/X11/xorg.conf
...
Section "Device"
  Identifier  "myDevice"
  Driver      "fbdev"
EndSection
```

Драйвер VGA

Этот драйвер поддерживает только разрешения 640 × 480 или 800 × 600 точек при глубине цвета 4 бита (то есть 16 цветов), поэтому его следует применять лишь при отсутствии лучшего. Более подробная информация сообщается в справке man vga.

12.5. Клавиатура и мышь

Есть несколько драйверов, с помощью которых X обменивается информацией с клавиатурой, мышью или сенсорной панелью. Новее и современнее остальных драйвер evdev для мыши и клавиатуры. Fedora и Ubuntu в версиях 11 и 9.04 соответственно уже перешли к использованию драйвера evdev для мыши и клавиатуры. Debian 5, openSUSE 11.1 и более старые дистрибутивы, напротив, используют драйвер xkbd для клавиатуры и mouse для мыши. В ноутбуках с сенсорной панелью применяется драйвер synaptics. Чтобы узнать, какие драйверы использует система X на вашем компьютере, лучше всего взглянуть в файл регистрации X:

```
root# grep LoadModule /var/log/Xorg.0.log
...
(II) LoadModule: "evdev"
(II) LoadModule: "synaptics"
```

Драйвер evdev (мышь и клавиатура)

Основное достоинство драйвера evdev по сравнению с более старыми драйверами xkbd и mouse заключается в том, что он без проблем распознает мыши и клавиатуры, подсоединяемые без выключения компьютера (методом *горячего подключения*). Для автоматической активизации устройств ввода применяются два следующих файла правил HAL:

- /usr/share/hal/fdi/policy/10osvendor/10-x11-input.fdi;
- /usr/share/hal/fdi/policy/10osvendor/10-x11-keymap.fdi.

Файл 10-x11-input.fdi активизирует драйвер evdev, когда HAL распознает мышь или клавиатуру. Благодаря 10-x11-keymap.fdi на любой клавиатуре автоматически устанавливается нужная раскладка. В Fedora в этом случае вызывается программа /usr/bin/fedora-setup-keyboard, интерпретирующая файл /etc/sysconfig/keyboard. Ubuntu, в свою очередь, использует сценарий /usr/lib/hal/debian-setup-keyboard, интерпретирующий /etc/default/console-setup.

Настройки мыши и клавиатуры в xorg.conf при использовании драйвера evdev задавать нельзя! Более подробно этот вопрос рассмотрен на следующем сайте: https://fedoraproject.org/wiki/Features/EvdevInputDriver.

Драйвер xkbd (клавиатура)

Драйвер xkbd статически конфигурируется в разделе InputDevice файла xorg.conf. Если этот раздел отсутствует, то клавиатура обычно функционирует, но только с американской раскладкой. В следующих строках показано, как задать для клавиатуры немецкую раскладку:

```
Section "InputDevice"
  Identifier  "myKeyboard"
  Driver      "Keyboard"
  Option      "XkbModel"    "pc105"
  Option      "XkbLayout"   "de"
  Option      "XkbVariant"  "nodeadkeys"
EndSection
```

К сожалению, ключевые слова Xkb*Xxx* плохо документированы. В следующих пунктах я обобщил известные мне возможности настройки.

- XkbRules определяет, как должны интерпретироваться настройки остальных параметров. Как правило, правильной настройкой здесь является xorg. Только для японских клавиатур PC-98 необходимо указывать xfree98.
- XkbModel описывает клавиатуру. Среди допустимых значений следует упомянуть следующие:
  - pc101 — раскладка США без клавиш Windows (стандартная настройка);
  - pc102 — международная раскладка без клавиш Windows;
  - pc104 — раскладка США с клавишами Windows;
  - pc105 — международная раскладка с клавишами Windows;
  - abnt2 — бразильская раскладка;
  - jp106 — японская раскладка;
  - pc98 — японская раскладка PC-98;
  - macintosh — Apple Macintosh;
  - powerpcps2 — Apple Power PC.
- XkbLayout описывает расположение клавиш на клавиатуре, которое зависит от страны. В качестве настроек допустимы обычные коды стран, например us (английский), de (немецкий), ru (русский).
- XkbVariant позволяет задавать дополнительные настройки раскладки клавиатуры. Из таких настроек наиболее применяема nodeadkeys. Благодаря ей символы ~ ^ ' ` можно вводить непосредственно, таким образом, они не используются для составления символов из иностранных языков.
- XkbOptions задает дополнительные параметры, передаваемые команде setxkbmap -option.

Чтобы оптимально использовать в Linux клавиатуру с различными специальными клавишами, можно установить программу LinEAK (*Linux support for Easy Access and Internet Keyboards*, пакет называется lineak*). Подробная информация по конфигурации содержится в справке `man lineakd`, а также на следующем сайте: http://lineak.sourceforge.net/.

Драйвер mouse (мышь)

Этот драйвер также конфигурируется в разделе `InputDevice`. Если раздел отсутствует, система X пытается сама угадать подходящую конфигурацию, и обычно ей это удается. В следующих строках показана минимальная конфигурация для мыши с колесиком.

```
Section "InputDevice"
  Identifier  "myMouse"
  Driver      "mouse"
  Option      "Protocol"     "Auto"
  Option      "Device"       "/dev/input/mice"
  Option      Buttons 5
  Option      "ZAxisMapping" "4 5"
EndSection
```

Для конфигурирования мыши предусмотрены следующие ключевые слова.

- `Protocol` указывает, как происходит обмен информацией между мышью и компьютером. На выбор предлагаются в том числе следующие варианты:
  - `Auto` — X пытается сама распознать протокол;
  - `ExplorerPS/2` — мышь с колесиком, подсоединяемая к интерфейсу PS/2;
  - `IMPS/2` — Microsoft-совместимая мышь с колесиком, подсоединяемая к интерфейсу PS/2;
  - `IntelliMouse` — Microsoft-совместимая мышь с колесиком, подсоединяемая к последовательному интерфейсу;
  - `PS/2` — стандартная мышь, подсоединяемая к интерфейсу PS/2;
  - `Usb` — USB-мышь.
- `Device` указывает, как мышь подсоединяется к компьютеру. Обычно применяются настройки `/dev/input/mousen` или `/dev/input/mice`, причем во втором случае параллельно интерпретируются все подключенные мыши, сенсорные панели и т. д.
- `Buttons` определяет, сколько кнопок у мыши. По умолчанию X ожидает, что у мыши три кнопки. Обратите внимание — каждое колесико считается за две кнопки. Иначе говоря, если у мыши три кнопки и одно колесико, то верная настройка — `5`.
- `ZAxisMapping` указывает, каким виртуальным кнопкам присваиваются имеющиеся колесики и их движения. Иными словами, если вы крутите колесико в том или ином направлении, то X интерпретирует это как нажатие определенной

кнопки. Если вы крутите его в обратном направлении, это соответствует нажатию другой кнопки.

- Благодаря `Emulate3Buttons` вы можете эмулировать отсутствующую среднюю кнопку мыши, одновременно нажимая левую и правую. Это, конечно, решение на самый крайний случай, но оно помогает эмулировать щелчок средней кнопкой при использовании двухкнопочной мыши. Кроме того, параметр `Emulate3Buttons` позволяет задавать в миллисекундах время, в течение которого нужно нажать обе кнопки. Если задать слишком краткий промежуток, то два не совсем синхронных щелчка будут интерпретироваться как разные нажатия. Однако и слишком больших промежутков задавать не стоит, так как это сильно замедляет реакцию системы на щелчки (ведь X «не знает», интерпретировать ли щелчок кнопкой мыши как один или ждать второго). Советую использовать такие настройки:

```
Option "Emulate3Buttons"
Option "Emulate3Timeout" "50"
```

MPX и XInput2

Если при работе с актуальной версией X подключить несколько мышей (например, подсоединить к ноутбуку с сенсорной панелью внешнюю мышь), то команды всех индикаторных устройств как бы будут «складываться» (`/dev/input/mice`). Другими словами, вы можете немного передвинуть указатель с помощью сенсорной панели, а затем продолжить движение с помощью мыши.

Предполагается, что новые версии X смогут различать несколько индикаторных устройств и обеспечивать независимое перемещение нескольких указателей (то есть, например, красной и зеленой стрелок мыши). Это основано на технологии Multi Pointer X и драйвере Xinput2 (кратко — XI2). Оба компонента входят в состав версии X.org 7.5. Однако пока совершенно неясно, какую пользу принесет поддержка нескольких мышей пользователю и есть ли на самом деле области, в которых такая поддержка была бы целесообразна. Более подробная информация содержится в следующей статье: http://lwn.net/Articles/337898/.

Драйвер synaptics (для сенсорной панели)

На большинстве ноутбуков установлены сенсорные панели фирмы Synaptic или совместимые с ними компоненты. В принципе протокол этих устройств эмулирует обычную мышь, так что для использования сенсорной панели в качестве мыши в системе X никаких специальных драйверов не требуется. Однако, чтобы можно было пользоваться различными дополнительными функциями сенсорной панели, вместо драйвера mouse обычно используется драйвер synaptic. Система X автоматически загружает этот драйвер при запуске, в том числе в таких дистрибутивах, в которых для работы с мышью и клавиатурой используется драйвер evdev.

Как и при работе с xkbd и mouse, конфигурировать `xorg.conf` вручную требуется лишь тогда, когда стандартные настройки X недостаточно эффективны. Приведу пример:

```
Section "InputDevice"
  Identifier  "Synaptics"
  Driver      "synaptics"
  Option      "Device"          "/dev/input/mice"
  Option      "Protocol"        "auto-dev"
  Option      "Emulate3Buttons" "yes"
EndSection
```

Параметр `Emulate3Buttons` необходим потому, что в большинстве сенсорных панелей отсутствует третья (средняя) кнопка. Теперь с помощью дополнительных параметров (см. `man synaptics`) можно активизировать всевозможные дополнительные функции. Чтобы полностью отключить сенсорную панель (если вы работаете с внешней мышью), замените раздел `Synaptics` в `xorg.conf` разделом для мыши либо используйте параметр `"TouchpadOff" "1"`.

Параметр `"SHMConfig" "on"` позволяет изменять параметры сенсорной панели на ходу, то есть без перезапуска X. Для этого применяется команда `synclient` или графические интерфейсы `gsynaptics`. Команда `synclient name=value` изменяет указанный параметр, `synclient -l` возвращает актуальные настройки, `synclient -m 100` отслеживает текущее состояние сенсорной панели с периодичностью 100 мс.

Gnome и KDE

Независимо от конфигурации X, Gnome и KDE позволяют отдельно настроить клавиатуру, мышь и сенсорную панель. Чтобы задать конфигурацию клавиатуры, запустите в Gnome программу `gnome-keyboard-properties`. В KDE используйте модуль Региональные и специальные возможности центра управления.

Обе программы позволяют настроить работу клавиши CapsLock, а также указать, должна ли одна из клавиш (например, клавиша Windows) работать в качестве «клавиши композиции» (compose key). Такая клавиша позволяет объединять два символа. Например, если нажать Compose+A+E, получится `AE`. Если вы хотите быстро переключаться между разными раскладками клавиатуры (например, при написании текста сразу на нескольких языках), добавьте к панели Gnome минипрограмму Tastaturindikator либо установите соответствующие настройки в разделе Раскладка клавиатуры в центре управления KDE.

12.6. Динамическое изменение конфигурации с помощью RandR

Расширение *Resize and Rotate Extension (RandR)* позволяет на ходу изменять конфигурацию X, если такая функция поддерживается графическим драйвером. Современные рабочие столы автоматически принимают новые рамочные условия (например, изменение разрешения экрана) и не требуют перезапуска системы.

Если хотите вносить изменения вручную, используйте команду `xrandr` (описание ее синтаксиса приводится на странице справки `man`). Графические интерфейсы, интегрированные в Gnome и KDE, значительно удобнее в использовании, чем `xrandr`.

Команда xrandr

При использовании команды `xrandr` без параметров или с параметром `-q` отображается текущее состояние системы X. Следующий вывод означает, что монитор использует сигнальный выход DVI и работает с разрешением 1680 × 1050 пикселов. Данные в скобках говорят о том, что изображение можно повернуть, отразить или инвертировать. В следующем списке показано, какие еще разрешения можно настроить. Важно также указывать названия сигнальных выходов (в данном случае VGA-0 и DVI-I-0), поскольку эти последовательности символов отличаются в зависимости от драйвера.

```
user$ xrandr
Screen 0: minimum 320 x 200, current 1680 x 1050, maximum 4080 x 4096
VGA-0 disconnected
DVI-I-0 connected 1680x1050+0+0 (normal left inverted right x axis y axis)
  434mm x 270mm
    1680x1050     60.0*+
    1400x1050     60.0
    1280x1024     75.0      60.0
    1280x960      60.0
    1152x864      75.0
    1024x768      75.0      70.1    60.0
    832x624       74.6
    800x600       72.2      75.0    60.3     56.2
    640x480       75.0      72.8    75.0     59.9
    720x400       70.1
```

Вы можете работать с `xrandr`, имея права обычного пользователя. Все сделанные изменения действительны до тех пор, пока вы не выйдете из системы. Данная команда понижает разрешение до 1280 × 1024:

```
user$ xrandr --size 1280x1024
```

Следующая команда активизирует оба выхода. На двух мониторах отображается одинаковая картинка. Благодаря параметру `--auto` мониторы работают с оптимальным разрешением и кадровой частотой.

```
user$ xrandr --output DVI-I-0 --auto --output VGA-0 --auto
user$ xrandr --output VGA-0 --off    (снова отключает выход VGA)
```

Инструмент gnome-display-properties

Инструмент Gnome для изменения конфигурации RandR называется `gnome-display-properties`. Обычно он запускается через меню Система ▸ Параметры ▸ Отображение (рис. 12.3).

Если вы работаете с несколькими мониторами, то по умолчанию оба показывают одну и ту же картинку (режим *cloning*), причем размер изображения рассчитывается исходя из наименьших значений разрешения по горизонтали и по вертикали. Если же вы снимете флажок Отображать экраны, то можно будет автономно конфигурировать оба монитора, в том числе их расположение относительно друг

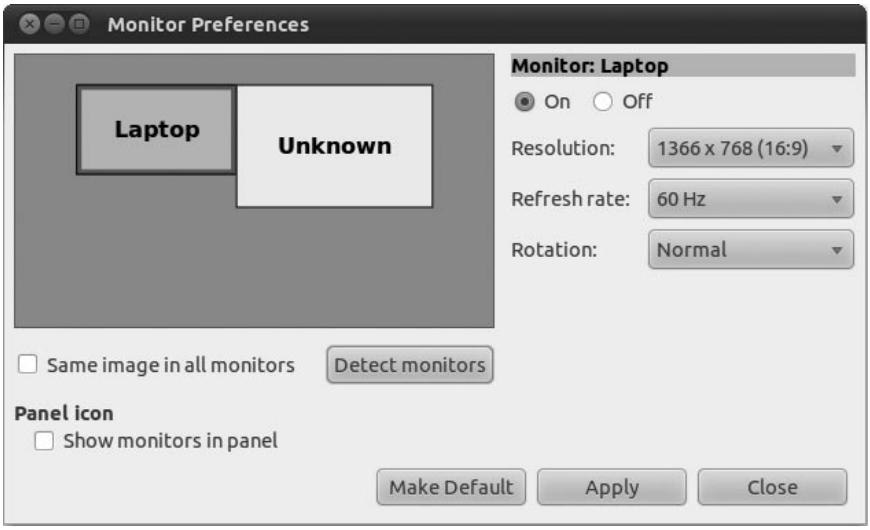

Рис. 12.3. Изменение настроек RandR в Gnome

друга. В зависимости от применяемого вами графического драйвера может потребоваться сначала увеличить размер виртуального экрана в `xorg.conf`. Чтобы внести такое изменение, требуются права администратора; изменение вступает в силу только после нового входа в систему.

Настройки `gnome-display-properties` сохраняются в файле `~/.config/monitors.xml`. Они действительны только для текущего пользователя и автоматически применяются при его следующем входе в Gnome (за исключением тех случаев, когда вы работаете с другим рабочим столом!).

Модуль KDE

Программа KDE, аналогичная `gnome-display-properties`, — это модуль Управление системой ▸ Отображение. В принципе функционально он не отличается от `gnome-display-properties`, но сохраняет настройки в файле `~/.kde/share/config/krandrrc`.

12.7. Конфигурация с двумя мониторами и проектором

К одной графической карте можно подключить два монитора — подобная конфигурация называется «Dual-Head». Существуют и такие графические карты, к которым можно подключать еще больше мониторов, но эти случаи мы рассматривать не будем. Один из вариантов конфигурации «Dual-Head» — подключение ноутбука к проектору или к внешнему монитору.

Проблемы. Многие программы Linux работают с несколькими мониторами с большим трудом: диалоговое окно выводится не на тот монитор, с которым вы

сейчас работаете, раскрывающиеся меню пересекают линию границы двух мониторов и т. д. Особенно тяжелый участок работы — панель Gnome, которая иногда отображается не на том мониторе, на котором вы хотите ее видеть. Конфигурационное диалоговое окно, которое помогло бы отрегулировать положение этой панели, отсутствует. Вам потребуется указать нужный монитор в базе данных gconf (измените integer 0 на integer 1).

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

Варианты конфигурации. В X предусмотрено два варианта конфигурации для работы с несколькими мониторами.

- Простейший способ — конфигурация через RandR. Для этого вам потребуется виртуальный монитор (Screen) такого размера, чтобы на нем умещались оба реальных монитора. Если, например, у вас два монитора с разрешениями соответственно 1280×1024 и 1600×1200 пикселов и вы хотите работать с ними одновременно, виртуальное разрешение должно составлять 2880×1200 пикселов. Если это условие выполняется, то можно активизировать второй монитор с помощью программы xrandr или других конфигурационных инструментов RandR. Такой метод работает с большинством распространенных графических драйверов.
- NVIDIA-специфичный вариант этой конфигурации называется TwinView. Конфигурация выполняется в файле xorg.conf (без применения RandR).
- При использовании расширения Xinerama также можно объединить два и более объекта Screen в большую общую картинку (в соответствии с xorg.conf). Для каждого объекта Screen можно задать любые настройки (разрешение, глубину цвета). Этот вариант особенно хорош в тех случаях, когда на одном компьютере стоит несколько графических карт. Существенный недостаток заключается в том, что 3D-функции, как правило, работают только на одном мониторе (независимо от количества графических карт). Xinerama считается устаревшей концепцией и в будущем, вероятно, поддерживаться не будет.

Конфигурация с двумя мониторами — вариант RandR

При использовании конфигурации с двумя мониторами и RandR необходимо достаточно высокое виртуальное разрешение. Однако большинство графических драйверов дают разрешение не больше, чем у основного монитора (у ноутбуков основным монитором считается встроенный, а не внешний), поэтому зачастую необходимо жестко задавать более высокое виртуальное разрешение. Соответствующие настройки указываются в файле xorg.conf. Если имеются параметры Identifier для Screen и Device, оставьте их без изменений. Обратите внимание, что в разделе Screen должно содержаться указание на подходящий раздел Device:

```
Section "Screen"
  Identifier  "Screen0"
```

```
  Device      "Device0"
  SubSection "Display"
    Virtual 2880 1200
  EndSubSection
EndSection
Section "Device"
  Identifier "Device0"
EndSection
```

После перезапуска X можно настроить оба монитора с помощью конфигурационных инструментов KDE и Gnome либо вручную, с помощью `xrandr`:

```
user$ xrandr
Screen 0: minimum 320 x 200, current 1280 x 800, maximum 2880 x 1200
VGA disconnected (normal left inverted right x axis y axis)
LVDS connected 1280x800+0+0 (normal left inverted right x axis y axis)
  331mm x 207mm
  1280x800     59.9*+
HDMI-1 connected 1280x800+0+0 (normal left inverted right x axis y axis)
  519mm x 324mm
  1280x800     59.9*+
user$ xrandr --output HDMI-1 --mode 1600x1200 --right-of LVDS
```

На примере следующей команды видно, какие возможности предоставляет RandR — испытания проводили с драйвером nouveau, а также двумя мониторами: 1680 × 1050 (DVI) и 1600 × 1200 пикселов (VGA). Определяется виртуальная рабочая область размером 3864 × 2415 пикселов. На меньшем мониторе общая рабочая область уменьшается в 2,3 раза и отображается размером 3864 / 2,3 = 1680. На большем мониторе демонстрируется фрагмент картинки размером 1600 × 1200 вокруг точки, в которой находится указатель мыши, в масштабе 1:1. Рядом с мышью можно видеть еще минимум 256 пикселов (только если стрелка не находится у края экрана). Если вам всегда хотелось узнать, каково это — работать с монитором размером 3864 × 2415 пикселов, попробуйте, причем без всяких дополнительных затрат!

```
root# xrandr -fb 3864x2415 --output DVI-I-0 --scale 2.3x2.3 \
             --output VGA-0 --pos 0x0 \
             --panning 3864x2415+0+0/3864x2415+0+0/256/256/256/256
```

Файл xorg.conf

Разумеется, конфигурацию с несколькими мониторами можно жестко задать и в `xorg.conf`. Принцип заключается в том, что вы перечисляете используемые мониторы в разделах `Monitor`. Для второго монитора вы указываете, как он будет расположен относительно первого. Допустимые ключевые слова — `RightOf`, `LeftOf`, `Below` и т. д. соответствуют параметрам `xrandr`. При необходимости используйте `Option "Position" "x y"`, чтобы точно задать координаты на виртуальном экране. В разделе `Device` укажите, какие сигнальные выходы `monitor-`*xxx* с какими мониторами будут соединены.

Следующая конфигурация предназначается для работы с драйвером nouveau. Для других драйверов потребуется изменить последовательность символов `monitor-xxx`, так как каждый драйвер использует собственную номенклатуру для обозначения сигнальных выходов. Просто выполните `xrandr`, чтобы выяснить, как называются сигнальные выходы в вашей системе. При работе с драйвером nouveau можно специально не устанавливать размер виртуального экрана, так как по умолчанию экран может иметь размер до 4080 × 4096 пикселов.

```
# /etc/X11/xorg.conf
Section "Monitor"
  Identifier     "dvi0"
EndSection
Section "Monitor"
  Identifier     "vga0"
  Option         "RightOf" "dvi0"
EndSection
Section "Device"
  Identifier     "device0"
  Driver         "nouveau"
  Option         "monitor-VGA-0"    "vga0"
  Option         "monitor-DVI-I-0"  "dvi0"
EndSection
```

ВНИМАНИЕ

Настройки, сделанные в xorg.conf, вступают в силу только после перезапуска X. Кроме того, они игнорируются, если вы изменили конфигурацию RandR с помощью инструментов Gnome или KDE. При необходимости удалите файл ~/.config/monitors.xml (Gnome) или ~/.kde/share/config/krandrrc (KDE)!

Проблемы с Intel

В большинстве чипсетов Intel (за исключением новейших моделей) трехмерная графика (DRI) функционирует лишь при условии, что виртуальное разрешение в любом направлении не превышает 2048 × 2048 точек. Отключите Compiz, или часть монитора у вас получится черной, с разной неверной информацией! Еще одна проблема, которую мне не удалось решить, — жесткая настройка конфигурации с несколькими мониторами в `xorg.conf` (однако удалась динамическая конфигурация с `xrandr`). Драйвер Intel просто игнорировал соответствующие параметры.

На сайте http://intellinuxgraphics.org/dualhead.html приводятся советы по конфигурации Intel. Однако приведенные там примеры `xorg.conf` не работали на моем тестовом ноутбуке, где используется чипсет GM45-Express.

Конфигурация TwinView с одним экраном

Если вы работаете с графическим драйвером nvidia, то вышеописанный метод применить не удастся. Вместо этого используйте nvidia-специфичный режим TwinView. Далее показаны важнейшие разделы простой конфигурации с TwinView.

При этом ЖК-монитор с разрешением 1920 × 1200 пикселов подключается к DVI-выходу графической карты, а второй ЖК-монитор с разрешением 1600 × 1200 пикселов подключается к CRT-выходу. TwinView дает виртуальное разрешение 3520 × 1200 пикселов.

Основные показатели монитора определяются автоматически. Благодаря `"TwinViewXineramaInfoOrder" "DFP"` DVI-выход графической карты считается основным. С помощью этого параметра на подключенном здесь мониторе выводятся, например, окно для входа в систему X, а также панели KDE и Gnome (по умолчанию, если на графической карте есть выход CRT, драйвер nvidia считает этот выход основным, что уже не отвечает требованиям времени).

```
# /etc/X11/xorg.conf
Section "Device"
  Identifier   "Device0"
  Driver       "nvidia"
  VendorName   "NVIDIA Corporation"
  BoardName    "GeForce 7600 GS"
EndSection
Section "Screen"
  Identifier   "Screen0"
  Device       "Device0"
  DefaultDepth  24
  Option       "TwinView" "1"
  Option       "TwinViewXineramaInfoOrder" "DFP"
  Option       "metamodes" "DFP: nvidia-auto-select +0+0, \
                            CRT: nvidia-auto-select +1920+0"
  SubSection "Display"
    Depth       24
  EndSubSection
EndSection
```

Конфигурация Xinerama с несколькими экранами

Для того чтобы пользоваться Xinerama, необходимо объявить по два раздела `Monitor`, `Device` и `Screen`. В разделе `ServerLayout` активизируем параметр Xinerama и указываем, как объекты на экране должны располагаться относительно друг друга. Минимальная конфигурация должна строиться по приведенному ниже образцу. Итак, нам требуется два раздела `Device`, хотя у нас всего одна графическая карта!

```
# два раздела Monitor и два раздела Device
Section "Monitor"
  Identifier "Monitor1"
  ...
EndSection
Section "Monitor"
  Identifier "Monitor2"
  ...
EndSection
Section "Device"
```

```
  Identifier "Videocard1a"
  Driver     "..."
  Screen      0
  ...
EndSection
Section "Device"
  Identifier "Videocard1b"
  Driver     "..."
  Screen      1
  ...
EndSection
# два раздела Screen, связывающие Device и Monitor
Section "Screen"
  Identifier "Screen1a"
  Device     "Videocard1a"
  Monitor    "Monitor1"
  DefaultDepth 24
  SubSection "Display"
    Viewport   0 0
    Depth      24
  EndSubSection
EndSection
Section "Screen"
  Identifier "Screen1b"
  Device     "Videocard1b"
  Monitor    "Monitor2"
  DefaultDepth 24
  SubSection "Display"
    Viewport   0 0
    Depth      24
  EndSubSection
EndSection
# Раздел ServerLayout, объединяющий разделы Screen
Section "ServerLayout"
  Identifier "Layout1"
  Screen      0 "Screen1a" LeftOf "Screen1b"
  Screen      1 "Screen1b"
  Option     "Xinerama" "on"
EndSection
```

Вы также можете задать в `xorg.conf` несколько разделов `ServerLayout` для различных сценариев работы, например `Layout0` для обычной работы с ноутбуком (с локальным монитором), `Layout1` — для показа презентаций с использованием проектора и `Layout2` — для работы с внешним монитором. Нужный раздел `ServerLayout` можно выбрать в разделе `ServerFlags` с помощью параметра `DefaultServerLayout`.

Советы о том, как подключить проектор

Каждый, кому хоть раз приходилось показывать презентацию со своего ноутбука с Linux, помнит это сосание под ложечкой: удастся ли синхронизировать картинку

на проекторе? В последние годы у меня с этим проблем не возникало — конфигурация RandR с `gnome-display-properties` каждый раз запускалась «с пол-оборота». Однако пару лет назад нередко приходилось быстро внести в `xorg.conf` несколько небольших изменений. Если у вас возникают проблемы с подключением проектора, твердо усвойте следующие правила.

- Сначала подключайте ноутбук к проектору, а только потом включайте! Как правило, это позволяет отключить экран ноутбука и активизировать внешний сигнальный выход. Если вам повезет, то сообщения о запуске системы будут выводиться уже на проектор. При необходимости вы еще сможете установить в KDE или Gnome разрешение, оптимальное для работы с проектором (обычно это 1024×768 точек).
- На некоторых ноутбуках внешний сигнальный выход можно активизировать прямо в BIOS.
- Протестируйте внешний выход ноутбука дома с любым монитором. Конечно, нельзя гарантировать, что проектор поведет себя так же, как и этот монитор, но такой тест все же позволяет заблаговременно обнаружить возможные проблемы.
- Чтобы обойти проблему с проектором, можно установить в `xorg.conf` разрешение 1024×768 точек, частоту развертки — примерно 53 КГц, а кадровую частоту снизить до 60 КГц. С такими показателями работает большинство проекторов (в том числе не самых новых):

```
Section "Monitor"
  ...
  HorizSync    31.5 - 53
  VertRefresh  57-63
EndSection
Section "Screen"
  ...
  DefaultDepth 24
  SubSection "Display"
    Modes    "1024x768"
  EndSubSection
EndSection
```

12.8. Трехмерная графика и видео

3D-графика. Раньше 3D-графика в Linux влачила жалкое существование: во-первых, кроме экранных заставок и пары игр она нигде не применялась, во-вторых, для работы с трехмерной графикой требовалось использовать непопулярные двоичные драйверы. Теперь ситуация с драйверами немного изменилась, но несколько лет назад появились революционные новинки, использующие 3D. Это не кровавые шутеры, как вы могли подумать, а трехмерные эффекты рабочего стола, исполь-

зующие функции графической карты для необычных трансформаций изображения. Так, они позволяют разделить окно на мелкие кусочки, проецировать рабочие плоскости на грани кубика и т. д. Польза от таких функций невелика, зато это очень эффектно выглядит. В данном разделе даются некоторые практические советы по работе с 3D- и видеофункциями в X.

Ссылки. Внимательно изучите глоссарий в разделе 12.1! Там объясняются важнейшие сокращения из 3D-номенклатуры для X (AIGLX, DRI, GLX, OpenGL и др.). Прочая базовая информация по теме 3D-графики в Linux находится в том числе на следующих сайтах:

- http://www.mesa3d.org;
- http://dri.freedesktop.org/wiki/;
- http://jonsmirl.googlepages.com/graphics.html (старый).

Конфигурация. В отличие от предыдущих случаев, для использования 3D-графики не требуется никакой особой работы по настройке конфигурации. При работе с NVIDIA и современными графическими картами ATI/AMD потребуется установить двоичный драйвер от поставщика. 3D-функции — если они доступны — будут автоматически активизироваться при запуске X. Если ваше оборудование не поддерживает 3D-функции, то они реализуются с помощью программного обеспечения (Mesa). В принципе это работает с большинством приложений, но скорость остается недостаточно высокой.

3D-тест. С помощью программы `glxinfo` проверим, все ли у нас получилось. Программа выдает множество подробностей относительно работающей системы GLX. С помощью команды `grep` можно отфильтровать важные строки:

```
root#  glxinfo | grep render
direct rendering: Yes
OpenGL renderer string: GeForce 7600 GS/PCI/SSE2
```

Это означает, что драйвер NVIDIA активен. Если при этом не работает драйвер, ускоряющий работу 3D (а только версия программы Mesa), вывод выглядит, как в двух следующих примерах:

```
root#  glxinfo | grep render
direct rendering: No
OpenGL renderer string: Mesa GLX Indirect
root#  glxinfo | grep render
direct rendering: Yes
OpenGL renderer string: Software Rasterizer
```

В дистрибутивах SUSE можно протестировать статус 3D и командой `3Ddiag`:

```
root# 3Ddiag
3Ddiag version 0.742
No 3D capable graphic chipset found!
Checking GL/GLU/glut runtime configuration:
  GL/GLU ... done (package Mesa)
  glut ... done (package freeglut)
```

Google Earth. Для испытания 3D-функций хорошо подходит Google Earth (рис. 12.4) Эту программу можно бесплатно скачать со следующего сайта: http://earth.google.de/download-earth.html.

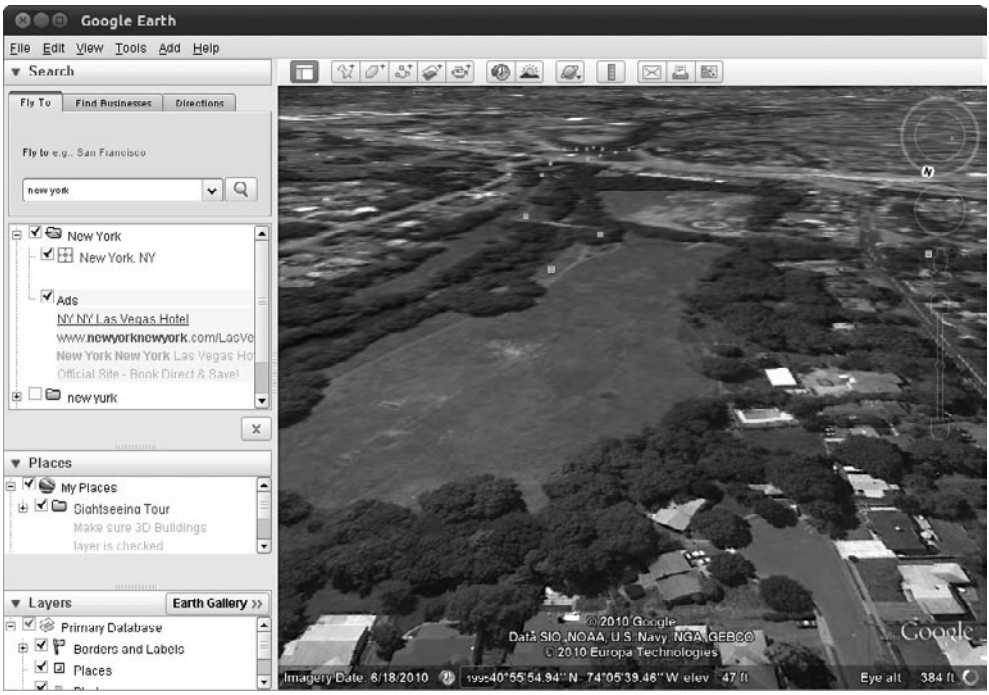

Рис. 12.4. Программа Google Earth

Для установки выполните программу `setup`. Права администратора для этого не требуются. Программа устанавливается в каталог `~/google-earth`.

```
user$ sh GoogleEarthLinux.bin  (Установка)
user$ google-earth/googleearth (Запуск программы)
```

Google Earth доступна только в 32-битном варианте. Если вы работаете с 64-битным дистрибутивом, вам потребуется установить некоторые 32-битные библиотеки (в Ubuntu — `ia32-libs` и `lib32nss-mdns`).

Трехмерный рабочий стол

Впервые функции с применением трехмерной графики, предоставляемые современными графическими картами, стали использоваться в программах для рабочего стола компании Apple: при этом можно гораздо эффективнее осуществлять простейшие операции, например перетаскивание окон. В мире Linux летом 2006 года всех опередила Novell, оснастившая дистрибутив *SUSE Linux Enterprise Desktop*

3D-функциями. Со временем трехмерный рабочий стол стал обычным явлением практически во всех дистрибутивах Linux.

Эффекты

В зависимости от конфигурации вы можете пользоваться следующими трехмерными эффектами.

- С помощью различных свойств (например, прозрачности и т. д.) анимировать такие эффекты, как появление, уменьшение, увеличение и закрытие окон.
- При перемещении окна временно изменять его форму, как будто оно резиновое (рис. 12.5).
- При переходе между окнами (Alt+Tab) распределять миниатюрные копии окон по рабочему столу (подобно функции Expose Mac OS X) или располагать веером (рис. 12.6).
- При переходе между различными рабочими поверхностями располагать их как грани вращающегося кубика или проецировать на цилиндр.
- Функция Масштаб позволяет сильно увеличить часть рабочего стола. Когда указатель мыши выходит за пределы видимого фрагмента, система сразу отображает другой фрагмент. Эта функция не только облегчает работу людям со слабым зрением, но и помогает просматривать некачественные сайты на большом мониторе с таким разрешением, при котором сайт выглядит красиво.

Разумеется, практическая ценность этих функций может показаться сомнительной. Через пару дней работы я отключил большую часть таких эффектов, ограничившись минимумом 3D-графики.

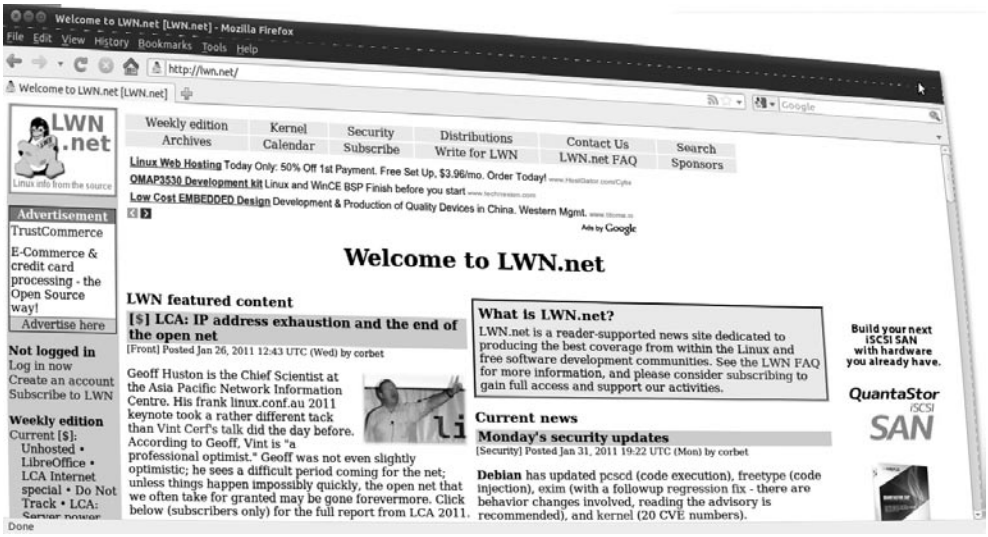

Рис. 12.5. Перемещение окна на трехмерном рабочем столе

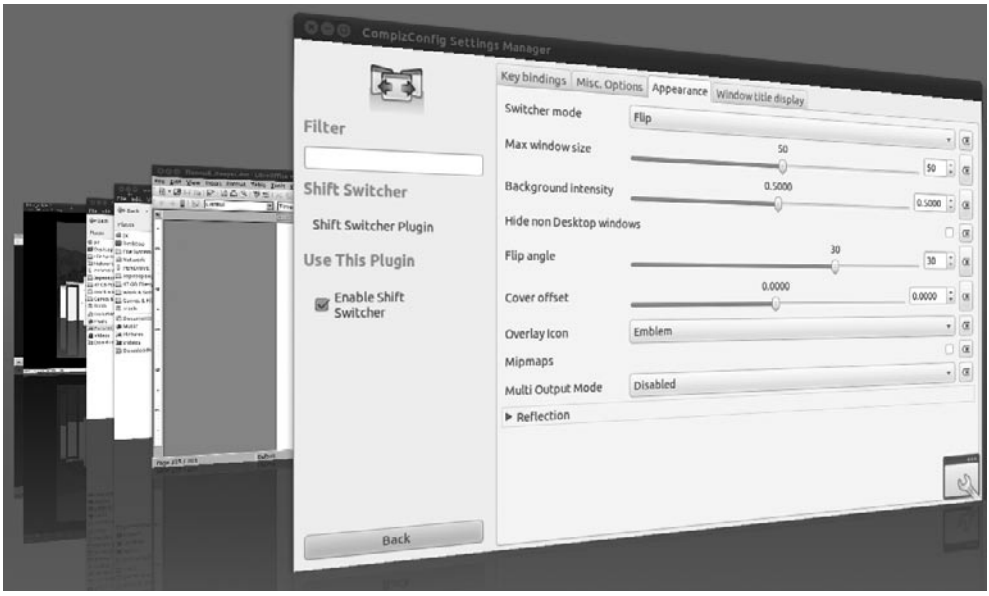

Рис. 12.6. Переход между окнами с помощью переключателя окон

Compiz

За реализацию 3D-эффектов отвечает диспетчер окон. В Gnome 2.n для этого применяется программа Compiz. В Gnome 3.0, предположительно, будет использоваться диспетчер окон «мама», способный работать с 3D-эффектами, таким образом, необходимость в Compiz отпадет (но сама возможность работы с программой, по-видимому, сохранится). KDE уже обходится без Compiz и сам реализует 3D-эффекты.

Итак, хотя значение Compiz в ближайшие годы должно снизиться, эта программа все еще является «мерой всех вещей» в мире трехмерных эффектов. Она стала родоначальником трехмерных эффектов рабочего стола в Linux и в настоящее время по умолчанию устанавливается почти во всех дистрибутивах.

По существу, Compiz состоит из двух программ.

- Сам `compiz` — это диспетчер окон. Он отвечает за то, какое окно сейчас отображается, какое окно находится в фокусе и готово для ввода информации, какие эффекты применяются при появлении/перемещении/закрытии окна и какие сочетания клавиш при этом действуют. За реализацию отдельных 3D-эффектов отвечают плагины.
- `Compiz-decorator` — это программа, «декорирующая» само содержимое окна, в том числе область заголовка и находящиеся на ней кнопки.

Более подробная информация по Compiz и ее плагинам находится на следующем сайте: http://www.compiz.org/.

Конфигурация

Во многих дистрибутивах как минимум основные 3D-эффекты активизируются автоматически, если действующий графический драйвер поддерживает такие

эффекты. В зависимости от дистрибутива и рабочего стола для активизации и конфигурации таких эффектов также предусмотрены следующие инструменты:

- Fedora (Gnome) — Система ▸ Параметры ▸ Эффекты рабочего стола;
- KDE 4 — модуль Оформление ▸ Канва ▸ Эффекты канвы в системных настройках;
- SUSE (Gnome) — модуль Эффекты канвы центра управления (`simple-ccsm`);
- Ubuntu — Система ▸ Параметры ▸ Оформление ▸ Визуальные эффекты.

Пользователи Compiz, у которых есть время и желание, могут, кроме того, присвоить каждому отдельному эффекту сочетание клавиш, запустив программу `ccsm` (`compizconfigsettings-manager`, рис. 12.7). Правда, работа с этой программой построена не очень логично. Если вы только начинаете работать с Compiz, я рекомендую пользоваться гораздо более понятной программой `simple-ccsm`.

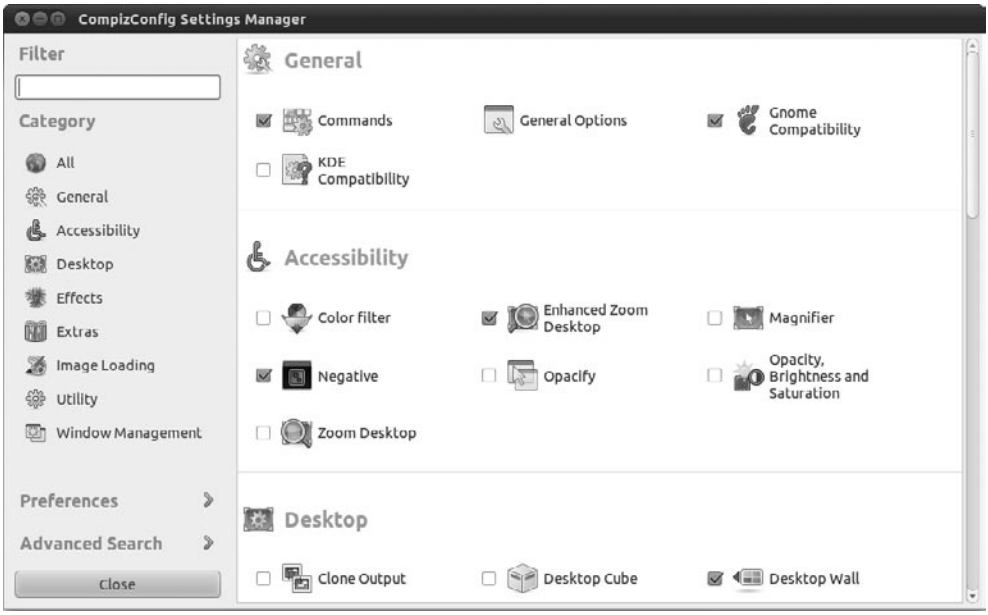

Рис. 12.7. Конфигурация Compiz — продвинутый уровень

XVideo

XVideo (V4L). У вас есть возможность добавить в систему программу XVideo (кратко — XV) с применением *разделяемой памяти (shared memory)*, выделяемой из общей памяти графической карты. Чтобы задействовать этот метод, вам потребуется модуль ядра v412 (Video4Linux версия 2). Этот метод очень эффективен, но работает только тогда, когда драйвер графической карты поддерживает XV. К сожалению, это правило выполняется не для всех графических карт.

Если ваш драйвер X поддерживает XV, это расширение X обычно активизируется автоматически. При необходимости можете добавить в раздел `Module` строку `Load "v4l"`. Чтобы проверить, есть ли в системе XV, выполните команду `xvinfo`.

Должен появиться длинный список данных о различных функциях XVideo. Если же в результате компьютер сообщит no adapters present (нет указанных адаптеров), это означает, что вы не сможете пользоваться XV.

Более подробная информация по XV и Video4Linux содержится на следующих сайтах:

- http://linuxtv.org/v4lwiki/;
- http://www.exploits.org/v4l/.

DGA. Если драйвер вашей графической карты не поддерживает XV, возможно, заработает DGA (прямой доступ для графики). DGA, как и XV, активизируется автоматически, если драйвер «знает» эту функцию. Можете проверить DGA программой `dga`. Нажатие клавиши B выполняет оценку производительности, Q завершает работу программы.

12.9. X в сети

Особенность X заключается в том, что весь протокол приспособлен для работы в сети. Иначе говоря, вы можете через сеть зайти в систему с любого компьютера и запустить графическую программу. Работа этой программы будет отображаться на локальном компьютере и может с него обслуживаться, хотя на самом деле программа выполняется на удаленном компьютере. В этом разделе будут показаны различные способы использования такого механизма.

Команда ssh

Проще и надежнее всего работать через сеть с помощью команды `ssh`, описанной в разделе 6.2. Для этого необходимо выполнить три условия: на удаленном компьютере должен работать SSH-сервер, SSH-порт 22 не должен блокироваться брандмауэром и при вызове `ssh` нужно применять параметр `-X`, чтобы программа правильно настроила переменные `DISPLAY`. С помощью команды `ssh -X root@localhost` можно выполнять программу X и на локальном компьютере, если у вас есть права администратора.

VNC

VNC означает «система для управления удаленным компьютером через сеть» (Virtual Network Computing) и позволяет другому пользователю увидеть на экране своего компьютера ваш рабочий стол и работать с ним. Таким образом, ваш коллега может через сеть взять на себя управление пользовательским интерфейсом вашего компьютера. Эта функция особенно полезна в тех случаях, когда у пользователя возникает проблема с компьютером и другой пользователь (находящийся не рядом) хочет помочь. VNC применяется не только в X, то есть в Linux, но и в большинстве других операционных систем (в том числе Microsoft Windows).

Существует множество вариантов VNC. В Linux чаще всего используются RealVNC и TightVNC. Общим знаменателем для многочисленных программ VNC

является протокол *RFB (удаленный кадровый буфер)*, который обеспечивает совместимость как минимум между основными функциями большинства программ VNC. По протоколу RFB передается ввод с мыши и клавиатуры, а также изменения информации на экране.

VNC — это клиентско-серверный протокол. Чтобы система VNC работала, на одном компьютере должен работать VNC-сервер. На втором компьютере запускается VNC-клиент (например, vncviewer, vinagre или krdc), устанавливающий соединение с сервером. Тогда в окне VNC-клиента отображается рабочий стол сервера.

По умолчанию передача данных между клиентом и сервером осуществляется по протоколу TCP/IP (порты 5900–5906). VNC-клиенты, написанные на Java и выполняемые в браузере, используют, как правило, порты 5800–5806. Эти порты не должны блокироваться брандмауэром! Обратите внимание — удаленное обслуживание возможно лишь при условии, что оба компьютера находятся в одной и той же локальной сети либо имеют внешние IP-адреса. Если же один или оба компьютера находятся в разных частных сетях, образуемых одним ADSL-маршрутизатором и многими WLAN-маршрутизаторами, то вам не повезло.

При запуске VNC-клиента вы сообщаете имя сети или адрес того компьютера, на котором работает VNC-сервер. Кроме того, нужно указать либо номер дисплея (`:n`), либо номер порта (`::nnnn`).

```
user$ vncviewer 192.168.0.17:0     (Отобразить экран X 0)
user$ vncviewer 192.168.0.17::5901 (Использовать порт 5901)
```

VNC с шифрованием SSH. Сам по себе протокол VNC ненадежен, так как данные передаются по нему в незашифрованном виде. Если при передаче данных для вас особенно важна безопасность, передайте поток данных через зашифрованный туннель либо используйте варианты VNC со встроенными функциями безопасности. Как направить поток данных протокола VNC через туннель SSH, описано на сайте http://linuxwiki.de/VNC.

VNC в Gnome и KDE. И в Gnome, и в KDE есть удобные пользовательские интерфейсы для подключения к VNC, которые, однако, в некоторых дистрибутивах необходимо устанавливать дополнительно. В Gnome, если вам необходима помощь, вы можете запустить удаленное управление, выполнив команду Система ▸ Параметры ▸ Удаленный рабочий стол или программу `vino-preferences`. Функции VNC-сервера реализуются с помощью библиотеки `vino-server` — по умолчанию для этого используется порт 5900. Коллега, оказывающий вам помощь, может воспользоваться любым VNC-клиентом, например vinagre.

В KDE VNC-соединение запускается с помощью меню Программы ▸ Система ▸ Общий рабочий стол или программой `krfb`. Коллега, оказывающий вам помощь, может воспользоваться любым VNC-клиентом или программой KDE `krdc`.

NX

Это технология, применяемая для ускорения и повышения надежности работы VNC и для реализации других технологий удаленного управления рабочим столом. NX позволяет использовать VNC с приемлемой скоростью даже при самом медленном

соединении. NX работает как прокси между обоими компьютерами, использующими систему X (клиентом и сервером). Программа NX архивирует данные, предназначенные для передачи, и использует кэш, чтобы избежать многократной передачи одних и тех же данных. Для шифрования информации NX применяет SSH.

Программа NX была разработана итальянской фирмой NoMachine. Основные библиотеки предоставляются по лицензии GPL. Клиент NX есть в свободном доступе, но его код закрыт. Только NX-сервер является коммерческой программой и бизнес-идеей компании NoMachine.

Если вы хотите настроить на основе NX бесплатный терминальный сервер, то для коммерческого NX-сервера есть две альтернативы с открытым кодом: FreeNX-Server, проверенный на практике сервер (правда, в последнее время его техническая поддержка прекратилась), и новый сервер Neatx, разработанный Google. Более подробная информация есть на следующих сайтах:

- http://www.nomachine.com/;
- http://freenx.berlios.de/;
- http://code.google.com/p/neatx/.

Nomad. Проект Nomad, реализуемый openSUSE или Novell, преследует те же цели, что и NX, но с поддержкой 3D-эффектов: http://en.opensuse.org/Nomad.

12.10. Шрифты (гарнитуры)

В системе X различаются масштабируемые и немасштабируемые шрифты.

- Еще несколько лет назад в X можно было работать только с немасштабируемыми шрифтами. Такие шрифты могут иметь только определенные, заранее заданные размеры. Они, правда, могут отображаться и с увеличением, и с уменьшением, но при этом будут выглядеть «мозаичными». Сегодня в повседневной практике немасштабируемые шрифты практически не применяются и присутствуют лишь в немногих программах (например, в `xterm`).
- Теперь система X может работать и с современными, свободно масштабируемыми шрифтами (TrueType, Type-1, OpenType). В приложениях KDE и Gnome используются только такие шрифты.

Исторически сложившееся различие между масштабируемыми и немасштабируемыми шрифтами также отражается в сосуществовании многих аналогичных команд управления. Например, вы можете получить список всех масштабируемых шрифтов командой `fc-list`, а команда `xlsfonts` выдает список всех немасштабируемых шрифтов.

Каталоги шрифтов. Обычно общедоступные файлы шрифтов располагаются в подкаталогах `/etc/X11/fonts` или `/usr/share/fonts`. Что касается немасштабируемых шрифтов, то для каждого размера и атрибута (например, полужирный, курсив) предусмотрен отдельный файл. Для масштабируемых шрифтов хватает одного файла для всех размеров.

Кроме того, вы можете устанавливать собственные шрифты в каталоге `~/.fonts`. Для этого нужно просто скопировать туда файлы шрифтов.

Библиотека fontconfig. За управление масштабируемыми шрифтами отвечает библиотека `fontconfig`. За интеграцию X и `fontconfig` отвечает библиотека Xft. Данные о конфигурации системы `fontconfig` содержатся в файле `/etc/fonts/fonts.conf`.

xorg.conf. Если шрифты были установлены в какие-либо каталоги, кроме стандартных, известных серверу X, то эти каталоги необходимо указать в разделе `Files` в файле `xorg.conf`.

```
# в /etc/X11/xorg.conf
Section "Files"
  FontPath "/usr/share/fonts/myown"
  ...
EndSection
```

Чтобы изменения конфигурации шрифтов вступили в силу, достаточно выполнить следующую команду. Если это не помогает, попробуйте заново войти в систему или повторно запустить X.

```
root# xset fp rehash
```

fc-list и xlsfonts. Команда `fc-list sort` возвращает список всех масштабируемых шрифтов. Те же функции для немасштабируемых шрифтов выполняет команда `xlsfonts`. Результаты выполнения второй команды длинные и непонятные, так как в списке содержатся отдельные записи по каждому размеру шрифта и по каждой кодировке. Программа `xfontsel` позволяет целенаправленно искать немасштабируемые шрифты, отвечающие критериям поиска.

gucharmap и xfd. Программа Gnome `gucharmap` отображает все символы масштабируемого шрифта и позволяет копировать в буфер обмена отдельные специальные символы. Программа оптимизирована под представление символов Unicode.

Чтобы просмотреть символы немасштабируемых шрифтов, лучше всего воспользоваться командой `xfd -fn 'имя_шрифта'`. На месте *`'имя_шрифта'`* должен быть указан точный синтаксис шрифта X. Лучше всего в качестве параметра применить строку из списка результатов `xlsfonts`.

Кодировки, поддержка Unicode. В принципе X может разобраться с любыми мыслимыми кодировками, в том числе с Unicode. Правда, для разных шрифтов могут поддерживаться различные кодировки. Способ поддержки кодировок зависит от вида шрифта.

- У масштабируемых шрифтов информация о кодировке интегрирована в файл шрифта. За соотнесение отдельных символов с кодами отвечают файлы кодировки, управляемые независимо от конкретного шрифта и находящиеся, как правило, в одном из следующих каталогов:
  - `/usr/share/fonts/encodings`;
  - `/usr/share/fonts/X11/encodings`.
- У немасштабируемых шрифтов для каждой поддерживаемой кодировки предусмотрен собственный файл шрифта. В Unicode существует относительно немного немасштабируемых шрифтов. Следующая команда выводит список всех немасштабируемых шрифтов, доступных на компьютере:

  ```
  user$ xlsfonts '*iso10646-1*'
  ```

Если известно, что имеется шрифт в кодировке Unicode, еще нельзя сказать, сколько символов в нем содержится. В настоящее время, по-видимому, не существует полных шрифтов Unicode, которые могли бы отображать все символы стандарта Unicode. В любом случае такие шрифты были бы очень объемными и, вероятно, не подходили бы для повседневного использования.

Установка дополнительных шрифтов

В дистрибутивах Linux при поставке могут содержаться только «общедоступные» шрифты. В этом разделе даются некоторые советы по установке собственных шрифтов. В принципе для этого необходимо выполнить два шага.

1. Файлы шрифтов необходимо скопировать в предназначенный для них каталог. Такие каталоги перечислены в `xorg.conf` либо даются в `/var/log/Xorg.0.log`. Кроме того, шрифты можно устанавливать в каталог `~/.fonts`.
2. Нужно обновить внутрисистемное управление шрифтами. Для этого в каталоге Font выполняется команда `fc-cache`. Она создает файлы `fonts.cache`, необходимые для системы `fontconfig` и библиотеки `Xft`.

KDE. В модуле Управление системой ▸ Установка шрифта программа KDE отображает все доступные шрифты. С помощью этого модуля вы также можете установить собственные шрифты в каталоге `~/.fonts`.

Шрифты Microsoft для Интернета. В течение достаточно долгого времени Microsoft предоставляла для загрузки множество высококачественных шрифтов TrueType. В этот набор входили шрифты Andale Mono, Arial, Comic Sans, Courier New, Georgia, Impact, Times New Roman, а также шрифты для Интернета Trebuchet MS, Verdana и Webdings. Они должны были обеспечить всем пользователям возможность просматривать в наилучшем качестве те сайты, при создании которых использовались эти шрифты.

Надо отметить, что тот сайт, с которого первоначально можно было скачивать шрифты, уже не работает, но те же шрифты можно скачивать с сайта corefonts, указанного ниже. Их можно бесплатно использовать, однако их коммерческое распространение запрещено, поэтому они не поставляются вместе с коммерческими дистрибутивами.

К сожалению, устанавливать шрифты в Linux неудобно, так как они упакованы в EXE-файлах и не передаются (и не могут передаваться) в ином виде. К счастью, есть инструменты для извлечения шрифтов из EXE-файлов. Подробное руководство по установке шрифтов в дистрибутивах, использующих RPM-пакеты, находится здесь: http://corefonts.sourceforge.net/.

В дистрибутивах есть сценарии, облегчающие скачивание и установку шрифтов.

- Debian, Ubuntu — в пакете `msttcorefonts` содержится сценарий `update-ms-fonts`. Он выполняется автоматически, скачивает файлы шрифтов и устанавливает их в каталог `/usr/share/fonts/truetype/msttcorefonts`.
- SUSE — в пакете `fetchmsttfonts` содержится сценарий для загрузки шрифтов. Он автоматически выполняется при установке пакета. Кроме того, файлы шрифтов находятся в каталоге `/usr/share/fonts/truetype`.

Сглаживание

По умолчанию в X используется сглаживание (anti-aliasing, коротко — AA), или хинтование (hinting), — техника, позволяющая отображать шрифты TrueType и Type-1 с минимальным количеством неровностей. При этом края букв представляются в полутонах серого. В таком случае отдельные буквы кажутся менее мозаичными и более отчетливыми.

На многих плоских экранах каждый пиксел состоит из трех расположенных рядом подпикселов — красного, зеленого и синего. На таких экранах с помощью варьирования цветов отдельных пикселов можно достичь еще лучшего качества изображения, чем при обычном сглаживании. Этот метод отображения называется *попиксельный рендеринг* (Sub-Pixel-Rendering); в Microsoft он именуется *ClearType*. Предлагаю вам две статьи по этой теме, которые будут как минимум любопытны, а возможно, и очень полезны для разработчиков, и находятся по следующим адресам:

- http://antigrain.com/research/font_rasterization/;
- http://behdad.org/text/.

Конфигурация. Функции сглаживания и попиксельного рендеринга управляются XML-файлами `/etc/fonts/fonts.conf` и `/etc/fonts/conf.d/*.conf`. В KDE и Gnome имеются удобные диалоги для настройки пользовательской конфигурации сглаживания и попиксельного рендеринга. В KDE для этого применяется модуль настройки системы Оформление ▸ Шрифты. В Gnome необходимо выполнить команду Система ▸ Параметры ▸ Оформление ▸ Шрифты или запустить программу `gnome-appearance-properties` (рис. 12.8). Настройки сохраняются в базе данных `gconf` (путь: `desktop/gnome/font_rendering`) и активизируются при запуске Gnome.

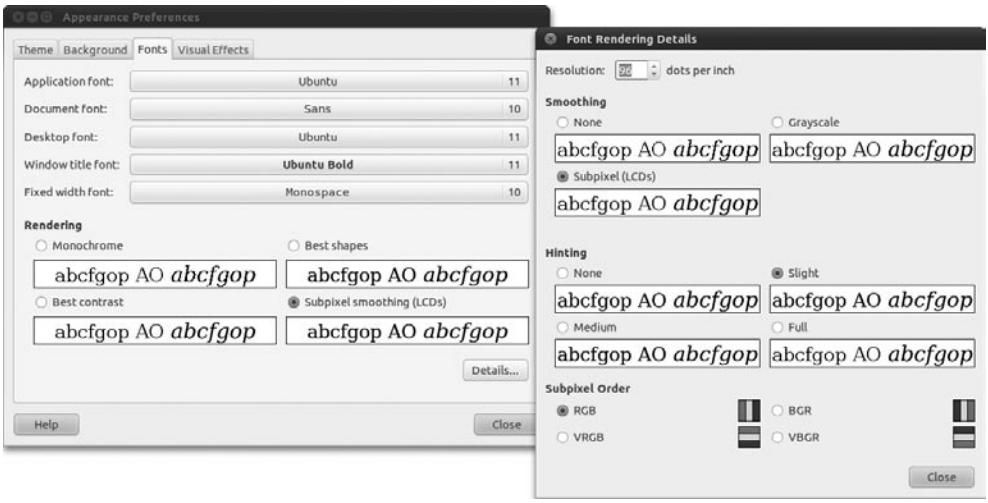

Рис. 12.8. Конфигурация сглаживания в Gnome

Настройка DPI

DPI означает dots per inch («точек на дюйм»). В DPI указывается разрешение экрана. Сразу приведу пример: если вы работаете с монитором, имеющим ширину 36 см (19-дюймовый монитор), а разрешение по горизонтали составляет 1280 пикселов, то на дюйме экрана размещается примерно 91 пиксел (1280/36 * 2,54).

Какую роль играет значение DPI? Неважно, какой размер экрана и разрешение графики, — чтобы было удобно читать текст на экране, размер шрифта должен варьироваться в зависимости от показателя DPI. Если система X Window и работающие в ней программы «знают» правильное значение DPI и способны его интерпретировать, шрифт никогда не будет слишком маленьким и нечитаемым или слишком крупным и неаккуратным.

Чтобы система X могла рассчитать правильное значение DPI, она должна знать размер монитора. Современные мониторы сообщают эту информацию X через DDC. Если этот механизм не работает, следует задать эту величину (в миллиметрах) с помощью ключевого слова `DisplaySize` в разделе `Monitor`:

```
Section "Monitor"
  ...
  DisplaySize 400 300
EndSection
```

В качестве альтернативы можно указать в диалогах KDE или Gnome, предназначенных для конфигурации шрифта, любое значение DPI. Кроме того, многие программы позволяют отдельно настраивать размер шрифта.

Для того чтобы узнать текущее значение DPI, выполните команду `xdpyinfo`. Далее приводится результат для 20-дюймового монитора:

```
root# xdpyinfo | grep -C 1 dimensions
screen #0:
  dimensions:   1600x1200 pixels (411x311 millimeters)
  resolution:   99x98 dots per inch
```

13 Администрирование файловой системы

В этой главе описаны различные стороны администрирования файловой системы. Глава рассчитана на опытных пользователей Linux и рассматривает такие вопросы.

- **Как взаимосвязаны компоненты файловой системы** — в этом разделе дается обзорное описание того, как связаны друг с другом различные фрагменты файловой системы Linux.
- **Имена устройств** — внутри системы Linux жесткие диски и другие носители данных, а также их сегменты запрашиваются через файлы устройств, например, `/dev/hda` или `/dev/sdc3`. В этом разделе рассмотрена соответствующая схема нумерации и номенклатуры.
- **Сегментирование жесткого диска** — это основная часть установки Linux. Иногда для установки может потребоваться добавить новый сегмент диска.
- **Типы файловых систем** — сложно найти операционную систему, которая поддерживала бы так же много типов файловой системы, как Linux. В этом разделе обобщены важнейшие варианты использования файловых систем.
- **Управление файловой системой** — в данном разделе будет рассказано, как можно вручную добавить к файловой системе отдельные сегменты с данными и как автоматизировать этот процесс (`/etc/fstab`).
- **Файловые системы Windows и Linux** — в нескольких разделах, посвященных этой теме, даются советы и указания по работе с файловыми системами ext3, ext4, xfs, vfat и ntfs.
- **CD/DVD** — для CD и DVD с данными также существуют собственные типы файловых систем, которые кратко представлены в этом разделе.
- **Внешние носители (USB, FireWire)** — если подключить к компьютеру диск FireWire или USB-флешку, обычно автоматически выводится окно файлового менеджера, через которое вы получаете доступ к данным. В этом разделе объясняется, что происходит «за кулисами» и как можно (при необходимости) пользоваться внешними носителями данных вручную.
- **Разделы подкачки** — если в Linux не хватает оперативной памяти для выполнения программ, то части памяти откладываются в так называемых разделах подкачки.
- **Сетевые файловые системы** — вы можете интегрировать в свое дерево каталогов каталоги с других компьютеров, работающих в той же локальной сети. В этой книге будут специально рассмотрены файловые системы CIFS (Windows/Samba) и NFS (UNIX/Linux).

- **RAID** — с помощью RAID (избыточных массивов недорогих/независимых жестких дисков) можно соединять друг с другом разделы нескольких дисков, чтобы таким образом построить более надежную и/или быструю общую систему. В этом разделе кратко рассматриваются основы RAID и описывается, как создать систему RAID-0 (с распределением данных).
- **LVM** — программа управления логическими томами (кратко — LVM) обеспечивает более гибкое управление разделами дисков. Например, она позволяет объединить разделы нескольких жестких дисков в виртуальный диск, изменять размеры разделов на ходу и т. д.
- **Зашифрованные файловые системы** — если вы хотите предотвратить доступ посторонних лиц к вашим данным (например, при краже компьютера), нужно зашифровать файлы или файловые системы. Для этого в Linux применяются различные методы, причем самый популярный из них в настоящее время основан на модуле ядра dm_crypt.
- **SMART** — технология самодиагностики, анализа и отчета, которая позволяет собирать статистические данные при эксплуатации жестких дисков и, таким образом, заблаговременно распознавать потенциальные проблемы с надежностью системы и не допускать потери информации.

Разумеется, об администрировании файловой системы можно рассказать гораздо больше, но размеры книги ограничены. Однако необходимо дать несколько дополнительных ссылок.

- **Использование файловой системы** — команды для копирования файлов или создания резервных копий, базовая информация о правах доступа к файлам и т. д. описаны в этой книге в главе 3.
- **Дисковые квоты** — здесь речь идет о системе, определяющей, сколько места на диске могут занимать файлы конкретных пользователей. Если заданная граница превышается, то пользователь больше не может сохранять новые файлы. Хорошее введение делается здесь: http://www.tldp.org/HOWTO/Quota.html.
- **Кластерные файловые системы** — кластерные, или глобальные, файловые системы объединяют данные нескольких компьютеров в виртуальные системы данных. Так можно создавать гигантские ЗУ для данных и параллельно использовать их содержимое на нескольких компьютерах. Опять же в Linux предусмотрено несколько методов создания таких областей памяти, например OCFS (кластерная файловая система Oracle) или GFS (глобальная файловая система):
  - http://oss.oracle.com/projects/ocfs2/;
  - http://sources.redhat.com/cluster/gfs/.

13.1. Как взаимосвязаны компоненты файловой системы

Порой взаимосвязи, возникающие при построении файловой системы, невероятно запутаны. В этом разделе я попытаюсь кратко и понятно представить важнейшие

взаимосвязи. Ради наглядности ограничимся рассмотрением встроенных жестких дисков и обычных файловых систем Linux. CD/DVD-приводы, внешние носители данных, управление логическими томами и системы RAID пока оставим в стороне.

Жесткие диски. Встроенные жесткие диски соотносятся с файлами-устройствами системы Linux. В новых дистрибутивах для всех жестких дисков применяются файлы-устройства вида `/dev/sda`, `/dev/sdb` и т. д. В старых дистрибутивах `/dev/hda`, `/deb/hdb` и т. д. используются для дисков IDE, а `/dev/sda`, `/dev/sdb` — для дисков SATA и SCSI.

Разделы. Чтобы расположить на одном жестком диске несколько независимых друг от друга файловых систем, нужно разбить этот диск на *разделы*. Разделы диска также соотносятся с файлами-устройствами, например `/dev/sda1` соответствует первому разделу первого жесткого диска SATA. Номенклатура файлов-устройств для разделов диска будет подробно рассмотрена позже.

Системный (загрузочный) раздел. При запуске Linux система в первую очередь обращается к системному (корневому) разделу. Номер устройства, или универсальный уникальный идентификатор (UUID) файловой системы, содержащейся в этом разделе, заносится (как часть параметра ядра) в конфигурационный файл GRUB. Подробно о запуске Linux и о конфигурации GRUB или LILO рассказано в главе 14.

Другие разделы. Кроме системного раздела, который требуется обязательно, могут быть и другие разделы, учитываемые уже при запуске Linux. Эти файлы перечислены в файле `/etc/fstab`. Он опять же должен находиться в системном разделе. Данный файл интерпретируется при выполнении процесса Init (подробнее — в разделе 14.10).

Определение совместимости. При включении разделов диска в дерево каталогов автоматически проверяется совместимость файловых систем. Если компьютер аварийно завершил работу, например из-за перебоев с электричеством, то файловая система автоматически восстанавливается или выполняются другие страховочные мероприятия, призванные предотвратить дальнейшие ошибки совместимости и связанные с ними потери данных. Соответствующий тест совместимости также автоматически запускается по истечении определенного периода использования системы. Детали протекания этого процесса зависят от дистрибутива и конфигурации конкретной системы.

Дерево каталогов — вместо букв, обозначающих приводы. В Windows принято запрашивать отдельные файловые системы через буквы, присвоенные различным дисководам (`A:`, `C:`, `D:` и т. д.), а в Linux все файловые системы объединяются в одном дереве каталогов. Доступ к системному разделу осуществляется через корневой каталог `/`. Стартовые пункты других файловых систем могут варьироваться в зависимости от дистрибутива и конфигурации. Обычно используются подкаталоги `/mnt` или `/media`, например `/media/dvd` для DVD с данными.

Добавление файловых систем. Вы можете, не останавливая работу системы, подключать к дереву каталогов новые файловые системы либо отключать те или иные файловые системы. При подключении внешнего носителя данных (например, USB-флешки) эта операция обычно выполняется автоматически. Если такой «автоматизм» не работает либо этот механизм специально был отключен, то администратор

может вручную подсоединять и отключать файловые системы командами `mount` и `unmount` соответственно.

Единственной неизменной частью остается системный раздел: его нельзя отключить от файловой системы в ходе эксплуатации компьютера. Это можно сделать только при завершении работы компьютера.

Типы файловых систем. В Linux поддерживается очень много типов файловых систем. Системный раздел должен относиться к одному из следующих типов систем: ext3, ext4 или xfs. Для остальных разделов выбор еще шире. Можно применить, например, файловые системы Windows, UNIX или Apple.

13.2. Названия устройств для жестких дисков и других носителей данных

В настоящее время связь компьютера с приводами обычно осуществляется в соответствии со стандартами IDE, SATA и SCSI. В табл. 13.1 поясняется значение этих и некоторых других сокращений.

Таблица 13.1. Сокращения

Сокращение	Значение
ATA	«Подключение по передовой технологии» (интерфейс для подключения жестких дисков)
ATAPI	Пакетный интерфейс ATA (расширение ATA для CD- и DVD-приводов)
IDE	Встроенный интерфейс привода (альтернативное название PATA)
PATA	Параллельный интерфейс ATA (традиционный ATA-интерфейс с параллельной передачей данных)
SATA	Последовательный интерфейс ATA (новый ATA-интерфейс с последовательной передачей данных)
SCSI	Интерфейс малых компьютерных систем (альтернатива IDE/SATA)

Внутренние свойства ядра

Внутри Linux доступ к встроенным и внешним жестким дискам и их разделам, к приводам CD и DVD, а также к другим носителям данных обеспечивается через файлы-устройства. Это особые файлы, выполняющие роль «интерфейсов» между Linux и аппаратным обеспечением компьютера (см. также раздел 3.10).

Такие файлы-устройства нужны только для управления системой, то есть, например, для изменения сегментирования диска или для подключения определенного раздела к файловой системе. При обычной эксплуатации системы вы имеете доступ ко всей файловой системе — через каталоги. При этом символ / означает начало файловой системы. Разрешается подключать новые файловые системы в любой части дерева каталогов, например это может быть дополнительный раздел Linux под названием `/data`, раздел Windows под названием `/media/win`.

В ядре имеется два основных семейства драйверов — для жестких дисков и для других носителей данных.

- **IDE** — драйвер IDE в наше время еще обслуживает старые IDE-диски и приводы IDE-DVD/CD. Код IDE в ядре уже не нов, и, по-видимому, не поддерживается.
- **SCSI** — через систему SCSI управляются не только все SCSI-устройства, но и все приводы, подключенные к шинным системам SATA, USB или FireWire.
- С 2007 года большинство жестких дисков IDE запрашивается через драйвер SCSI. Чтобы обеспечить такое взаимодействие, модуль libata дополняет SCSI-систему ядра функциями PATA. Преимущества очевидны: теперь почти все носители данных обрабатываются одинаково, на основе одного и того же базового кода. Только некоторые старые или экзотические материнские платы либо чипсеты несовместимы с libata и по-прежнему требуют IDE-драйвер.

Названия устройств

Любой жесткий диск, управляемый модулем ядра SCSI (то есть в большинстве компьютеров *все* жесткие диски и flash-накопители называются по образцу `/dev/sdxy` — `/dev/sda`, `/dev/sdb` и т. д. по порядку). Что касается устройств SATA, в данном случае все устройства по очереди связываются каналами и в качестве названия получают букву. На современных материнских платах обычно предусмотрено минимум 6–8 каналов. Если, например, два жестких диска подсоединяются к SATA-каналам 1 и 3, эти устройства получают имена `/dev/sda` и `/dev/sdb`. Если позже третий жесткий диск подсоединяется к каналу 2, название второго диска изменяется с `/dev/sdb` на `/dev/sdc`.

Если применяется libata, то IDE-устройства также получают названия в порядке `/dev/sda`, `/dev/sdb` и т. д. Обратите внимание на важное отличие по сравнению с обычной номенклатурой IDE — если на одном компьютере и к первому, и ко второму каналу IDE подключено два *ведущих* (master) диска, но к первому IDE-каналу не подключено *ведомое* устройство (slave), то устройства будут иметь названия `/dev/hda` и `/dev/hdc`. При использовании ядра libata, напротив, те же диски получат названия `/dev/sda` и `/dev/sdb`!

Что касается устройств SCSI, то порядок зависит от ID-номеров этих устройств. Пропуски в последовательности номеров ID не учитываются. Иначе говоря, устройства SCSI получат названия от `/dev/sda` до `/dev/sdc`. Как и при работе с устройствами SATA, мы в таком случае можем позже изменить конфигурацию и, следовательно, названия устройств: если будет добавлено четвертое устройство с ID-номером 3, оно получит название `/dev/sdc`; устройство с номером 5 теперь будет запрашиваться как `/dev/sdd`. Если одновременно подключаются устройства с шинами различных систем, от BIOS и от используемых PCI-разъемов зависит, какая шинная система будет учитываться в первую очередь.

Внешние USB- и FireWire-устройства обрабатываются как устройства SCSI, причем вместо *x* используется первая свободная буква. При присвоении букв порядок, в котором были подключены устройства, имеет определяющее значение. CD- и DVD-приводы получают собственные названия устройств — в зависимости от дистрибутива это `/dev/scdn` или `/dev/srn` (табл. 13.2).

Таблица 13.2. Названия устройств

Устройство	Значение
/dev/sda	Первый жесткий диск
/dev/sdb	Второй жесткий диск
...	
/dev/scd0 или /dev/sr0	Первый CD/DVD-привод
/dev/scd1 или /dev/sr1	Второй CD/DVD-привод
...	

IDE-устройства

Если применяется традиционная IDE-система (например, при работе со старыми компьютерами или дистрибутивами), то жесткие диски IDE запрашиваются через файлы-устройства в форме `/dev/hda`, `/dev/hdb` и т. д. Порядок расположения IDE-устройств определяется внутренними кабельными соединениями: `/dev/hda` обозначает первое устройство на первом IDE-канале; `/dev/hdb` означает второе устройство на первом IDE-канале; `/dev/hdc` и `/dev/hdd` обозначают соответственно ведущее и ведомое устройства на втором IDE-канале. Вполне возможно, что два устройства получат названия `/dev/hda` и `/dev/hdc`, а `/dev/hdb` останется неиспользованным — именно в тех случаях, когда и к первому, и ко второму каналу подключены ведущие устройства. Приводы CD-ROM и DVD, подключенные к шине IDE, обрабатываются как жесткие диски.

Номера разделов

При нумерации разделов, независимо от того, о чем идет речь — об IDE или SCSI, — действует правило, в соответствии с которым цифры от 1 до 4 зарезервированы для основных или дополнительных разделов, а цифры от 5 и далее — для логических разделов в рамках дополнительных. Поэтому достаточно часто в нумерации возникают пробелы. Например, если на жестком диске есть основной, дополнительный и три логических раздела, они будут иметь номера 1, 2, 5, 6 и 7. В табл. 13.3 приведено несколько примеров.

Таблица 13.3. Пример нумерации разделов

Устройство	Значение
/dev/sda	Первый диск SCSI/SATA (или первый IDE-диск при использовании ядра libata)
/dev/sda1	Первый основной раздел этого диска
/dev/sdd3	Третий основной раздел четвертого диска SCSI/SATA
/dev/sdd5	Первый логический раздел четвертого диска SCSI/SATA
/dev/sdd15	Одиннадцатый логический раздел четвертого диска SCSI/SATA

Количество разделов, которые могут быть на диске, ограничено: во-первых, исторически сложилось, что на диске может быть максимум четыре основных либо три основных и один дополнительный раздел. С другой стороны, в Linux количество используемых логических разделов ограничено 59 (традиционная IDE-си-

стема) или 11 (SCSI/SATA/USB/Firewire/IDE с libata). В итоге имеем максимальное общее количество 63 или 15.

Альтернативные названия устройств

Как уже было сказано, при постепенном добавлении в систему новых жестких дисков SCSI или SATA названия устройств, подключенных ранее, могут изменяться. При замене традиционной системы IDE ядром libata названия устройств также изменяются. Присвоение названий внешним устройствам вообще сложно спрогнозировать. Эти названия зависят от того порядка, в котором устройства подключались.

Для того чтобы можно было обеспечить стандартный доступ к отдельным устройствам или разделам (например, для сценария резервного копирования), несмотря на такую вариативность названий устройств в каталоге `/dev/disk` содержатся дополнительные ссылки на все носители данных, упорядоченные в соответствии с различными критериями.

- `/dev/disk/by-id/id` использует ID, состоящие из обозначения шинной системы, названия устройства и номера серии или модели.
- `/dev/disk/by-label/label` применяет название, данное файловой системе.
- `/dev/disk/by-path/path` использует название пути, состоящее из названия интерфейса PCI, шинной системы и номера раздела. Осторожно: если в следующий раз вы подключите USB- или FireWire-устройство к другому USB-разъему, путь изменится!
- `/dev/disk/by-uuid/uuid` использует номера UUID файловой системы. UUID — это уникальный ID-номер, присваиваемый файловой системе при форматировании. Он позволяет безошибочно идентифицировать файловые системы, в том числе после изменения конфигурации оборудования.

Количество ссылок в каталогах `/dev/disk` бывает разным. Каталоги `/dev/disk/by-label` и `by-uuid`, например, содержат ссылки только на те разделы, которые имеют названия или UUID. За автоматическое создание ссылок отвечает система udev (см. раздел 3.10). Например, следующая команда `ls` демонстрирует ссылки тестовой системы с жестким диском SATA, USB-флешкой и CF-картой (также подключаемой через USB). Чтобы вывод было удобнее читать, я добавил перед строками небольшие отступы и удалил информацию, касающуюся прав доступа.

```
user$ cd /dev/
user$ ls -lR disk/
disk/by-id:
  scsi-SATA_ST3320620AS_5QF194H9             -> ../../sda
  scsi-SATA_ST3320620AS_5QF194H9-part1       -> ../../sda1
  scsi-SATA_ST3320620AS_5QF194H9-part2       -> ../../sda2
  scsi-SATA_ST3320620AS_5QF194H9-part3       -> ../../sda3
  scsi-SATA_ST3320620AS_5QF194H9-part4       -> ../../sda4
  scsi-SATA_ST3320620AS_5QF194H9-part5       -> ../../sda5
  usb-Generic_USB_CF_Reader_058F312D81B      -> ../../sdc
  usb-Generic_USB_MS_Reader_058F312D81B      -> ../../sde
  ...
disk/by-path:
  pci-0000:00:1d.7-usb-0:5:1.0-scsi-0:0:0:1  -> ../../sdc
```

```
  pci-0000:00:1d.7-usb-0:5:1.0-scsi-0:0:0:3   -> ../../sde
  pci-0000:00:1f.2-scsi-0:0:0:0               -> ../../sda
  pci-0000:00:1f.2-scsi-0:0:0:0-part1         -> ../../sda1
  pci-0000:00:1f.2-scsi-0:0:0:0-part2         -> ../../sda2
  pci-0000:00:1f.2-scsi-0:0:0:0-part3         -> ../../sda3
  pci-0000:00:1f.2-scsi-0:0:0:0-part4         -> ../../sda4
  pci-0000:00:1f.2-scsi-0:0:0:0-part5         -> ../../sda5
disk/by-uuid:
  008f06ef-28be-45c9-acbc-20cda51f712b        -> ../../sda2
  06efc09c-9a3e-4668-81d7-8925c380889e        -> ../../sda5
  366CA8D16CA88D65                            -> ../../sda1
  e35139a5-5871-48fe-9191-df0d003e4ed5        -> ../../sda3
```

13.3. Секционирование жесткого диска

При установке почти во всех дистрибутивах Linux предоставляются простые в обслуживании инструменты для секционирования жесткого диска. Но лишь в немногих дистрибутивах эти инструменты можно использовать не только при установке, но и при текущей работе — например, в SUSE таким инструментом является модуль YaST Система ▸ Разметка диска. Кроме того, если вы хотите внести изменения в схему разделов после установки, вам на выбор предлагается целый спектр инструментов для секционирования: к важнейшим из них относятся текстовые команды `fdisk` и `parted`, а также графическая программа `gparted`.

О том, как называются и нумеруются разделы диска в Linux, рассказано в разделе 13.2.

ВНИМАНИЕ

Программы для секционирования могут уничтожить всю информацию на жестком диске! Прежде чем применять их, полностью прочитайте этот раздел! Никогда не изменяйте раздел диска, используемый в данный момент, то есть раздел, файловая система которого подключена к дереву каталогов — mounted.

Для запуска Windows Vista или более новых версий Windows требуется, чтобы раздел начинался с сектора 2048, таким образом, первый мегабайт жесткого диска должен быть свободен для использования основной загрузочной записи (MBR) и загрузчика операционной системы (раньше стартовым пунктом и для Windows, и для Linux был сектор 64).

Если вы собираетесь часто вносить изменения в схему разделов диска, вам просто необходимо научиться работать с LVM (см. раздел 13.15): LVM добавляет виртуальный слой между физическими разделами жесткого диска и разделами, используемыми для размещения файловых систем, тем самым необычайно упрощая постепенное внесение изменений.

Основные правила

Независимо от того, какой инструмент вы применяете, необходимо учитывать некоторые основные правила.

- Во время работы нельзя вносить изменения в системный раздел. Если вы, к примеру, хотите увеличить его, лучше всего запустить компьютер с помощью «жи-

вого диска». Для этой цели особенно хороши мини-дистрибутивы, оптимизированные под секционирование жестких дисков, например GParted-Live-CD, Parted Magic и SystemRescueCd:

- http://gparted.sourceforge.net/livecd.php;
- http://partedmagic.com/;
- http://www.sysresccd.org/.

- В принципе можно изменить размер только последнего раздела жесткого диска. Разделы нельзя «менять местами».
- Если вы изменяете размер последнего раздела на диске, то размер содержащейся в нем файловой системы не изменяется автоматически! Для этого нужны дополнительные команды, которые отличаются в зависимости от типа файловой системы.
- На одном жестком диске может быть не более 15 разделов. Один из них является расширенным и может принимать в себя другие разделы, но не данные или файловую систему напрямую. Таким образом, максимальное количество разделов уменьшается до 3 основных и 11 логических.
- Если вы изменяете схему разделов жесткого диска, который в данный момент используется (например, потому, что один из разделов этого диска — системный раздел работающей Linux), то программа секционирования дополнительно попросит вас перезапустить компьютер.

  Причина ошибки в том, что ядро Linux не может заново считать таблицу разделов диска при работе системы. Таким образом, изменения сохраняются, но вступают в силу только после перезапуска. Вам *придется* перезапустить Linux, чтобы работать с новой схемой сегментов диска.

Сектора, дорожки, цилиндры и блоки

Исторически сложилось так, что для измерения жестких дисков применяются единицы со следующими названиями.

- **Сектор** (512 байт) — мельчайшей единицей жесткого диска является сектор, состоящий из 512 байт (предполагается, что в будущем эта единица может быть увеличена до 4096 байт, то есть до 4 Кбайт).
- **Дорожка** (32 256 байт) — на всех современных жестких дисках «дорожка» состоит из 63 секторов, то есть из 63 * 512 = 32 256 байт. Первоначально по количеству секторов можно было узнать, сколько их на одной окружности жесткого диска. Сегодня это число является условным. Максимально допустимое значение — 63, так как уже в течение многих лет на хранение количества секторов отводится всего 6 бит.
- **Цилиндр** (8 225 280 байт, около 7,8 Мбайт) — размер цилиндра равен произведению размера дорожки на количество читающих головок. При этом самым большим жестким дискам, используемым в Linux, присваивается условное количество головок, равное 255, таким образом, размер цилиндра оказывается равен 255 * 32 256 = 8 225 280 байт. Опять же, число 255 является максимально возможным, так как для поля предусмотрено всего 8 бит.

Чтобы узнать количество цилиндров, необходимо разделить мощность жесткого диска на размер цилиндра. Как правило, на жестком диске размером 1 Тбайт имеется 1 000 000 000 000 = 212 байт места, что составляет около 121 600 цилиндров. (Производители жестких дисков охотнее считают в десятичной системе, а не в двоичной, так как мощности кажутся «больше». На самом деле, если считать в двоичной системе, то в одном терабайте будет 240 = 1 099 511 627 776 байт, то есть примерно на 10 % «больше», чем в десятичном терабайте!)

Раньше эти термины отражали физическую архитектуру жесткого диска. Нынешняя архитектура уже давно не такая — даже у традиционных жестких дисков, не говоря уже о SSD. Так или иначе в некоторых программах для секционирования, в частности в `fdisk`, расчеты по-прежнему проводятся в этих единицах. Разумеется, при работе с Linux границы цилиндров не имеют никакого значения; раздел может начинаться прямо в центре цилиндра (даже условного)!

Что такое блок? Будьте внимательны, когда речь заходит о блоках. Обычно при этом имеется в виду размер блока, предусмотренный в ядре Linux, — 1024 байт (в частности, при работе с командами `df`, `du` и `fdisk`). В файловых системах для внутренней организации данных предусматриваются значительно более крупные блоки: обычно используется величина в 4096 байт = 4 Кбайта либо кратная ей величина.

Настройка производительности SSD. По причинам, связанным с производительностью, соответствие границ разделов и цилиндров не всегда оптимально, как минимум в тех случаях, когда условные цилиндры состоят из 63 * 255 секторов, как в Linux. Windows Vista и более новые версии системы Windows работают с цилиндрами, имеющими размер 63 * 240 секторов, то есть размер цилиндра кратен 4 Кбайтам. Если в будущем появятся жесткие диски с секторами размером 4 Кбайта, то при использовании такого расчета можно будет гарантировать, что каждый блок файловой системы будет соответствовать сектору жесткого диска. Для SSD, вероятно, было бы еще удобнее работать с единицами по 56 * 226 секторов, чтобы размер цилиндра был кратен 128 Кбайт.

В любом случае сомнительно, что код ядра будет откорректирован таким образом и достигнутое в результате увеличение производительности оправдает затраченные усилия. Базовая информация по таким расчетам, напоминающим современную магию чисел, дается на сайте http://tinyurl.com/c74an4=http://thunk.org/tytso/blog/2009/02/20/aligning-filesystems-to-an-ssds-erase-block-size/.

Корректировка размера расширенного раздела

Инструменты для секционирования, описанные в предыдущем разделе, а также вспомогательные инструменты, применяемые при установке Linux, отличаются в одном принципиальном отношении. Некоторые инструменты оставляют раздел таким же по размеру, каким он был при создании. Эти программы мы отнесем к типу 1. Таковы, например, `fdisk`, `parted`, программы установки Fedora, Red Hat и SUSE, а также программы секционирования для Windows.

Другие программы, напротив, автоматически корректируют размер увеличенного раздела таким образом, чтобы все логические разделы могли разместиться именно в этом физическом разделе (тип 2). Таковы программы установки Debian, Mandriva и Ubuntu.

Сами по себе хороши оба метода, сложности возникают только при их одновременном применении. Например, с помощью инструмента второго типа вы создаете новый логический раздел. При этом программа уменьшает размер расширенного раздела. Если вы теперь попытаетесь создать новый логический раздел с помощью инструмента типа 1, то получите сообщение, что расширенный раздел уже заполнен. Ни одна программа первого типа, за исключением `parted`, не может изменять размеры расширенных разделов, если в них уже находятся логические разделы.

Что делать: еще раз запустите программу типа 2 и измените существующее разбиение на разделы. Кроме того, с помощью инструмента секционирования второго типа можно создать достаточно большой логический раздел-заполнитель. Таким образом, расширенный раздел автоматически увеличится. Затем вы удалите раздел-заполнитель программой секционирования первого типа и сможете воспользоваться свободным местом в расширенном разделе.

Программа fdisk

Это одна из самых старых программ Linux, предназначенных для секционирования. Ее пользовательский интерфейс несколько старомоден, однако программа отличается высоким качеством и по достоинству оценена теми, кто не первый год работает с Linux.

Запуск

Программа `fdisk` всегда может обрабатывать только один жесткий диск — тот, чье название устройства было задано при запуске программы (например, `/dev/sdc` для третьего жесткого диска SATA/SCSI). Если вместо этого сообщить программе параметр `-l`, то `fdisk` выдаст список всех разделов всех жестких дисков. После запуска нажатие клавиши `M` *(menu)* дает краткий обзор команд, которыми можно воспользоваться. Нажатие `P` *(print)* выдает список разделов, которые в настоящий момент имеются на выбранном жестком диске.

Если жесткий диск состоит более чем из 1024 цилиндров (а таковы все современные жесткие диски), то `fdisk` выдает предупреждение. Оно касается только загрузочного раздела Linux и имеет значение лишь в том случае, если вы пользуетесь древнейшей материнской платой (дата BIOS до 1998 года) или применяете версию LILO из каменного века. Иными словами, такое предупреждение можно игнорировать практически в 100 % случаев.

Если вы хотите, чтобы `fdisk` считала не в цилиндрах, а в секторах, откорректируйте единицу измерения нажатием клавиши `U`.

Создание нового раздела

С помощью клавиши `N` *(new)* создаются новые разделы жесткого диска. При этом можно создать максимум четыре основных раздела. Если требуется управлять

более чем четырьмя основными разделами, то один из них должен быть «расширенным». Тогда в области, занимаемой расширенным разделом, можно создать до 11 логических разделов. Если создаваемые разделы жесткого диска должны относиться к различным типам (основные, расширенные или логические), команда `fdisk` выдает дополнительный запрос о типе раздела.

Когда будет определено, какого типа должен быть новый раздел, программа спросит, с какого места он должен начинаться (как правило, с первого свободного цилиндра) и какого размера должен быть (то есть в каком цилиндре он заканчивается). Можно указывать размер и более удобным способом — в виде `+`*n*`M` или `+`*n*`G` (то есть `+50G` означает, что размер раздела должен составлять 50 Гбайт).

Определив новый раздел, можно отобразить всю таблицу разбиения на разделы с помощью команды `P`. Кроме того, можно определять новые разделы и удалять разделы, определенные ранее, и т. д.

Идентификационные номера разделов

Новые разделы, создаваемые `fdisk`, всегда относятся к типу *Linux native* (идентификационный номер 83). Если вам требуется раздел другого типа, нужно изменить номер нового раздела, нажав T *(type)*. Чаще всего используются разделы следующих типов (идентификационные номера указаны в шестнадцатеричной системе):

- 82 — раздел подкачки Linux;
- 83 — файловая система Linux (любая файловая система Linux — ext, reiser, xfs и т. д.);
- 8e — раздел Linux LVM;
- Fd — раздел Linux RAID.

Чтобы получить список всех доступных ID-номеров, выполните команду `L`. В полученном списке также будут указаны коды для разнообразных других операционных систем (MS DOS, Windows, UNIX и т. д.).

Сохранение сделанных изменений

Программа `fdisk` выполняет все изменения только при нажатии клавиши W *(write)*. Перед этим можно проверить нажатием V *(verify)*, верна ли вся внутрисистемная информация о диске. Это делается в качестве перестраховки. Обычно в ответ на такую команду система просто выдает количество секторов по 512 байт, которые не относятся ни к основным, ни к логическим разделам диска (то есть являются неиспользованными).

Если вы начинаете запутываться, то в любой момент можете завершить работу `fdisk` нажатием клавиши Q *(quit)* или сочетания Ctrl+C — тогда ваш жесткий диск останется без изменений (табл. 13.4).

ПРИМЕЧАНИЕ

Разделы, создаваемые fdisk, еще пусты, то есть в них нет файловой системы! Команды для создания файловой системы зависят от того, какая именно система вам требуется. Например, команда mkfs.ext3 создает файловую систему ext3. Если вы хотите вручную создать раздел подкачки, посмотрите в разделе 13.13, как это делается.

Таблица 13.4. Клавиши для выполнения команд fdisk

Клавиша	Значение
D	Удалить раздел (delete)
L	Отобразить идентификационный номер раздела (list)
M	Онлайн-справка (menu)
N	Создать новый раздел (new)
P	Отобразить список разделов (print)
Q	Завершить работу программы (не изменяя таблицу разбиения) (quit)
T	Изменить тип раздела (для разделов подкачки)
U	Изменить единицы измерения с секторов на цилиндры и наоборот (unit)
V	Проверить таблицу разбиения (verify)
W	Изменить таблицу разбиения (write)

Увеличение раздела

В принципе `fdisk` не может увеличить размер существующего раздела. Единственное исключение — вы можете изменить размер последнего раздела на жестком диске (или последнего логического раздела в расширенном разделе) при условии, что за этим разделом еще есть свободное место. В таком случае можно удалить этот раздел и создать на его месте новый, большего размера.

Команда `fdisk` изменяет только таблицу разбиения, не затрагивая сами данные, расположенные на диске. Это означает, что файловая система на увеличенном разделе не растет вместе с ним. Таким образом, часть раздела остается неиспользованной. Увеличить файловую систему можно лишь в некоторых дистрибутивах (раздел 13.7).

Вообще изменять размер раздела или файловой системы очень опасно, и пусть лучше этим занимаются профессионалы Linux! Если вам необходимо произвести такую операцию, предварительно выполните резервное копирование!

Пример

В следующем примере показано, как на жестком диске размером 500 Гбайт с тремя основными разделами создать расширенный раздел максимального размера, включающий, в свою очередь, логический раздел размером 50 Гбайт. Нажатие клавиши `P` позволяет получить информацию о текущем состоянии диска. Размер диска составляет 255 * 63 * 60 801 * 512 байт, то есть около 500 Гбайт. На диске есть три основных раздела: 2 Гбайт для `/boot`, раздел подкачки, имеющий размер еще 2 Гбайт и, наконец, около 150 Гбайт выделяется на системный раздел.

```
root# fdisk /dev/sdb
Количество цилиндров на диске – 60801.
Это не ошибка, но данное количество превышает 1024,
следовательно, могут возникнуть определенные проблемы с:
1) программами, работающими в ходе загрузки (например, старые версии LILO);
2) программами для загрузки и секционирования, предназначенными
   для работы в старых ОС (например, DOS FDISK, OS/2 FDISK).
Команда (m for help): p
Диск /dev/sdb: 500.1 ГБ, 500107862016 байт
```

```
255 головок, 63 секторов/дорожек, 60801 цилиндров
Единицы = цилиндры по 16065 * 512 = 8225280 байт
   Device   Boot    Start     End     Blocks    Id     System
/dev/sdb1               1     244    1959898+   83     Linux
/dev/sdb2             245     488    1959930    82     Linux swap
/dev/sdb3             489   18725  146488702+   83     Linux
```

Нажатие N создает новый расширенный раздел. В качестве начального цилиндра выделяется первый свободный цилиндр, конечным становится последний цилиндр на жестком диске.

```
Команда (m for help): n
Команда действие
  e   дополнительный
  p   основной раздел (1-4)
e
Выбранный раздел 4
Первый цилиндр (18726-60801, по умолчанию 18726): <Return>
Применить значение по умолчанию 18726
Последний цилиндр или +размер или +размерM или +размерK (18726-60801,
  default 60801): <Return>
Применить значение по умолчанию 60801
```

Нажатие клавиши N теперь делает расширенный раздел логическим. Указан размер 50 Гбайт. Таблица разделов диска снова отображается при нажатии клавиши P.

```
Команда (m for help): n
Первый цилиндр (18726-60801, по умолчанию 18726): <Return>
Применить значение по умолчанию 18726
Последний цилиндр или +размер или +размерM или +размерK (18726-60801,
  default 60801): +50G
Команда (m for help): p
Диск /dev/sdb: 500.1 ГБ, 500107862016 байт
255 головок, 63 секторов/дорожек, 60801 цилиндров
Единицы = цилиндры по 16065 * 512 = 8225280 байт
   Device   Boot    Start      End     Blocks    Id   System
/dev/sdb1               1      244    1959898+   83   Linux
/dev/sdb2             245      488    1959930    82   Linux swap
/dev/sdb3             489    18725  146488702+   83   Linux
/dev/sdb4           18726    60801  337975470     5   Extended
/dev/sdb5           18726    24805   48837568+   83   Linux
```

Нажатие клавиши W сохраняет изменения, внесенные в таблицу разделов. Поскольку во время выполнения команд некоторые разделы диска используются, системе не удается правильно считать таблицу. Иначе говоря, компьютер необходимо перезапустить, чтобы можно было использовать новый раздел.

```
Команда (m for help): w
Таблица разделов диска была изменена!
Вызов ioctl() для повторного считывания таблицы разбиения
Считать таблицу не удалось, ошибка 16: устройство или ресурс заняты
Перезагрузите систему, чтобы гарантировать успешное обновление таблицы разбиения
Синхронизация дисков.
```

Программа parted

Важнейшее отличие между fdisk и parted заключается в том, что fdisk предназначена исключительно для секционирования, а parted, напротив, насколько это возможно, учитывает содержимое разделов. Команду parted можно использовать, например, для того, чтобы одновременно изменять размеры разделов и содержащихся в них файловых систем без потери данных. Разумеется, чтобы этот механизм работал, parted необходимо обращаться к различным внешним командам или библиотекам. Подробное описание многочисленных функций parted дается по следующему адресу: http://www.gnu.org/software/parted/.

К сожалению, работать с этой программой еще сложнее, чем с fdisk. Другой недостаток — все сделанные изменения применяются незамедлительно, а не тогда, когда вы прикажете их сохранить. Поэтому я предпочитаю работать с fdisk или с графической программой для секционирования.

Запуск

При запуске parted необходимо указать устройство — жесткий диск. При нажатии клавиши H отобразятся команды, предлагаемые на выбор. Ввод H *команда* выдает короткий справочный текст по отдельным командам. Нажатие клавиши P позволяет отобразить таблицу разбиения — в данном случае имеем жесткий диск размером 500 Гбайт, полностью занятый тремя RAID-разделами.

```
root# parted /dev/sdb
(parted) p
Диск /dev/sdb: 500GB
Размер сектора (логический/физический): 512B/512B
Таблица разделов диска: msdos
Number    Start    End      Size     Type      File system    Flags
 1        32,3kB   1003MB   1003MB   primary   ext3           boot, raid
 2        1003MB   3002MB   1999MB   primary   linux-swap     raid
 3        3002MB   500GB    497GB    primary                  raid
```

Обслуживание

С помощью команд mkpart и rm можно соответственно создавать и удалять разделы. Первая команда создает новый раздел и помещает в нем желаемую файловую систему, если parted найдет нужные для этого программы и библиотеки (выбор зависит от конкретной файловой системы).

Вам потребуется сообщить командам, создающим разделы дисков, координаты начала и конца этих разделов. При этом используется сокращение GB — гигабайт. Программа parted обеспечивает следование разделов точно «друг за другом», без зазоров. Если вы создаете логический раздел, то перед этим необходимо соответствующим образом увеличить расширенный раздел. Команды, изменяющие размеры разделов, ожидают в качестве параметра так называемый *младший* номер устройства, то есть, например, 7 для /dev/sda7.

Если вы собираетесь использовать новый раздел в качестве раздела подкачки либо части системы RAID или LVM, то нужно соответствующим образом настроить тип раздела. Необходимая команда выглядит так: set *номер_раздела атрибут*. В том числе могут применяться boot, swap, lvm и raid.

Пример

С помощью следующих команд на жестком диске размером 500 Гбайт создаются три раздела RAID. Загрузочный раздел имеет размер 1 Гбайт, раздел подкачки — 2 Гбайта. Третий раздел занимает все оставшееся место. Поскольку в качестве конечной позиции указано -50MB, последние 50 Мбайт на диске остаются свободными (такой резерв нужен на тот случай, если придется выполнить точно такое же секционирование на другом диске, номинальный размер которого также составляет 500 Гбайт. Даже конструктивно идентичные жесткие диски обычно содержат неодинаковое количество блоков данных, что это связано с внутренним управлением ошибками).

```
root# parted /dev/sdb
(parted) print
Диск /dev/sdb: 500GB
Размер сектора (логический/физический): 512B/512B
Таблица разделов диска: msdos
Number   Start  End  Size  Type  File system  Flags
(parted) mkpart primary ext3 0     1GB
(parted) mkpart primary ext3 1GB   3GB
(parted) mkpart primary ext3 3GB -50MB
(parted) set 1 raid on
(parted) set 2 raid on
(parted) set 3 raid on
(parted) print
Диск /dev/sdb: 500GB
Размер сектора (логический/физический): 512B/512B
Таблица разделов диска: msdos
Number  Start    End     Size    Type     File system  Flags
 1      32,3kB   1003MB  1003MB  primary  ext3         raid
 2      1003MB   3002MB  1999MB  primary  ext3         raid
 3      3002MB   500GB   497GB   primary  ext3         raid
(parted) quit
```

Программа sfdisk

По сравнению с fdisk и parted команда sfdisk достаточно проста. С ее помощью можно построить список разделов диска и заново сегментировать жесткий диск, основываясь на таблице разбиения, предоставляемой в текстовой форме. Применять sfdisk удобно прежде всего в тех случаях, когда вы хотите в точности скопировать схему разбиения одного диска на другой, например для конфигурации RAID.

Пример. В следующем примере команда sfdisk-d /dev/sda узнает список разбиения первого диска, который может прочитать программа sfdisk. Символ | передает этот список второй команде sfdisk, которая соответствующим образом форматирует второй жесткий диск:

```
root# sfdisk -d /dev/sda | sfdisk /dev/sdb
```

Когда я испытывал эту команду, sfdisk иногда жаловалась, что второй жесткий диск уже используется. Убедившись, что это не так (в том числе выполнив dmesg | grep sdb), я применил со второй командой sfdisk параметр --force. Потребовалось дополнительно перезапустить компьютер, чтобы ядро восприняло новое секционирование второго диска:

```
root# sfdisk -d /dev/sda | sfdisk --force /dev/sdb
root# reboot
```

Параметр --force следует применять и в тех случаях, когда sfdisk сообщает, что раздел заканчивается не точно на границе цилиндра. В Linux и в большинстве других операционных систем границы цилиндров на жестком диске определяются произвольно и не влияют на работу системы.

Программа gparted

Для parted существует графический пользовательский интерфейс gparted (рис. 13.1). Эта программа может изменять только те разделы диска, которые в данный момент не используются, то есть не подключены к дереву каталогов. Все используемые разделы помечаются в программе символом замка, и кнопки для их обработки отключены. Расширенный раздел остается закрыт как минимум до тех пор, пока активен как минимум один логический раздел, расположенный внутри него. На практике это означает, что зачастую приходится использовать gparted вместе с «живым диском» Linux, иначе почти все важные операции будут заблокированы.

В отличие от parted, программа gparted запоминает все заданные действия, но сначала не выполняет их. Команда меню Правка ▸ Отменить отменяет выбранные операции, а Правка ▸ Принять наконец выполняет их.

К сожалению, gparted не может создавать разделы LVM или RAID. Соответствующие флажки отображаются, но с ними нельзя ничего сделать.

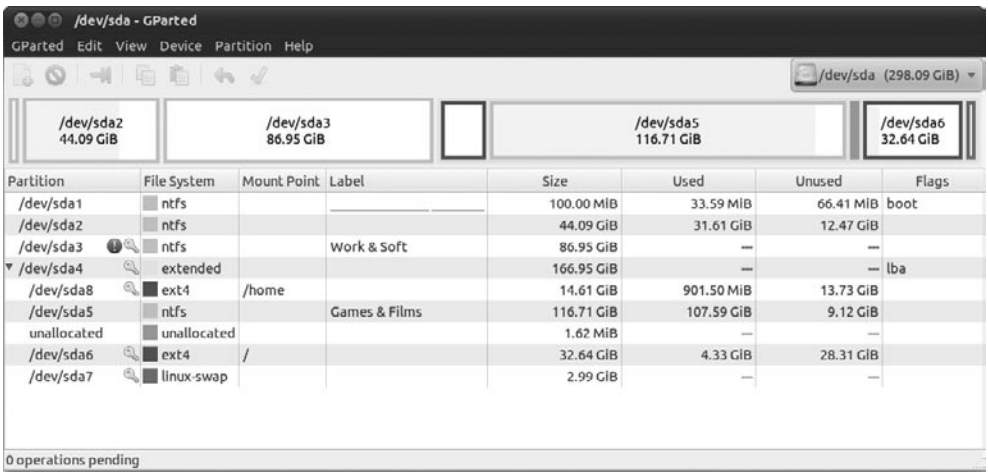

Рис. 13.1. Программа gparted

В SUSE в качестве альтернативы `gparted` можно использовать модуль YaST Система ▸ Разметка диска. В большинстве других дистрибутивов подобные инструменты отсутствуют, что тем более удивительно, так как в установочной программе предусмотрены соответствующие функции.

13.4. Типы файловых систем

В этом разделе мы кратко рассмотрим типы файловых систем, которые могут использоваться в Linux. Наиболее важные файловые системы — ext2-ext4, xfs, vfat, ntfs и iso9660 — мы подробнее рассмотрим далее в этой главе. Чтобы узнать, какой или какие типы файловых систем вы сейчас используете, необходимо выполнить команду `df -T`.

Linux

Файловые системы Linux приспособлены для установки Linux и работы с ней. В повседневной работе вы даже не будете замечать, какую именно файловую систему сейчас используете. Простейшие команды, например `ls` или `cp`, управление правами доступа и др. — все это работает независимо от файловой системы.

Файловые системы отличаются по признакам, представляющим интерес в первую очередь для опытных пользователей либо для тех, кто работает с сервером. К этим признакам относятся: скорость обработки внушительных по размеру файлов или большого количества сравнительно небольших файлов, эффективность выполнения операций считывания и записи, нагрузка, оказываемая на процессор, функция журналирования (меры, предпринимаемые после аварийного прекращения работы системы), функции квотирования (возможность ограничить максимальное потребление памяти на пользователя), совместимость с NFS, дополнительные затраты энергии на управление системой, поддержка дополнительных прав доступа (ACL), совместимость с SELinux и т. д.

- **ext** — на начальном этапе развития Linux была доминирующей система ext2 (расширенная файловая система, версия 2). С 2002 года ей на смену пришла система ext3, которая во многом совместима с ext2, но к тому же поддерживает функции журналирования, а при работе с версией ядра 2.6 и выше — и ACL. Максимальный размер файла составляет 2 Тбайт, максимальный размер файловой системы — 8 Тбайт. В конце 2008 года было официально заявлено о выпуске версии ext4, которая обладает обратной совместимостью с ext3, но многие функции внедрены более эффективно, чем ранее. Кроме того, максимальный размер файловой системы равен 1 Эбайт (1 048 576 Тбайт), и можно рассчитывать, что на какое-то время такого объема хватит.
- **reiser** — система получила название от имени своего основателя — Ганса Райзера (Hans Reiser) и была первой системой с функциями журналирования, обращавшейся за данными к ядру Linux. Версия 3.n в SUSE даже считалась стандартной в течение некоторого времени. Основные преимущества reiser в сравнении с ext3 — более высокая скорость работы и эффективность разме-

щения при работе с мелкими файлами (а в файловой системе, как правило, большинство файлов мелкие). Со временем, правда, разработка reisefers приостановилась. Уже давно было заявлено о выходе версии 4, которая все еще не готова, а поддержка версии 3 прекратилась.

- **xfs** — первоначально файловая система xfs разрабатывалась для рабочих станций фирмы SGI, функционировавших в операционной системе IRIX. Xfs особенно хороша для работы с крупными файлами, в частности идеально подходит для работы с потоковым видео. Система поддерживает квотирование и расширенные атрибуты (ACL).
- **jfs** — аббревиатура JFS расшифровывается «Журналируемая файловая система». Первоначально она была разработана для IBM, а затем адаптирована для Linux. Jfs никогда не пользовалась в Linux особым признанием и в настоящее время влачит жалкое существование, уступая другим файловым системам. Подробную информацию о ней вы найдете на сайте http://jfs.sourceforge.net/.
- **brtfs** — если на то будет воля ведущих разработчиков ядра, файловую систему brtfs в Linux ждет блестящее будущее. Эта система была разработана в Oracle с нуля. Она включает функции поддержки модуля отображения устройств (device-mapper) и RAID. Brtfs наиболее сходна с системой ZFS, разработанной компанией Sun. К ее самым интересным функциям относится проверка файловой системы на ходу, а также поддержка SSD (*твердотельные диски* — это жесткие диски, работающие на основе флеш-памяти). К сожалению, работа над brtfs в обозримом будущем не завершится. В Fedora, уже начиная с версии 11, предусмотрена возможность установки brtfs, но пользоваться ею я рекомендую только разработчикам файловых систем! Информацию о системе вы найдете по адресу: http://btrfs.wiki.kernel.org/index.php/Main_Page.

Не существует «быстрейшей» или «наилучшей» файловой системы — оценка зависит от того, для чего вы собираетесь использовать систему. Начинающим пользователям Linux, работающим с локальным компьютером, рекомендуется работать с ext3, а администраторам серверов — с ext4. Конечно, с ext4 скорость работы повыше, чем с ext3, но при этом в системе ext4 значительно хуже дело обстоит с надежностью данных — вы вполне можете потерять информацию при внезапном отключении системы.

UNIX

Если вы установили на компьютере вторую UNIX-подобную операционную систему, то при обмене данными (из одной ОС в другую) вам пригодятся следующие файловые системы.

- **sysv** — применяется в ОС SCO, Xenix и Coherent.
- **ufs** — используется в FreeBSD, NetBSD, NextStep и SunOS. Linux может только считывать информацию из таких файловых систем, но не может вносить изменения в данные. Для доступа к сегментам с BSD дополнительно потребуется расширение BSD disklabel. Аналогичное расширение существует и для таблиц разбиения SunOS.

- **ZFS** — это относительно новая система, разработанная Sun для Solaris. Поскольку код ZFS не соответствует лицензии GPL, ее нельзя интегрировать с ядром Linux. По этой причине Linux поддерживает эту файловую систему лишь опосредованно, через FUSE. Более подробная информация дается здесь: http://www.wizy.org/wiki/ZFS_on_FUSE.

Windows, Mac OS X

Следующие файловые системы будут полезны при обмене информацией с MS DOS, Windows, OS/2 и Macintosh.

- **vfat** — используется в Windows 9x/ME. Linux может считывать информацию из таких разделов и вносить в нее изменения. Драйверы системы vfat позволяют работать и со старыми файловыми системами MS DOS (8 + 3 символов).
- **ntfs** — система применяется во всех современных версиях Windows: от NT и выше. Linux может считывать и изменять ее файлы.
- **hfs и hfsplus** — эти файловые системы используются в компьютерах Apple. Linux может считывать и изменять ее файлы.

CD-ROM/DVD

На CD и DVD с данными обычно используются собственные файловые системы.

- **iso9660** — файловая система для CD-ROM описана в стандарте ISO-9660, допускающем только короткие названия файлов. Длинные названия поддерживаются в различных операционных системах по-разному, с помощью многообразных несовместимых друг с другом расширений. Система Linux способна работать как с расширением Rockridge, обычным в UNIX, так и с расширением Joliet, разработанным Microsoft.
- **udf** — этот формат (универсальный формат диска) появился и развился как наследник ISO 9660.

Сетевые файловые системы

Файловые системы не обязательно должны находиться на локальном диске — они могут подключаться к компьютеру и через сеть. Ядро Linux поддерживает различные сетевые файловые системы, из которых чаще всего применяются следующие.

- **smbfs/cifs** — помогают подключать сетевые каталоги Windows или Samba к дереву каталогов (см. раздел 13.12).
- **nfs** — это важнейшая в UNIX сетевая файловая система (см. раздел 13.12).
- **coda** — эта система очень напоминает NFS. В ней имеется множество дополнительных функций, но она не очень распространена.
- **ncpfs** — работает на базе протокола ядра NetWare; он используется Novell Netware.

Виртуальные файловые системы

В Linux существует несколько файловых систем, предназначенных не для сохранения данных на жестком диске (или другом носителе), а только для обмена информацией между ядром и пользовательскими программами. Эти файловые системы помечены в файле `/proc/filesystems` как `nodev`. Далее мы коротко рассмотрим важнейшие из них.

- **devpts** — эта файловая система обеспечивает доступ к *псевдотерминалам* (сокращенно — PTY) через `/dev/pts/*` в соответствии со спецификацией UNIX-98. (Псевдотерминалы эмулируют последовательный интерфейс. В системах UNIX/Linux такие интерфейсы используются эмуляторами терминалов, например `xterm`. При этом, как правило, применяются такие устройства, как `/dev/ttypn`. В спецификации UNIX-98, напротив, определяются новые устройства. Более подробная информация сообщается в текстовом терминале `HOWTO`.)
- **proc и sysfs** — файловая система proc служит для отображения служебной информации, касающейся управления ядром и процессами. В дополнение к этому файловая система sysfs строит взаимосвязи между ядром и оборудованием. Обе файловые системы подключаются на позициях `/proc` и `/sys`. Подробнее об этом рассказано в разделе 15.3.
- **tmpfs** — эта система построена на основе разделяемой памяти в соответствии с System V. Она обычно подключается на позиции `/dev/shm` и обеспечивает эффективный обмен информацией между двумя программами.

  В некоторых дистрибутивах (например, Ubuntu) каталоги `/var/run` и `/var/lock` также создаются с помощью файловой системы tmpfs. Файлы из этих каталогов применяются некоторыми сетевыми демонами для того, чтобы сохранять идентификационные номера процессов, а также информацию о доступе к файлам. Благодаря tmpfs эти данные теперь отражаются в RAM. Метод гарантирует высокую скорость, а также то, что после отключения компьютера в каталогах `/var/run` или `/var/lock` не останется никаких файлов.
- **usbfs** — файловая система usbfs, начиная с версии ядра 2.6 и выше, дает информацию о подключенных USB-устройствах. Обычно она интегрирована в файловую систему proc. О поддержке USB-устройств в Linux подробнее рассказано в подразделе «Интерфейсы и системы шин» раздела 9.6.

Прочие файловые системы

В заключение расскажу здесь еще о некоторых файловых системах, а также о ключевых словах, которым не нашлось места в предыдущих абзацах.

- **auto** — на самом деле файловой системы под таким названием не существует. Однако слово `auto` можно использовать в `/etc/fstab` или с командой `mount` для указания файловой системы. В таком случае Linux попытается самостоятельно распознать файловую систему. Этот метод работает с большинством важнейших файловых систем.

- **autofs, autofs4** — это тоже не файловые системы, а расширения ядра, автоматически выполняющие команду `mount` для выбранных файловых систем. Если файловая система не используется в течение некоторого времени, то для нее автоматически выполняется команда `umount`. Этот метод удобен прежде всего в тех случаях, когда из многих NFS-каталогов одновременно активно используются всего несколько.

  Для выполнения таких операций при запуске системы сценарий `/etc/init.d/autofs` автоматически выполняет программу `automount`. Она конфигурируется с помощью файла `/etc/auto.master`. Соответствующие программы автоматически устанавливаются, например, в Red Hat и Fedora. В любом случае autofs активизируется только после конфигурации `/etc/auto.master` или `/etc/auto.misc`. Более подробная информация содержится в следующем документе: http://tldp.org/HOWTO/Automount.html.
- **cramfs и squashfs** — файловые системы Cram и Squash предназначены только для чтения. Они используются для того, чтобы «упаковать» как можно больше заархивированных файлов во флеш-память или ПЗУ (постоянное запоминающее устройство).
- **fuse** — FUSE означает «Файловая система в пользовательском пространстве» (Filesystem in Userspace) и позволяет разрабатывать и использовать драйверы файловых систем вне ядра. Следовательно, FUSE всегда применяется с внешним драйвером файловой системы. FUSE работает, в частности, с драйвером NTFS ntfs-3g.
- **gfs и ocfs** — *Глобальная файловая система* (Global File System) и *Кластерная файловая система* от Oracle (Oracle Cluster File System) позволяют строить гигантские сетевые файловые системы, к которым могут параллельно обращаться множество компьютеров в один и тот же момент.
- **jffs и yaffs** — *Журналируемая файловая система для флеш-носителей* (Journaling Flash File System) и *Альтернативная файловая система для флеш-носителей* (Yet Another Flash File System) специально оптимизированы для работы с твердотельными дисками и флеш-носителями. С помощью специальных алгоритмов они пытаются равномерно использовать все ячейки памяти (технология «выравнивание износа»), чтобы избежать преждевременного отказа системы.
- **loop** — используется для работы с псевдоустройствами. Псевдоустройство (loopback device) — это адаптер, способный обращаться к обычному файлу как к блочному устройству. Благодаря ему в любом файле можно расположить любую файловую систему, а затем подключить ее к дереву каталогов с помощью `mount`. Отвечающая за это функция ядра — поддержка псевдоустройств — реализуется в модуле loop.

  Существуют разнообразные способы применения псевдоустройств. В частности, они могут использоваться при создании дисков в оперативной памяти для начальной инициализации (Initial RAM disk) для GRUB или LILO, при реализации зашифрованных файловых систем или тестировании ISO-образов для CD (см. раздел 3.6).

- **none** — само собой разумеется, что и none — это не файловая система. Однако эта функция предоставляет редко используемую возможность подключить локальный каталог в другой точке дерева каталогов. Для этого необходимо задать в команде `mount` или в файле `/etc/fstab` тип файловой системы `none`, а также поставить дополнительный параметр `bind`. Этот метод напоминает использование символьной ссылки, но внутри системы реализуется совершенно иначе. Использовать такой метод целесообразно, например, при конфигурации сервера NFS4 (см. раздел 20.2).
- **unionfs/aufs** — концепция unionfs или ее варианта aufs позволяет как бы накладывать друг на друга несколько файловых систем, причем система, расположенная выше всех, имеет приоритет. Системы unionfs и aufs применяются в некоторых живых системах. Linux запускается прямо с CD или DVD. На систему CD/DVD, предназначенную только для чтения, накладывается файловая система диска оперативной памяти. Снаружи «видна» только одна файловая система, которая составляется из базовой структуры CD/DVD и изменений, производимых в файловой системе диска оперативной памяти.
- **Зашифрованные файловые системы** — в Linux применяются различные методы шифрования содержимого файловой системы. Некоторые из этих методов основаны на применении специальных файловых систем (например, CryptoFS или eCryptfs).

Чтобы узнать, какие файловые системы в настоящее время интегрированы в ядро или загружены в виде модулей, посмотрите файл `/proc/filesystems`. Какие еще модули ядра с дополнительными файловыми системами есть в распоряжении, указано в каталоге `lib/modules/`*n*`/kernel/fs/`.

Ссылки

Актуальная информация об имеющихся в распоряжении файловых системах находится в каталоге filesystems документации ядра. Что касается базовой информации, рекомендую почитать файл `HOWTO` о файловых системах. Однако имейте в виду, что последние изменения вносились в эти документы еще летом 2000 года!

- http://www.kernel.org/doc/Documentation/filesystems/.
- http://www.tldp.org/HOWTO/Filesystems-HOWTO.html.

13.5. Управление файловой системой

После установки файловой системы вам, как правило, не придется беспокоиться об управлении файловой системой. Через различные каталоги вы получите доступ если не ко всем разделам жесткого диска, то по крайней мере к большинству из них. Если вы вставите CD или DVD либо подключите внешний носитель данных, файловые системы этих устройств автоматически подключатся к дереву каталогов. Все будет работать как по волшебству.

В этом разделе мы заглянем за кулисы системы и подробно рассмотрим команды mount и umount, а также файл /etc/fstab.

- Команды mount и umount всегда выполняются в тех случаях, когда к дереву каталогов подключается (либо отключается от него) раздел диска или носитель с данными. Разумеется, имея права администратора, вы сможете выполнять эти команды и сами, например в тех случаях, когда автоматика отказывает или когда вы работаете без графической системы рабочего стола (Gnome или KDE).
- Конфигурационный файл /etc/fstab управляет тем, какие файловые системы автоматически подключаются к дереву каталогов при запуске компьютера и какие настройки при этом действуют. Файл /etc/fstab автоматически конфигурируется при установке Linux. Если такая конфигурация вас не устраивает либо позже требования к системе изменятся, этот файл потребуется изменить в редакторе. В этом разделе я опишу синтаксис данного файла.

Как это ни странно, практически невозможно найти конфигурационный инструмент, который можно было бы использовать вместо команды mount или изменения /etc/fstab вручную. Редкий пример такого инструмента — модуль YaST Система ▸ Разбивка диска (SUSE). Многообещающе выглядит и новая программа Gnome Palimpsest, которая поставляется во все большем количестве дистрибутивов (в Fedora — с версии 11, в Ubuntu — с версии 9.10). Эта программа позволяет подключать и отключать разделы диска и отображает SMART-статус приводов (см. также раздел 13.16).

Особую группу составляют внешние накопители, например USB-накопители или жесткие диски FireWire. Во многих дистрибутивах такие носители также автоматически интегрируются в файловую систему при подключении к компьютеру. О том, как работать с внешними носителями данных, подробно рассказано в разделе 13.11.

Определение текущего состояния файловой системы

Команда df. Если вам нужно узнать, как в настоящее время организована ваша система Linux, проще всего выполнить команду df. Она указывает, где именно к вашей системе подключены жесткие диски, носители данных и т. д., а также сколько места еще свободно на конкретном жестком диске.

Команда mount. Используемая без дополнительных параметров, она сообщает еще более подробную информацию о файловых системах. В итоговом списке будут представлены и различные виртуальные файловые системы. В следующем примере результат выполнения данной команды разбит на колонки, чтобы его было удобнее читать.

```
user$ mount
/dev/mapper/vg-ubuntu on/           type ext3  (rw,relatime,errors=remount-ro)
/dev/mapper/vg-myhome on/myhome     type ext3  (rw,relatime)
/dev/mapper/vg-virt   on/virt       type ext3  (rw,relatime)
```

```
/dev/sda3            on/boot        type ext3  (rw,relatime)
Tmpfs                on/lib/init/rw type tmpfs (rw,nosuid,mode=0755)
Varrun               on/var/run     type tmpfs (rw,nosuid,mode=0755)
Varlock              on/var/lock    type tmpfs (rw,noexec,nosuid,nodev,...)
Udev                 on/dev         type tmpfs (rw,mode=0755)
...
```

Информация, напоминающая результат выполнения mount, содержится и в файлах /etc/mtab и /proc/mounts. В каждом из них хранится список всех носителей данных, которые в данный момент подключены к системе, с указанием типа файловой системы и использованных параметров mount. Файл /etc/mtab изменяется всякий раз, когда к дереву каталогов подключается новая файловая система либо отключается одна из файловых систем. Синтаксис mtab такой же, как и у /etc/fstab (см. ниже). В /proc/mounts, кроме того, содержатся параметры, которые прямо не указываются в /etc/fstab или с командой mount.

Как подключать и отключать файловые системы вручную (mount и umount)

После установки современного дистрибутива Linux система конфигурируется так, что команда mount понадобится вам лишь пару раз: все файловые системы Linux будут подключены к дереву каталогов. При вставке CD/DVD или внешнего носителя данных автоматически открывается новое окно файлового менеджера KDE или Gnome. Хотя все и работает, словно по волшебству, система раз за разом выполняет команду mount, чтобы подключать файловые системы к дереву каталогов либо отключать их.

Синтаксис mount выглядит следующим образом:

```
mount [options] device directory
```

Среди параметров указывается в том числе тип файловой системы [-t *xxx*]. Название устройства определяет раздел диска или привод (см. раздел 13.2). В качестве каталога можно указать любой каталог файловой системы, используемой в данный момент. (Он уже должен существовать! При необходимости создайте его с помощью команды mkdir!)

Как правило, команду mount может выполнять только администратор. Однако в /etc/fstab можно разрешить всем пользователям выполнять ее в некоторых разделах диска (параметр user или users).

Примеры. Продемонстрирую на примерах, как работать с mount. В первом примере мы открываем доступ ко всем данным раздела Windows-9x/ME через каталог /windows:

```
root# mkdir /windows
root# mount -t vfat /dev/sda1 /windows
```

Следующая команда подключает привод CD-ROM с диском с данными (файловая система ISO-9660) к общей файловой системе в каталоге /media/cdrom.

Устройство /dev/scd0 означает, что привод запрашивается через SCSI-систему ядра. В некоторых дистрибутивах вместо этого нужно указать устройство /dev/sr0.

```
root# mount -t iso9660 /dev/scd0 /media/cdrom
```

Если параметры привода CD-ROM (тип файловой системы, название устройства, каталог) внесены в файл /etc/fstab, то для подключения привода к дереву каталогов достаточно следующей команды:

```
root# mount/media/cdrom
```

remount. С помощью команды mount -o remount можно изменить настройки уже подключенной файловой системы. Например, следующая команда активизирует параметр exec для DVD — и вы можете выполнять программы, содержащиеся на DVD:

```
root# mount /media/dvd -o remount,exec
```

Если при подключении системного раздела в ходе запуска компьютера возникают проблемы, то раздел подключается в режиме «только для чтения». Однако, чтобы устранить причину ошибки, например исправить запись в файле /etc/fstab, часто требуется вносить изменения в файловую систему. Для этого необходимо выполнить следующую команду. С ее помощью системный раздел подключается заново и вы можете вносить в него изменения.

```
root# mount -o remount,rw /
```

Команда unmount. Чтобы отключить файловую систему от дерева каталогов, выполните команду umount:

```
root# umount /media/dvd
```

Автоматическое подключение файловых систем (/etc/fstab)

На самом деле было бы очень утомительно заново подключать различные разделы дисков к дереву каталогов при каждом запуске системы и всякий раз, вкладывая CD в привод, выполнять команду mount и указывать различные параметры. Чтобы облегчить себе работу, используйте файл /etc/fstab: в нем указывается, какие носители данных должны подключаться к файловой системе при запуске компьютера. В любом случае в fstab должна содержаться информация о системном разделе и файловых системах, необходимых для внутрисистемного управления.

Пример. В том или ином дистрибутиве максимально краткий файл fstab может выглядеть так:

```
# Пример двух строк в /etc/fstab
/dev/sda2  /      ext3  defaults   1   1
none       /proc  proc  defaults   0   0
...
```

В первой строке указано, что второй раздел первого жесткого диска используется в качестве системного каталога. В зависимости от того, в каком разделе жест-

кого диска установлен Linux, вам потребуется указать вместо sda2 имя устройства раздела, содержащего Linux!

Во второй строке к файловой системе подключается система управления процессами. На самом деле файлов и каталогов, расположенных в /proc, на жестком диске нет — мы видим лишь копии файлов, управление оригиналами которых происходит внутри ядра.

Синтаксис /etc/fstab

Из предыдущих примеров уже можно понять, каков формат fstab: в каждой строке в шести столбцах описывается носитель данных (раздел, файловая система).

Первый столбец. В первом столбце содержится название устройства носителя данных. О номенклатуре разделов жесткого диска рассказано в разделе 13.2. Другие примеры, касающиеся разделов Linux и Windows, приводов CD-ROM и т. д., будут приведены далее в этой главе.

Вместо названия устройства вы можете также указать имя (*имя тома* — в Red Hat и Fedora) или идентификационный номер файловой системы (в Ubuntu). В данном случае действует синтаксис LABEL=string или UUID=*nnn-nnn*. С помощью команды blkid можно узнать название и уникальный идентификатор раздела диска. Для изменения этих данных в различных файловых системах применяются специальные инструменты, например tune2fs.

```
root# blkid /dev/sda9
/dev/sda9: UUID="5a954fc1-00c6-4c25-a943-d4220eff350d" TYPE="ext4"
```

Преимущество этикетки или UUID по сравнению с именованием устройства заключается в том, что данные остаются правильными и в тех случаях, когда название устройства изменяется. Это может особенно легко произойти с USB-носителями. В зависимости от того, какие носители данных применялись раньше, внешний жесткий диск может запрашиваться через /dev/sdc, а в следующий раз — через /dev/sde.

К сожалению, программа fstab достаточно сложна в использовании, особенно при применении уникальных идентификационных номеров. Кроме того, возможны проблемы, если у вас параллельно установлены несколько дистрибутивов Linux. Как правило, при каждой установке нового дистрибутива определенные разделы диска форматируются заново. В таком случае они получают новые уникальные идентификаторы. Дистрибутивы, установленные ранее, «перестают узнавать» такие разделы, и вам предстоит нелегкая задача — проставить в fstab новые идентификационные номера.

Второй столбец. Во втором столбце определяется, к какому каталогу в дереве файлов подключается носитель данных. Каталоги, указанные во втором столбце, уже должны существовать. Они необязательно должны быть пустыми, но, подключив носитель, вы все равно не будете иметь доступа к файлам, содержащимся в этих каталогах, хотя получите доступ к файлам, расположенным на подключенном носителе.

Третий столбец. В третьем столбце указывается файловая система. В табл. 13.5 в алфавитном порядке перечислены важнейшие файловые системы. Вы можете

указать и несколько файловых систем, разделяя их названия запятыми. Например, для работы с CD/DVD-приводами можно указать `iso9660,udf`, так как на CD и DVD, как правило, применяются обе эти файловые системы. Команда `mount` автоматически выбирает из файловых систем, предоставляемых на выбор, ту, которая подходит для работы. Обратите внимание, что названия файловых систем нельзя разделять пробелами!

Таблица 13.5. Файловые системы

Файловая система	Использование
auto	Автоматическое определение файловой системы (CD-ROM, дискеты)
brtfs	Файловая система brtfs
cifs	Сетевой каталог Windows (Samba)
devpts	Псевдотерминалы, соответствующие спецификации UNIX-98
ext2, -3, -4	Файловая система ext версий 2, 3 и 4
iso9660	CD-ROM, DVD
nfs	Сетевой каталог UNIX (NFS)
ntfs	Файловая система Windows
proc	Управление процессами (/proc)
reiserfs, reiser4	Файловая система reiser версий 3.n или 4
smbfs	Сетевой каталог Windows (Samba)
swap	Разделы или файлы подкачки
sysfs	Управление системой (/sys)
tmpfs	Обмен данными между программами по системе System V (разделяемая память)
udf	Универсальный формат диска (DVD, CD-RW)
usbfs	Управление USB-устройствами
vfat	Файловая система Windows-9x/ME

Четвертый столбец. В четвертом столбце определяются параметры для доступа к носителям данных. Если задается несколько параметров, они разделяются запятыми. Пробелов быть не должно. В табл. 13.6 перечислены важнейшие универсальные параметры `mount`. Если вы не хотите использовать никаких параметров, используйте вариант `defaults`.

Таблица 13.6. Основные параметры команды mount

Параметр	Значение
defaults	Использовать параметры, заданные по умолчанию
dev	Интерпретация обозначений символьных и блочных устройств
exec	Разрешение выполнения программ (например, для CD/DVD-приводов)
noauto	Не подключать носитель данных к дереву каталогов при запуске системы
nodev	Игнорирование обозначений символьных и блочных устройств
noexec	Запрет на выполнение программ
nosuid	Запрет интерпретации битов доступа suid и guid
ro	Только для чтения (защита от внесения изменений)

Параметр	Значение
sw	Своп (файл или раздел подкачки)
suid	Интерпретация битов доступа suid и guid
sync	Не буферизовать доступ, позволяющий внесение изменений (надежнее, но медленнее)
owner	Владелец файла вправе выполнять команды (u)mount
user	Любой пользователь вправе выполнять mount, но выполнять umount может только тот пользователь, который выполнил mount последним
users	Любой пользователь вправе выполнять команды (u)mount

Пятый столбец. В пятом столбце содержится информация о программе `dump`, и пока этот столбец игнорируется. Здесь принято указывать для системного раздела `1`, а для остальных разделов и носителей данных — `0`.

Шестой столбец. В шестом столбце указано, следует ли проверять файловые системы при запуске компьютера, и если да, то в каком порядке. В большинстве дистрибутивов здесь указывается `1` для системного раздела, и `0` — для всех остальных разделов. Это означает, что при запуске системы только системный раздел проверяется на наличие ошибок и при необходимости восстанавливается.

Если вы хотите, чтобы автоматически проверялись и другие разделы, укажите для этих разделов значение `2`. Для всех файловых систем и носителей данных, которые нельзя или не следует проверять, нужно указать `0` (например, для разделов Windows, CD-ROM, DVD, дискет, виртуальных файловых систем, разделов подкачки и др.).

Если в пятом и шестом столбце `/etc/fstab` нет никаких записей, то система считает, что здесь проставлены нули.

13.6. Основы файловых систем

На следующих страницах речь пойдет в основном о файловых системах ext2, ext3, ext4 и xfs. Прежде чем описать их создание и администрирование, рассмотрим некоторые базовые данные, не зависящие от конкретной файловой системы.

Журналирование

Все распространенные файловые системы поддерживают функции журналирования. Если описать журналирование предельно просто, то это функция, дополнительно регистрирующая в специальном файле начало и конец каждой операции с файлом. Благодаря этому протоколу позже можно проверить, полностью ли была выполнена та или иная операция с файлом. Если операция не была закончена, ее можно выполнить повторно (в области работы с базами данных принят термин *транзакция*, а не *операция*). В самых совершенных системах журналирования также предусмотрена возможность занесения в журнал и конкретных изменений, сделанных в файлах. Это замедляет темп работы, но позволяет более полно реконструировать процесс в будущем.

Теперь, если операцию над файлом не удалось завершить, это видно из протокола. При простом журналировании изменения теряются (то есть чудес эта функция не совершает), однако вы можете просмотреть файл таким, каким он был до внесения изменений.

Основное преимущество функций журналирования заключается в том, что при следующем запуске компьютера система очень быстро возвращается в рабочее состояние и ее можно использовать практически сразу же. Это совсем иная ситуация, нежели ранее, когда после аварийного отключения компьютера или перебоев с электричеством требовалось тщательно проверить файловую систему и исправить возможные ошибки. Этот процесс длился несколько минут, а на особенно больших жестких дисках он мог затянуться на несколько часов.

ВНИМАНИЕ

При внезапном отключении электричества функция журналирования также не гарантирует полного восстановления файловой системы. Проблема в жестком диске: на нем в целях повышения эффективности при записи информации используется внутренний буфер обмена, поэтому файловая система может получить от диска подтверждение того, что данные получены и сохранены. На самом деле данные еще нужно записать из буфера обмена на жесткий диск, а на это может потребоваться несколько секунд. Если в этот краткий промежуток времени выскочат пробки, данные из буфера обмена будут потеряны. (На некоторых жестких дисках такое кэширование можно отключить. Из-за этого процесс записи на диск замедлится, но столь незначительно, что на практике этим можно пренебречь.)

Независимо от того, используется ли при записи кэш, поведение жесткого диска после внезапного отключения электричества предугадать сложно. Может быть и так, что жесткий диск запишет не информацию, а случайные биты, до того как пишущая головка внесет информацию на диск. Эта тема обсуждается по следующему адресу: http://lwn.net/Articles/191352/.

Иными словами, система журналирования файлов — вещь хорошая, но не исключает потери данных при неожиданном отключении электричества. Если ваши данные вам дороги, купите UPS (устройство бесперебойного питания), которое даст вам время корректно завершить работу компьютера и при отключении электричества.

Автоматическая проверка файловой системы

Если при запуске Linux обнаружит, что в прошлый раз работа компьютера была завершена некорректно, она проверит файловую систему системного раздела, а также другие разделы, указанные в файле `/etc/fstab` (будет ли проводиться такая проверка, определяется в зависимости от шестого столбца в `/etc/fstab` — см. также подраздел «Синтаксис /etc/fstab» предыдущего раздела). Благодаря применению функций журналирования такая проверка занимает всего несколько секунд.

Кроме того, в некоторых файловых системах (в том числе во всех версиях ext) предусмотрена регулярная проверка системы на предмет ошибок из-за несовместимости. Такие достаточно длительные тесты проводятся при запуске компьютера, если со времени последнего теста истек заданный промежуток времени или было выполнено заданное количество процессов `mount`.

После введения функций журналирования многократно приводились аргументы, что при применении таких функций отпадает необходимость регулярной проверки совместимости. Это справедливо, но, к сожалению, не совсем: файловая

система может стать несовместимой и из-за аппаратных ошибок жесткого диска — и вероятность таких ошибок возрастает по мере увеличения размеров жестких дисков. Например, в техническом паспорте своего жесткого диска (имеющего размер 1 Тбайт) я нашел указание, что вероятность возникновения ошибок в двоичных разрядах *(невосстановимые ошибки чтения для указанного объема прочитанных битов)* не превышает 1 к 10^{15}. Звучит так, как будто этой величиной действительно можно пренебречь. Однако, если учесть, что на таком жестком диске умещается $8 * 10^{12}$ бит, становится ясно, что при регулярном использовании этого носителя — без каких-либо повреждений — возникновение ошибок данных вполне вероятно. Если регулярно проверять совместимость файловой системы, таких ошибок все равно не предотвратить, зато вы сможете идентифицировать ошибки в работе системы и исправить их (как минимум в тех случаях, когда они затрагивают разделы, критически важные для внутрисистемного управления).

К сожалению, ни одна из файловых систем, описанных в этой книге, не является абсолютно отказоустойчивой. Однако в настоящее время ведутся активные работы по созданию файловых систем, которые могли бы распознавать аппаратные ошибки по неправильным контрольным суммам, а благодаря избыточному хранению данных даже исправлять такие ошибки. Кстати, отличным введением в область исследований, которые сегодня ведутся на ниве создания файловых систем, послужат две следующие статьи:

- http://lwn.net/Articles/190222/;
- http://lwn.net/Articles/196292/.

Вернемся к вопросу проверки файловой системы при запуске компьютера. Детали управления этим процессом часто зависят от конкретного дистрибутива. Как правило, такая проверка проводится без вмешательства пользователя и всего лишь замедляет процесс загрузки. Серьезные неприятности начинаются тогда, когда обнаруживаются неисправимые ошибки. В таком случае раздел диска загружается в режиме «только для чтения». После входа в систему с правами администратора вы можете вручную повторить проверку файловой системы и интерактивно указать, как программа проверки должна поступать с дефектными данными. Как правило, таким образом удается привести систему в состояние совместимости, даже если часть данных спасти не получится.

Проверка файловой системы вручную

Чтобы проверить файловую систему вручную, выполните команду `fsck`. В ходе контроля проверяемый раздел не должен использоваться, то есть предварительно необходимо выполнить `unmount`.

При работе компьютера так или иначе не получится проверить системный раздел, так как для него нельзя выполнить команду `unmount`. Вместо этого нужно с правами администратора выполнить команду `touch/forcefsck` и перезапустить компьютер. Кроме того, файл `forcefsck` создается при выполнении команды `shutdown` с дополнительным параметром `-F`.

Если файл `/forcefsck` существует, то почти во всех дистрибутивах при перезапуске системы автоматически выполняется проверка файловой системы. Если

проверка не начнется, загрузите компьютер с помощью восстановительной системы или с «живого диска» (Knoppix) и выполните команду `fsck` оттуда.

Максимальный размер

Раньше время от времени всплывал вопрос о том, каков может быть максимальный размер файла. Ответ зависит от того, какое ядро, процессор с какой архитектурой, какую библиотеку `glibc` и какую файловую систему вы используете. Все современные дистрибутивы поддерживают расширение библиотеки `glibc LFS` (LFS означает *«поддержка больших файлов»*). Таким образом, с точки зрения Linux, допустимый размер файла 2^{63} байт представляется практически безграничным. Во-вторых, в файловых системах различных типов задаются разные лимиты максимального размера файла (или файловой системы). В табл. 13.7 обобщены соответствующие данные. Помните, что 1 Тбайт = 1024 Гбайт.

Таблица 13.7. Названия устройств разделов жесткого диска

Файловая система	Максимальный размер файла, Тбайт	Максимальный размер файловой системы, Тбайт
brtfs	16 777 216	16 777 216
ext3	2	32 (при величине блока 8 Кбайт)
ext4	16	1 048 576 (или 1 Эбайт)
reiserfs	8	16
reiser4	8	Неизвестно
xfs	9 437 134	9 437 134
ZFS	16 777 216	16 777 216

Изменение типа файловой системы

Не существует инструментов, которые позволили бы преобразовать файловую систему одного типа в другой (например, из ext3 в reiserfs). Единственный способ — создать нужную файловую систему в новом разделе и скопировать туда все файлы.

13.7. Файловая система ext (ext2, ext3, ext4)

В мире файловых систем Linux доминируют различные версии ext. Проведу короткий экскурс в историю.

- ext, то есть первая версия файловой системы ext, недолго применялась на заре Linux (в 1992 году). Максимальный размер файловой системы составлял 2 Гбайт.
- ext2 была основной файловой системой Linux с 1993 по 2001 год. В этой версии максимальный размер файловой системы составлял 8 Тбайт.

- Важнейшими нововведениями, появившимися в системе ext3, были функции журналирования и поддержка контрольных списков доступа (с версиями ядра 2.6 и выше). Победное шествие ext3 с 2002 года было обусловлено не в последнюю очередь полной совместимостью: имеющиеся системы ext2 не требовалось форматировать заново, с них можно было перейти на ext3 с минимальными усилиями. Если файловая система корректно отключалась командой `unmount`, она даже могла далее использоваться как система ext2.
- С 2006 года началась разработка ext4. В конце 2008 года было официально объявлено, что эта система готова и многие современные дистрибутивы уже используют ext4 по умолчанию (Fedora — с версии 11, openSUSE — с версии 11.2, Ubuntu — с версии 9.10).

  Важнейшие нововведения: максимальный размер файловой системы достиг 1 Эбайт (1 048 576 Тбайт), время изменения файлов также протоколируется точнее, чем ранее. Так называемые *экстенты* позволяют запрашивать прилегающие друг к другу блоки данных файловой системы как группы, благодаря чему существенно упрощается управление крупными файлами. Кроме того, во многом была оптимизирована скорость работы: и удаление крупных файлов, и проверка файловой системы теперь проводятся в разы быстрее, чем в ext3.

  Не остались без внимания и проблемы совместимости: миграция с ext3 на ext4 проходит без малейших проблем. Однако имейте в виду, что при такой миграции обратного пути уже нет!

Совместимость различных версий файловой системы ext выражается и в том, что многие инструменты администрирования до сих пор содержат в имени команды номер версии 2, хотя могут применяться и при работе с новыми версиями (например, `tune2fs`).

/etc/fstab. Записи в файле `/etc/fstab`, касающиеся файловых систем ext3 и ext4, обычно выглядят так же, как в следующем примере.

```
# /etc/fstab: Файловые системы Linux
/dev/sdb8   /       ext4    defaults        1  1
/dev/sdb9   /boot   ext3    defaults        0  0
/dev/sdb9   /data   ext4    acl,user_xattr  0  0
```

GRUB. Версия GRUB 0.97 несовместима с ext4! Если хотите загрузиться прямо с системного раздела, содержащего систему ext4, найдите пропатченную версию GRUB 0.97 (поставляется, например, с Ubuntu 9.04, Fedora 12 и openSUSE 11.2, но не поставляется с Fedora 11) либо воспользуйтесь GRUB 2. Если в вашем дистрибутиве применяется оригинальная система GRUB 0.97 без патча для ext4, то вам потребуется отдельный раздел диска с файловой системой в формате ext2 или ext3!

Журналирование

Файловая система ext (версия 3 и выше) поддерживает функции журналирования. Необходимый для этого файл обычно использует специальные индексные

дескрипторы и, следовательно, не виден в файловой системе. В нем содержится информация только о тех файлах, которые были неполностью сохранены на жестком диске. Когда изменения будут выполнены, запись считается *зафиксированной* и может быть заменена новыми записями. Можно (но не принято) записывать файлы журналирования на отдельном устройстве.

В файловой системе ext различается три метода журналирования.

- `data=ordered` — в этом режиме в журнале сохраняются метаданные, то есть данные о файлах, но не информация, содержащаяся в файлах. В журнале файл обозначается как зафиксированный только тогда, когда он полностью сохранен на жестком диске. После аварийного завершения работы файловая система восстанавливается очень быстро, так как по сведениям журнала можно сразу узнать, какие файлы были сохранены неполностью. Однако такие файлы нельзя восстановить.

  В режиме `data=ordered` журнал каждые 5 секунд синхронизируется с жестким диском. В результате в ext3 все данные по тем или иным файлам физически сохраняются на жестком диске. Такой стандартный метод не очень эффективен, зато очень надежен: даже при общем коллапсе системы или отключении электричества серьезные потери информации практически исключены. В ext3 за `data=ordered` замечен неприятный побочный эффект: при каждом вызове функции `fsync` синхронизируется не только определенный файл, но и вся файловая система. Это может существенно замедлять работу системы.

  В ext4 журнал также синхронизируется с системой каждые 5 секунд, но изменения вносятся в файлы гораздо позже, с использованием технологии отложенного выделения. Чтобы немедленно сохранить файл, необходимо специально вызвать функцию `fsync` (правда, в ext4 для выполнения команды `fsync` не требуется синхронизировать всю файловую систему целиком, поэтому функция выполняется гораздо быстрее).

- `data=writeback` — этот режим напоминает `ordered`. Единственное отличие заключается в том, что работа журнала и операции с файлами не всегда протекают синхронно. Файловая система помечает в журнале данные как фиксированные, не дожидаясь окончательного сохранения информации на диск. В случае аварийного отключения последующая целостность данных гарантирована достаточно надежно. Однако не исключено, что в измененных файлах будут содержаться старые данные. Эта проблема не возникает, если процессы сохранения в пользовательских программах — в соответствии со стандартом POSIX — завершаются командой `fsync`.

- `data=journal` — в отличие от двух предыдущих режимов, при этом в журнале сохраняются не только метаданные, но и сами файлы. Все изменения приходится сохранять дважды (сначала в журнале, а затем в конкретном файле), поэтому система ext3 работает значительно медленнее. Данный метод позволяет восстановить после аварийного отключения и те файлы, изменения которых уже были записаны в журнале, но еще не сохранились в файле.

В принципе информация из журнала переносится на жесткий диск каждые 5 секунд. Этот промежуток можно изменить с помощью параметра `commit` команды `mount`. Если у вас установлен и сконфигурирован пакет `laptop-mode` и ноутбук работает от батареи, то промежуток `commit` будет значительно дольше.

Внутри системы работает демон журналирования `kjournald`, интегрированный в ядро и предназначенный для регулярного обновления файла журнала. Этот процесс запускается автоматически, как только к дереву каталогов командой `mount` подключается файловая система ext3 или ext4.

Отложенное выделение

Важнейшее нововведение системы ext4, влияющее на скорость ее работы, — это так называемое *отложенное выделение* — функция, действующая и во многих других современных файловых системах (например, brtfs, HFS+, reiser4, xfs и ZFS). Отложенное выделение заключается в том, что при внесении изменений в блоки с данными эти блоки резервируются не сразу, а в момент физического сохранения данных на диске — на это может потребоваться до полуминуты. Такой метод имеет два значительных достоинства: во-первых, операции сохранения данных можно выполнять группами, благодаря чему повышается скорость работы и снижается степень фрагментации файловой системы. Во-вторых, временные файлы, которые иногда существуют в течение всего нескольких секунд, зачастую вообще не сохраняются физически.

К сожалению, у отложенного выделения есть и недостатки: основная проблема заключается в том, что метаданные (то есть информация о состоянии файла) часто записываются в систему еще до того, как соответствующие изменения заносятся в файл. При использовании исходного варианта драйвера ext4 случалось так, что измененный, но еще не синхронизированный файл после аварийного отключения системы и последующего восстановления оказывался пуст. Эта проблема особенно часто происходила с конфигурационными файлами. (Многих пользователей устроит, если файл после восстановления системы просто сохранится в исходном состоянии. Однако полная потеря содержимого файла, а вместе с ним и конфигурации программы неприемлема.)

Теодор Цо (Theodore Ts'o), главный разработчик всех версий ext, считает, что потери данных обусловлены лишь тем, что во многих программах не выполняется команда `fsync`. Однако, в соответствии со стандартом POSIX, только выполнение этой команды гарантирует, что изменения действительно будут сохранены. В версии ядра 2.6.30 все же появились различные изменения драйвера ext4, призванные свести проблему к минимуму: если при изменении имеющихся файлов используются функции `rename` или `ftruncate` (как обычно), то ext4 отказывается от отложенного выделения. Вы можете полностью отключить его, задав параметр `nodelalloc` команды `mount`. Правда, при этом значительно снижается эффективность работы и часть достижений ext4 в области производительности сводится на нет.

Если вы желаете глубже изучить техническую сторону проблемы, то вас заинтересуют перечисленые ниже сайты. Чтобы их читать, необходимо хорошо знать

английский язык и иметь достаточно времени. Однако чтение того стоит: вы подробно изучите, какие концепции лежат в основе современных файловых систем, причем эти сайты будут полезнее иной лекции по информатике!

- http://thunk.org/tytso/blog/ — блог разработчика ext Теодора Цо;
- http://lwn.net/Articles/322823/ — о потере данных в ext4;
- http://lwn.net/Articles/326471/ — журналирование в ext3/ext4, fsync;
- http://lwn.net/Articles/327601/ — отчет о работе семинара Linux Storage and Filesystem Workshop.

Стандартная работа системы и дополнительные параметры

Если режим журналирования и распределения данных специально не установить в команде `mount` или файле `/etc/fstab`, то по умолчанию будут действовать следующие настройки:

- ext3 до версии ядра 2.6.29 — `data=ordered`;
- ext3 после версии ядра 2.6.30 — `data=writeback`;
- ext4 — `data=ordered` с отложенным выделением.

Изменение стандартного режима работы файловой системы ext3 в версии ядра 2.6.30 вызвало немало споров. Еще неизвестно, воспримут ли это изменение все дистрибутивы. Более вероятно, что в отдельных дистрибутивах конфигурация ядра будет изменена так, что ext3 будет работать как раньше, то есть в режиме `data=ordered`. В принципе режим журналирования можно изменить при каждом процессе `mount`, то есть данная настройка свободно задается при форматировании файловой системы.

Если хотите узнать, какой режим журналирования применяется в настоящее время, прочитайте сообщения ядра. В следующем примере три раздела используют систему ext3, и один раздел — ext4. Первое сообщение `EXT3-fs` относится к системному разделу.

```
root# dmesg | grep EXT
EXT3-fs: подключенная файловая система, работающая в режиме data = ordered
EXT3 FS в sda3, внутренний журнал
EXT3-fs: подключенная файловая система, работающая в режиме data = ordered
EXT4-fs (sda4): включение барьеров
EXT4-fs (sda4): включение отложенного выделения
EXT4-fs: включение файловых экстентов
EXT4-fs: включение mballoc
EXT4-fs (sda4): подключенная файловая система, работающая в режиме data = ordered
EXT4-fs (sda4): внутренний журнал в sda1:8
```

Чтобы специально выбрать определенный режим журналирования, укажите с командой `mount` или в файле `/etc/fstab` параметр `data=`*xxx*. В системе ext4 также можно отключить отложенное выделение параметром `nodelalloc`.

Администрирование

Создание файловой системы

Файловые системы ext2, ext3 и ext4 форматируются с помощью команд `mkfs.ext2`, `mkfs.ext3` и `mkfs.ext4`.

В следующем примере в *логическом томе* размером 20 Гбайт (то есть в разделе диска, управляемом LVM) создается файловая система ext4. Команда `mke2fs` сама задает размер блока данных 4 Кбайт и количество индексных дескрипторов, равное 1 310 720. Это означает, что в данной файловой системе можно расположить максимум 1,3 миллиона файлов. Таким образом, средний размер файла составит 16 Кбайт. Если вы хотите сохранять файлы большего или меньшего размера, можно указать с помощью `-i` *n*, через сколько байт должен располагаться индексный дескриптор (если средний размер файла будет меньше *n*, то размер файловой системы лимитируется не дисковым пространством, отведенным под раздел, а количеством индексных дескрипторов). Обратите внимание, что изменить абсолютное количество индексных дескрипторов вы уже не сможете. В большинстве случаев вам подойдет значение, задаваемое `mkfs.ext4` по умолчанию.

```
root# mkfs.ext4 /dev/mapper/vg1-test
mke2fs 1.41.4 (27-Jan-2009)
Обозначение файловой системы=
Тип ОС: Linux
Размер блока=4096 (log=2)
Размер фрагмента=4096 (log=2)
1310720 индексных дескрипторов, 5242880 блоков
262144 блоков (5.00%) зарезервировано для суперпользователя
Внешний блок данных=0
Максимальные блоки файловой системы=4294967296
160 групп блоков
32768 блоков в группе, 32768 фрагментов в группе
8192 индексных дескрипторов в группе
Резервные копии суперблоков, сохраненные в блоках:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000
Запись таблиц индексных дескрипторов: готово
Создание журнала (32768 блоков): готово
Запись суперблоков и учетной информации файловой системы: готово
Файловая система автоматически проверяется через каждые 35 операций подключения,
либо раз в 180 дней, в зависимости от того, что наступит ранее. Чтобы изменить,
выполните
tune2fs -c или -t .
```

Преобразование ext3 в ext4

Если хотите преобразовать имеющуюся файловую систему ext3 в ext4, просто выполните команду, указанную ниже. Команду `tune2fs` можно выполнять на ходу; но чтобы

использовать новые функции ext4, файловую систему необходимо заново подключить к дереву каталогов. Команда `mount -o remount` не работает.

```
root# tune2fs -O extents /dev/sda5
root# umount /dev/sda5
root# mount /dev/sda5 /home
```

При постепенном преобразовании файловой системы ext3 есть недостаток — имеющиеся файлы не будут использовать экстенты (а новые — будут). В данном случае вам поможет программа дефрагментации `e4defrag`, воспользоваться которой пока нельзя.

Проверка файловой системы

Файловые системы ext регулярно проверяются на наличие ошибок при запуске компьютера, а именно после выполнения определенного количества операций `mount` (по умолчанию 36) либо по истечении определенного времени (6 месяцев) в зависимости от того, какое из условий будет выполнено раньше. Обратите внимание: в некоторых дистрибутивах максимальное количество операций `mount`, после которых запускается проверка, либо временной интервал имеют большее значение или это значение равно 0 (проверка не выполняется). Кроме того, в большинстве дистрибутивов конфигурация `fstab` такова (если она вообще задается), что проверяется только системный раздел (это касается шестого столбца в `fstab` — см. подраздел «Синтаксис /etc/fstab» раздела 13.5).

Несмотря на применение функций журналирования, проверять файловую систему рекомендуется как минимум один-два раза в год. Во-первых, так заблаговременно распознаются ошибки оборудования. Во-вторых, не исключено, что в драйверах файловой системы могут быть еще неизвестные ошибки. Чем раньше будут распознаны ошибки, которые могут возникнуть в результате, тем меньше будет потенциальный ущерб.

Чтобы проверить файловую систему вручную, можно просто выполнить команду `fsck.ext2/ext3/ext4`. Во время контроля проверяемый раздел не может использоваться, то есть при необходимости сначала выполните `unmount`.

```
root# fsck.ext4 -f /dev/mapper/vg1-test
e2fsck 1.41.4 (27-Jan-2009)
Процесс 1: Проверка индексных дескрипторов, блоков и величин
Процесс 2: Проверка структуры каталогов
Процесс 3: Проверка связей между каталогами
Процесс 4: Проверка счетчика обращений
Процесс 5: Проверка общей информации о группе
/dev/mapper/vg1-test: 21357/1310720 файлов (1.3% без взаимосвязей),
    2062135/5242880 блоков
```

Обычно после проверки оказывается, что все в порядке. В противном случае в каталоге `/lost+found` каждого раздела сохраняются остатки тех файлов, которые уже невозможно восстановить. Если это текстовые файлы, то из их остатков вы, возможно, сможете извлечь крупицы полезной информации.

Настройка интервала для автоматической проверки системы

Действующие интервалы для автоматической проверки файловой системы можно определить и изменить с помощью команды tune2fs. Здесь с использованием параметра -c вы указываете максимальное количество операций mount, а с помощью параметра -i — временной интервал в днях:

```
root# tune2fs -l /dev/mapper/vg1-test
...
Mount count:              1
Maximum mount count:      35
Last checked:             Wed Jul 15 11:45:54 2009
Check interval:           15552000 (6 months)
...
root# tune2fs -c 100 -i 90 /dev/mapper/vg1-test
Установка максимального количества подключений 100
Установка интервала между проверками 7776000 секунд
```

Настройка названий разделов

С помощью команды e2label можно узнать, настроить или изменить внутреннее название файловой системы ext3 *(имя тома файловой системы)*:

```
root# e2label /dev/sda1 mylabel
```

Это имя можно задать в первом столбце файла /etc/fstab вместо названия устройства.

Настройка уникального идентификационного номера

При создании файловая система автоматически получает уникальный идентификатор (UUID), узнать который можно, выполнив команду vol_id или /lib/udev/vol_id. При необходимости эти номера можно изменить с помощью команды tune2fs -U. Такое изменение можно внести на ходу, использовать команду umount не требуется.

```
root# tune2fs -U random /dev/sda1              (Случайные UUID)
root# tune2fs -U f7c49568-8955-4ffa-9f52-9b2ba9877021 /dev/sda1  (Собственные UUID)
```

Изменение размера файловой системы

Команда resize2fs позволяет увеличить или уменьшить размер файловой системы ext. Обратите внимание, что при увеличении системы нужно сначала увеличить раздел или логический том, в котором она находится, а при уменьшении файловой системы нужно сначала уменьшить файловую систему, а потом — раздел диска или логический том. В следующем примере мы увеличим логический том с помощью команды lvextend (подробно об администрировании LVM рассказано в разделе 13.15).

```
root# lvextend -L 40G /dev/mapper/vg1-test
  Расширение тестирования логического тома до 40,00 Гбайт
  Тестирование логического тома выполнено, изменение размера
```

```
root# resize2fs /dev/mapper/vg1-test
resize2fs 1.41.4 (27-Jan-2009)
Файловая система на /dev/mapper/vg1-test подключена к /test;
  Необходимо изменить размер онлайн
old desc_blocks = 2, new_desc_blocks = 3
Выполнение изменения размера онлайн /dev/mapper/vg1-test
  на 10485760 (4k) блоков.
Файловая система на /dev/mapper/vg1-test теперь имеет размер 10485760 блоков.
```

Размер файловой системы можно увеличить, не выключая компьютер. Чтобы уменьшить размер файловой системы, ее необходимо отключить от дерева каталогов.

Фрагментация файловой системы

Под *фрагментацией* понимается состояние диска, при котором файлы сохраняются не в последовательно идущих друг за другом блоках, а в произвольном порядке, по всему разделу. Такое состояние может возникнуть, если попеременно удалять, сохранять, удлинять и укорачивать файлы. Фрагментация может значительно замедлить операции доступа к данным.

Драйверы ext2/3/4 пытаются, насколько это возможно, избежать фрагментации. Это, однако, удается сделать лишь в том случае, когда файловая система заполнена не более чем на 90 %.

Пока не существует инструментов, которые позволили бы дефрагментировать файловые системы ext. В настоящее время подобный инструмент для файловой системы ext4 находится в разработке. Эта программа — если она все же будет готова — позволит проводить дефрагментацию диска на ходу.

Доступ к файловым системам ext2 и ext3 из Windows

Вы можете обращаться к данным, сохраненным в Linux, и из системы Windows. Для этого существует несколько программ. По собственному опыту я рекомендую работать с Explore2fs. Это своего рода файловый менеджер, с помощью которого можно считывать файлы из систем ext2/3, но нельзя их изменять. Информацию о программе вы найдете на сайте http://www.chrysocome.net/explore2fs.

На первый взгляд драйвер файловой системы ext кажется еще элегантнее (http://www.fs-driver.org/). После его установки вы можете без малейших сложностей работать в Windows с любыми данными из разделов, отведенных под Linux, и даже изменять эти данные. При этом обязательно учитывайте, что, покидая одну систему и переходя в другую, нужно полностью завершать работу Windows или Linux (а не переводить ее в ждущий или спящий режим). Иначе вы рискуете повредить файловую систему и потерять данные.

Ни Explore2fs, ни драйвер файловой системы ext не совместимы с ext4.

13.8. Файловая система xfs

Файловая система xfs была разработана в 1994 году фирмой SGI для ее рабочей станции, действовавшей на базе UNIX-подобной файловой системы IRIX. Позже

эта файловая система была адаптирована для Linux, а теперь официально входит в состав ядра (версии 2.6 и выше). Система считается хорошо продуманной, стабильной и удобна в первую очередь для работы с очень большими (мультимедийными) файлами. Более подробная информация о файловой системе xfs сообщается здесь:

- `man xfs`;
- http://en.wikipedia.org/wiki/XFS;
- http://oss.sgi.com/projects/xfs/faq.html.

ВНИМАНИЕ

При работе с xfs не забывайте о двух важных особенностях. Во-первых, файловые системы xfs можно увеличивать командой xfs_growfs, а уменьшать нельзя. Во-вторых, эта система использует раздел, начиная с первого байта, и, в отличие от других файловых систем Linux, не оставляет места для загрузочного сектора. Поэтому при установке GRUB или LILO в загрузочный сектор раздела системы xfs уничтожаются части этой файловой системы! Если вам требуется установить GRUB или LILO, делайте это в ведущем секторе или в загрузочном секторе другого раздела.

/etc/fstab. Записи для рассматриваемой файловой системы в каталоге `/etc/fstab` обычно выглядят, как в следующем примере. Дополнительные параметры `mount` используются очень редко. Они перечислены в `man mount`.

```
# /etc/fstab
/dev/sdb13   /data   xfs   defaults  0  0
```

Создание файловой системы xfs

Чтобы создать в разделе диска файловую систему xfs, просто выполните команду `mkfs.xfs`:

```
root# mkfs.xfs /dev/sdc1
meta-data=/dev/sdc1    isize=256    agcount=16, agsize=152742 blks
         =             sectsz=512   attr=0
data     =             bsize=4096   blocks=2443872, imaxpct=25
         =             sunit=0      swidth=0 blks, unwritten=1
naming   =version 2    bsize=4096
log      =internal log bsize=4096   blocks=2560, version=1
         = sectsz=512  sunit=0 blks
realtime =none         extsz=65536  blocks=0, rtextents=0
```

Осталось выполнить команду `mount` и можно приступать к работе с файловой системой.

```
root# mount -t xfs /dev/sdc1 /test
```

Проверка файловой системы

Целостность файловой системы xfs автоматически проверяется при каждой операции `mount` (для этого просто интерпретируется протокол журналирования). Чтобы проверить систему вручную, выполните команду `xfs_check`. Это можно сделать,

когда файловая система еще не подключена. Если команда обнаружит ошибки, можете попытаться исправить их с помощью команды `xfs_repair`.

Если хотите обеспечить совместимость с другими файловыми системами, применяйте команду `fsck.xfs`. Однако помните, что она не выполняет никаких задач и всегда возвращает результат `OK`.

Изменение параметров файловой системы

Команда `xfs_growfs` позволяет увеличить файловую систему xfs на ходу (файловая система должна быть подключена к дереву каталогов!). Для выполнения команды необходимо, чтобы раздел диска, в котором расположена файловая система, уже был увеличен. Команда `xfs_admin` позволяет изменить различные параметры файловой системы, например ее название (`Label`) и уникальный идентификационный номер. Для этого файловую систему сначала нужно отключить от дерева каталогов (`unmount`).

13.9. Файловые системы Windows

У многих пользователей Linux на компьютере параллельно установлена версия Windows. Внешние носители данных (USB-флешки, карты памяти от цифровых фотоаппаратов) также часто используют файловые системы Windows. Далее будет рассказано, как, работая в Linux, обращаться к данным, сохраненным в файловых системах Windows, — независимо от того, находится эта информация в одном из разделов внутреннего жесткого диска либо на внешнем носителе.

Строго говоря, есть два типа файловых систем Windows.

- **FAT, VFAT, exFAT** — существуют многочисленные варианты FAT. Раньше других появились FAT12 для дискет, FAT16 для файловых систем размером менее 2 Гбайт, а также FAT32 для файловых систем размером до 8 Тбайт и для файлов до 4 Гбайт.

  В Windows 95 появилось дополнение VFAT, которое позволило сохранять во всех вариантах FAT названия, превышающие лимит 8 + 3 символов, известный со времен MS DOS. Длинные названия файлов система сохраняет как последовательности символов в Unicode. В более новых версиях Windows все имена файлов (в том числе короткие) сохраняются в таком виде. Подобный метод позволяет гарантировать, что регистр названий файлов сохранится и не возникнет никаких проблем с кодировками из-за применения различных кодовых страниц.

  Обычный раздел Windows, располагающийся на жестком диске, обычно содержит комбинацию FAT32 и VFAT (сокращенно — VFAT32). В дальнейшем, говоря о VFAT (или используя запись `mount/fstab` при обозначении файловой системы), я буду иметь в виду любые комбинации из FAT12/16/32 и VFAT.

  Существует относительно новый вариант FAT — система exFAT, которая была разработана для масштабных операций с флеш-памятью, но пока еще не

стала слишком популярной. В этой системе могут существовать файлы размером до 16 777 216 Тбайт, поддерживаются списки контроля доступа и транзакции.

- **NTFS (New Technology File System)** — появилась в Windows NT и поддерживается во всех современных версиях Windows. По сравнению с FAT, NTFS отличается повышенной надежностью (права доступа, журналирование и т. д.), а также содержит разнообразные дополнительные функции. Размер файловой системы практически не ограничен (16 777 216 Тбайт).

Поддержка в Linux. Linux способна работать с системами (V)FAT и NTFS — считывать из них информацию и вносить изменения. В файловой системе exFAT есть временный драйвер для Linux «только для чтения», пока не входящий в состав ядра.

ВНИМАНИЕ

Если вы хотите внести изменения в файловую систему Windows на компьютере, поддерживающем мультисистемную загрузку, то сначала полностью завершите работу Windows (а не переводите систему в спящий или ждущий режим) и только потом запустите Linux. Иначе вы можете спровоцировать несовместимость файловой системы и потерять данные.

В Linux почти никогда не приходится создавать файловые системы Windows, хотя такое возможно. С помощью команды `mkfs.vfat` можно отформатировать раздел диска для системы VFAT, с помощью `mkfs.ntfs` — для NTFS (обычно `mkfs.ntfs` находится в пакете `ntfsprogs`, который требуется дополнительно установить).

Преобразование текстовых файлов. Независимо от применяемой файловой системы обмен текстовыми файлами между Linux и Windows вызывает проблемы, так как в каждой из операционных систем применяются различные кодировки и обозначения конца строки. Эти проблемы решаются с помощью различных инструментов конвертации (см. главу 5).

Права доступа. В VFAT вообще не существует концепта прав доступа. В NTFS права доступа хотя и поддерживаются, но иначе, чем в UNIX/Linux. Возникает следующий вопрос: какие пользователи Linux получат определенные права доступа к файлам Windows? Ответ на этот вопрос дают параметры `uid`, `gid` и `umask`/`fmask`/`dmask`. Они определяют владельцев, их групповую отнесенность и биты доступа к файловой системе Windows — права для всех файлов этой системы будут одинаковы, независимо от того, как определены права доступа NTFS.

Файловая система VFAT

Стандартные настройки. Для начала коротко обобщим стандартные настройки драйвера файловой системы VFAT: драйвер самостоятельно распознает тип FAT (FAT12/-16/-32). Имена файлов Windows в Linux отображаются в кодировке латиница-1 (ISO8859-1). Пользователь, выполняющий команду `mount`, имеет право читать все файлы во всех каталогах, а также вносить изменения в эти файлы. Все остальные пользователи имеют право читать файлы, но не могут вносить изменения.

Разумеется, все эти настройки можно изменять с помощью параметров.

/etc/fstab. Как правило, запись в `/etc/fstab` для раздела локального диска с системой VFAT выглядит так:

```
# /etc/fstab
/dev/sda1    /media/win1    vfat    utf8,uid=1000   0 0
```

В результате имеем: пользователь с номером 1000 имеет право изменять все файлы, а специальные символы, содержащиеся в длинных именах файлов Windows (включающие более 8 + 3 символов), отображаются в Linux в кодировке UTF-8.

Следующая строка `fstab` не подключает раздел с Windows к дереву каталогов автоматически (`noauto`). Однако благодаря `users` каждый пользователь может выполнить команду `mount`. Кроме того, файлы Windows относятся к группе, являющейся стандартной для определенного пользователя, а не к группе, актуальной в настоящий момент.

```
# /etc/fstab
/dev/sda1    /media/win1    vfat    noauto,users,gid=users,utf8   0  0
```

Файловая система NTFS (ntfs-3g)

Раньше существовало много разных ntfs-драйверов для Linux. В последнее время определился доминирующий драйвер — ntfs-3g. Он поддерживает доступ для чтения и изменения файлов, а также может работать с потоками. Правда, этот драйвер не позволяет читать и изменять зашифрованные файлы, а также создавать заархивированные файлы (хотя дает возможность читать заархивированные файлы). Почти во всех крупных дистрибутивах драйвер ntfs-3g устанавливается по умолчанию.

В отличие от большинства других драйверов файловых систем, ntfs-3g внедряется не как модуль ядра, а как FUSE-драйвер. FUSE означает *«файловая система в пользовательском пространстве»*. Это модуль ядра, обменивающийся информацией с внешними программами. Иными словами, FUSE позволяет установить драйвер файловой системы вне ядра.

Файл /etc/fstab

Как правило, строка `fstab`, служащая для автоматического интегрирования раздела NTFS в дерево каталогов, выглядит так:

```
# /etc/fstab
/dev/sda1   /media/win   ntfs-3g   uid=1000,gid=1000  0 0
```

В большинстве дистрибутивов можно обозначить файловую систему просто ntfs (а не ntfs-3d).

Потоки

Потоки — характерная черта файловой системы NTFS: файл NTFS может состоять из нескольких потоков. При этом каждый поток функционально аналогичен обыч-

ному файлу. При простом доступе к файлу автоматически считывается или изменяется стандартный поток.

При работе с драйвером ntfs-3g параметр `streams_interface` управляет доступом к потокам. При стандартных настройках `xattr` потоки обрабатываются как атрибуты файла. Доступ к потокам обеспечивают команды `get` или `setfattr` из пакета attr (см. также раздел 3.8 о списках контроля доступа и расширенных атрибутах (EA/ACL)). Команда `getfattr -d -e text` возвращает список всех атрибутов, причем их содержимое представляется в виде текста.

```
root# mount /dev/sda1 /media/win
root# cd /media/win
root# cat > streamtest
abc (Ctrl)+(D)
root# setfattr -n user.stream1 -v "efg" streamtest
root# setfattr -n user.stream2 -v "xyz" streamtest
root# cat streamtest
abc
root# getfattr -d -e text streamtest
# file: streamtest
user.stream1="efg"
user.stream2="xyz"
root# cd
root# umount /media/win
```

Кроме того, можно работать с настройкой `streams_interface=windows`. Она активизирует типичный для Windows метод записи в форме *имя_файла*:*имя_потока*.

```
root# mount -o streams_interface=windows /dev/sda1 /media/win
root# cd /media/win
root# cat streamtest
abc
root# cat streamtest:stream1
efg
```

Администрирование

В пакете `ntfsprogs` содержатся различные команды, помогающие при администрировании файловых систем NTFS (табл. 13.8).

Таблица 13.8. Команды из пакета ntfsprogs

Команда	Значение
mkntfs	Создает файловую систему NTFS
ntfsclone	Копирует файловую систему NTFS
ntfsinfo	Сообщает информацию о файловой системе NTFS
ntfslabel	Дает заголовок разделу с NTFS
ntfsresize	Изменяет размер файловой системы NTFS
ntfsundelete	Пытается восстановить удаленные файлы

13.10. CD, DVD, дискеты

CD и DVD с данными

В принципе, управление CD и DVD не отличается от управления жесткими дисками. Однако здесь есть два важных отличия: во-первых, диск в приводе можно заменить, а обычный жесткий диск заменить на ходу нельзя. Во-вторых, на CD и DVD с данными используются особые файловые системы: ISO9660 или UDF.

ISO9660 и UDF. Сначала коротко рассмотрим эти файловые системы в целом. *ISO9660* представляет собой общепризнанный стандарт для CD с данными. Поскольку эта система имеет несколько фундаментальных ограничений, для нее известны также некоторые расширения. Характерное для UNIX расширение Rockridge позволяет сохранять длинные названия файлов и права доступа. Характерное для Windows расширение Joliet обеспечивает возможность использовать в именах файлов символы Unicode. Расширение El-Torito позволяет запускать компьютер прямо с CD.

UDF (универсальный дисковый формат) появился после ISO9660. Он используется на многих DVD (хотя вполне может использоваться и формат ISO9660). Отличия от ISO9660 заключаются в том, что файлы могут иметь размер более 2 Гбайт, имена файлов без установки каких-либо дополнений могут состоять из символов Unicode общим количеством до 255, лучше поддерживаются RW-носители (пакетная запись) и т. д.

Названия устройств CD/DVD. В табл. 13.9 указано, какие названия устройств применяются при доступе к приводу CD/DVD. Название устройства зависит от того, как подключен привод (IDE, SCSI, SATA, USB или FireWire) и применяется ли современное ядро с поддержкой libata. Вероятнее всего, вам встретится `/dev/scd0`. Только в очень старых компьютерах вам, может быть, попадется `/dev/hda`, `/dev/hdb` и т. д. Кроме того, существуют файлы-устройства `/dev/cdrom`, `/dev/dvd` или `/dev/dvd-recorder`. В данном случае мы имеем дело со ссылками на настоящие файлы-устройства.

Таблица 13.9. Названия устройств CD/DVD

Название устройства	Значение
/dev/scd0 или /dev/sr0	Первый привод CD/DVD
/dev/scd1 или /dev/sr1	Второй привод CD/DVD
/dev/hda	IDE-привод (контроллер 1/ведущий, обычная система IDE)
/dev/hdb	IDE-привод (контроллер 1/ведомый, обычная система IDE)
/dev/hdc	IDE-привод (контроллер 2/ведущий, обычная система IDE)
/dev/hdd	IDE-привод (контроллер 2/ведомый, обычная система IDE)

Автоматическая эксплуатация. В большинстве дистрибутивов конфигурация локальной системы задается таким образом, что при вкладывании в привод CD или DVD автоматически открывается окно файлового менеджера, отображающее содержимое носителя. В любой момент можно извлечь CD/DVD, нажав кнопку на приводе или щелкнув на значке соответствующего привода в контекстном меню.

Такое удобство обеспечивается благодаря «закулисному» управлению оборудованием в Linux и работе специальной службы KDE или демона Gnome (см. подраздел «Система горячего подключения» раздела 9.6 и раздел 13.11).

Ручное управление. Если вы работаете с консолью либо с локальной системой, где не предусмотрено автоматическое управление CD/DVD, вам потребуется вручную подключать диск к дереву каталогов после того, как он окажется в приводе. Как обычно, названия устройств и каталогов при этом могут варьироваться в зависимости от оборудования и дистрибутива.

```
root# mount -t iso9660 -o ro /dev/scd0 /media/dvd   (ISO9660-CD/DVD)
root# mount -t udf -o ro /dev/scd0 /media/dvd       (UDF-DVD)
```

По умолчанию все файлы и каталоги открыты для чтения всем пользователям. Если вы хотите непосредственно запускать программы, находящиеся на CD или DVD, добавьте параметр `exec`. Чтобы интернациональные имена файлов правильно обрабатывались, нужно использовать параметр `iocharset=utf8` или просто `utf8`.

Прежде чем извлечь CD/DVD, необходимо выполнить команду `unmount`:

```
root# umount /media/dvd
```

СОВЕТ

Вместо unmount вы можете выполнить eject. В таком случае CD не только отключается от файловой системы, но и извлекается из компьютера. Если в компьютере находится несколько носителей данных, которые можно извлечь (CD, DVD, ЗУ на магнитной ленте), эти варианты интерпретируются по очереди: извлекается первый в очереди носитель данных. Вы также можете задать нужное устройство, указав название устройства или точку подключения к системе.

Device is busy (Устройство занято). Если команда `unmount` возвращает ошибку Device is busy (Устройство занято), это означает, что данные с CD-ROM используются другой программой. Такая реакция возникает и в том случае, когда какой-либо из каталогов диска открыт в одной из оболочек. Выполните в нем команду `cd`, чтобы перейти в домашний каталог. Если требуется найти процесс, из-за которого возникает такая ошибка, воспользуйтесь командой `fuser` — выполните `fuser -m /cdrom`.

Еще одной причиной такой ошибки может быть NFS: если привод CD-ROM используется через NFS на другом компьютере, выполнить `unmount` зачастую не удается и тогда, когда тот компьютер уже давно открыл доступ к приводу. В таких случаях требуется перезагрузить NFS-сервер и (иногда) даже сам компьютер.

/etc/fstab. В большинстве дистрибутивов применяется описанное выше автоматическое управление, и поэтому в `/etc/fstab` отсутствует запись, касающаяся CD/DVD-привода (она не нужна). Однако если вы часто вручную подключаете CD/DVD к дереву каталогов, такая запись вам пригодится. Она будет выглядеть примерно как в следующем образце:

```
# /etc/fstab
/dev/scd0   /media/dvd  udf,iso9660   users,noauto,ro   0  0
```

Теперь достаточно будет команд `mount /media/dvd` или `mount /media/dvd`, чтобы интегрировать CD/DVD в дерево каталогов либо отключить диск от дерева. Эти команды может выполнять любой пользователь.

Аудио-CD, видео-DVD

Аудио-CD. Работа с аудио-CD отличается от обработки CD с данными. Они не подключаются к дереву каталогов командой `mount`, а напрямую считываются специальными программами (например, в KDE и Gnome это программы Amarok и Rhytmbox соответственно). Можно выполнить и цифровое считывание аудиотреков (например, для последующего преобразования их в файлы Ogg-Vorbis).

Видео-DVD. Как правило, при работе с видео-DVD используется файловая система UDF.

Запись CD и DVD. Для записи CD и DVD используйте в KDE программу K3B, в Gnome — Brasero, а в консоли — команду `wodim` (см. раздел 3.6).

Дискеты

При повседневной работе с Linux дискеты уже не играют никакой роли. Однако, если вы вдруг окажетесь в ситуации, когда нужно считать информацию с дискеты, вам потребуется подключить ее к файловой системе с помощью `mount` и, как обычно, обратиться к ее файлам. Для дискет чаще всего используется название устройства `/dev/fd0`. На них могут находиться различные файловые системы. Обычно дискеты для MS DOS/Windows работают с файловой системой VFAT.

```
root# mount -t vfat /dev/fd0 /media/floppy
```

С помощью команд из пакета `mtools` (например, `mcopy` или `mdir`) можно считывать информацию с дискет MS DOS/Windows либо изменять ее и без применения `mount`. С помощью команды `fdformat` выполняется низкоуровневое форматирование. Чтобы дополнительно создать файловую систему, выполните команду `mkfs.vfat`.

13.11. Внешние носители данных (USB, FireWire и др.)

USB-флешки, карты памяти цифровых фотоаппаратов, жесткие диски FireWire и eSATA, а также другие внешние носители имеют важный общий признак: они на ходу подключаются к компьютеру, на ходу же и отключаются. Система работает почти со всеми такими носителями, как с SCSI.

Автоматическое управление

Локальные системы (KDE, Gnome) практически всех дистрибутивов реагируют на подключение внешнего носителя так: открывается новое окно файлового менеджера (иногда с запросом о подтверждении), обеспечивающее удобный доступ к файлам подключенного внешнего носителя. Часто на рабочем столе появляется значок, обозначающий носитель и позволяющий открыть контекстное меню, через которое файловую систему можно специально отключить от дерева каталогов.

ВНИМАНИЕ

Обратите внимание: необходимо специально отключать все разделы внешнего носителя из дерева каталогов, а только потом извлекать кабель! В большинстве дистрибутивов для этого нужно щелкнуть на значке носителя и выбрать Eject (Извлечь), Safely Remove (Безопасное извлечение) или подобную запись в меню. Таким образом, вы гарантируете, что все операции записи будут завершены, а потом на самом деле отключится устройство. Если пренебречь этим шагом, вы рискуете повредить файловую систему и потерять данные!

В KDE и Gnome возможна ситуация, в которой несколько пользователей параллельно входят в систему. В таком случае права доступа к новому подключенному внешнему носителю, как правило, получает пользователь, вошедший в систему раньше других. Этот частный случай по-разному решается в различных дистрибутивах (или вообще не решается), поэтому могут возникнуть проблемы. Таким образом, старайтесь не менять пользователя, когда работаете с внешними носителями!

Внутрисистемная обработка горячего подключения

Управление горячим подключением в современных дистрибутивах осуществляется в тесном взаимодействии ядра, системы udev, системы обмена информацией D-Bus и программы PolicyKit (см. также раздел 4.4 и подраздел «Система горячего подключения» раздела 9.6). В более старых версиях вы, возможно, встретите программы `supermount`, `magicdev` или `subfs/submount`, но все они не очень хорошо работают.

Управление вручную

При работе в текстовом режиме или с локальной системой, в которой не предусмотрено автоматическое управление носителями, вам потребуется самостоятельно выполнить команду `mount`. Для этого сначала определите, какое название имеет ваше устройство (как правило, это `/dev/sd`*x*, где *x* — первая свободная буква по алфавиту).

Обзор всех носителей данных (включая жесткие диски, но исключая приводы CD и DVD) выводит команда `fdisk -l`. В следующем примере `/dev/sdf1` — первый и единственный раздел на USB-флешке.

```
root# fdisk -l
...
Диск /dev/sdf: 256 MB, 256901120 байт
16 головок, 32 секторов/дорожек, 980 цилиндров
Единицы = цилиндры по 512 * 512 = 262144 байт
Device     Boot    Start    End    Blocks    Id   System
/dev/sdf1     *        1    980    250864    e    W95 FAT16 (LBA)
```

USB-флешки и карты памяти также можно форматировать как Superfloppy. Это означает, что на диске не создается таблица разделов. В таком случае весь

привод запрашивается как устройство /dev/sda (вместо обычного способа именования устройства с помощью /dev/sda1, когда указывается номер определенного раздела).

Если знать номера устройств, дальше все просто: создается новый каталог и выполняется следующая команда mount:

```
root# mkdir /media/memorystick
root# mount /dev/sdf1 /media/memorystick
```

На внешних носителях могут использоваться файловые системы различных типов. На практике на внешних жестких дисках и USB-флешках чаще всего применяется система VFAT. То же касается карт памяти для различных электронных приборов.

После того как вы считаете или запишете все нужные файлы, выполните команду unmount, как обычно. Ни в коем случае не отключайте кабель USB или FireWire, пока не выполните unmount, иначе рискуете потерять данные!

```
root# umount /media/memorystick
```

Файл /etc/fstab

Лишь администратор вправе выполнять команду mount. Если обычные пользователи должны иметь возможность самостоятельно подключать к дереву каталогов внешние носители и отключать их из системы, то вам потребуется вставить в fstab соответствующую строку с параметром users. Для USB-флешки с файловой системой VFAT эта строка может выглядеть так:

```
# /etc/fstab: USB-Stick
/dev/sdf1  /media/memorystick  vfat  users,gid=users,utf8,noatime,noauto 0 0
```

Теперь каждый пользователь может подключить USB-флешку к дереву каталогов с помощью команды mount /media/memorystick, а потом читать и изменять данные, содержащиеся на ней. Однако для этого метода характерны два серьезнейших недостатка.

- В зависимости от того, в каком порядке подключаются устройства, их названия изменяются. Если USB-флешка подключается в качестве второго или третьего устройства, ее название может быть, например, /dev/sdg и доступ к устройству через каталог /media/memorystick будет закрыт.
- Наоборот, вышеуказанная запись fstab может быть использована для доступа к другому устройству, что, возможно, не планировалось.

Таким образом, лучше всего задавать имя устройства в /etc/fstab не непосредственно, а через ссылку by-uuid. Чтобы узнать номер UUID, выполните команду blkid:

```
root# blkid /dev/sdf1
/dev/sdf1: UUID="4550-9BD2" TYPE="vfat"
```

Соответствующую запись fstab мы разделим на две строки, так как она не помещается на одной:

```
# /etc/fstab: USB-Stick
/dev/disk/by-uuid/4550-9BD2 /media/memorystick vfat \
users,gid=users,utf8,noatime,noauto 0 0
```

В принципе вы можете начать строку fstab с записи UUID=4550-9BD2. Команда mount в таком случае будет работать, как в предыдущем примере, но с unmount возникнут проблемы: вместо уникального идентификатора UUID команда mount запишет носитель данных в /etc/mtab с актуальным названием устройства. При этом у unmount строки /etc/fstab и /etc/mtab не совпадут, в результате чего возникнет ошибка.

13.12. Сетевые файловые системы (NFS, CIFS)

Сетевые каталоги Linux (NFS)

Сетевая файловая система — это самый обычный путь, по которому в UNIX/Linux каталоги передаются с локального компьютера в сеть. В этом разделе описано, как использовать сетевые каталоги с точки зрения клиента. О том, как настроить конфигурацию NFS-сервера, рассказано в разделе 20.1.

В этом разделе мы рассмотрим только самую популярную версию системы NFS — третью. В Linux не так давно стала поддерживаться и версия NFS4. Однако эта версия, несмотря на многочисленные усовершенствования по сравнению с третьей, пока не очень распространена. Если вы желаете работать с NFS4, посмотрите в разделе 20.2 соответствующую информацию о конфигурации сервера и клиента.

Необходимые условия

Для того чтобы работать с NFS, вы должны выполнить несколько предварительных условий.

- Компьютер, на котором вы собираетесь читать или изменять файлы, должен быть доступен в сети (проверьте это командой ping).
- В сети должен быть установлен NFS-сервер, сконфигурированный таким образом, чтобы у вас было право доступа к нужному каталогу. Конфигурация сервера NFS подробно описана в разделе 20.1.
- На локальном (клиентском) компьютере нужно установить программы, необходимые для работы с NFS. В большинстве дистрибутивов они установлены по умолчанию. В Debian и Ubuntu необходимо специально установить пакет nfs-common.
- Если на вашем компьютере или на компьютере, где работает NFS-сервер, включены брандмауэры, они не должны блокировать обмен данными в сети NFS. Система NFS использует не только порты 111 и 2049, но и другие порты, назначаемые случайным образом.

Команда mount

Если эти условия выполнены, получить доступ к каталогу NFS несложно. Следующая команда позволяет интегрировать каталог /data внешнего компьютера mars в папку /media/nfsdata в дереве каталогов. Разумеется, каталог /media/nfsdata должен существовать до того, как вы выполните команду mount!

```
root# mount -t nfs mars:/data /media/nfsdata
```

Теперь вы можете обращаться ко всем данным из каталога mars:/data так же, как и к данным на своем локальном компьютере. Сможете ли вы изменять эти данные, зависит от конфигурации NFS-сервера.

Вместо имени компьютера можно указать и его IP-адрес, то есть, например, 192.168.0.10:/data. Это решение на крайний случай, если в сети нет сервера имен, а имя NFS-сервера не внесено в /etc/hosts.

Команда unmount вновь удаляет каталог NFS из локальной файловой системы. Если при этом сетевое соединение уже разорвано, нужно выполнять unmount с параметром -f. В ином случае вам потребуется очень долго ждать, пока не завершится выполнение mount.

```
root# umount /media/nfsdata
```

Файл /etc/fstab

Чтобы при запуске компьютера автоматически интегрировать каталоги NFS в файловую систему, добавьте в файл /etc/fstab строку, построенную по следующему образцу. В четвертом столбце можно использовать NFS-специфичный параметр bg. Благодаря этому mount запускается в фоновом режиме и пытается подключить сетевой каталог, если такого каталога еще нет в распоряжении. Это следует делать прежде всего во время подключения сетевых каталогов при запуске компьютера.

```
# Дополнение в /etc/fstab
jupiter:/data   /externaldata  nfs  user,exec,bg  0 0
```

В большинстве дистрибутивов все каталоги, указанные в /etc/fstab, подключаются в ходе процесса Init-V. SUSE — исключение из этого правила: здесь данный процесс выполняется сценарием Init-V nfs, который необходимо специально активизировать:

```
root# insserv nfs
```

ВНИМАНИЕ

NFS не учитывает кодировки, в которых записаны названия файлов, и интерпретирует эти названия просто как последовательности байт. NFS предполагает, что все пользователи работают с одной и той же кодировкой. Если это не так, то символы, не входящие в ASCII, могут представлять проблему. Например, пользователь NFS работает с дистрибутивом Linux, в котором поддерживается кодировка UTF-8, и сохраняет файл grundstück.txt. Другой пользователь работает с дистрибутивом, в котором поддерживается кодировка латиница-1, и он увидит файл, сохраненный первым пользователем, под названием grundst??ck.txt, где символы ?? соответствуют последовательности байт, присвоенной в UTF-8 символу ü.

Что делать? Нужно либо использовать NFS 4, либо убедиться, что все пользователи будут работать с одинаковой кодировкой, либо вообще не указывать в названиях файлов специальных символов!

Сетевые каталоги Windows (CIFS)

Протокол SMB (Блок сообщений сервера) — это аналог NFS из мира Windows. Обычно пользователь обращается к сетевым каталогам Windows прямо из файлового менеджера с локального компьютера.

В качестве альтернативы можно использовать файловую систему cifs или ее устаревший вариант smbfs, чтобы подключать сетевые каталоги Windows к локальному дереву каталогов подобно тому, как это делается в NFS. Кроме того, вы имеете доступ ко всем файлам, как если бы они были расположены в вашей файловой системе. Это удобно и работает хорошо, пока каталог доступен в сети.

Интеграция сетевых каталогов Windows в файловую систему — статический процесс. Метод, описанный здесь, целесообразен лишь в том случае, когда сетевая конфигурация остается неизменной в течение достаточно долгого времени. Если же, наоборот, к сети все время подключаются одни компьютеры и от нее отключаются другие, причем меняются их имена и IP-адреса, то лучше обращаться к сетевым каталогам динамически, через файловый менеджер.

Обратите внимание, что в любом случае придется установить Samba. Это комплект программ, предназначенных для совместного использования в сети каталогов, файлов, принтеров и т. д. по протоколу SMB. Во многих дистрибутивах Samba разделен на две части: клиентскую и серверную. Пока вы работаете так, как описано в этом разделе — только обращаетесь к сетевым каталогам, но не собираетесь предоставлять доступ к тем или иным каталогам своего компьютера через сеть, достаточно установить клиентские пакеты. О том, как конфигурировать Samba для работы с сервером, рассказано в главе 20.

Для доступа к сетевым каталогам Windows необходимо, чтобы протокол SMB, работающий между двумя компьютерами, не блокировался брандмауэром!

Сравнение smbfs и cifs

В Linux существует две файловые системы, предназначенные для работы с сетевыми каталогами Windows: SMBFS (Файловая система SMB) и более новая система CIFS (Общая межсетевая файловая система). Сетевая система CIFS является усовершенствованной разработкой Microsoft, поэтому рекомендую остановиться именно на ней. Например, совместимые с CIFS Samba-сервера передают информацию о правах доступа к файлам, совместимым с UNIX/Linux, а при работе с SMBFS этого не происходит.

Команда mount

Чтобы интегрировать сетевые каталоги Windows в свою локальную файловую систему, воспользуйтесь командой `mount`. С помощью параметра `-t` команды `mount` вы можете задать файловую систему, с которой собираетесь работать. Большая часть остальных параметров в обеих файловых системах одинакова, но для cifs существуют некоторые дополнительные параметры. Внутри системы команда `mount` выполняет `mount.cifs` или `smbmount`. Подробности обо всех параметрах сообщаются в `man mount.cifs`, а также в `man smbmount`.

Если хотите подключить внешний каталог, введите две следующие команды (в зависимости от того, открывается ли доступ к Windows на основе имени пользователя).

```
root# mount -t cifs //venus/myshare /media/winshare
root# mount -t cifs -o username=name //venus/myshare /media/winshare
```

В результате открытый каталог с компьютера venus будет подключен к файловой системе Linux под именем myshare. Файлы из открытого каталога будут предоставлены в `/media/winshare`. Разумеется, этот каталог должен существовать еще до выполнения `mount`. Перед выполнением команды система запросит у вас пароль. Вы можете указать пароль и напрямую:

```
root# mount -t cifs -o username=name,password=pw //venus/myshare /media/winshare
```

СОВЕТ

Если вам часто приходится вручную подключать и отключать каталоги, попробуйте поработать с программой KDE Smb4k. Она отображает все серверы Windows и Samba, доступные в сети, и облегчает интеграцию отдельных каталогов в дерево каталогов локальной системы. Более подробно эта программа описана на сайте http://smb4k.berlios.de/.

Файл /etc/fstab

Чтобы автоматически подключить сетевые каталоги к файловой системе при запуске компьютера, дополните `/etc/fstab` строкой, показанной в следующем примере. Подробная информация о значении и конфигурации `/etc/fstab` находится в разделе 13.5.

```
# Дополнение в /etc/fstab
//venus/myshare   /media/winshare  cifs  defaults  0 0
```

В большинстве дистрибутивов все каталоги CIFS, упомянутые в `/etc/fstab`, подключаются к системе при выполнении процесса Init-V. Исключением, как и в случае с NFS, является SUSE: здесь за аналогичный процесс отвечает сценарий Init-V, который необходимо специально активизировать:

```
root# insserv smbfs
```

В вышеуказанной форме операция подключения каталогов работает только тогда, когда не требуется аутентификации на сервере Windows или Samba. Если, наоборот, аутентификация требуется, создайте файл `/etc/.winshare-pw`, в котором будет содержаться логин и пароль для вашего каталога. Файл построен так:

```
username=name
password=xxxx
```

Если хранить пароль в незашифрованном виде, то сразу возникает риск, что пароль узнают злоумышленники. Следующая команда ограничивает доступ к файлу, позволяя немного снизить риск. Она дает возможность читать и изменять файл с паролем только администратору:

```
root# chmod go-rw /etc/.winshare-pw
```

Кроме того, добавьте к записи fstab параметр credentials. При подключении каталога будут считываться файлы аутентификации из .winshare-pw.

```
# Дополнение в /etc/fstab
//venus/myshare  /media/winshare  cifs  credentials=/etc/.winshare-pw  0 0
```

13.13. Разделы и файлы подкачки

Если оперативной памяти недостаточно для выполнения всех программ и в вашем распоряжении есть файлы и разделы подкачки, Linux задействует их в качестве дополнительной памяти *(страничная организация памяти)*. Таким образом, Linux может использовать больше памяти, чем доступно в RAM.

Обычно раздел подкачки создается при установке. Чтобы проверить, есть ли в вашем распоряжении виртуальная память и если есть, то сколько (имеется или действительно применяется), выполните команду free. В следующем примере имеем 1519 Мбайт RAM и 2000 Мбайт виртуальной памяти. В настоящее время, согласно примеру, используется 401 Мбайт RAM для работы с программами и данными, остаток используется как файловый буфер (кэш). Виртуальная память пока не используется.

```
root# free -m
             total       used       free     shared    buffers     cached
Mem:          1519       1479         39          0         67       1010
-/+ buffers/cache:        401       1117
Swap:         2000          0       2000
```

Если компьютер работает длительное время, ему рано или поздно придется использовать виртуальную память, даже если у вас в распоряжении еще достаточно RAM. Причина проста: ядро управляет кэшем для предоставления доступа к файлам «для чтения»; если позже понадобится какой-либо файл, его можно считать из кэша. Как только кэш превышает по размеру свободную оперативную память, Linux выгружает блоки памяти, которые давно не использовались, в раздел подкачки. Это совсем не означает, что осталось мало памяти. Linux просто пытается использовать имеющуюся память как можно более эффективно.

Файл /etc/fstab

Далее показаны две записи о разделах подкачки; эти записи находятся в файле /etc/fstab. Благодаря параметру pri оба раздела Linux используются равномерно. Это повышает скорость системы (как при работе с распределением данных (striping) или RAID-0, см. раздел 13.14), если разделы находятся на двух взаимно независимых жестких дисках. Если имеется только один раздел подкачки, укажите defaults вместо pri=0.

```
# /etc/fstab: разделы подкачки
/dev/sda9  swap   swap   pri=1   0  0
/dev/sdc7  swap   swap   pri=1   0  0
```

Управление работой виртуальной памяти

Если памяти в RAM начинает не хватать, ядро Linux использует достаточно сложный алгоритм, позволяющий решить, следует ли открыть доступ к кэш-памяти для решения новых задач либо лучше выгрузить в раздел подкачки последние неиспользованные области памяти. Параметр ядра `/proc/sys/vm/swappiness` позволяет вам указать, что должно делать ядро в случае выбора — уменьшить кэш или выгрузить данные (что такое параметры ядра и как их настраивать, рассказано в разделе 15.4).

Стандартная настройка для `swappiness` равна `60`, диапазон возможных значений: от 0 до 100. Значение `0` означает, что ядро по возможности не будет использовать *постраничную организацию памяти*; `100` говорит о том, что области памяти, не используемые долгое время, будут отправляться в раздел подкачки как можно раньше. Более подробно параметр `swappiness` описан на следующих сайтах:

- http://lwn.net/Articles/83588/;
- http://kerneltrap.org/node/3000.

На практике вы заметите работу разделов подкачки только тогда, когда оставите компьютер работать на всю ночь и за это время программа обратится к большому количеству файлов на диске (это может быть, например, сценарий для резервного копирования или программа для составления поискового индекса). Поскольку обращений к файлам много, кэш значительно увеличивается. Если в системе действует настройка `swappiness=60` (или выше), то ядро часами будет выгружать неиспользуемую память. Например, это может касаться страниц виртуальной памяти OpenOffice или Gimp. Если на следующий день вы захотите продолжить работу с OpenOffice, потребуется пара секунд, чтобы все эти страницы вернулись из раздела подкачки обратно в оперативную память. Если установить значение `swappiness=0`, то ждать в таком случае не придется.

Сколько нужно виртуальной памяти

Раньше часто можно было встретить рекомендацию оставлять на виртуальную память примерно в два раза больше места, чем есть в оперативной. Однако с ростом объемов памяти это практическое правило все реже срабатывает. Если вы работаете с Linux в основном на локальном компьютере, вам вполне хватит и относительно небольшого раздела подкачки (например, раздел подкачки размером 512 Мбайт при оперативной памяти около 2 Гбайт).

Особый случай представляют собой ноутбуки, которые вы, возможно, захотите перевести в спящий режим (однако у меня это получалось плохо). В таком случае вся оперативная память сохраняется в разделе подкачки и ноутбук переходит в спящий режим. Для этого, разумеется, необходимо, чтобы раздел подкачки был больше, чем вся оперативная память (например, в полтора раза).

Специфические требования предъявляются и к крупным серверным системам. Например, Oracle для работы со своим сервером базы данных (версия 10.2) при

расчете размеров раздела подкачки рекомендует использовать различные коэффициенты — в зависимости от доступного объема RAM:

- до 2 Гбайт — коэффициент 2;
- 2–8 Гбайт — коэффициент 1;
- более 8 Гбайт — коэффициент 0,75.

В 32-битных системах максимальный размер раздела подкачки составляет 2 Гбайта. Если вам требуется больше виртуальной памяти, можно использовать и несколько разделов подкачки. В таком случае гораздо целесообразнее перейти к работе с 64-битным дистрибутивом.

Время от времени возникает вопрос: можно ли (а, возможно, и следует ли) отказаться от раздела подкачки, если на компьютере установлена очень большая оперативная память. В принципе Linux может работать и без виртуальной памяти, но есть и важный аргумент в пользу создания раздела подкачки: если какая-то программа выйдет из-под контроля или по каким-то причинам потребует больше памяти, чем ожидалось, доступная память когда-нибудь будет израсходована. Это может привести к аварийному завершению следующего процесса, на который потребуется дополнительная память. Это может быть любой процесс — необязательно программа, вышедшая из-под контроля.

Строго говоря, раздел подкачки не решает этой проблемы, ведь и виртуальная память когда-нибудь закончится. Но из-за нарастающей загруженности виртуальной памяти программы в таком случае будут работать все медленнее и медленнее и вы заблаговременно поймете, что с компьютером что-то неладно. Прежде чем система откажет, вы успеете завершить программу, вызвавшую ошибку, командой `kill`. Если вы желаете подробнее изучить эту тему, обратитесь к одному из следующих сайтов:

- http://www.thomashertweck.de/linuxram.html;
- http://kerneltrap.org/node/3202.

Создание нового раздела подкачки

Если вы создали слишком маленький раздел подкачки или вам по каким-либо причинам понадобится еще один, создайте новый раздел (например, командой `fdisk`). В качестве типа диска укажите *Linux swap* (`82` в `fdisk`). Отформатировав раздел с помощью команды `mkswap`, вы можете активизировать его командой `swapon`. Если все получится, дополните файл `etc/fstab`.

Чтобы не замедлять работу системы, старайтесь создавать на жестком диске только один раздел подкачки. В идеальном случае такой раздел должен находиться на малоиспользуемом или неиспользуемом диске.

Файлы подкачки

Для того чтобы не использовать раздел подкачки, вы можете выгружать память в отдельный файл подкачки. Конечно, это решение на крайний случай, так как

из-за такой операции замедляется доступ к файловой системе. Единственное достоинство файла подкачки заключается в том, что для него не требуется отдельный раздел.

Обычно файлы подкачки сохраняются в разделе /dev. В первую очередь создаем командой dd пустой файл заданного размера. При этом в качестве источника данных используется /dev/zero. Из этого устройства можно считать сколько угодно нулевых байт. Размер нужно указывать в блоках, причем каждый блок равен 1024 байт. Кроме того, файл подкачки, подобно разделу подкачки, форматируется командой mkswap и активизируется с помощью swapon. В следующем примере мы создадим маленький файл подкачки размером около 1 Мбайт:

```
root#  dd bs=1024 if=/dev/zero of=/swapfile count=1000
1000+0 records in
1000+0 records out
root#  mkswap /swapfile 1000
Setting up swapspace, size = 1019904 bytes
root#  sync
root#  swapon -v /swapfile
swapon on device /swapfile
```

Файлы подкачки можно указать в fstab, как и разделы подкачки:

```
# Дополнение в /etc/fstab
/swapfile  none    swap   sw   0 0
```

13.14. RAID

RAID — избыточный массив независимых жестких дисков. В этом разделе мы рассмотрим администрирование программного RAID-массива для Linux на базе пакета mdadm. Ради экономии места здесь будут описаны только уровни RAID-1 и 0.

Обратите внимание — в Интернете есть много устаревших руководств по RAID. В них описывается конфигурация, основанная на raidtools (этот инструментарий в современных дистрибутивах Linux уже не используется).

Основы

Пакет mdadm. Если вы создали группу дисков RAID уже в ходе инсталляции, установите пакет mdadm. В нем содержится одноименная команда для администрирования RAID.

Рекомендуется пользоваться mdadm и при установке почтового сервера (MTA — агент пересылки сообщений), чтобы в случае проблем с RAID система сообщила об этом администратору по электронной почте. Если вы еще не занимались темой почтового сервера, пока не устанавливайте такой агент. Вместо этого (в Debian и Ubuntu) при установке укажите с помощью apt-get параметр --no-install-recommends.

md_mod. Внутри Linux за работу программного RAID отвечает драйвер поддержки многодисковых устройств (Multi Devices Driver Support). В некоторых дистрибутивах этот драйвер интегрирован прямо в ядро, в других случаях при

запуске системы автоматически загружается модуль md_mod (раньше он назывался просто md). В любом случае в dmesg должны содержаться соответствующие сообщения. Убедитесь, что существует псевдофайл /proc/mdstat. В нем хранится информация о текущем состоянии системы RAID.

Модуль md_mod создает логический уровень между драйвером, отвечающим за доступ к жесткому диску, и драйвером файловой системы (например, ext4). Этот модуль создает на нескольких разделах диска новое логическое устройство, к которому может обращаться драйвер файловой системы (/dev/mdn). После завершения конфигурации RAID вы используете не определенный раздел диска, а раздел /dev/mdn, где создаете свою файловую систему.

mdadm.conf. Основным конфигурационным файлом RAID является /etc/mdadm/mdadm.conf. В этом файле кроме некоторых глобальных настроек RAID должны содержаться данные обо всех активных группах дисков RAID. Можно также с помощью /usr/share/mdadm/mkconf создать полностью новый конфигурационный файл. Это целесообразно делать в тех случаях, когда такой файл потерялся либо вы работаете с живым или восстановительным диском.

При такой конфигурации используется несколько необычный метод. Сначала, выполнив команду mdadm, вы создаете желаемые группы RAID или модифицируете их. Кроме того, вы дополняете имеющийся файл mdadm.conf в соответствии с действующей конфигурацией. Чтобы узнать основные показатели действующих групп RAID, выполните команду mdadm --examine -scan, а потом вставьте эти показатели с помощью >> в имеющийся конфигурационный файл.

```
root# mdadm --examine --scan >> /etc/mdadm/mdadm.conf
```

Если в файле mdadm.conf уже содержатся определенные ранее группы RAID, удалите эти сведения с помощью текстового редактора, чтобы какая-нибудь группа RAID не оказалась определена дважды. В следующих строках показано, как может быть построен файл mdadm.conf.

```
# Файл /etc/mdadm/mdadm.conf
DEVICE partitions
CREATE owner=root group=disk mode=0660 auto=yes
HOMEHOST <system>
MAILADDR root
ARRAY /dev/md0 level=raid1 num-devices=2 UUID=36c426b0:...
ARRAY /dev/md1 level=raid1 num-devices=2 UUID=71dfc474:...
ARRAY /dev/md2 level=raid1 num-devices=2 UUID=e0f65ea0:...
```

Статус. Актуальная информация о статусе RAID содержится в упоминавшемся выше файле /proc/mdstat. В следующем примере мы имеем три группы RAID-1, каждая из которых состоит из двух разделов. Все три группы активны и работают без ошибок: [UU] означает, что первый и второй раздел находятся в состоянии up (то есть без проблем).

```
root# cat /proc/mdstat
Personalities : [raid0] [raid1] [linear] [multipath]
                [raid6] [raid5] [raid4] [raid10]
md0 : active raid1 sda1[0] sdb1[1]
```

```
      979840 blocks [2/2] [UU]
md1 : active raid1 sda2[0] sdb2[1]
      1951808 blocks [2/2] [UU]
md2 : active raid1 sda3[0] sdb3[1]
      387730624 blocks [2/2] [UU]
unused devices: <none>
```

Разумеется, было бы неудобно постоянно заглядывать в этот файл и проверять, все ли в порядке. Гораздо лучше перепоручить эту задачу программе `mdadm --monitor`. Обычно она запускается сценарием Init-V `/etc/init.d/mdadm`. В зависимости от дистрибутива может потребоваться сначала соответствующим образом сконфигурировать пакет `mdadm`. В Ubuntu, например, для этого нужно выполнить следующую команду:

```
root# dpkg-reconfigure mdadm
```

В процессе конфигурации `mdadm` система отобразит четыре диалоговых окна. На первом этапе можно активизировать автоматическую проверку избыточности — один раз в месяц (в 1:06 в первое воскресенье месяца данные разделов RAID сравниваются друг с другом). Такой контроль помогает заблаговременно обнаружить ошибки в тех сегментах или файлах, которые уже давно не читались и не изменялись. Внутри системы проверка избыточности осуществляется с помощью команды `checkarray`, запускаемой сценарием Cron `/etc/cron.d/mdadm`.

На втором и третьем этапе вы активизируете наблюдение за состоянием RAID и указываете, на какой электронный адрес посылать предупреждения либо сообщения о возникновении ошибок. Внутри системы такое наблюдение осуществляется с помощью команды `mdadm --monitor`. Она выполняется при запуске системы с помощью `/etc/init.d/mdadm`, если в `/etc/default/mdadm` есть настройка `START_DAEMON=true`. Электронный адрес сохраняется в файле `/etc/mdadm/mdadm.conf`. Если возникнет проблема, программа `mdadm` отошлет сообщение с уведомлением на адрес администратора. Чтобы этот механизм работал, на компьютере необходимо установить почтовый агент (MTA)! Нужный электронный адрес вы можете настроить в файле `/etc/mdadm/mdadm.conf` с помощью переменной `MAILADDR`.

На четвертом этапе вы, наконец, указываете, должен ли сервер при перегрузке запускаться и в том случае, когда он обнаруживает в разделе RAID ошибку. Запускаться в таком случае должны прежде всего корневые серверы.

GRUB и RAID. Версия GRUB 0.97 совместима только с RAID-1. Если вы работаете с GRUB 0.97, но для системного раздела выбираете другой уровень RAID, вам потребуется отдельный загрузочный раздел. С GRUB 2 ситуация обстоит лучше: если в `grub.cfg` загружен модуль raid, то GRUB может считать данные ядра и файла `Initrd` прямо из группы RAID и отдельного загрузочного раздела для этого не требуется.

Для того чтобы компьютер, имеющий на диске системный раздел RAID, загружался и в том случае, когда жесткий диск неисправен, GRUB нужно установить в загрузочном секторе каждого жесткого диска. При работе с GRUB 2 для этого просто выполняется команда `grub-install /dev/mdn`. В GRUB 0.97 установка на каждом диске производится вручную.

Администрирование

Создание группы RAID-0

Чтобы создать группу RAID-0, вам потребуется как минимум два неиспользуемых раздела. Лучше, если эти разделы будут одинаковыми по размеру, но это необязательно. В зависимости от уровня RAID при неравном размере разделов скорость работы может быть не оптимальной или же части большого раздела могут оставаться незаполненными.

Выбранные разделы должны быть обозначены как разделы RAID. Если для секционирования диска использовалась программа `fdisk`, то для идентификационного номера раздела необходимо установить шестнадцатеричное значение `fd`, используя команду `T`. При работе с программой `parted` выполняется команда `set` *`номер_раздела`* `raid on`.

Далее разделы с названиями устройств `/dev/sda3` и `/dev/sdc1` объединяются в систему RAID-0. Форматировать эти разделы не требуется. Команда `fdisk -l` отображает примерную конфигурацию:

```
root# fdisk -l /dev/sda /dev/sdc
Disk /dev/sda: 320.0 GB, 320072933376 bytes
  Device Boot    Start   End    Blocks   Id    System
/dev/sda1            1   973   7815591   83    Linux
/dev/sda2          974  1034    489982+  82    Linux swap / Solaris
/dev/sda3         1035  2251   9775552+  fd    Linux raid autodetect
Disk /dev/sdc: 320.0 GB, 320072933376 bytes
  Device Boot    Start   End    Blocks   Id    System
/dev/sdc1            1  1217   9775521   fd    Linux raid autodetect
```

Всего одной команды `mdadm` достаточно для того, чтобы создать группу RAID-0 из двух разделов — `/dev/sda3` и `/dev/sdc1`.

```
root# mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sda3 /dev/sdc1
mdadm: array /dev/md0 started.
```

Затем требуется поместить файловую систему в новый виртуальный раздел `/dev/md0`. Его можно подключить к файловой системе Linux командой `mount`. Раздел запрашивается через каталог `/striped`, но, разумеется, вы можете его переименовать.

```
root# mkfs.ext4 /dev/md0
root# mkdir /striped
root# mount /dev/md0 /striped/
```

Если все получится, нужно занести информацию о новом разделе в файл `/etc/fstab`. Во всех новых дистрибутивах Linux система RAID автоматически инициализируется процессом Init-V при перезапуске.

```
# in /etc/fstab
/dev/md0    /striped   ext4   defaults   0  0
/
```

Кроме того, необходимо дополнить конфигурационный файл одной строкой mdadm.conf, описывающей новую группу RAID-0. Команда mdadm --examine --scan возвращает строку в синтаксически правильном виде.

Создание группы RAID-1

Группа RAID-1 создается тем же способом, что и RAID-0. Только команда для создания системы RAID выглядит несколько иначе и содержит --level=1 вместо --level=0:

```
root# mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sda3 /dev/sdc1
mdadm: array /dev/md0 started.
root# mkfs.ext4 /dev/md0
```

Если вы уже создали в /dev/md0 раздел RAID-0, нужно отключить раздел от дерева каталогов и деактивизировать командой mdadm -stop. Только после этого вы сможете выполнить команду mdadm --create. Тем не менее mdadm распознает, что разделы /dev/sda3 и /dev/sdc1 были использованы, и потребует подтвердить, что вы действительно хотите заново настроить /dev/md0.

Тестирование группы RAID-1

Чтобы протестировать, как действует группа RAID-1, еще до того как вы сохраните там важные данные, пометьте этот раздел как неисправный:

```
root# mdadm /dev/md0 --fail /dev/sdc1
```

Если при старте системы была запущена команда mdadm --monitor, то администратор, работающий на локальном компьютере, сразу же получит по электронной почте соответствующее уведомление. После этого вы сможете работать с группой как и раньше, но теперь все изменения будут сохраняться на свободном пространстве, оставшемся на диске. Теперь /proc/mdstat отображает статус U_. Это говорит о том, что один раздел работает (U означает up), а один отсутствует (_).

```
root# cat /proc/mdstat
md0 : active raid1 sda3[1]
      979840 blocks [2/1] [_U]
```

Чтобы снова добавить /dev/sdc1 к /dev/md0, нужно специально удалить его как неисправный.

```
root# mdadm --remove /dev/md0 /dev/sdc1
root# mdadm --add /dev/md0 /dev/sdc1
```

Теперь автоматически начнется повторная синхронизация обеих разделов, на что в зависимости от размера разделов может понадобиться достаточно много времени (ориентировочно около 20 минут на 100 Гбайт). Правда, в ходе синхронизации вы можете продолжать работу. Файловая система будет работать несколько медленнее.

```
root# cat /proc/mdstat
md0 : active raid1 sda3[1] sdc1[2]
      485454656 blocks [2/1] [U_]
```

```
      [>....................] recovery = 3.0% (14577856/485454656)
                             finish=72.8min speed=107724K/sec
root# mdadm --detail /dev/md0   (Пока синхронизация продолжается)
...
          State : clean, degraded, recovering
 Active Devices : 1
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 1
 Rebuild Status : 75% complete
...
  Number    Major    Minor    RaidDevice    State
    0         3        3         0          active sync       /dev/sda3
    1         0        0         -          removed
    2        22        2         1          spare rebuilding  /dev/sdc1
root# mdadm --detail /dev/md0  (После завершения синхронизации)
...
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0
...
  Number    Major    Minor    RaidDevice    State
    0         3        3         0          active sync   /dev/sda3
    1        22        2         1          active sync   /dev/sdc1
```

Замена неисправного жесткого диска из RAID-1

К счастью, это бывает редко, но если на жестком диске действительно появляются неисправности и mdadm пометит отдельные разделы как дефектные, все разделы этого диска необходимо специально удалить из соответствующих групп RAID.

```
root# mdadm --remove /dev/md0 /dev/sdc1
...
```

Кроме того, вам нужно как можно скорее заменить диск. На новом жестком диске должно быть достаточно места, чтобы создать разделы такого же размера, как и на имеющихся жестких дисках.

Будьте внимательны — из группы нужно извлечь именно неисправный диск, а не рабочий! Этот совет кажется банальным, но если на компьютере два или более конструктивно одинаковых диска, найти нужный диск не так просто, как кажется. Уникальным признаком жесткого диска является только его серийный номер! Чтобы узнать, какой серийный номер какому названию устройства соответствует, используйте команды hdparm или smartctl. Для выполнения обеих команд необходимо установить одноименные пакеты.

```
root# smartctl -i /dev/sdc
...
Device Model:  SAMSUNG HD403LJ
Serial Number: S0NFJ1MPA07356
```

```
root# hdparm -i /dev/sdc
/dev/sdb:
...
Model=SAMSUNG HD403LJ, FwRev=CT100-12,
SerialNo=S0NFJ1MPA07356
```

После замены жесткого диска вам следует создать на новом диске разделы, которые будут не меньше, чем уже имеющиеся разделы RAID. При этом вам очень пригодится команда `sfdisk` (см. подраздел «sfdisk» раздела 13.3). Разделы необходимо пометить как относящиеся к RAID (шестнадцатеричный идентификационный код `fd`). Когда эта подготовительная работа будет завершена, останется добавить разделы нового жесткого диска к группам RAID:

```
root# mdadm --add /dev/md0 /dev/sdc1
...
```

Теперь ядро начнет синхронизировать разделы нового жесткого диска с имеющимися данными RAID. Статус синхронизации отслеживается с помощью команды `cat /proc/mdstat`.

СОВЕТ

Настоятельно рекомендую вам поучиться исправлять RAID на тестовой системе, никуда не торопясь. Дефект жесткого диска можно сымитировать, пометив диск как неисправный командой mdadm --fail или на время отключив диск от сети (конечно же, не на ходу!).

Деактивизация группы RAID

Команда `mdadm --stop` деактивизирует группу RAID. Предварительно нужно отключить от дерева каталогов файловую систему этой группы с помощью команды `umount`.

```
root# umount /mount-verzeichnis/
root# mdadm --stop /dev/md0
```

Повторная активизация группы RAID

Теперь, если после выполнения `mdadm --stop` вы не внесли в разделы группы никаких изменений, можно снова собрать и активизировать группу RAID командой `mdadm --assemble` — потери данных исключаются.

```
root# mdadm --assemble /dev/md0 /dev/sda3 /dev/sdc1
mdadm: /dev/md0 has been started with 2 drives.
```

Анализ разделов

Во всех разделах жесткого диска, которые вы объединили в группы RAID с помощью `mdadm`, в специальных блоках сохраняется контекстная информация (метаданные). Эту информацию можно считать с помощью команды `mdadm --query`, например чтобы узнать статус неизвестной системы.

```
root# mdadm --query /dev/sda3
/dev/sda3: is not an md array
/dev/sda3: device 0 in 2 device active raid1 md0. Use mdadm --examine
```

```
  for more detail.
root# mdadm --query /dev/md0
/dev/md0: 9.32GiB raid1 2 devices, 0 spares. Use mdadm --detail for more detail.
/dev/md0: No md super block found, not an md component.
```

Команда `mdadm --examine` возвращает подробную информацию о разделе, входящем в группу RAID:

```
root# mdadm --examine /dev/sda3
/dev/sda3:
          Magic : a92b4efc
        Version : 00.90.03
           UUID : ae4e4fbb:aaaf9d27:008c228b:0d7abb31
  Creation Time : Thu Nov 9 16:55:35 2006
     Raid Level : raid1
   Raid Devices : 2
  Total Devices : 2
Preferred Minor : 0
    Update Time : Thu Nov 9 17:02:39 2006
          State : clean
 Active Devices : 2
Working Devices : 2
  Number    Major    Minor    RaidDevice    State
0   0         3        3          0         active sync   /dev/sda3
1   1        22        1          1         active sync   /dev/sdc1
```

Аналогично `mdadm --detail` выдает подробную информацию о группе RAID:

```
root# mdadm --detail /dev/md0
/dev/md0:
      Version : 00.90.03
Creation Time : Thu Nov 9 16:55:35 2006
   Raid Level : raid1
   Array Size : 9775424 (9.32 GiB 10.01 GB)
  Device Size : 9775424 (9.32 GiB 10.01 GB)
 Raid Devices : 2
Total Devices : 2
…
```

Удаление метаданных RAID

Как правило, бывает полезно сохранять метаданные RAID в неиспользуемых секторах раздела. Однако если позже вы пожелаете задействовать этот жесткий диск для других целей, метаданные RAID могут представлять проблему: установочные программы Linux и `mdadm` распознают остатки конфигурации RAID и ни за что «не признают», что теперь эти разделы нужно использовать иначе. Вам пригодится следующая команда, которую необходимо применить ко всем разделам RAID:

```
root# mdadm --zero-superblock /dev/sda3
```

Если вы пробовали работать с BIOS-RAID, можете удалить соответствующие метаданные на всех жестких дисках с помощью команды `dmraid -r -E`.

13.15. Менеджер логических томов (LVM)

Менеджер логических томов — это логический слой, расположенный между файловой системой и разделами диска. Принцип, приоритеты и порой совершенно неясная номенклатура LVM здесь рассматриваться не будут. Поговорим об администрировании LVM.

Помощь при конфигурации

В некоторых дистрибутивах предоставляются инструменты для администрирования LVM, не требующие остановки системы. В Fedora и Red Hat при конфигурации используется `system-config-lvm`, в SUSE — модуль YaST Система ▸ LVM.

Хотя эти программы и помогают при конфигурации, для работы с ними требуется глубоко понимать концепции, связанные с LVM. Обратите внимание, что при изменении размера, как правило, увеличиваются или уменьшаются только сами логические тома, а не содержащиеся в них файловые системы. Размер файловой системы необходимо изменять *до* уменьшения и *после* увеличения соответствующих логических томов.

Модуль dm_mod. Внутри системы за управление LVM отвечает модуль ядра `dm_mod`. В некоторых дистрибутивах функции LVM скомпилированы прямо внутри ядра, поэтому не отображаются при выводе результата команды `lsmod`.

GRUB. Если LVM создан уже при установке системы, то системный раздел также может находиться в логическом томе. В любом случае с LVM совместима только версия GRUB 2. Если вы работаете с GRUB 0.97, вам для этого потребуется отдельный загрузочный раздел, без LVM.

RAID. Можно комбинировать LVM и RAID. Обычно для этого создается группа RAID, а затем полученное устройство `/dev/mdn` используется в качестве физического тома (PV).

Особый случай представляет собой RAID-0. Этот вариант RAID поддерживается непосредственно LVM. Чтобы пользоваться данной функцией, вам потребуется создать на одном или нескольких жестких дисках по физическому тому. Эти тома объединяются в группу томов (Volume Group). Теперь с помощью команды `lvcreate -i` *n* вы можете создать логический том, данные которого будут разделены между несколькими (*n*) физическими томами.

Команды

Для администрирования LVM применяется целый спектр команд. Их названия начинаются с `pv`, `vg` или `lv` в зависимости от того, для работы с чем они предназначены — с *физическими томами* (`pv`), *группами томов* (`vg`) или *логическими томами* (`lv`). Важнейшие команды перечислены в табл. 13.10. Они входят в состав пакета `lvm2`, который сначала нужно установить.

Таблица 13.10. Обзор команд для работы с LVM

Команда	Функция
lvcreate	Создает в группе томов новый логический том
lvdisplay	Сообщает подробную информацию по определенному логическому тому
lvextend	Увеличивает логический том
lvreduce	Уменьшает логический том
lvremove	Удаляет логический том
lvrename	Переименовывает логический том
lvscan	Перечисляет логические тома
pvcreate	Обозначает раздел или устройство как физический том
pvdisplay	Сообщает подробную информацию по физическому тому
pvremove	Удаляет обозначение «физический том» неиспользуемого физического тома
pvscan	Перечисляет физические тома
vgchange	Изменяет атрибуты группы томов
vgcreate	Создает новую группу томов из одного или нескольких физических томов
vgdisplay	Сообщает подробную информацию о группе томов
vgextend	Увеличивает группу томов на один том
vgmerge	Объединяет две группы томов
vgreduce	Уменьшает группу томов, убирая из нее неиспользуемый том
vgrename	Переименовывает группу томов
vgscan	Перечисляет все группы томов

Примеры

В следующих примерах показано, как применять некоторые команды LVM. При этом предполагается, что при установке LVM не создавался. Теперь дополнительный жесткий диск `/dev/sdc` должен использоваться через LVM. Секционирование диска выглядит так:

```
root# fdisk -l /dev/sdc
Disk /dev/sdc: 320.0 GB, 320072933376 bytes
  Device Boot    Start    End     Blocks    Id    System
  /dev/sdc1          1   1217    9775521    8e    Linux LVM
  /dev/sdc2       1218   2434    9775552+   8e    Linux LVM
```

Чтобы инициализировать LVM, выполните команды `modprobe` и `vgscan`. Как только система LVM будет создана, модуль ядра LVM автоматически выполнится при запуске компьютера. Иначе говоря, систему требуется инициализировать вручную только в первый раз:

```
root# modprobe dm_mod
root# vgscan
  Считывание всех физических томов (подождите, пожалуйста...)
  Группы томов не найдены
```

По методическим соображениям мы сначала создадим LVM в разделе `/dev/sdc1`, а потом добавим к нему `/dev/sdc2`. Если и так ясно, что вы собираетесь использовать для LVM весь жесткий диск, гораздо проще обозначить сам диск или как минимум

его раздел максимального размера с помощью команды pvcreate как предназначенный для LVM.

```
root# pvcreate /dev/sdc1
  Физический том "/dev/sdc1" успешно создан
```

Теперь нужно объединить все физические тома в группу. В этом примере у нас сначала есть только один физический том, но выполнить такой шаг необходимо. Команде vgcreate также нужно сообщить желаемое название группы томов. В этом примере группа томов получает название myvg1:

```
root# vgcreate myvg1 /dev/sdc1
  Группа томов "myvg1" успешно создана
```

Теперь myvg1 представляет собой своего рода пул данных, который, правда, пока не прменяется. Для его использования вам нужно будет создать в myvg1 логический том, то есть своего рода виртуальный раздел. При этом необходимо передать команде lvcreate три параметра: желаемый размер и название логического тома, а также название существующей группы томов:

```
root# lvcreate -L 2G -n myvol1 myvg1
  Создан логический том "lvol0"
```

Одновременно команда создает файл /dev/myvg1/myvol1. При этом используется ссылка на файл /dev/mapper/myvg1-myvol1. Теперь логический том можно применять под любым из обоих названий устройств как обычный раздел жесткого диска.

Чтобы создать файловую систему в *логическом томе*, используйте, например, команду mkfs.ext4 или mkfs.xfs:

```
root# mkfs.ext4 /dev/myvg1/myvol1
```

С помощью команды mount можете проверить, все ли получилось:

```
root# mkdir /test
root# mount /dev/myvg1/myvol1 /test
```

Причина, по которой стоит пользоваться логическими томами, заключается в том, что появляется возможность постепенно увеличивать файловую систему, не секционируя диск заново. В следующем примере созданная выше файловая система (dev/myvg1/myvol1 посредством /test) увеличивается с 2 до 3 Гбайт. Команда df позволяет узнать мощность /test перед внесением изменений:

```
root# df -h -T /test
Filesystem   Typ   Size   Used   Available   Used%   Mounted on
/dev/mapper/myvg1-myvol1
             ext4  2,0G   760M        1,2G     40%   /test
```

Для этого сначала нужно увеличить логический том. Чтобы это сделать, сообщим название устройства и новый размер команде lvextend. Дополнительно следует соответствующим образом увеличить и файловую систему ext4.

```
root# lvextend -L 3G /dev/myvg1/myvol1
  Увеличение логического тома myvol1 до 3,00 GB
  Размер логического тома myvol1 успешно изменен
root# resize2fs /dev/myvg1/myvol1
```

Команда df доказывает, что все сработало:

```
root# df -h -T /test
File system   Typ   Size   Used   Available   Used%   Mounted on
/dev/mapper/myvg1-myvol1
              ext4  3,0G   760M   2,1G          27%   /test
```

В принципе логический том можно и уменьшить. Для этого нужно сначала обязательно отключить ту или иную файловую систему от дерева каталогов, проверить с помощью fsck.ext4 и, наконец, уменьшить командой resize2fs. Только теперь можно уменьшить лежащий в основе логический том командой lvreduce.

Пока в пуле памяти (*группы томов*) еще есть место, виртуальные разделы (логические тома) можно с легкостью увеличить. Но что делать, если группа томов заполнена? В таком случае на любом жестком диске вашего компьютера нужно создать новый раздел, пометить этот раздел как *физический том* и добавить его в группу томов командой vgextend.

На примере двух следующих команд показано, как этот механизм работает с разделом /dev/sdc2. Таким образом, общий объем myvg1 достигает около 19 Гбайт, из которых почти 16 Гбайт свободно:

```
root# pvcreate /dev/sdc2
  Физический том "/dev/sdc2" успешно создан
root# vgextend myvg1 /dev/sdc2
  Группа томов "myvg1" успешно расширена
root# vgdisplay myvg1
...
  Размер группы томов         18,64 GB
  Распределено PE / Размер    640 / 2,50 GB
  Свободно PE / Размер        4132 / 16,14 GB
...
```

13.16. SMART

Аббревиатура SMART означает *«Система слежения, анализа и отчетности»* и используется почти со всеми имеющимися на рынке жесткими дисками IDE, SATA и SCSI. Благодаря SMART на диске периодически сохраняются различные параметры, которые позволяют сделать вывод о потенциальных неисправностях жесткого диска, а также о том, сколько он еще проработает. Через специальный интерфейс параметры SMART можно считывать. Регулярное отслеживание этих параметров операционной системой — это своего рода механизм заблаговременного обнаружения проблем. С его помощью неисправности можно предупредить задолго до того, как они, возможно, вызовут потери данных.

В этом разделе сделан обзор инструментов, используемых в Linux для считывания параметров SMART. Более подробная информация сообщается в «Википедии», а также на следующих сайтах:

- http://smartmontools.sourceforge.net/;
- http://www.linuxjournal.com/article/6983;
- http://www.linux-user.de/ausgabe/2004/10/056-smartmontools/.

Для работы SMART есть некоторые предварительные условия.

- Диск должен поддерживать систему SMART. Чтобы определить, так ли это, выполните, например, команду `hdparm -I/dev/sdx`.
- Диск должен быть внутренним или типа eSata. С внешними дисками, подключаемыми через USB или FireWire, функции SMART, к сожалению, не используются.
- Если жесткий диск управляется через аппаратный контроллер RAID, функции SMART могут использоваться только в отдельных случаях (подробности в справке `man smartctl` по параметру `-d`).

Определение состояния (smartctl)

Для считывания данных SMART и для того, чтобы произвести самодиагностику SMART, используется команда `smartctl`. В такой простейшей форме команда сообщает различную статусную информацию. Если `smartctl -i` выдает в последней строке сообщение SMART support is Disabled (Поддержка SMART отключена), активизируйте SMART командой `smartctl -s`.

В большинстве дистрибутивов программа `smartctl` входит в состав пакета `smartmontools`, который зачастую требуется установить дополнительно. В некоторых дистрибутивах при этом автоматически инсталлируется почтовый сервер (MTA), позволяющий при необходимости рассылать уведомления SMART по электронной почте. На сервере такую систему иметь полезно, а на локальном компьютере, как правило, нет. В Ubuntu можно отменить установку почтового сервера, если сообщить команде `apt-get` параметр `--no-install-recommends`.

```
root# smartctl -i /dev/sdb
Версия smartctl 5.37 Copyright (C) 2002-6 Bruce Allen
Модель устройства: ST3500320AS
Серийный номер: 5QM1GG0W
Версия прошивки: SD1A
Пользовательская мощность: 500.107.862.016 байт
Устройство: не входит в базу данных smartctl [подробности: -P showall]
Версия ATA: 8
Стандарт ATA: неизвестно. ed. Код версии: 0x29
Местное время: Пт Мар 13 14:34:47 2009 среднеевропейское время
Поддержка SMART: доступна - устройство поддерживает SMART.
Поддержка SMART: включена
```

Команда `smartctl -H`, или `smartctl --health`, указывает, нормально ли работает жесткий диск и будет ли он работать через 24 часа. Если в качестве результата `smartctl` в данном случае не выдаст `PASSED`, нужно *немедленно* начать полное резервное копирование!

```
root# smartctl -H /dev/sda
...
SMART overall-health self-assessment test result: PASSED
```

Команда `smartctl -A`, или `smartctl -attributes`, выдает полный список специфичных заводских атрибутов жесткого диска. Для этих атрибутов не существует строгого стандарта, но важнейшие из них поддерживаются многими производителями жестких дисков. При интерпретации значений очень важны два столбца: `VALUE` указывает актуальное значение, а `THRESH` — предельную величину. Если актуальная величина превышает предельную, ждите проблем — жесткий диск свое уже отработал.

Базовым нормальным значением для большинства показателей является `100`. Например, значение `Power_On_Hour` у нового жесткого диска равно `100`. По истечении определенного периода эксплуатации (в часах) значение снижается до `99` и т. д. Чтобы узнать, сколько часов уже эксплуатировался жесткий диск, посмотрите значение в столбце `RAW_VALUE`. У тестового диска это `3360`, что равно примерно 420 рабочим дням по 8 часов. На некоторых жестких дисках срок эксплуатации измеряется в минутах или секундах. В таком случае, чтобы узнать верное значение, укажите `-v 9,minutes` или `-v 9,seconds` соответственно.

Следующий, несколько сокращенный вывод взят из оценки состояния жесткого диска SATA, проработавшего около девяти месяцев. Признаков проблем нет. Текущая эксплуатационная температура (`ID 194`, столбец `RAW_VALUE`) составляет 35 °C.

```
root# smartctl -A /dev/sda
...
Vendor Specific SMART Attributes with Thresholds:
ID#  ATTRIBUTE_NAME        VALUE  WORST  THRESH  TYPE     UPDATED  RAW_VALUE
  1  Raw_Read_Error_Rate   109    099    006     Pre-fail Always   24386832
  3  Spin_Up_Time          096    095    000     Pre-fail Always   0
  4  Start_Stop_Count      100    100    020     Old_age  Always   167
  5  Reallocated_Sector_Ct 100    100    036     Pre-fail Always   0
  7  Seek_Error_Rate       065    060    030     Pre-fail Always   3391200
  9  Power_On_Hours        100    100    000     Old_age  Always   451
...
198  Offline_Uncorrectable 100    100    000     Old_age  Offline  0
199  UDMA_CRC_Error_Count  200    200    000     Old_age  Always   0
```

Команда `smartctl -l error` сообщает информацию о пяти последних возникших ошибках. Часто результат просто пуст (`No Errors Logged`). Единичные неповторяющиеся ошибки — это, как правило, не повод для беспокойства.

```
root# smartctl -l error /dev/sda
SMART Error Log Version: 1
No Errors Logged
```

Проведение самодиагностики

SMART предусматривает различные варианты самодиагностики системы, позволяющие еще точнее определить, в каком состоянии находится жесткий диск. Такие тесты начинаются с `smartctl -t short/long`. Краткий тест длится несколько минут,

а подробный (long) в некоторых случаях может занять несколько часов. Тест проводится в фоновом режиме, то есть вы можете продолжать работу. После того как тест будет закончен, его результаты можно отобразить командой smartctl -l selftest. В столбце Remaining указано, сколько еще времени будет выполняться тест. Если значение выше 0 %, то тест еще не окончен! Значение LifeTime указывает, сколько времени проработал диск; LBA определяет место (сектор), где возникла первая ошибка.

В следующем выводе показано выполнение трех тестов самодиагностики; первый был проведен почти сразу же после подключения диска (через 50 часов эксплуатации), два оставшихся — примерно через 2600 часов.

```
root# smartctl -t short /dev/sda
Num   Test_Description   Status                  Remaining   LifeTime   LBA
# 1   Extended offline   Completed without error    00%       2592     -
# 2   Short offline      Completed without error    00%       2591     -
# 3   Short offline      Completed without error    00%         40     -
```

Автоматическое наблюдение (smartd)

Программа smartctl — это очень интересный инструмент, предназначенный для сбора данных о жестком диске. Однако для регулярного наблюдения за всеми дисками программа неудобна. При необходимости регулярного наблюдения аналогичную задачу выполняет программа smartd. Это демон (системная служба). Команды для автоматического запуска отличаются от дистрибутива к дистрибутиву и перечислены в разделе 4.5.

Команда smartd управляется через /etc/smartd.conf. В некоторых дистрибутивах сценарий Init-V также интерпретирует файлы /etc/sysconfig/smartmontools или /etc/default/smartmontools. В этих файлах имеются дополнительные командные параметры для smartd. В Debian и Ubuntu потребуется внести в smartmontools настройку start_smartd=yes, иначе программа не запустится!

Обычная конфигурация компьютера с двумя жесткими дисками SATA (/dev/sda и /dev/sdb) выглядит следующим образом:

```
# Файл /etc/smartd.conf
/dev/sda -d sat -H -m root -M test
/dev/sdb -d sat -H -m root -M test
```

Это означает, что «здоровье» обоих жестких дисков будет проверяться каждые полчаса (как при использовании smartctl -H). Если при этом обнаружится ошибка, smartd отошлет по электронной почте сообщение локальному пользователю root (для этого вам в любом случае потребуется локальный почтовый сервер). Параметр -d sat обозначает жесткие диски как устройства SATA; -M test служит для того, чтобы узнать, работает ли в принципе рассылка электронной почты. Начните со smartd:

```
root# /etc/init.d/smartd start
```

Если вы получите тестовое электронное сообщение, удалите из конфигурации -M test. В файле smartd.conf, входящем в комплект вместе с smartmontools, есть еще несколько примеров конфигурации.

Пакет smart-notifier

Вы можете получать визуальные сообщения о возникновении проблем, угрожающих жесткому диску. Для этого установите пакет `smart-notifier` и измените `smartd.conf` согласно следующему образцу:

```
# /etc/smartd.conf
DEVICESCAN -m root -M test -M exec /usr/share/smartmontools/smartd-runner
```

Теперь для сообщений SMART будет выполняться команда `smartd-runner`. Она пересылает уведомление `smartd` через систему связи D-Bus. Чтобы предупреждение `smartd` отображалось и на действующем рабочем столе, в фоновом режиме должна запуститься программа `smart-notifier`. Это условие выполняется во всех локальных системах, интерпретирующих `/etc/xdg/autostart`.

Обратите внимание, что для параметра `-M exec` в `smartd.conf` также нужен параметр `-m`. Иными словами, вы не можете обойтись без электронного сообщения. Напротив, параметр `-M test` вы можете удалить после того, как убедитесь, что система уведомления функционирует.

Программа Palimpsest

В новых дистрибутивах текущее состояние SMART можно узнать с помощью программы Gnome Palimpsest, что очень удобно (рис. 13.2). Для этого выберите привод и щелкните на ссылке Показать информацию.

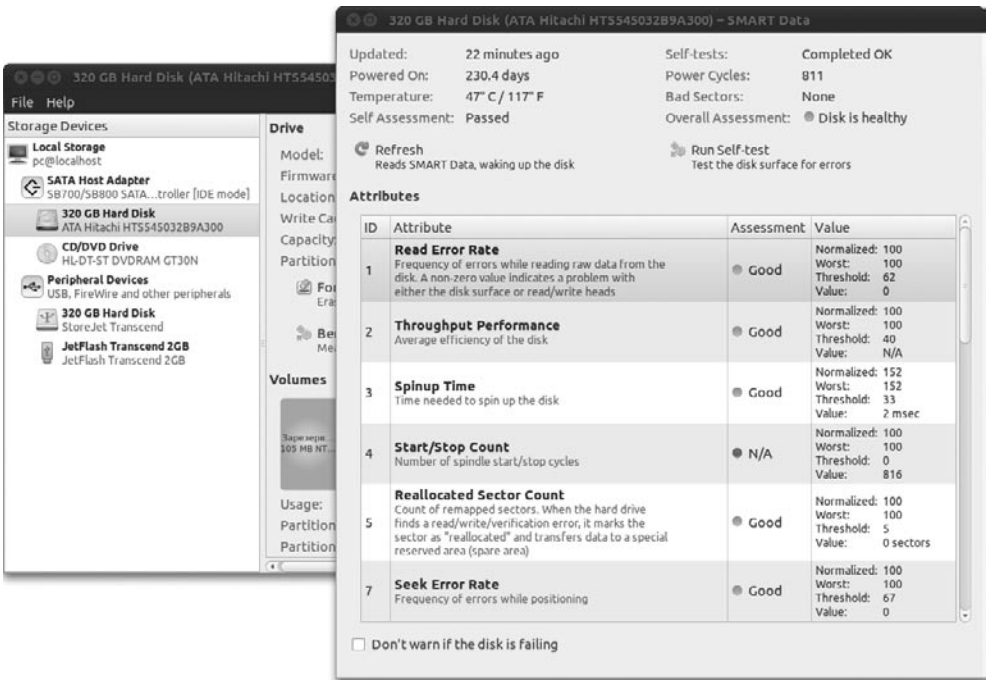

Рис. 13.2. Информация о SMART в Palimpsest

13.17. Шифрование

Иногда ноутбуки или USB-флешки теряются или попадают в руки к ворам. В таком случае вы теряете не только устройство, но и хранящиеся на нем данные, что вдвойне плохо. Чужие люди могут получить доступ к данным онлайн-банкинга, номерам страховки, историям болезни, коммерческим или военным тайнам. Не нужно этого допускать. Достаточно провести несложное шифрование файловой системы, чтобы надежно защитить свои данные. В этом разделе сообщается некоторая базовая информация по работе с зашифрованными файлами и файловыми системами.

Шифрование отдельных файлов

Gpg. Отдельный файл удобнее всего зашифровать командой `gpg`. Если вы введете `gpg -c`, система попросит вас дважды ввести пароль, затем зашифрует указанный файл и сохранит результат под именем `file.gpg`. При этом по умолчанию применяется алгоритм шифрования CAST5. Теперь исходный файл можно удалить. Команда `gpg -d` снова восстанавливает файл.

```
user$ gpg -c file
Введите пароль: **********
Повторите пароль: **********
user$ gpg -d файл.gpg > file
Введите пароль: **********
```

Команда `gpg` может использовать при кодировании или декодировании открытый или закрытый ключ, может ставить на файлах цифровые подписи, управлять ключами и т. д. Описание бесчисленных параметров в справке `man` занимает около 50 страниц! Но вручную команда `gpg` применяется редко. Чаще ее используют почтовые клиенты, чтобы (более или менее автоматически) ставить на электронные сообщения подписи или шифровать эти сообщения.

Шифрование файловой системы (USB-флешка, внешний жесткий диск)

dm_crypt и LUKS. Уже давно разработаны многочисленные методы для шифрования файловых систем: CryptoFS, eCryptfs, Enc-FS, Loop-AES и LUKS. Некоторые из этих методов применяются до сих пор, другие — зачастую из-за недостаточной безопасности — уже заброшены. В настоящее время самым популярным методом является LUKS (Linux Unified Key Setup).

Метод LUKS основан на модуле ядра `dm_crypt`, который обогащает применяемый в LVM модуль отображения устройств функциями шифрования. Модуль `dm_crypt` — это логический слой, расположенный между зашифрованными исходными данными жесткого диска и файловой системой, которую видит пользователь. Модуль поддерживает различные алгоритмы шифрования, его можно комбинировать с LVM — часто так и делается, но это совсем не обязательно. Модуль `dm_crypt` можно использовать и в такой системе, которая не работает с LVM!

LUKS добавляет к зашифрованным данным заголовок, в котором содержится метаинформация. В заголовке среди прочего указано, каким методом были зашиф-

рованы данные. LUKS очень упрощает интеграцию зашифрованных носителей данных в Linux.

Cryptsetup. Чтобы создавать зашифрованные файловые системы, воспользуйтесь командой `cryptsetup` из одноименного пакета. В следующих строках показано, как сначала отформатировать USB-флешку (`/dev/sdh1`) как криптоустройство (`luksFormat`), а потом активизировать устройство под произвольно выбранным именем `mycontainer` (`luksOpen`). Разумеется, степень защиты ваших данных прямо пропорциональна сложности пароля, который может состоять и из нескольких слов (`passphrase`). Рекомендуется, чтобы в пароле было минимум 20 символов.

Кроме того, вы можете использовать `/dev/mapper/mycontainer` как раздел жесткого диска или логический том, то есть создать файловую систему, подключить ее к дереву каталогов и т. д. После выполнения команды `unmount` можно снова деактивизировать криптоустройство (`luksOpen`), чтобы открыть доступ к `/dev/sdh1`. Только после этого извлекайте флешку.

```
root# cryptsetup luksFormat /dev/sdh1
Безвозвратно заменить данные /dev/sdh1
Вы уверены? (Введите YES): YES
Введите пароль LUKS: **********
Подтвердите пароль: **********
Команда выполнена успешно.
root# cryptsetup luksOpen /dev/sdh1 mycontainer
Введите пароль LUKS: **********
root# mkfs.ext3 /dev/mapper/mycontainer
root# mount /dev/mapper/mycontainer /test
root# touch /test/xy
root# umount /test/
root# cryptsetup luksClose mycontainer
```

Разумеется, вместо USB-флешки можно использовать и раздел внешнего или внутреннего жесткого диска, устройство RAID или логический том вашей системы LVM. Для этого просто замените `/dev/sdh1` названием устройства раздела или логического тома.

По умолчанию `cryptsetup` использует алгоритм шифрования AES, длина ключа в котором составляет 128 бит. Вы можете в этом убедиться с помощью команды `cryptsetup luksDump`: она выдает метаинформацию о криптоустройстве; LUKS сохраняет эту метаинформацию в специальном секторе носителя данных.

```
root# cryptsetup luksDump /dev/sdh1
LUKS header information for /dev/sdh1
Version:        1
Cipher name:    aes
Cipher mode:    cbc-essiv:sha256
Hash spec:      sha1
Payload offset: 1032
MK bits:        128
...
```

Если хотите использовать другой алгоритм шифрования или более длинный ключ, сообщите нужные данные с помощью параметров `-c` и `-s` команде `cryptsetup`

luksFormat. Чтобы узнать, какие алгоритмы можно применить, посмотрите cat /proc/crypto. В настоящее время наиболее надежными считаются алгоритмы AES и TwoFish. Обратите внимание, что алгоритмы шифрования сейчас активно исследуются и ситуация с ними быстро меняется: они часто оказываются не такими надежными, как это казалось поначалу. Краткое описание алгоритмов приводится на следующем сайте: http://de.gentoo-wiki.com/wiki/DM-Crypt. Еще более подробно этот вопрос рассматривается в «Википедии».

Используя команду cryptsetup luksAddKey, можно обезопасить доступ к устройству LUKS с помощью восьми различных паролей. Это позволяет совместно работать с носителем, для доступа к которому каждый из пользователей применяет собственный пароль.

Команда luksformat. Эта команда немного упрощает создание зашифрованного раздела или носителя данных. Сначала она выполняет cryptsetup luksFormat, а затем mkfs.vfat. Если вы хотите работать с файловой системой другого типа, укажите его с помощью параметра -t.

СОВЕТ

После выполнения команды (зачастую и при возникновении ошибки) криптоустройство /dev/mapper/luksformatn может «отстать» и не сразу среагировать. Прежде чем удалить носитель данных и еще раз попытаться создать криптоустройство, выполните команду cryptsetup luksClose luksformatn.

Работа с локальным компьютером. Если вы подключаете к компьютеру внешний носитель данных, отформатированный под LUKS, и начинаете работать с Gnome или KDE, то носитель данных автоматически распознается как криптоустройство. Открывается окно, в котором вам следует указать пароль для шифрования (рис. 13.3). После этого носитель данных подключается к файловой системе. Имя контейнера для /dev/mapper — luks_crypto_uuid. При отключении носителя нужно выполнить luksClose.

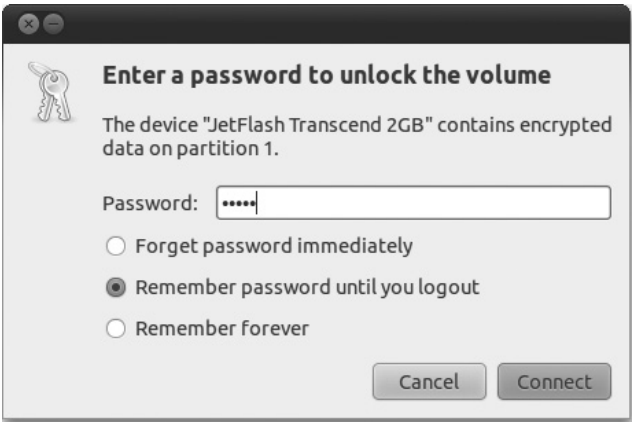

Рис. 13.3. Использование криптоустройства в Gnome

Файл crypttab. Если вы создали зашифрованную файловую систему в разделе локального диска, то, вероятно, захотите, чтобы эта система подключалась к дереву каталогов при запуске компьютера. В пакете cryptsetup уже содержатся сцена-

рии, необходимые для автоматизации такого процесса. Для работы сценариев необходимо, чтобы криптоустройство было указано в файле /etc/crypttab.

Файл построен очень просто: в первом столбце указывается название для /dev/mapper, во втором — имя устройства, в третьем — файл, из которого должен быть считан ключ (например, с USB-флешки), или none, если пароль шифрования вводится в интерактивном режиме; в четвертом столбце указываются параметры.

В следующем примере устройство /dev/sda7 нужно создать под именем /dev/mapper/cdisk1. Пароль необходимо указать при запуске компьютера, а криптоустройство создается с LUKS (другие параметры описаны в man crypttab).

```
# Файл /etc/crypttab
# Название модуля  Устройство  Файл ключей  Параметры
cdisk1             /dev/sda7   none         luks
```

Чтобы не только активизировать криптоустройство, но и подключить файловую систему к дереву каталогов, нужно дополнить /etc/fstab. Следующая строка позволяет использовать файловую систему через каталог /media/private-data:

```
# Файл /etc/fstab
...
/dev/mapper/cdisk1   /media/private-data   ext3   defaults   0 0
```

После этого перезапустите компьютер и проверьте, все ли работает.

Недостатки. Шифрование раздела имеет и свои недостатки. Во-первых, все операции с файлами совершаются заметно медленнее, чем обычно — и более того, тем медленнее, чем сложнее (и надежнее) применяемый метод шифрования. По возможности используйте быстрый процессор с несколькими ядрами. Во-вторых, при загрузке системы необходимо вводить пароль. Это не просто неудобно — главное, что в ваше отсутствие компьютер нельзя будет перезапустить. В принципе с зашифрованными разделами стоит работать только на локальном компьютере и в исключительных случаях на сервере.

TrueCrypt. У представленных здесь методов, основанных на dm_crypt и LUKS, есть интересная альтернатива — TrueCrypt (http://www.truecrypt.org/). Эта шифровальная программа доступна также для Windows и Mac OS X, таким образом, она значительно упрощает обмен данными между Linux и другими системами. Исходный код открыт, но некоторые его части несовместимы с лицензией GPL, поэтому в распространенных дистрибутивах TrueCrypt отсутствует.

ecryptfs (Ubuntu). В Ubuntu есть возможность зашифровать весь домашний каталог (версия 9.10 и выше) или его подкаталог Private (Ubuntu 8.10 и 9.04). Зашифрованный каталог автоматически подключается к файловой системе, а при выходе снова отключается. Внутри системы для этого применяется не dm_crypt и LUKS, а файловая система ecryptfs.

Шифрование целой системы

Файл /etc/crypttab, рассмотренный в предыдущем разделе, позволяет сделать еще один небольшой шаг от шифрования раздела локального диска (например, /home) к шифрованию всей системы, в том числе системного раздела. Важно учесть две детали: во-первых, GRUB не может получить доступ к зашифрованным данным,

поэтому обязательно должен быть отдельный, незашифрованный загрузочный раздел. Во-вторых, для доступа к системному разделу нужно ввести пароль шифрования; необходимые для этого функции должны быть интегрированы в виде сценариев в файл `Initrd` (все необходимые файлы содержатся в пакете `cryptsetup`).

Отдельно нужно рассказать, кому вообще может потребоваться зашифровать всю систему: в принципе будет вполне достаточно зашифровать только личные файлы, находящиеся в каталоге `/home`. Однако для вора, укравшего ноутбук и желающего заполучить содержащиеся в нем данные, может быть очень информативен и системный раздел: в `/var/cache` или `/var/tmp` могут содержаться остатки отосланных электронных писем, распечатанных документов, прочитанных файлов PDF и т. д; в `/var/log` документируется, кто и когда работал с компьютером; в разделе подкачки содержатся всевозможные выгруженные блоки данных с информацией, важной с точки зрения безопасности, и т. д. Таким образом, если вы хотите максимально защитить данные (например, личные), хранящиеся на компьютере, попробуйте зашифровать всю систему.

Установка

В большинстве крупных дистрибутивов в установочной программе предусмотрена функция, позволяющая зашифровать всю файловую систему (в openSUSE — только в версии 11.2 и выше). В любом случае в дистрибутивах, где на выбор предлагается несколько способов или видов установки, нужно, как правило, останавливаться на традиционном варианте; установочные программы, запускаемые с живого CD или DVD, в силу принципа их работы не подходят. В следующем списке перечислены подходящие средства установки для важнейших дистрибутивов:

- Debian — все стандартные CD или DVD (в том числе Netinstall, но не живые диски);
- Fedora — установка с DVD (но не живые диски);
- openSUSE — установка с DVD в версии 11.2 и выше (но не живые диски);
- Ubuntu — установка с CD Alternate в текстовом режиме (но не с CD для локальной системы).

В Debian и Ubuntu выберите вариант секционирования диска Шифрование ▸ LVM-система, в Fedora откройте диалоговое окно секционирования, а в нем установите флажок Зашифровать систему. В openSUSE секционирование нужно выполнить вручную, а затем создать незашифрованный загрузочный раздел и зашифрованный раздел LVM. После этого вы сможете создать в физическом томе LVM любые другие разделы, в том числе системный.

Вообще, в Интернете есть множество руководств, описывающих создание зашифрованной системы «задним числом» или помогающих «перехитрить» систему живого диска и все-таки зашифровать ее, хотя это и не предусматривалось. Но зачем усложнять жизнь, если она и так непроста?

Организация системы

В большинстве дистрибутивов зашифрованные системы построены одинаково: для GRUB создается незашифрованный загрузочный раздел, а другой раздел — зашифрованный — работает как физический том для LVM. Таким образом, все раз-

делы, созданные с применением LVM (раздел подкачки, системный раздел, разделы с данными) автоматически шифруются. Кроме того, не требуется задавать для каждого раздела собственный пароль; достаточно будет общего пароля для всей системы LVM. На рис. 13.4 схематически показано строение такой системы, причем обозначения устройств и логических томов взяты из Fedora.

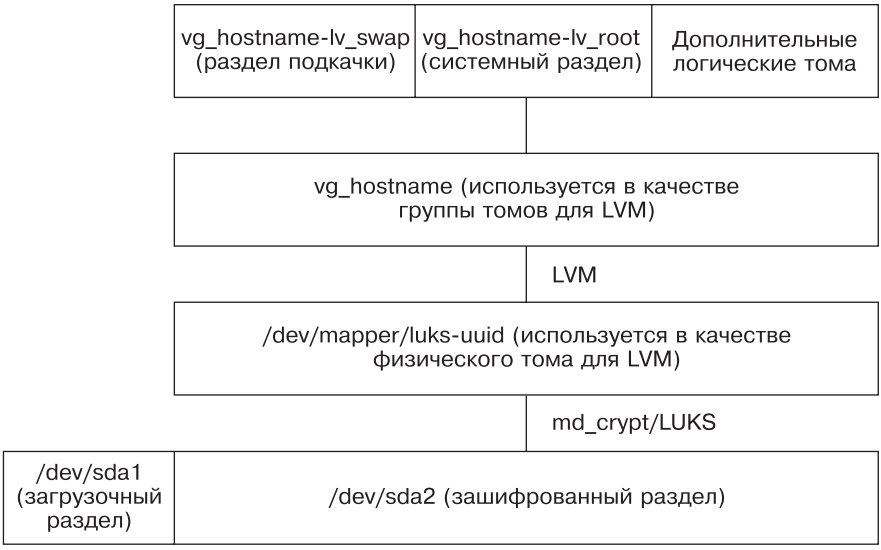

Рис. 13.4. Полностью зашифрованная система Linux

Конечно, вариант организации системы, показанный на рис. 13.4, далеко не единственный. Еще один вариант позволяет отказаться от LVM и отдельно зашифровать каждый раздел. При этом для раздела подкачки можно использовать случайный ключ, который будет заново создаваться в `/dev/urandom` при каждом запуске системы.

Можно представить себе и другой метод задания ключа: не вводить пароль интерактивно, а считывать ключ из файла, расположенного на USB-флешке, при загрузке системы. В таком случае флешка служит своего рода «аппаратным ключом», необходимым для загрузки компьютера. Некоторые устройства для чтения карт памяти также могут работать с Linux; однако интеграция подобных устройств с системой шифрования выполняется вручную.

14 Запуск системы

В этой главе речь пойдет о запуске системы Linux. Запуск системы зависит от загрузчика GRUB, отвечающего за то, чтобы ядро Linux запустилось. GRUB — это крошечная программа, установленная в загрузочном секторе жесткого диска или раздела. В настоящее время используется две версии GRUB: широко распространена устаревшая версия GRUB 0.9n, однако в некоторых дистрибутивах (например, в Ubuntu 9.10) уже задействована новая версия 2. В качестве альтернативы GRUB можно применять и LILO.

При запуске ядро выполняет инициализацию (распознает аппаратное обеспечение, устанавливает соединение с жесткими дисками и т. д.). Если при этом возникают проблемы, их иногда можно устранить, откорректировав загрузочные параметры. Наконец, ядро запускает программу `init`. Она выполняет различные инициализационные задачи, в частности подключает файловые системы, запускает сетевые службы и т. д. В зависимости от дистрибутива `init` может входить в состав системы Init-V или быть частью более нового процесса Upstart.

Среди сетевых служб, запускаемых Init-V, в некоторых конфигурациях предусмотрен так называемый *демон служб Интернета*. Эта программа следит за сетевыми портами. Если оттуда поступают запросы, демон запускает ту или иную программу, которая предназначена для отклика на конкретный запрос.

Информация, сообщаемая в этой главе, поможет вам и в тех случаях, когда компьютер запустить не получается. При выполнении процессов Init-V или Upstart запускается также и графический пользовательский интерфейс X. О системе X подробно рассказано в главе 12.

14.1. GRUB

Грандиозный унифицированный системный загрузчик (кратко — GRUB) — это маленькая программа, считывающая данные ядра Linux при запуске компьютера, помещающая их в оперативную память и выполняющая команды. Наиболее распространенная версия имеет необычное название «GRUB 0.9n legacy». Добавка «legacy» указывает на то, что активная разработка этой версии прекращена, кроме того, данная версия уже не поддерживается (централизованно). Правда, во многих дистрибутивах GRUB неоднократно дорабатывалась по усмотрению производи-

телей; в результате некоторые функции GRUB отличаются от дистрибутива к дистрибутиву!

Вместо того чтобы поддерживать старую версию GRUB, разработчики с лета 2005 года трудились над совершенно новой версией, называемой GRUB 2. В принципе, новая версия уже работает, но еще не совсем готова и очень плохо документирована. Из-за того, что старая версия поддерживается плохо, разработчики Ubuntu решили не дожидаться полной готовности GRUB 2 и окончательно перевести версию Ubuntu уже на GRUB 2. Остальные крупные дистрибутивы только собираются последовать за Ubuntu.

В этой главе мы сначала рассмотрим версию GRUB 0.9n legacy, так как она более распространена. Общая информация по нововведениям, появившимся в GRUB 2, дается в разделе 14.7.

Официальная документация по GRUB выдается по запросу справочной командой `info grub`. Более подробная информация содержится на официальной странице GRUB: http://www.gnu.org/software/grub/.

Особенности загрузки системы

BIOS. Прежде чем подробно обсудить установку и конфигурацию GRUB, коротко рассмотрим, что же происходит при загрузке системы. Когда вы включаете компьютер, первым делом инициализируется BIOS. В это время на экран обычно выводится пара системных сообщений, например сколько в компьютере оперативной памяти.

Затем BIOS загружает содержимое первого сектора первого жесткого диска в оперативную память и выполняет этот код. Этот специальный раздел жесткого диска называется *основной загрузочной записью (MBR)*. Подробная информация по MBR и загрузочным секторам разделов жесткого диска дается в разделе 14.4.

Загрузчик Windows. Если на компьютере установлена ОС Windows, то в MBR располагается крошечная программа. Она находит раздел, помеченный как «активный», а затем запускает загрузчик Windows, расположенный в загрузочном секторе данного раздела. Если на компьютере установлено несколько версий Windows, то загрузчик Windows позволяет выбрать одну из них.

Загрузчик Linux. Если на компьютере установлена и Linux, MBR обычно заменяется кодом загрузчика GRUB. В таком случае GRUB может либо запустить Linux, либо совершить условный переход для запуска Windows.

Альтернативный метод заключается в том, чтобы не трогать MBR и установить GRUB в загрузочном секторе системного раздела Linux, пометив этот раздел как активный. Такой метод хотя и не противоречит правилам MBR, менее надежен и поэтому почти не используется.

Основная загрузочная запись имеет размер всего 512 Мбайт, поэтому загрузчик не сможет полностью поместиться в MBR. Чтобы можно было обойти это ограничение, MBR может вместить ровно такой фрагмент кода, который позволит вам запустить оставшуюся часть загрузчика уже с жесткого диска. Соответственно, код GRUB подразделяется на две или три части: `stage1` находится в основной загрузочной записи и предназначена для того, чтобы загрузить первые сектора части

stage1_5 или stage2. В части stage1_5 содержится дополнительный код, обеспечивающий доступ к файлам различных файловых систем. Наконец, в stage2 содержится сам загрузчик.

Когда запустится загрузчик, появится меню, в котором будут на выбор представлены все операционные системы, определенные в процессе конфигурации GRUB (обычно это Windows и Linux). Теперь с помощью клавиш для управления курсором можно выбрать интересующую вас операционную систему и запустить ее, нажав Enter. Часто GRUB настроен так, чтобы по истечении определенного промежутка времени операционная система загружалась автоматически.

ВНИМАНИЕ

На жестком диске всегда есть только одна основная загрузочная запись, но может быть несколько операционных систем. При установке как Linux, так и Windows MBR перезаписывается. Если GRUB настроен так, что программа может запускать и Windows, то эта операционная система, к сожалению, не учитывает, что на компьютере кроме нее есть еще и Linux. Поэтому после установки Windows вам потребуется откорректировать GRUB, а для этого лучше всего использовать живой диск или систему-реаниматор. Сначала устанавливайте Windows, а потом Linux!

К сожалению, и при установке GRUB не исключены ошибки (но, к счастью, они очень редки). В особо тяжелом случае вы не сможете запустить ни одну из операционных систем. Некоторые советы насчет того, как решить такую проблему, даются в разделе 14.6.

Запуск Linux. Если вы определяете в загрузчике, что необходимо запустить Linux, то загрузчик должен поместить в оперативную память файл ядра Linux и запустить этот файл. Обычно файл ядра Linux называется /boot/vmlinuz (последняя буква z указывает на то, что ядро заархивировано). Иными словами, загрузчик должен быть в состоянии загрузить из файловой системы весь указанный файл.

Параметры ядра. Обычно ядру сообщается несколько параметров, но как минимум один: имя устройства системного раздела (например, root=/dev/sdb13). Это делается для того, чтобы ядро «знало», какой из разделов является системным. Когда ядро запустится, управление будет передано программе Linux /sbin/init, отвечающей за инициализацию системы Linux и подробно описанной в разделе 14.10. Например, она отвечает за запуск сетевых служб.

Доступ к модулям ядра. Ядро Linux имеет модульную структуру. Это означает, что в самом ядре содержатся только самые элементарные функции. Дополнительные функции, необходимые для доступа к компонентам аппаратного обеспечения, для считывания и изменения файлов из различных файловых систем и т. д., напротив, находятся в модулях, которые при необходимости загружаются из файловой системы и, таким образом, дополняют ядро.

Для того чтобы процесс запуска прошел успешно, ядро должно иметь возможность получать доступ к системному разделу. Если этот раздел расположен в файловой системе, не поддерживаемой ядром напрямую, или на жестком диске SCSI, для которого в ядре нет подходящего драйвера, возникает проблема «курицы и яйца»: ядро не может обратиться к файловой системе и загрузить из нее модули, которые понадобились бы ядру, чтобы прочитать файлы файловой системы...

Файл Initrd. Решение заключается в том, что GRUB должен загружать не только ядро, но и файл Initrd. Это специальный файл, в котором содержатся все модули ядра, необходимые для запуска системы. Ядро временно использует этот файл

как псевдодиск, то есть оно может загрузить все модули сразу же после запуска псевдодиска (сокращение Initrd означает *диск в оперативной памяти для начальной инициализации*).

Обычно файл `Initrd` называется `/boot/initrd` или `/boot/initrd.gz`. В большинстве дистрибутивов имеются инструменты, позволяющие создать такой файл `Initrd`, который бы подошел к применяемому аппаратному обеспечению и к файловой системе, находящейся в системном разделе (команда `mkinitrd`).

Установка и конфигурация GRUB

Когда на страницах этой книги мы говорим об установке программы, обычно имеется в виду установка пакета с CD для Linux. В этой главе все иначе. Под установкой GRUB понимается процесс, в ходе которого стартовый код GRUB записывается в загрузочный сектор жесткого диска. Предполагается, что в файловой системе Linux установлен программный пакет GRUB.

При конфигурации GRUB необходимо настроить файл `/boot/grub/menu.lst` так, чтобы можно было запустить любую операционную систему из тех, что установлены на компьютере. По умолчанию GRUB обращается к этому файлу.

EFI. Уже несколько лет Intel и некоторые производители материнских плат и компьютеров пытаются продвигать расширяемый интерфейс встроенного ПО (кратко — EFI) как новую альтернативу BIOS. Современные версии Linux и Windows уже совместимы с EFI.

К сожалению, EFI пока не распространен в мире ПК, поэтому мы не будем рассматривать его более подробно. Только Apple использует EFI уже несколько лет. Если вы хотите параллельно установить на компьютере Apple Mac OS X и Linux, посмотрите следующий сайт: http://wiki.onmac.net/index.php/Triple_Boot_via_BootCamp.

Обратите внимание, что версия GRUB 0.97 legacy не совместима с EFI. В GRUB 2, напротив, существует версия как для BIOS (пакет grub-pc), так и для EFI (пакет grub-efi).

14.2. Работа с GRUB (с точки зрения пользователя)

Если установка GRUB прошла успешно, то после перезапуска на экране отображается меню для выбора нужной операционной системы (рис. 14.1). В зависимости от конфигурации внешний вид окна GRUB может быть разным. В некоторых дистрибутивах GRUB изукрашен яркой графикой и теряет свой спартанский облик. Чтобы можно было использовать различные дополнительные функции загрузчика, покиньте графический режим, нажав клавишу Esc.

Теперь любую запись в GRUB можно отредактировать командой `E` (edit). Команда `C` (command line) позволяет выйти из меню. Так вы снова окажетесь в своеобразной оболочке, в которой можно вводить команды с клавиатуры. Обзор команд выводится с помощью `help`. Более подробную информацию по конкретной команде можно узнать, набрав `help` *команда*. В этом разделе будут показаны некоторые

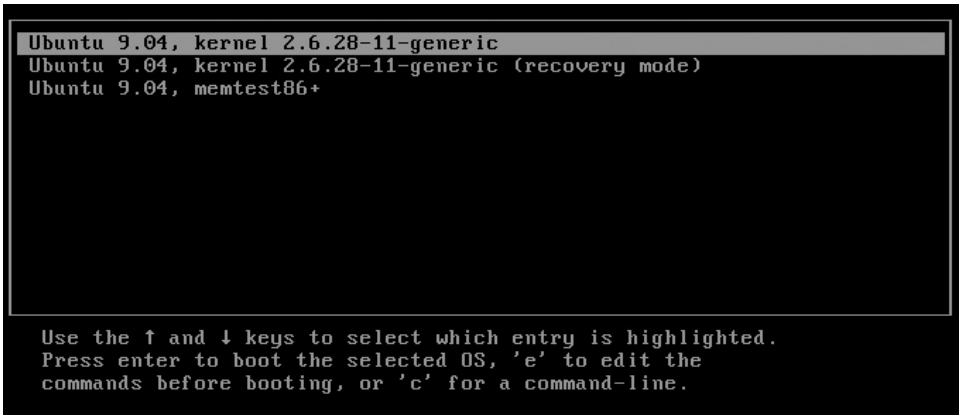

Рис. 14.1. Меню GRUB

команды. Нажав клавишу Esc, вы можете в любое время вернуться в загрузочное меню.

GRUB можно обезопасить с помощью пароля. В таком случае пользоваться интерактивными функциями загрузчика вы сможете только после того, как нажмете клавишу P и введете пароль.

В GRUB, как правило, используется клавиатура с раскладкой США.

Передача параметров загрузки ядра Linux

Функции редактирования используются в GRUB для того, чтобы при запуске Linux сообщить дополнительные параметры, касающиеся загрузки ядра (например, чтобы избежать аппаратных проблем). Для этого выберите нужную запись меню и командой E перейдите в режим редактирования. Теперь вы увидите несколько строк, которые будут выглядеть примерно так:

```
root (hd1,12)
kernel /boot/vmlinuz root=/dev/sdb13
initrd /boot/initrd
```

Поставьте курсор в строку kernel и снова нажмите клавишу E, чтобы изменить эту строку; в конце укажите дополнительные параметры, касающиеся загрузки ядра. Чтобы подтвердить внесенные изменения, нажмите клавишу Enter. Нажав Esc, вы снова выйдете в меню загрузки, где сможете запустить Linux. Изменение параметров ядра будет действительно только для конкретного сеанса работы, то есть они не сохраняются.

Интерактивное выполнение команд

Из меню GRUB можно перейти в режим интерактивного ввода команд — для этого нажмите C. В данном режиме пользователь вводит вручную различные команды.

Таким образом, можно запустить операционную систему Linux даже тогда, когда нужная запись в меню GRUB отсутствует или является ошибочной. Для того чтобы интерактивно вводить команды, вы должны знать, в каком разделе диска находится Linux и как называются нужные вам команды GRUB (подробно данный механизм описан далее в этой главе). Например, следующие команды позволяют запустить дистрибутив Linux, установленный в разделе `/dev/sdb13`.

```
grub> root (hd1,12)
grub> kernel /boot/vmlinuz root=/dev/sdb13
grub> initrd /boot/initrd
grub> boot
```

При вводе названий файлов GRUB автоматически дополняет названия файлов из файловой системы, указанной администратором или находящейся на заданном диске (рис. 14.2). Введя команду `cat`, вы даже можете отобразить некоторые текстовые файлы. Кроме того, в командном режиме имеется еще несколько интересных возможностей работы с GRUB, но мы не будем их здесь рассматривать ради экономии места.

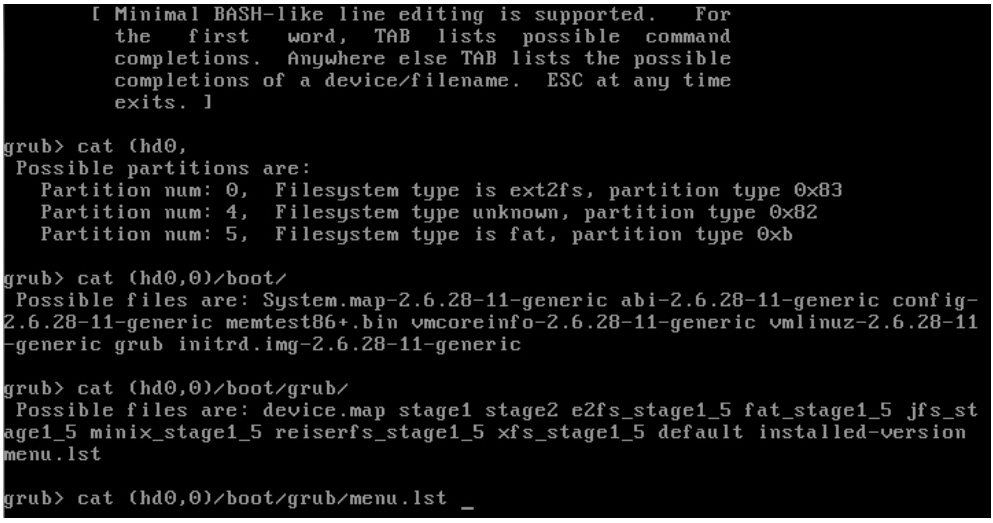

Рис. 14.2. Пример интерактивного использования GRUB

Закрепление изменений, внесенных в меню

Как правило, GRUB считывает загрузочное меню из файла `/boot/grub/menu.lst`. Этот файл содержит такие команды, как `title`, `root`, `kernel`, `chainloader` и т. д. Итак, если вы хотите закрепить изменения, внесенные в меню GRUB, вам нужно запустить Linux, найти файл меню GRUB и изменить его в редакторе. Загрузчик автоматически примет сделанные изменения при перезапуске. Организация файла меню GRUB более подробно описана в следующем разделе.

14.3. Конфигурация GRUB (файл меню)

В этом разделе описаны важнейшие элементы файла меню GRUB, который обычно называется `/boot/grub/menu.lst`. В Red Hat и Fedora для этого используется файл `grub.conf`. На него указывают символьные ссылки `/boot/grub/menu.lst` и `/etc/grub.conf`, так что вы не ошибетесь, если и в двух указанных дистрибутивах загрузите в редактор и измените файл `menu.lst`. В файле меню строки комментариев отмечены символом #. В некоторых дистрибутивах таких комментариев в файле меню очень много и читать файл практически невозможно.

ВНИМАНИЕ

В большинстве дистрибутивов Linux файл меню GRUB обновляется при каждом обновлении ядра. Это позволяет гарантировать, что после перезапуска будет использоваться новейшая версия ядра. В любом случае не исключено, что при автоматическом обновлении конфигурации GRUB система удалит внесенные вами изменения. Обращайте внимание на комментарии в файле menu.lst, так как они отличаются от дистрибутива к дистрибутиву!

В системах Debian и Ubuntu за обновление конфигурации GRUB отвечает сценарий update-grub.

Все названия файлов и разделов здесь приведены только в качестве примеров! Разумеется, вы должны использовать имена устройств системного и загрузочного разделов именно из вашей системы. Если Linux уже запущена, вы можете узнать названия этих разделов с помощью команды df. Кроме того, файл ядра Linux (vmlinux-xxx) или файл Initrd (initrd-xxx) на вашем компьютере называются не так, как в этой книге.

Изменения, внесенные в файл `menu.lst`, вступят в силу только после завершения установки GRUB в загрузочный сектор вашего жесткого диска или другого носителя данных. Подробно об установке GRUB рассказано в следующем разделе.

Для создания или конфигурирования GRUB вы можете пользоваться специальными инструментами, имеющимися в отдельных дистрибутивах; в частности, в SUSE для этого предназначен модуль YAST Система ▸ Конфигурация загрузчика.

Обозначение жестких дисков и разделов

В GRUB применяется собственная номенклатура для обозначения жестких дисков и их разделов (табл. 14.1). Основное правило — нумерация всегда начинается с нуля.

Таблица 14.1. Названия разделов GRUB

Название устройства GRUB	Значение
(hd0)	Первый жесткий диск (соответствует /dev/sda)
(hd0,0)	Первый раздел первого жесткого диска (соответствует /dev/sda1)
(hd2,7)	Восьмой раздел третьего жесткого диска

В зависимости от того, как секционирован диск, между обычными названиями устройств, а также между названиями разделов GRUB могут возникать незаполненные промежутки. Предположим, что на жестком диске расположен основной раздел, а также дополнительный, в котором содержится два логических раздела. Названия устройств этих разделов будут следующими: `/dev/sda1`, `/dev/sda2`, `/dev/sda5` и `/dev/sda6`. Соответствующие названия разделов GRUB — `(hd0,0)`, `(hd0,1)`, `(hd0,4)` и `(hd0,5)`.

Если на компьютере установлены жесткие диски двух типов: SATA и SCSI, то их нумерация зависит от настроек BIOS. CD- и DVD-приводы в нумерации не учитываются!

Файл devices.map. На внутрисистемном уровне GRUB использует файл `/boot/grub/devices.map` для соотнесения приводов и названий устройств загрузчика. Этот файл создается при первом выполнении GRUB. Правда, он не обновляется автоматически при добавлении новых приводов. При необходимости `devices.map` можно просто удалить, а после этого выполнить GRUB. В таком случае система создаст файл заново. Этот процесс может занять около минуты.

В особо тяжелых случаях можете также попробовать самостоятельно изменить файл. Однако учитывайте, что вносимые изменения должны соответствовать информации, получаемой GRUB от BIOS при запуске компьютера. Формат файла таков:

```
# Пример /boot/grub/devices.map
(hd0) /dev/sda
(hd1) /dev/sdb
(fd0) /dev/fd0
```

Глобальная область в menu.lst

Файл меню GRUB состоит из глобальной области, в которой содержатся различные базовые настройки, а также из нескольких записей меню, каждая из которых начинается со строки `title`. Например, в следующих строках показана глобальная область `menu.lst`:

```
# Глобальная область файла /boot/grub/menu.lst
default 2                    # третья запись меню используется по умолчанию
timeout 30                   # ожидать 30 секунд, пока не запустится
                             # система, заданная по умолчанию
color yellow/blue red/white  # Выделять записи меню цветом
```

В следующих абзацах описаны ключевые слова GRUB, используемые в глобальной области `menu.lst`.

- `default` — указывает номер записи меню, используемой по умолчанию. Счет начинается с 0! Вместо номера вы также можете указать `default saved`. В таком случае по умолчанию используется запись, применявшаяся при последнем запуске. Чтобы этот механизм работал, в каждой записи меню должно содержаться ключевое слово `savedefault` (подробнее об этом чуть ниже). Если в `menu.lst` отсутствует запись `default`, то первая запись в меню обозначает систему, запускаемую по умолчанию.
- `fallback` — задает номер записи меню, используемой, если запись, установленная для работы по умолчанию, содержит ошибку. Когда запись `fallback` отсутствует, GRUB, сталкиваясь с подобными ошибками, переходит в интерактивный режим.
- `timeout` — указывает, в течение какого времени (в секундах) GRUB будет ожидать выбора из меню. По истечении этого времени автоматически запускается операционная система, указанная в качестве стандартной. Если вы хотите,

чтобы GRUB ждала бесконечно, не запуская автоматически операционную систему, заданную по умолчанию, поставьте перед строкой `timeout` символ комментария #.

- `hiddenmenu` — заставляет GRUB не отображать меню. По истечении периода, указанного в `timeout`, запускается стандартная система. До наступления этого момента пользователь может отобразить меню, нажав Esc, а потом, как обычно, выбрать из него одну из записей.
- `password -- md5 code` — защищает GRUB паролем. Команды меню можно использовать и без пароля, но интерактивные функции GRUB предоставляются только после ввода пароля. Как создать код пароля, рассказано в разделе 14.5.
- `color fg/bg menufg/menubg` — управляет цветами, используемыми в меню GRUB. При этом команда `fg` задает цвет текста, а команда `bg` — цвет фона всего экрана. Соответственно, команды `menufg` и `menubg` задают цвет выбранной записи меню. Если не использовать команду `color`, то меню GRUB отображается в черном и белом цвете.
- `splashimage` — имеется лишь в некоторых дистрибутивах Linux, в которых GRUB был соответствующим образом дополнен (например, в Fedora). Иначе говоря, `splashimage` не является официальной функцией GRUB. Это ключевое слово позволяет задавать для меню фоновый рисунок, который должен иметь размер 640 × 480 пикселов, быть в формате XPM (8 бит на пиксел) и находиться в архиве GZIP. В следующей строке показано, как использовать `splashimage`:

  ```
  splashimage=(hd1,1)/boot/grub/splash.xpm.gz
  ```

  Данный метод подробно описан в документе по адресу: http://ruslug.rutgers.edu/~mcgrof/grub-images/.
- `gfxmenu` — еще одно неофициальное дополнение, предназначенное для отображения графических меню. Оно применяется, в частности, в дистрибутивах Novell и SUSE. Графический файл создается с помощью команды `mkbootmsg`. Эта команда, а также файл `gfxboot.html`, где содержится документация по ней, находятся в пакете gfxboot. Так или иначе, чтобы самостоятельно создать сплеш-файл, вам придется потрудиться.

Записи меню в menu.lst

После глобальной области в `menu.lst` следуют записи меню, соответствующие различным операционным системам. Каждая запись меню вводится словом `title`. Текст, введенный после `title`, является содержанием строки меню. При этом испытания, которые я проводил, показали, что поддерживаются только символы US-ASCII, то есть международные специальные символы использовать нельзя. Мне не удалось найти никакой документации, где давалась бы подробная информация по кодировкам для `menu.lst`.

Остальные строки (вплоть до следующей команды `title` или до конца файла) — это команды GRUB, выполняемые в том порядке, в котором они записаны. (Если вы испытываете команды в интерактивном режиме, вам дополнительно потребуется выполнить `boot`. Эту команду нельзя указывать в файле меню.)

Запуск Linux

Чтобы запустить Linux, необходимо указать с помощью `root` раздел, в котором расположены ядро и файл диска оперативной памяти для начальной инициализации. Этот раздел будет считаться в GRUB активным. Команды `kernel` и `initrd` точно указывают место, где находятся эти файлы, а также задают возможные параметры загрузки ядра.

Обратите внимание, что при указании параметров ядра (в частности, при указании корневого устройства ядра) используется номенклатура Linux. Корректная запись в данном случае будет выглядеть так: `root=/dev/sdb13`. Кроме того, есть и другие способы записи, например `root=LABEL=label` и `root=UUID=n`, причем в таких случаях необходимо указывать обозначение или идентификационный номер раздела (см. также раздел 14.9).

В дальнейшем учитывайте, что названия устройств в вашей системе могут отличаться от `vmlinuz` и `initrd` (базовая информация о файле `Initrd` сообщается в разделе 14.5).

```
# Выполнить запись меню в /boot/grub/menu.lst
# Linux в /dev/sdb13
title Linux
  root (hd1,12)
  kernel /boot/vmlinuz root=/dev/sdb13
  initrd /boot/initrd
```

Вы можете обойтись и без команды `root`. В таком случае потребуется указать для каждого файла нужный раздел диска:

```
# Выполнить Linux в /dev/sdb13
title Linux
  kernel (hd1,12)/boot/vmlinuz root=/dev/sdb13
  initrd (hd1,12)/boot/initrd
```

UUID

В Ubuntu предусмотрена возможность указывать раздел с файлами ядра и `initrd` не только с помощью `root`, но и с помощью ключевого слова GRUB `uuid`. Правда, такой синтаксис действует лишь в Ubuntu, поскольку разработчики этого дистрибутива соответствующим образом модифицировали GRUB. Запись с применением `uuid` идеально подходит для установки Ubuntu на USB-носителях.

```
# Запуск Linux вариант Ubuntu; загрузочный раздел имеет
# UUID=2a021cf3-..., корневой раздел управляется LVM
title Ubuntu 9.04
  uuid    2a021cf3-2b34-4cd4-b741-42b8fb1db89c
  kernel  /vmlinuz-2.6.28-13-generic root=/dev/mapper/vg1-ubuntu904 ro quiet
  initrd  /initrd.img-2.6.28-13-generic
```

Указание параметров ядра Linux

В зависимости от используемого оборудования или дистрибутива при запуске Linux обычно требуется передать системе параметры загрузки ядра (подробнее см. в разделе 14.9). Они отвечают в том числе за то, как будут отображаться

сообщения системы Init-V. В файле `menu.lst` такие параметры нужно просто указать в конце строки `kernel`. В следующих строках приведен соответствующий пример:

```
# Запуск Linux в /dev/sdb13 (с дополнительными параметрами ядра)

title Linux
  root (hd1,12)
  kernel /boot/vmlinuz root=/dev/sdb13 vga=normal
  initrd /boot/initrd
```

Чтобы узнать, какие параметры нужны для работы с вашим оборудованием или дистрибутивом, посмотрите конфигурационный файл GRUB, создаваемый при установке системы. Вот несколько примеров:

- Debian 5 — `root=/dev/xxx ro quiet`;
- Fedora 12 — `root=/dev/xxx ro rhgb quiet LANG=... SYSFONT=... KEYTABLE=...`;
- Ubuntu 9.10 — `root=UUID=xxx ro quiet splash`;
- SUSE 11.2 — `root=/dev/xxx resume=/dev/xxx splash=silent showopts vga=n`.

Запуск Windows

Если вы хотите запустить Windows, укажите активный раздел не с помощью `root`, а применив `rootnoverify`. Благодаря команде `chainloader +1` система прочитает и выполнит первый сектор этого раздела. В Windows 9x/ME это приводит к запуску операционной системы. В более новых версиях Windows, напротив, запускается загрузчик Windows, который, в свою очередь, уже загружает систему (сам GRUB не может запускать новейшие версии Windows).

```
# Запуск Windows в /dev/sda1
title Windows
  rootnoverify (hd0,0)
  chainloader +1
```

Запустить Windows удается лишь в том случае, если она находится на первом жестком диске. Если это не так, то следующие строки помогут вам виртуально «поменять диски местами». Я такой вариант не тестировал.

```
# Запуск Windows в /dev/sdb1
title Windows
  rootnoverify (hd1,0)
  map (hd0) (hd1)
  map (hd1) (hd0)
  chainloader +1
```

ВНИМАНИЕ

Если вы зашифровали раздел с Windows с помощью BitLocker, а отдельный загрузочный раздел для Windows отсутствует (именно так обычно бывает при первичной установке Windows 7), то GRUB не находит загрузчик Windows и, таким образом, Windows не может запуститься. В этом случае повторите весь процесс в обратном направлении и интегрируйте GRUB в загрузчик Windows (см. раздел 14.5).

Запуск другого загрузчика

Такие загрузчики, как GRUB или LILO, обычно устанавливаются в MBR первого жесткого диска. Однако имеется возможность устанавливать их в загрузочных секторах любых разделов (см. раздел 14.4). С помощью центрального GRUB, расположенного в MBR, можно опосредованно запустить другой GRUB, установленный в том или ином разделе. В `menu.lst` применяются те же ключевые слова, что и при запуске Windows:

```
# Запуск загрузчика в загрузочном секторе /dev/sda7
title Boot-Loader in /dev/sda7
  rootnoverify (hd0,6)
  chainloader +1
```

Такой метод может быть целесообразен, если требуется параллельно установить на одном жестком диске несколько дистрибутивов Linux (см. раздел 14.5).

Как пометить последнюю выбранную операционную систему

Если в `menu.lst` содержится много записей, стоит помечать запись, которую GRUB выбрал последней. Для того чтобы активизировать такую функцию, укажите в глобальной области `menu.lst` параметр `default saved` и дополните все записи меню ключевым словом `savedefault`:

```
# Запуск Linux в /dev/sdb13 пометить сделанный выбор
title Linux
  root (hd1,12)
  kernel /boot/vmlinuz root=/dev/sdb13
  initrd /boot/initrd
  savedefault
```

ВНИМАНИЕ

Ни в коем случае не применяйте savedefault на тех компьютерах, где в BIOS расположено два или более жестких диска, объединенных в группу RAID (BIOS-Software-RAID)! Вы рискуете потерять данные либо сбить синхронность работы системы!

Тестирование конфигурации GRUB

Если хотите быстро и без перезапуска проверить, нет ли в измененном файле меню GRUB синтаксических ошибок, запустите GRUB и выполните в нем следующую команду:

```
root# grub
grub> configfile (hd1,12)/boot/grub/menu.lst
```

Вместо `(hd1,12)` следует указать название GRUB для того раздела жесткого диска, где находится файл меню загрузчика. Если все получится, то GRUB отобразит меню. Правда, вы не сможете запустить ни одну из операционных систем, поскольку у вас уже работает Linux.

Сценарий update-grub (Debian и Ubuntu)

В Debian и Ubuntu для обновления конфигурации GRUB предусмотрен сценарий оболочки `update-grub`. Он ищет в каталоге `/boot` файлы `vmlinuz-*` и добавляет для каждого такого файла ядра отдельную запись в меню. При этом также учитываются файлы `initrd-*`, соответствующие той или иной версии ядра. Сценарий `update-grub` автоматически выполняется после каждого обновления ядра.

Сценарий учитывает некоторые настройки, которые указываются в `menu.lst`, как комментарии. Отдельные переменные подробно описаны в файле справки `man update-grub`. Самая важная переменная называется `kopt` и указывает, какие параметры сообщаются ядру. Если вы хотите внести в `menu.lst` изменения, касающиеся этой переменной, не изменяйте записи в `menu.lst` напрямую — изменяйте саму переменную `kopt`. Затем еще раз выполните `update-grub`.

```
# /etc/boot/menu.lst
...
# Настройки для update-grub
#
# kopt=root=/dev/sda13 ro (Параметры ядра)
# groot=(hd0,12)          (Место, в котором установлен GRUB)
# alternative=true        (Добавление записей о других операционных системах)
# lockalternative=false   (Опустить записи о других операционных системах)
# defoptions=             (Параметры, касающиеся лишь ядра, используемого
по умолчанию)
# lockold=false           (Опустить параметры, касающиеся других операционных систем)
# xenhopt=                (Параметры Xen)
# xenkopt=console=tty0    (Параметры ядра для ядра Xen)
# altoptions=(single-user mode) single  (Параметры ядра для альтернативного ядра)
# howmany=all             (Максимальное количество записей, касающихся ядра)
# memtest86=true          (Добавить запись Memtest)
# updatedefaultentry=false  (Изменить ключевое слово, используемое в menu.lst
по умолчанию)
…
```

14.4. Установка GRUB

В этом разделе описана установка GRUB в загрузочный сектор жесткого диска, раздела или другого носителя данных. Обычно подобная установка происходит одновременно с установкой Linux, поэтому *не* требуется повторно устанавливать GRUB. Для проведения конфигурации достаточно внести изменения в `/boot/grub/menu.lst` так, как это было описано в предыдущем разделе.

Заново устанавливать GRUB нужно только в тех случаях, когда он стерт (например, после переустановки Windows), копия GRUB не работает или вы хотите заменить на GRUB другой загрузчик. Как правило, установка GRUB производится с «живой» системы (например, в Knoppix), так как без рабочего загрузчика невозможно запустить дистрибутив Linux, установленный на жестком диске вашего компьютера. Советы о том, как устанавливать GRUB и «ремонтировать» систему с помощью Knoppix, даются в разделе 14.7.

Базовая информация о загрузочном секторе

Прежде чем я расскажу, как на самом деле производится установка GRUB, необходимо ненадолго заглянуть внутрь BIOS и MS DOS. Интерпретация (одного или нескольких) загрузочных секторов происходит по принципу, появившемуся не один десяток лет назад. В дальнейшем предполагается, что вы знаете о существовании нескольких типов разделов.

Основная запись диска (MBR). Это первый сектор жесткого диска. Он занимает 512 байт и обычно содержит крошечную программу (не больше 446 байт). Далее следует таблица разбиения диска для четырех основных разделов (64 байт) и цифровая подпись (2 байт).

Загрузочный сектор раздела. Такой сектор есть не только в MBR, но и в каждом разделе, и в действительности он может занимать до 16 секторов жесткого диска (8192 байт). В большинстве файловых систем загрузочный сектор раздела не используется, то есть сами данные начинаются только с последующих секторов. Но есть и исключения. Например, файловая система XFS использует все сектора. Если при этом информация загрузочного сектора раздела XFS будет стерта, то файловая система разрушится!

Запуск компьютера. При запуске BIOS считывает запись MBR первого жесткого диска, загружает ее в оперативную память и проверяет, содержатся ли в двух последних байтах шестнадцатеричные коды 55 AA. Эти коды служат для идентификации носителей с возможностью загрузки. Если коды совпадают, то выполняется минипрограмма с загрузочного сектора. На компьютере, где установлена система MS DOS или Windows (не Linux), такая программа «узнает», какие из системных разделов помечены как активные (обычно это первый раздел). Затем эта программа загружает еще одну программу — из загрузочного сектора активного раздела — и выполняет ее. И уже эта программа отвечает за запуск Windows или MS DOS.

Если на компьютере установлено несколько жестких дисков, можно настроить в BIOS, в каком порядке будут запрашиваться жесткие диски при загрузке. Таким образом, в современных компьютерах система может загружаться с внешнего диска или USB-флешки. BIOS можно настроить и так, чтобы загрузка производилась с CD или DVD.

Существует несколько стратегий, позволяющих обеспечить мирное сосуществование Windows и Linux.

- Самый обычный способ заключается в том, чтобы установить загрузчик Linux в MBR и выбирать, какую систему запускать, — Windows или Linux.
- Если на компьютере уже установлена современная версия Windows (новее 9x/ME), можно настроить загрузчик Windows так, чтобы он запускал GRUB. Достоинство в том, что MBR даже не нужно трогать. Недостаток состоит в том, что процесс установки относительно сложен. Кроме Ubuntu с WUBI, мне не известен ни один дистрибутив Linux, который мог бы выполнять такую операцию. Всегда требуется работать вручную (см. раздел 14.5).
- Третий вариант заключается в том, чтобы установить GRUB в загрузочный сектор основного раздела и пометить этот раздел как активный. При использовании такого варианта есть то же существенное достоинство, что и в предыдущем случае: не нужно трогать MBR. Недостаток в том, что метод работает лишь с основ-

ными разделами (но не с логическими) и лишь с теми файловыми системами, которые не касаются загрузочного сектора вашего раздела. Поскольку эти ограничения существуют, мы этот вариант далее рассматривать не будем.

Создание резервной копии MBR

Если при установке GRUB с MBR что-то пойдет не так, в самом тяжелом случае вы не сможете запустить ни Windows, ни Linux. Тогда придется прибегнуть к «живой» системе Linux, например Knoppix, или использовать установочный CD/DVD для Windows, чтобы исправить загрузочный сектор. «Ремонт» будет наименее проблематичен, если у вас будет резервная копия MBR. Для этого нужно выполнить одну из следующих команд (в зависимости от того, под каким именем устройства запрашивается первый жесткий диск вашего компьютера):

```
root# dd if=/dev/sda of=/boot/bootsektor.scsi bs=1 count=446
```

Для того чтобы восстановить загрузочный сектор, выполните следующую команду:

```
root# dd if=/boot/bootsektor.scsi of=/dev/sda bs=1 count=446
```

Вышеуказанные команды считывают или изменяют только 446 байт MBR. Причина в том, что в MBR содержится и таблица разбиения основных разделов. Если вы изменяете всю MBR (то есть 512 байт), то в межоперационное время могут быть потеряны изменения, внесенные в таблицу разбиения разделов. Тогда компьютер, конечно, загрузится, но если вам немного не повезет, вы потеряете целые разделы с данными! Строение MBR подробно описано здесь: http://de.wikipedia.org/wiki/Master_Boot_Record.

Установка в MBR жесткого диска

В дальнейшем предполагается, что вы уже создали конфигурационный файл GRUB `/boot/grub/menu.lst`. Тогда в `/boot/grub` должны находиться файлы (`stage1`, `stage2` а также `*_stage1_5`). Если их нет, скопируйте туда файлы GRUB. Точный путь может отличаться от дистрибутива к дистрибутиву. Например, файлы GRUB в Red Hat или Fedora находятся в каталоге `/usr/share/grub/i386-redhat`. При необходимости просмотрите список файлов пакета GRUB:

```
root# cp /usr/lib/grub/* /boot/grub
```

Затем запустите GRUB и выполните в нем следующую команду `setup`. Вместо `(hd1,12)` нужно указать имя устройства GRUB того раздела вашего жесткого диска, в котором находится каталог `/boot`. Обратите внимание — часто каталог `/boot` находится в собственном разделе, а не в системном! Переменная `(hd0)` обозначает место, куда должен быть установлен GRUB, то есть загрузочный сектор первого жесткого диска.

```
root# grub
grub> root (hd1,12)
grub> setup (hd0)
grub> quit
```

grub.conf. В SUSE при установке дистрибутива создается файл `/etc/grub.conf`. В нем содержатся команды, касающиеся установки GRUB. Установку можно повторить с помощью команды `grub < /etc/grub.conf`.

В актуальных версиях Fedora и Red Hat также существует файл `/etc/grub.conf`, но в данном случае он представляет собой ссылку на `/boot/grub/grub.conf`. В этих дистрибутивах `menu.lst` — это тоже ссылка на `grub.conf`.

Установка в загрузочный сектор жесткого диска

GRUB можно установить не в MBR, а в загрузочный сектор любого жесткого диска. Правда, обычно такая установка оказывается неэффективной, поскольку этот загрузочный сектор при запуске системы не учитывается. Однако в трех случаях подобная установка все же может быть полезна.

- Если GRUB запускается опосредованно, через загрузчик Windows (см. подраздел «Запуск GRUB с помощью загрузчика Windows» раздела 14.5).
- Если GRUB запускается опосредованно, через загрузчик Linux (например, GRUB), который уже находится в MBR, а вы не хотите его трогать. Этот вариант возможен прежде всего в тех случаях, когда вы намереваетесь параллельно использовать несколько экземпляров Linux (см. подраздел «Управление несколькими системами, установленными на одном компьютере» раздела 14.5).
- GRUB устанавливается в загрузочный сектор основного раздела, и вы помечаете этот раздел как «активный» с помощью команды `fdisk` (клавиша A, команда `toggle a bootable flag`). В таком случае программа, находящаяся в MBR, учитывает загрузочный сектор активного раздела. Данный метод не работает для логических разделов, а также в тех случаях, когда в MBR уже установлен GRUB или другой загрузчик.

Обычно для такой установки применяется системный раздел Linux. Иначе говоря, если вы установили Linux в раздел `/dev/sda7` и хотите установить GRUB в загрузочный сектор этого же раздела, выполните приведенные ниже команды. Единственное отличие по сравнению с установкой в MBR состоит в том, что в `setup` вы указываете не `(hd0)`, а нужный раздел.

```
root# grub
grub> root (hd1,12)
grub> setup (hd0,6)      (Установка в загрузочный сектор /dev/sda7)
grub> quit
```

ВНИМАНИЕ

В некоторых файловых системах загрузочный сектор раздела нельзя использовать с помощью загрузчика или других программ. К таким системам относится XFS. Если установить GRUB в загрузочный сектор раздела XFS, то файловая система будет разрушена! По этой причине в таких системах установка в загрузочный сектор не применяется.

Установка на USB-носитель

В BIOS современных компьютеров обычно предусмотрена возможность загружать систему с USB-носителя. В принципе вполне можно установить GRUB

в загрузочный сектор флешки и загружать с его помощью Windows, Linux и т. д. Теоретически все просто, но на практике часто возникают проблемы. Есть две основные причины проблем.

- Материнская плата должна правильно распознавать USB-носитель уже при загрузке и работать с ним как со средством загрузки. Обратите внимание и на то, что флешку можно отформатировать двумя способами: как «супердискету» (superfloppy) или как жесткий диск. Какой вариант будет использоваться, зависит от BIOS.

  Учитывайте также, что необходимо активизировать поддержку USB в BIOS (обычно для этого предназначается специальный параметр BIOS). В отличие от Linux, GRUB может обращаться к USB-носителям только через BIOS!

- Если BIOS опознает USB-носитель как загрузочный диск, то при этом (как минимум в некоторых версиях BIOS) изменится порядок, в котором GRUB будет «видеть» носители с данными. Теперь первым диском (`hd0`) будет считаться USB-носитель, встроенные жесткие диски будут запрашиваться через (`hd1`), (`hd2`) и т. д. При необходимости соответствующим образом измените `/boot/grub/devices.map` перед установкой GRUB.

Оптимальная стратегия — сначала попробовать установить GRUB на флешке. Когда Linux запущена, GRUB опознает флешку под названием (`hd`*n*`+1`), где *n* — это последний внутренний жесткий диск. Если ваша система Linux, как в предыдущих примерах, находится в разделе `/dev/sdb13` и в компьютере имеется два внутренних диска, флешка будет называться (`hd2`). Для того чтобы установить GRUB в MBR флешки, используйте следующие команды:

```
root# grub
grub> root (hd1,12)
grub> setup (hd2)      (Установка в MBR USB-носителя)
grub> quit
```

В идеальном случае после перезапуска компьютер обнаружит GRUB в MBR на флешке и, как и предполагалось, загрузит операционные системы, указанные в `menu.lst`. Если при запуске операционных систем возникнут сложности, перейдите из меню GRUB в интерактивный режим, нажав клавишу C, а затем, воспользовавшись командой `cat` и клавишей табуляции, узнайте, под какими названиями GRUB «видит» жесткие диски. Нажатие Esc выводит вас обратно в меню, где с помощью клавиши E можно изменить команды загрузки и испробовать их еще раз.

Установка Linux на жестких дисках, подключаемых через USB. До сих пор предполагалось, что USB-флешка будет использоваться только для запуска загрузки. Но ситуация осложняется, если и сама Linux находится на большой флешке или на внешнем жестком диске, подключаемом через USB. В большинстве дистрибутивов можно без труда установить USB-носитель, но при запуске могут возникнуть проблемы. Вам придется преодолеть три основных препятствия.

- **GRUB** — как было указано выше, уже при установке GRUB на USB-носитель могут возникать проблемы. Вы можете попробовать, получается ли такая установка. При необходимости поэкспериментируйте с настройками BIOS и обозначениями приводов и попробуйте настроить эти свойства в файле `menu.lst` вручную.

- **USB-модули для ядра** — ядро уже в момент запуска должно быть «в состоянии» обратиться к USB-носителю. Чтобы это получилось, в файле `Initrd` должны храниться все необходимые USB-модули. Базовые сведения о том, как построены файлы `Initrd` и как они создаются, сообщаются в следующем разделе.
- **Названия устройств** — в зависимости от того, как загружается компьютер — с подключенными внешними жесткими дисками (USB) или без них, — названия устройств жестких дисков могут изменяться (`/dev/sda` на `/dev/sdb`). Поэтому целесообразно использовать в файле `/etc/fstab` и при задании корневых параметров в строке `kernel` в `menu.lst` не названия устройств, а номера UUID (см. также разделы 13.5 и 14.9). Если вы работаете с Ubuntu, то в `menu.lst` с помощью `uuid` также можно выбрать тот раздел, в котором будут находиться ядро и файл `Initrd`.

При работе с современными материнскими платами, как правило, можно без проблем использовать Linux, установленную прямо на USB-носителе. Однако, чтобы все заработало, порой приходится потрудиться и потратить время. Если вы только начинаете работать с Linux, не рекомендую использовать этот вариант.

14.5. Внутренняя организация GRUB и особые случаи

В следующих разделах рассмотрены различные детали, важные при работе с GRUB, внутренняя организация этой программы и особые случаи, которые могут возникнуть при работе. В числе прочего мы разберем защиту с помощью пароля, управление несколькими системами, установленными на одном компьютере, создание файлов `Initrd` и т. д.

Защита паролем

Гибкость, характерная для GRUB, может не только приносить пользу, но и представлять опасность. Например, пользователь GRUB может воспользоваться командой `cat` и считать все файлы, сохраненные в разделах с файловыми системами ext2, ext3, reiserfs или vfat. Поэтому обычно следует защищать GRUB паролем.

Для этого запустите команду `grub` и выполните вместе с ней `md5crypt`. Система потребует от вас ввести пароль. GRUB ответит зашифрованной последовательностью символов:

```
root# grub
grub> md5crypt
Password: ******
Encrypted: $1$FWk/60$QfckeBVBoaWNBm274USHO0
```

Теперь с помощью ключевого слова `password` добавьте эту последовательность символов в файл меню GRUB:

```
# в глобальной области /boot/grub/menu.lst
password --md5 $1$FWk/60$QfckeBVBoaWNBm274USHO0
```

Благодаря этому пользователи GRUB могут выбирать и выполнять любые команды меню, но доступ к другим интерактивным возможностям загрузчика для них закрыт. Если пользователи хотят изменить параметры загрузки ядра или выполнять другие команды, им сначала потребуется нажать клавишу P, а потом указать пароль.

Возможно, вы захотите защитить паролем отдельные записи меню. Для этого введите прямо в строку `title` команду `lock`. Теперь такую строку меню можно будет использовать лишь после ввода пароля. Если вы хотите, чтобы GRUB вообще нельзя было использовать, не указав пароль, добавьте `lock` во все записи меню.

Запуск GRUB с помощью загрузчика Windows

Если вы устанавливаете на жестком диске и Windows, и Linux, то GRUB обычно устанавливается в MBR. Тогда загрузчик запускает либо Windows, либо Linux. Такой метод был описан выше в этой главе.

Если вы работаете с современной версией Windows (не с Windows 9x/ME), то можно применить и обратный метод: вы не трогаете MBR и устанавливаете GRUB в загрузочный сектор вашего системного или загрузочного раздела (см. также раздел 14.4). Затем загрузчик Windows конфигурируется так, чтобы он запускал GRUB. Достоинство этого метода заключается в том, что GRUB будет работать и после установки другой версии Windows. Обычно этого не происходит, так как Windows стирает информацию, находящуюся в MBR.

Сначала в Linux запускается команда `grub`, и с ее помощью загрузчик GRUB устанавливается в загрузочный сектор системного или загрузочного раздела. Для выполнения следующих команд необходимо, чтобы в `/boot/grub/*` уже находились все важные файлы GRUB, в том числе `menu.lst`. Чтобы узнать названия устройств системного раздела, используйте команду `df`.

```
root# df /
Файловая система 1K-блоки Использовано Доступно Использовано% Подключено к
/dev/sda3        14417392      4647540  9037488           34% /
root# grub
grub> root (hd0,2)
grub> setup (hd0,2)
```

Windows NT, 2000, XP

Дальше процесс будет зависеть от того, с какой версией Windows вы работаете. Если вы используете Windows XP или более старую версию, скопируйте загрузочный сектор GRUB в файловую систему Windows с помощью указанной ниже команды. Предполагается, что файловая система доступна через каталог `/media/windows`.

```
root# mount /media/floppy
root# dd if=/dev/sdb13 of=/media/windows/grubfile bs=512 count=1
```

Затем перезапустите компьютер и загрузите Windows. Измените конфигурационный файл загрузчика Windows так, чтобы можно было запустить загрузчик GRUB,

находящийся в `grubfile`. Для этого просто вставьте в текстовый файл `C:\boot.ini` следующую строку:

```
C:\grubfile="GRUB"
```

Теперь, после перезапуска GRUB, вы увидите, что GRUB будет добавлен в список имеющихся операционных систем отдельной строкой. Если выбрать этот параметр, то загрузчик Windows запустит GRUB. Там вы сможете выбрать нужный дистрибутив Linux.

Windows Vista, 2008 и 7

В Windows Vista и во всех более новых версиях Windows используется иной загрузчик. Он конфигурируется с помощью двоичного файла, для изменения которого из всего арсенала команд подходит только `bcdedit`, да и при ее использовании придется потрудиться. Чтобы не мучиться с `bcdedit`, просто установите бесплатную программу EasyBCD с сайта http://neosmart.net/software.php (рис. 14.3).

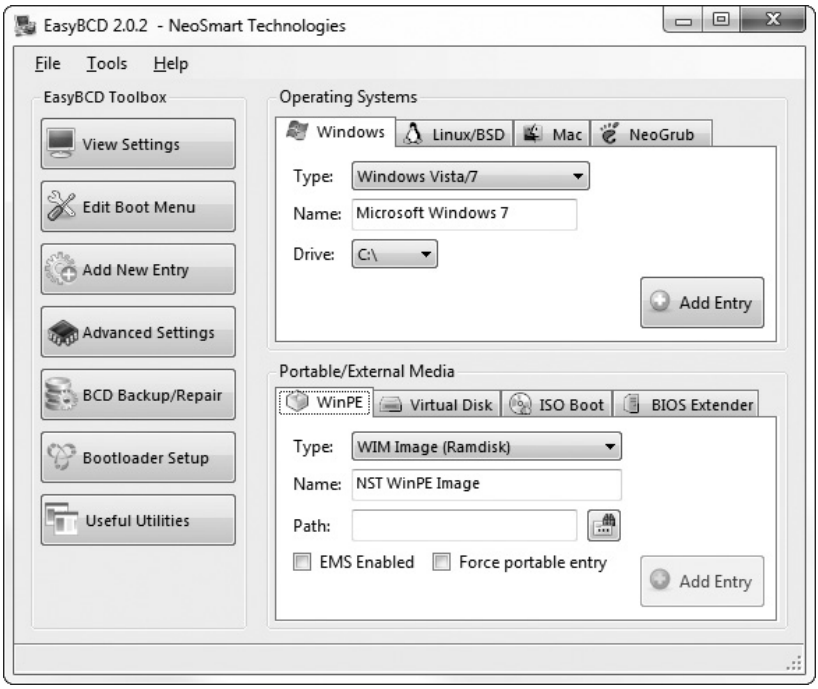

Рис. 14.3. Окно программы EasyBCD

В этой программе, нажав кнопку Добавить/Удалить записи, вы вставляете новую запись в меню загрузки Windows. При этом нужно выбрать в диалоговом окне вкладку Linux, на ней — менеджер загрузки GRUB, а затем прямо указать жесткий диск и номер раздела, в который вы установили GRUB. В отличие от Windows XP и других версий, не требуется предварительно копировать загрузочный сектор раздела в файловую систему NTFS локального компьютера.

Но будьте внимательны при указании номера жесткого диска и раздела: у жестких дисков нумерация начинается с 0, а у разделов — с 1. Дополнительные разделы, а также неиспользуемые основные разделы в нумерации не учитываются! Например, если ваш диск секционирован так, как показано в следующем примере, укажите для hdb13 диск 1 и раздел 10:

```
\dev\hdb1 основной раздел, Linux      1
\dev\hdb2 дополнительный раздел
\dev\hdb3 (не используется)
\dev\hdb4 (не используется)
\dev\hdb5 логический раздел подкачки  2
\dev\hdb6 логический раздел, Linux    3
...
\dev\hdb13 логический раздел, Linux  10
```

Управление несколькими системами, установленными на одном компьютере

Если вы хотите установить на компьютере несколько операционных систем, то лучше всего начать с Windows.

ПРИМЕЧАНИЕ

В принципе можно установить сначала Linux, а потом Windows. Но Windows гораздо «прихотливее» относится к месту для установки. Проблемы сводятся к минимуму, если установить Windows в первый раздел первого жесткого диска. Последующие установки этой системы протекают без проблем, при условии что вы установили загрузчик Windows.

При установке на компьютер первого дистрибутива Linux настройте GRUB так, чтобы с его помощью можно было запускать и Windows, и Linux. Во многих ситуациях это означает, что загрузчик понадобится установить в MBR, но описанный выше вариант с применением загрузчика Windows также вполне уместен.

Самое интересное начнется при установке следующих дистрибутивов Linux. Можно выбрать один из трех вариантов.

- **Перезаписать GRUB** — в таком случае вы заменяете имеющийся экземпляр GRUB новым. Будьте внимательны, иначе программа автоматически выберет именно этот способ.

  Установочные программы некоторых дистрибутивов, включая Debian, Ubuntu, SUSE, но исключая Fedora, достаточно хитроумны — они анализируют все разделы жесткого диска. Если они обнаруживают, что на диске уже есть дистрибутив Linux, то автоматически создают новую запись в меню. В идеальном случае, завершив новую установку, вы сможете запустить все операционные системы, установленные на компьютере.

- **Изменение имеющейся конфигурации GRUB** — при установке Linux вы должны прямо отказаться от создания загрузчика. В большинстве установочных программ такая возможность предусмотрена, хотя иногда она бывает не на виду.

  Разумеется, это означает, что вы не сможете сразу же пользоваться только что установленным дистрибутивом Linux. Вместо этого запустите первую систему

Linux, установленную на вашем компьютере, и добавьте в ее конфигурационный файл `menu.lst` строки, нужные для запуска нового дистрибутива Linux. Для этого важно уже достаточно хорошо разбираться в Linux.

- **Установка GRUB в системном разделе и последующий опосредованный запуск** — в таком случае вы выбираете установку GRUB в загрузочный сектор нового системного раздела. Подобная возможность предусмотрена практически в любой установочной программе. Но вам, вероятно, придется эту возможность немного поискать.

  В таком случае вы также не сможете сразу же пользоваться новым экземпляром Linux. Сначала потребуется добавить в конфигурационный файл загрузчика вашего первого дистрибутива Linux несколько строк, чтобы центральный GRUB мог запустить GRUB нового дистрибутива (см. раздел 14.3). Однако внести такое изменение значительно проще, чем при предыдущем варианте установки. Если вы с самого начала планируете установить несколько дистрибутивов Linux сразу, можете заблаговременно подготовить в центральном конфигурационном файле GRUB несколько записей для новых разделов, например по следующему образцу.

```
Дополнение в /boot/grub/menu.lst
title запуск GRUB в загрузочном секторе /dev/sda7
  rootnoverify (hd0,6)
  chainloader +1
title запуск GRUB в загрузочном секторе /dev/sda8
  rootnoverify (hd0,7)
  chainloader +1
title запуск GRUB в загрузочном секторе /dev/sda9
  rootnoverify (hd0,8)
  chainloader +1
```

Надо отметить, что я всегда пользовался третьим вариантом — а мне не раз и не два доводилось параллельно устанавливать на компьютере несколько дистрибутивов. Основное преимущество третьего варианта перед двумя предыдущими заключается в том, что для каждого дистрибутива предусматривается отдельный конфигурационный файл GRUB. Это особенно важно при обновлениях ядра: почти в любом дистрибутиве при таком обновлении изменяется файл `menu.lst`. Однако при использовании вариантов 1 и 2 такие изменения обычно никак не влияют на систему, так как есть только один действующий конфигурационный файл GRUB сразу для всех дистрибутивов.

Учтите и следующий момент: в документации GRUB 2 открытым текстом не рекомендуется устанавливать программу в системный раздел и делается указание на то, что установка в MBR (вариант 1) более надежна и, следовательно, предпочтительна. При испытаниях, которые я проводил, возникали проблемы при попытке передать управление от GRUB 1 к GRUB 2, находящихся в системном разделе. Следовательно, если вы устанавливаете на своем компьютере сразу несколько дистрибутивов, одни из которых использует GRUB 1, а другие — GRUB 2, то в MBR должен быть установлен экземпляр GRUB 2.

Пример использования варианта 3. Третий вариант лучше всего продемонстрировать на примере. Допустим, в разделе `/dev/sda1` на вашем компьютере находится

Windows XP. Теперь вам нужно установить вторую операционную систему — openSUSE. Для этого создается дополнительный раздел /dev/sda2. Системный раздел SUSE оказывается в первом логическом разделе /dev/sda5, а раздел подкачки — в /dev/sda6. GRUB устанавливается в MBR /dev/sda. Соответствующий конфигурационный файл GRUB — /boot/grub/menu.lst — будет выглядеть, как в следующем листинге:

```
default 0
timeout 30
title Запуск SUSE
  kernel (hd0,4)/boot/vmlinuz root=/dev/sda5 vga=0x317
  initrd (hd0,4)/boot/initrd
title Запуск Windows
  rootnoverify (hd0,0)
  chainloader +1
```

Теперь требуется установить еще один дистрибутив Linux, предположим, Fedora. Его системный раздел называется /dev/sda7, а раздел подкачки остается без изменений. Установка GRUB производится в загрузочном секторе /dev/sda7.

При следующем перезапуске GRUB автоматически загружает SUSE, то есть тот дистрибутив Linux, который был установлен раньше. Теперь добавим в файл /boot/grub/menu.lst три строки, позволяющие запустить GRUB системы Fedora. Дополненный конфигурационный файл GRUB /boot/grub/menu.lst (находящийся в файловой системе дистрибутива SUSE) теперь выглядит так:

```
default 0
timeout 30
title Запуск SUSE
  root (hd0,4)
  kernel /boot/vmlinuz root=/dev/sda5 vga=0x317
  initrd /boot/initrd
title Запуск GRUB системы Fedora (/dev/sda7)
 rootnoverify (hd0,6)
  chainloader +1
title Запуск Windows
  rootnoverify (hd0,0)
  chainloader +1
```

При следующем перезапуске компьютер предлагает начать работу с Windows, SUSE или Fedora. Если вы выберете Fedora, то запустится GRUB этого дистрибутива, находящийся в /dev/sda7. Он будет интерпретировать собственный конфигурационный файл GRUB, находящийся в файловой системе Fedora.

ПРИМЕЧАНИЕ

При работе с несколькими операционными системами спорным вопросом является обращение с каталогом /home. Казалось бы, логично использовать для всех дистрибутивов общий раздел /home, чтобы личная информация и настройки были в вашем распоряжении независимо от того, в какой операционной системе вы сейчас работаете. Однако на практике в подобном случае часто возникают конфликты и проблемы совместимости, поэтому я советую так не поступать. Компромиссное решение — создать собственный раздел для хранения данных и поместить там избранные каталоги (например, ~/.mozilla), проставив символьные ссылки из соответствующих каталогов /home, чтобы можно было использовать все личные данные вместе.

LVM и RAID

GRUB 0.9.n не в состоянии запускать системы Linux, загрузочные файлы которых находятся в разделах LVM или RAID. В таком случае вам потребуется отдельный раздел `/boot`, который расположен вне областей LVM или RAID (такое ограничение снимается только в GRUB 2).

Независимо от этого, при установке с применением LVM или RAID вы должны учитывать, что функции или модули, необходимые для доступа к разделам LVM и RAID, содержатся или прямо в ядре, или в файле `Initrd`.

Файловые системы ext4 и brtfs

GRUB пока несовместим с новыми файловыми системами ext4 и brtfs. Для ext4 уже существует неофициальная заплатка, интегрированная, например, в Ubuntu 9.04. Если она отсутствует (например, в Fedora 11) или в вашем системном разделе используется файловая система brtfs, вам потребуется отдельный загрузочный раздел с файловой системой ext3.

Создание файла Initrd

В файле меню GRUB для каждого дистрибутива Linux всегда указывается два файла: ядро Linux и относящийся к нему файл `Initrd`. Оба файла обычно находятся в каталоге `/boot`, но их названия отличаются от дистрибутива к дистрибутиву. В файле `Initrd` расположены те модули, которые нужны ядру для работы еще до того, как оно сможет обратиться к файловой системе, чтобы при необходимости считать оттуда другие модули. GRUB предоставляет ядру файл `Initrd` в виде псевдодиска, находящегося в оперативной памяти.

Обычно в файле `Initrd` имеется модуль ядра с драйвером системы, необходимым для работы системного раздела. Если системный раздел находится на жестком диске SCSI, то в файле `Initrd` должен находиться драйвер для вашей SCSI-карты. Если вы применяете LVM или RAID (см. главу 13), то и в этом случае нужны специальные драйверы.

В файле `Initrd` должны содержаться модули ядра, версии которых строго совпадают с версией самого ядра. Поэтому всякий раз, когда вы устанавливаете или самостоятельно компилируете новое ядро, требуется заново создавать и файл `Initrd`, который подойдет именно для этого ядра. При обновлении ядра этот процесс обычно осуществляет программа, выполняющая обновление. Если же вы, напротив, самостоятельно устанавливаете новое ядро, то вам самим придется создать и файл `Initrd`.

Файл `Initrd` необязателен. Если в ядре вашей системы содержатся все компоненты, необходимые для процесса загрузки, то систему можно запустить и без файла `Initrd`. Однако для этого ядро нужно скомпилировать особым образом (почти все модули ядра Linux можно интегрировать прямо в ядро, правда, файл ядра от этого увеличится). Тогда в файле GRUB не нужна строка `initrd`.

В большинстве новых дистрибутивах файлов `Initrd` как таковых уже нет. Их роль выполняют файлы `initramfs`, строение которых описано несколько ниже. Однако,

поскольку и в параметрах GRUB/LILO, и в командах для создания файлов используется термин Initrd, а ядро правильно интерпретирует файл, хотя он и назван некорректно, мы и в дальнейшем будем употреблять термин Initrd, пусть это и заведомо неверно.

К сожалению, процесс создания файлов Initrd нисколько не стандартизирован. В каждом дистрибутиве для этого применяются собственные инструменты. В файлах Initrd содержатся не только модули ядра, но и сценарии, предназначенные для инициализации оборудования. Их выполнение длится сравнительно долго, и это мешает разработчикам — слишком тормозится процесс загрузки. По этой причине планируется провести коренную реконструкцию системы Initrd. Первопроходцем в данном отношении стала система Fedora 12 с программой Dracut.

Сценарий update-initframs (Debian, Ubuntu)

В Debian и Ubuntu созданием и администрированием файлов Initrd занимается сценарий update-initramfs. В простейшем случае требуется всего лишь указать параметр -u, чтобы обновить версию файла Initrd до соответствия новейшей версии ядра. Если вы хотите обновить файл Initrd до другой версии ядра, укажите номер версии с помощью параметра -k. Задание -k all позволит обновить файлы Initrd для всех установленных версий ядра.

С помощью параметров -c или -d сценарий update-initramfs создает новый файл Initrd, удаляя имеющийся. В таком случае нужно обязательно указать версию ядра с помощью параметра -k.

```
root# update-initramfs -c -k 2.6.28-13-generic
update-initramfs: Generating /boot/initrd.img-2.6.28-13-generic
```

«За кулисами» происходит следующее: чтобы создавать файлы Initrd, сценарий update-initramfs обращается к сценарию mkinitramfs. Базовая конфигурация осуществляется в файле /etc/initramfs-tools/initramfs.conf, а также в файлах каталога /etc/initramfs-tools/conf.d. Кроме того, к файлу Initrd добавляются все модули, перечисленные в /etc/initramfs-tools/modules (одна строка — один модуль).

Сценарий mkinitramfs в ходе стандартной конфигурации (с использованием MODULES=most в initramfs.conf) создает очень большие файлы Initrd, в которых содержатся бесчисленные дополнительные модули, в том числе все важнейшие файловые системы Linux, различные драйверы USB, SCSI и SATA, а также сетевые драйверы и драйверы NFS.

Если вы обращаетесь напрямую к mkinitramfs (это требуется редко), следует как минимум передать имя нового файла Initrd (параметр -o). Если необходимо создать файл Initrd не для текущей версии ядра, то дополнительно укажите нужную версию:

```
root# mkinitramfs -o myinitrd 2.6.28-13-generic
```

Команда mkinitrd (Red Hat, Fedora)

Эта команда создает файл Initrd, содержащий все модули, необходимые для доступа к системному разделу (модули файловых систем, LVM и RAID). Если

системный раздел находится на диске SCSI, то mkinitrd берет нужный модуль SCSI из файла /etc/modprobe.conf.

Модули, которые не опознаются программой автоматически, необходимо прямо указать с помощью параметра --with=*имя_модуля*. При этом для каждого модуля требуется собственный параметр --with. Если вы хотите, чтобы определенный модуль загрузился *раньше* SCSI-модулей, используйте вместо --with параметр --preload=*имя_модуля*.

Кроме этого, в качестве параметров необходимо передать название файла псевдодиска и точную версию ядра. Если вы хотите заменить уже имеющийся файл псевдодиска новым файлом, вам понадобится параметр -f. Еще несколько параметров описано в тексте справки man mkinitrd.

```
root# mkinitrd /boot/initrd-2.6.29.4-162.fc11.i686.PAE.img \
      2.6.29.4-162.fc11.i686.PAE
```

В Fedora 12 и выше, а также, предположительно, в RHEL 6 и выше вместо mkinitrd будет применяться новая команда dracut.

Сценарий mkinitrd (SUSE)

В SUSE команда mkinitrd работает несколько иначе, чем в Red Hat. Как правило, этой команде не требуется передавать никаких параметров. Она автоматически создает файлы initrd для всех файлов ядра, которые находит в каталоге /boot. Новые файлы получают названия вида /boot/initrd-*nnn*, где *nnn* — версия ядра. Кроме того, mkinitrd создает ссылку, указывающую из /boot/initrd на файл initrd, подходящий к vmlinuz.

Если вы хотите создать только определенный файл initrd, то можете указать версию ядра и файла initrd с помощью параметров -k и -i (по умолчанию — в каталоге /boot). Команда mkinitrd интерпретирует переменную INITRD_MODULES из файла /etc/sysconfig/kernel. В этой переменной указаны все модули, необходимые для загрузки, и она может выглядеть, например, так:

```
# в /etc/sysconfig/kernel
INITRD_MODULES="processor thermal ata_piix ata_generic piix ide_pci_generic fan
                jbd ext3 edd"
```

Дополнительные модули указываются с помощью параметра -m. Более подробную информацию о mkinitrd выдает параметр -h, а также справка man mkinitrd и исходный текст сценария (файл /sbin/mkinitrd).

Просмотр файла initrd

Файлы initrd для версии ядра 2.6.n внутри системы представлены как файлы initramfs. Initrd — это сжатый архивный файл (с расширением CPIO), в состав которого входят различные файлы и каталоги. Если вы хотите просмотреть содержимое архива, действуйте так:

```
root# cd /boot
root# cp initrd-n.n initrd-test.gz
root# gunzip initrd-test
root# mkdir test
```

```
root# cd test
root# cpio -i < ../initrd-test
root# ls -lR
```

Ссылки. Подробная информация о внутреннем строении системы initramfs сообщается в файле `ramfs-rootfs-initramfs.txt`, входящем в состав документации ядра: http://www.kernel.org/doc/Documentation/filesystems/ramfs-rootfs-initramfs.txt.

Особые случаи и сложности, которые могут возникать при работе с файлами Initramfs, обсуждаются здесь: http://lwn.net/Articles/191004/.

Обновления ядра

Если дистрибутив обновляет ядро (при этом изменяется название файла ядра), то вам следует аналогичным образом изменить и файл меню GRUB, а также создать файл `initrd`, подходящий для нового ядра. В большинстве дистрибутивов эти процессы выполняются автоматически, в рамках управления обновлениями, так что после перезапуска компьютер автоматически начинает использовать новое ядро (однако в некоторых дистрибутивах в меню GRUB остается запись и для старого ядра, чтобы при проблемах с обновлениями можно было продолжать использовать систему со старым ядром).

Проблем не избежать, если файл меню GRUB будет находиться не в `/boot/grub`, а в совершенно другом разделе (так как у вас на компьютере несколько дистрибутивов Linux и всего один активный файл меню GRUB). В таком случае вам придется самостоятельно вносить изменения в файл меню GRUB после каждого обновления. Такая же ситуация возникает, если вы обновляете ядро сами.

14.6. GRUB — экстренные меры

Что делать, если GRUB не работает? В худшем случае после перезапуска вы не сможете включить ни Linux, ни какую-либо другую операционную систему. В этом разделе собраны некоторые рекомендации, которые, возможно, помогут вам реанимировать компьютер.

Возможные проблемы

Записи меню GRUB не работают. GRUB запускается, но, когда вы выбираете одну из записей меню, компьютер не реагирует, аварийно завершает работу и т. д. Возможно, в записи меню указаны неверные данные. В таком случае попытайтесь вручную ввести команды, необходимые для запуска Linux (см. раздел 14.2). Если после этого удастся запустить систему, исправьте файл меню.

GRUB не отображает меню. После запуска GRUB на экране появляется приглашение ко вводу, а меню не отображается. Иначе говоря, вы оказываетесь в режиме интерактивного ввода команд GRUB. Очевидно, загрузчик не может найти файл меню. В таком случае можете ввести команды для запуска Linux вручную. Затем в Linux еще раз откройте файл меню и заново установите GRUB.

GRUB аварийно завершает работу или не реагирует на команды. Наиболее вероятная причина этой проблемы заключается в том, что в процессе установки что-то не заладилось с файлами `stage1` и `stage2`. Теперь запустить компьютер обычным способом невозможно. Используйте один из следующих вариантов.

- Запустите с установочного CD для Linux аварийную систему и попытайтесь с ее помощью заново установить GRUB.
- Используйте «живой» диск Linux (например, Knoppix) и попытайтесь запустить Linux прямо оттуда. Затем еще раз попробуйте исправить установленную программу GRUB.
- Восстановите загрузочный сектор жесткого диска с помощью установочного CD/DVD для Windows. После этого вы как минимум сможете работать с Windows (с Linux — не сможете).

Восстановление программы GRUB с помощью «живой» системы

Включив «живую» систему, запустите консоль, и выполните в ней команду `su -l` или `sudo -s`, чтобы получить права администратора. Дальше потребуется найти раздел, в котором находится ваш дистрибутив Linux, в частности каталог `/boot`.

```
root# mkdir /test
root# mount /dev/sda3 /test
```

В искомом разделе должен находиться каталог `/boot`, в котором расположены файлы ядра Linux (`vmlinuzxxx`), подкаталог `/boot/grub` и файл меню GRUB `/boot/grub/menu.lst`. Если у вас есть отдельный загрузочный раздел, в нем не будет каталога `/boot`. Файл `vmlinuzxxx` и подкаталог `grub` в таком случае будут находиться прямо в корневом каталоге раздела.

Когда эти подготовительные работы будут завершены, дело останется за малым — установить GRUB в загрузочный сектор жесткого диска, чтобы при запуске компьютера на его основе считывались все остальные файлы GRUB из каталога `/dev/sda3`. При этом `(hd0,2)` — обычное для GRUB обозначение раздела `/dev/sda3` (см. раздел 14.3), а `(hd0)` — обозначение для всего первого жесткого диска, в загрузочный сектор которого должен быть записан GRUB.

```
root# grub
grub> root (hd0,2)  (Место для системного или загрузочного раздела)
grub> setup (hd0)   (Цель установки GRUB: запись MBR первого жесткого диска)
grub> quit
```

14.7. GRUB 2

В настоящее время (на конец 2009 года) выбор версии GRUB во многих дистрибутивах представляет собой выбор из двух зол: либо использовать версию GRUB 0.97 legacy, которая официально давно уже не поддерживается и во многом устарела,

либо остановиться на версии GRUB 2, которая не совсем доработана и плохо документирована. Пока большинство дистрибутивов склоняется к первому варианту, поэтому для работы требуется отдельный раздел `/boot` (LVM, RAID и т. д.).

Первым дистрибьютером, не побоявшимся неизведанного будущего, стала фирма Canonical, которая начала применять GRUB 2 в версии Ubuntu 9.10, поэтому в данном разделе мы рассмотрим именно ту версию GRUB, которая поставляется с Ubuntu 9.10. В следующих изданиях книги будут учтены специфические детали, касающиеся других дистрибутивов.

В данном разделе, говоря о GRUB 2, мы на самом деле будем рассматривать версию GRUB 1.97 — версия GRUB 2.0 еще не готова, и не вполне ясно, появится ли она вообще (ведь до GRUB 1 дело также не дошло — разработки остановились на версии 0.97).

Нововведения

Далее перечислены важнейшие нововведения, появившиеся в GRUB 2 и отсутствовавшие в официальной версии GRUB 0.97 (по состоянию на октябрь 2009 года). С некоторыми дистрибутивами поставляются дополненные заплатками версии GRUB 0.97, в которых также имеются некоторые из функций, перечисленных ниже, но эти версии неофициальные и для вариантов из различных дистрибутивов характерны отдельные специфические черты.

- GRUB 2 совместим с LVM и программным RAID, то есть для работы экземпляров LVM и RAID больше не требуется собственного раздела `/boot`.
- GRUB 2 совместим с ext4.
- GRUB 2 позволяет запрашивать разделы через UUID файловой системы.
- GRUB 2 совместим с альтернативами BIOS: EFI, coreboot (бывший LinuxBIOS).
- При соответствующей конфигурации GRUB 2 может отображать в записях меню любые символы Unicode.
- Внутренняя структура GRUB 2 реализована совершенно по-новому. Загрузчик теперь обходится без этапа `stage1_5`. Дополнительные функции реализуются в форме модулей, динамически загружаемых программой на время выполнения конкретной задачи. Такое нововведение должно обеспечить относительно простое расширение системы и техническую поддержку.
- Конфигурация GRUB заметно усложнилась. Конфигурационный файл `grub.cfg` — результат работы различных сценариев. И в самом файле может содержаться сценарный код на языке, напоминающем оболочку (shell), или на языке LUA.

Кроме этих, уже реализованных нововведений до окончательного выхода версии 2.0 предстоит дополнить общую картину еще несколькими идеями. Сюда, в частности, относится графическое представление загрузочного меню, внешний вид которого можно изменять с помощью *темизации*. При этом в записях меню могут содержаться любые символы UTF-8. Для отображения текстов можно использовать собственные файлы шрифтов. Даже предусмотрен выбор той или иной записи меню с помощью мыши.

В идеальном случае графическое разрешение, выбранное GRUB в ходе общего процесса загрузки и при запуске системы X, должно сохраняться, что, теоретически, должно было бы позволить запуск системы без помех. Получится ли реализовать эту идею на практике — покажет время.

Если внимательно рассмотреть идеи, заложенные разработчиками в GRUB 2, создается впечатление, что этот загрузчик задумывался как миниатюрная операционная система. Сомнительно, что дистрибьютерам и пользователям Linux действительно нужен столь многофункциональный загрузчик.

Недостатки

По некоторым показателям GRUB 2 все же уступает GRUB 0.97.

- GRUB 2 нельзя защитить паролем. Из-за этого возникает серьезный риск, связанный с безопасностью, так как искушенные пользователи GRUB теперь могут получить доступ практически к любому файлу на компьютере.
- В GRUB 2 отсутствуют какие-либо сетевые функции, доступ к сетевым устройствам невозможен.
- GRUB 2 пока недостаточно документирован; это, в частности, осложнило мне работу при написании данного раздела.
- Разработка GRUB 2 еще не завершена. Версия 2.0, точная дата выхода которой еще не известна, должна содержать много новых функций.

К тому же нельзя не отметить еще одну серьезную недоработку: в GRUB 2 совершенно отсутствует поддержка новой файловой системы brtfs (в GRUB 0.97 для этой цели предусмотрена ранняя, хотя и неофициальная версия от разработчиков).

Работа с программой

С точки зрения пользователя, и внешний вид GRUB, и работа с ним практически не изменились. Как и раньше, нажатие клавиши E позволяет изменять отдельные записи меню, а клавиши C — интерактивно выполнять другие команды.

Компоненты и пакеты. Как правило, файлы, необходимые для работы GRUB 2, распределены на несколько пакетов: `grub-common` содержит различные конфигурационные файлы и команды, не зависящие от конкретных платформ, в `grub-pc` собраны файлы, необходимые для работы BIOS. Если на компьютере вместо BIOS применяется EFI, Coreboot и др., то вместо `grub-pc` вам потребуется один из следующих пакетов: `grub-efi-amd64`, `grub-efi-ia32` или `grub-coreboot`.

Для работы `grub-pc` сначала нужно установить пакет `os-prober`. Одноименная команда ищет во всех доступных разделах жесткого диска операционные системы. Результат выполнения `os-prober` поступает в автоматически создаваемое меню GRUB.

Вы также можете установить пакет `grub2-splashimages`. В нем содержится несколько растровых рисунков, которые можно использовать в качестве фона для меню GRUB. Однако этот пакет будет излишним, так как GRUB 2 (в отличие от GRUB 0.97) может считать любой файл в формате PNG или JPEG.

И, наконец, в пакете grub-rescue-pc содержатся IMG- и ISO-файлы, позволяющие сохранить восстановительную систему GRUB на флешке или на CD. В самом крайнем случае вы сможете запустить GRUB с флешки или диска, а затем вручную ввести команды загрузчика или изменить заданные по умолчанию записи меню, чтобы запустить систему.

Ссылки. На следующих сайтах сообщается дополнительная информация по GRUB:

- http://grub.enbug.org/ — «Вики», мануал;
- http://grub.gibibit.com/ — темы и шрифты;
- http://lwn.net/Articles/338337/ — статья в Linux Weekly News;
- http://lists.gnu.org/archive/html/grub-devel/ — почтовый архив разработчиков GRUB.

Базовая конфигурация

Файл grub.cfg

Конфигурация GRUB 2 выполняется совершенно иначе, чем в случае с GRUB 0.97. Основным конфигурационным файлом является /boot/grub/grub.cfg. Он не предназначен для внесения изменений вручную, поэтому права доступа к нему установлены как «только для чтения».

Если хотите изменить конфигурацию, измените файл в каталоге /etc/grub.d или /etc/default/grub (второй вариант характерен для Ubuntu), а затем выполните команду update-grub. В следующем листинге показана автоматически составленная версия grub.cfg (ради экономии места она значительно сокращена). Вас не должна путать строка insmod ext2 — этот модуль GRUB отвечает за работу всех файловых систем ext, в том числе ext3 и ext4.

```
# Файл /boot/grub/grub.cfg
# Не редактируйте этот файл. Он был автоматически сгенерирован
# /usr/sbin/grub-mkconfig с применением шаблонов из /etc/grub.d
# и настроек из /etc/default/grub
# из /etc/grub.d/00_header
# Управление переменными
load_env
set default="0"
if [ ${prev_saved_entry} ]; then
  saved_entry=${prev_saved_entry}
  save_env saved_entry
  prev_saved_entry=
  save_env prev_saved_entry
fi
# Выбор раздела с установочными файлами GRUB
insmod ext2
set root=(hd0,2)
search --no-floppy --fs-uuid --set 23511f16-402c-4364-b281-0baed06f8a6b
# Активизация графического режима, загрузка кодировки
```

```
if loadfont /usr/share/grub/unicode.pf2 ; then
  set gfxmode=640x480
  insmod gfxterm
  insmod vbe
fi
if [ ${recordfail} = 1 ]; then
  set timeout=-1
else
  set timeout=1
fi
# aus /etc/grub.d/05_debian_theme
set menu_color_normal=cyan/blue
set menu_color_highlight=white/blue
# из /etc/grub.d/10_linux
menuentry "Ubuntu, Linux 2.6.31-12-generic" {
  recordfail=1
  save_env recordfail
  set quiet=1
  insmod ext2
  set root=(hd0,2)
  search --no-floppy --fs-uuid --set 23511f16...
  linux /boot/vmlinuz-2.6.31-13-generic root=UUID=23511f16... ro quiet splash
  initrd /boot/initrd.img-2.6.31-13-generic
}
# из /etc/grub.d/20_memtest86+
menuentry "Memory test (memtest86+)" {
  linux16 /boot/memtest86+.bin
}
menuentry "Memory test (memtest86+, serial console 115200)" {
  linux16 /boot/memtest86+.bin console=ttyS0,115200n8
}
# из /etc/grub.d/30_os-prober
menuentry "Windows 7 (loader) (on /dev/sda1)" {
  insmod ntfs
  set root=(hd0,1)
  search --no-floppy --fs-uuid --set 2ca80f2ba80ef35e
  chainloader +1
}
```

Команда update-grub

Команда `update-grub` создает новую версию `grub.cfg`. Эта команда автоматически выполняется после каждого обновления ядра, чтобы `grub.cfg` работал с новым ядром.

По существу, `update-grub` — это просто сценарий, единственная задача которого заключается в том, чтобы вызвать программу `grub-mkconfig`. Она, в свою очередь, интерпретирует конфигурационные файлы и сценарии, описанные ниже. При этом, в частности, создаются записи меню GRUB для всех файлов ядра из каталога `/boot`. Кроме того, просматриваются все доступные разделы. Если там записаны другие

операционные системы, то для них также создаются записи меню GRUB. Поэтому на компьютерах с большим количеством разделов команда update-grub выполняется достаточно долго.

Файл /etc/default/grub

В файле /etc/default/grub содержатся некоторые глобальные настройки GRUB. Не забывайте, что после каждого изменения этого файла вам следует выполнить команду update-grub! В Ubuntu 9.10 в конфигурационном файле содержатся следующие настройки:

```
# Файл /etc/default/grub
GRUB_DEFAULT=0
#GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX=""
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
# GRUB_TERMINAL=console
# GRUB_GFXMODE=640x480
# GRUB_DISABLE_LINUX_UUID=true
# GRUB_DISABLE_LINUX_RECOVERY="true"
```

В GRUB_DEFAULT указывается, какая запись меню GRUB должна использоваться по умолчанию. Как и в GRUB 0.97, допускается настройка "saved", при которой активизируется запись меню, использованная последней. Обратите внимание, что количество записей может изменяться при каждом вызове update-grub и, в частности, при обновлениях ядра. По этой причине такая настройка, как GRUB_DEFAULT=5, нужна достаточно редко.

Переменная GRUB_TIMEOUT указывает, как долго GRUB будет ожидать ввода с клавиатуры, прежде чем активизирует вариант, заданный по умолчанию; GRUB_HIDDEN_TIMEOUT действует подобным образом, только меню не отображается; GRUB_HIDDEN_TIMEOUT_QUIET определяет, отображается ли при этом счетчик оставшегося времени.

Переменная GRUB_DISTRIBUTOR интерпретируется сценарием 10_linux (см. ниже) и указывает название текущего дистрибутива (например, Ubuntu).

Показатели GRUB_CMDLINE_LINUX и GRUB_CMDLINE_LINUX_DEFAULT также учитываются 10_linux и указывают, какие параметры следует передать ядру. Переменные GRUB_CMDLINE_LINUX действуют при каждом запуске, а GRUB_CMDLINE_LINUX_DEFAULT — только если при запуске работают установки, заданные по умолчанию (но не в *режиме восстановления*).

По умолчанию меню GRUB отображается в графическом режиме с разрешением 640 × 480 пикселов. Если вам требуется более высокое разрешение, можете настроить его с помощью переменной GRUB_GFXMODE (при этом, разумеется, учитывайте параметры графической карты). Если вы хотите вообще отказаться от графического режима, активизируйте предусмотренную для этого настройку GRUB_TERMINAL=console. Обе переменные интерпретируются сценарием 00_header. Обратите внимание, что при использовании стандартных настроек картинка в текстовом и графическом режиме внешне одинакова.

Как правило, GRUB указывает корневой каталог запускаемому ядру Linux, сообщая UUID-номер. Если вы предпочитаете в таком случае указывать название устройства (например, `/dev/sda1`), активизируйте строку `GRUB_DISABLE_LINUX_UUID=true`. Эта настройка будет действовать лишь при запуске активного дистрибутива (сценарий `10_linux`), а при работе с другими дистрибутивами будет отключена.

Обычно `update-grub` также создает записи меню для запуска Ubuntu в *восстановительном режиме*. При этом Ubuntu запускается в режиме работы с одним пользователем, не отображая экран-заставку. Если вы хотите отказаться от этих восстановительных записей, активизируйте строку `GRUB_DISABLE_LINUX_RECOVERY="true"`.

Каталог /etc/grub.d

В каталоге `/etc/grub.d` содержится несколько исполняемых сценарных файлов (табл. 14.2). Если следует создать новую версию `grub.cfg`, команда `update-grub` по порядку выполняет все сценарии, находящиеся в `grub.d`. Правда, из-за применения такого метода конфигурация становится очень запутанной: в коде то и дело встречаются команды вида `cat « EOF`, которые переадресуют все следующие строки (до сокращения EOF, означающего «конец файла») в стандартный ввод. Часто сами эти строки содержат сценарный код, который интерпретируется GRUB, уже при запуске системы.

Таблица 14.2. Файлы каталога /etc/grub.d/

Файл	Функция
00_header	Основные настройки GRUB
05_debian_theme	Оптическое оформление
10_linux	Записи меню, используемые при запуске актуального дистрибутива
20_memtest86+	Запись меню для запуска Memtest86 (теста оперативной памяти)
30_os-prober	Записи меню для запуска других операционных систем
40_custom	Образец для создания собственных конфигурационных файлов

В этом списке наиболее интересны сценарии `10_linux` и `30_os-prober`.

Первый передает каждой версии ядра, указанной в каталоге `/boot/`, две записи меню GRUB: одну для обычного запуска и другую — для запуска работы с одним пользователем в восстановительном режиме, без экрана-заставки. Записи в меню сортируются по номерам версий, самая новая версия располагается в начале списка.

Второй вызывает сценарий `os-prober`, который возвращает список всех операционных систем (Linux, Windows, Mac OS X), расположенных на всех доступных разделах жесткого диска. Для каждой из этих операционных систем создается отдельная запись в меню, причем для запуска дистрибутивов Linux применяется имеющаяся конфигурация GRUB или LILO. При этом происходит вызов многочисленных сценариев, входящих в состав пакета `os-prober` (см. команду `dpkg -L os-prober`). Когда я испытывал эти функции, система не только безошибочно опознавала любые дистрибутивы Linux и версии Windows, но и отлично запускала их.

Если вы хотите добавить в grub.cfg несколько записей, вам потребуется записать в каталог grub.d ряд сценариев. Порядок выполнения сценариев определяется их стартовыми номерами (если эти номера одинаковы, то по алфавиту). Не забудьте указать бит *execute*. Кроме того, обращаю ваше внимание, что файл нужно не только встроить в grub.cfg в неизменном виде, но и направить результат его выполнения (стандартный вывод) в grub.cfg! В файле-примере 40_custom продемонстрирован возможный метод: к файлу применяется команда tail (параметр $0). Благодаря параметру -n +3 команда tail отображает файл, начиная с третьей строки, то есть пропускает две первые строки.

```
#!/bin/sh
exec tail -n +3 $0
# Это файл-пример, в котором показано, как добавлять пользовательские записи
...
```

Синтаксис и внутренняя организация

Пока, к сожалению, нет возможности детально описать синтаксис grub.cfg, так как разработчики GRUB на данный момент документировали не все ключевые слова. Наилучшим источником информации послужит следующий сайт, который, правда, уже давно не поддерживается и не совсем актуален: http://grub.enbug.org/FranklinPiat/grub.cfg.manpage.

В дальнейшем я опишу только те ключевые слова, которые встречаются в примерах из этого раздела.

Переменные

С помощью команды set *имя_переменной=значение* присваиваются переменные. Для считывания переменных используйте запись $*имя_переменной*. Если вы интерактивно выполняете команды GRUB, то команда echo $*имя_переменной* отображает имя переменной, а set возвращает все определенные переменные.

Кроме того, отдельные переменные имеют особые значения. К их числу относятся efault, timeout, color_*xxx*, menu_color_*xxx* и в особенности root: при любом обращении к файлу автоматически считывается раздел диска, определенный с помощью root.

GRUB может сохранять переменные на время работы системы. Для этого сначала нужно создать в Linux файл /boot/grub/grub-editenv:

```
root# grub-editenv /boot/grub/grubenv create
```

Теперь GRUB может сохранить в этом файле переменную на время работы системы с использованием команды save_env *имя_переменной* либо считать все переменные из этого файла с помощью load_env. Сначала нужно настроить переменную root, чтобы она указывала на раздел с файлом окружения. Вы также можете читать или изменять переменные GRUB в Linux с помощью команды grub-editenv.

Разделы

В принципе для названия разделов в GRUB 2 применяется та же номенклатура, что и в GRUB 0.97 (см. раздел 14.3). Важное отличие заключается в том, что теперь

первый раздел имеет номер 1 (а не 0). Непостижимо, но это изменение касается только разделов и не касается дисков: первый диск по-прежнему имеет номер 0. Таким образом, `(hd1,5)` обозначает пятый раздел второго жесткого диска.

Работа с UUID

В `grub.cfg` часто используется такая последовательность команд:

```
set root=(hd1,1)
search --no-floppy --fs-uuid --set 12345678...
```

Первая команда инициализирует переменную `root`. Вторая команда ищет файловую систему, имеющую указанный UUID. Если поиск завершится успешно, GRUB, пользуясь параметром `--set`, сохраняет название соответствующего раздела в переменной `root`.

Такая двойственность — своего рода страховочная мера. Она гарантирует, что GRUB обнаружит раздел и в том случае, если файловая система недавно была заново отформатирована (получила другой UUID) или устройство (например, USB-флешка) получило другой номер из-за изменения кабельного соединения.

Модули

С помощью команды `insmod` *имя* GRUB на время работы системы загружает новые модули с дополнительными функциями. GRUB ищет файлы модулей `name.mod` в каталоге `/boot/grub` того раздела, который задан в переменной `root`. Самые важные модули — `raid`, `raid5rec`, `raid6rec` и `mdraid` (программный RAID), `lvm`, `gfxterm` (графическая консоль), `vbe` (графическая система), а также `jpeg`, `tga` и `png` для считывания графических файлов.

Записи меню GRUB вводятся ключевым словом `menuentry`. Последующий текст стоит в кавычках и может содержать интернациональные символы.

ВНИМАНИЕ

Если записи меню содержат ошибки, они просто игнорируются GRUB при работе системы и не отображаются. Соответствующие сообщения об ошибках не выводятся, поэтому найти ошибку становится очень сложно. Лучше всего при работе GRUB перейти в командный режим, нажав клавишу C, и самостоятельно ввести команды, предназначенные для конкретной строки меню.

Запуск Linux

Запись меню GRUB 2, предназначенная для запуска Linux, в минимальном варианте выглядит следующим образом:

```
menuentry "Linux" {
  set root=(hd0,3)
  linux /boot/vmlinuz-n.n.n root=... ro quiet splash
  initrd /boot/initrd.img-n.n.n
}
```

Команда `set root` указывает раздел, в котором расположен файл ядра и файл `initrd`. Ключевые слова `linux` и `initrd` определяют имя файла относительно данного раздела. Вам обязательно потребуется `root` для указания системного раздела и `ro`, чтобы к системному разделу сначала был открыт доступ «только для чтения».

Все остальные параметры зависят от конкретного дистрибутива (также см. раздел 14.3).

Если в вашей системе есть отдельный загрузочный раздел, укажите его с помощью команды `set root`. Для `linux` и `initrd` загрузочный каталог не указывается:

```
menuentry "Linux - Mit eigener Bootpartition" {
  set root=(hd0,2)
  linux /vmlinuz-n.n.n root=... ro quiet splash
  initrd /initrd.img-n.n.n
}
```

Если раздел с файлами ядра и `initrd` входит в состав системы LVM и/или программного массива RAID, вам потребуется дополнительно загрузить соответствующие модули GRUB. При работе с RAID-5 или RAID-6 еще добавляется модуль `raid5rec` или `raid6rec`. В `set root` теперь можно указать системный раздел по образцу `(lv`*имя*`)` или `(md`*n*`)`.

```
menuentry "Linux - Mit Software-RAID" {
  insmod raid mdraid
  set root=(md0)
  linux /boot/vmlinuz-n.n.n root=... ro quiet splash
  initrd /boot/initrd.img-n.n.n
}
menuentry "Linux - Mit LVM" {
  insmod lvm
  set root=(vg1-root)
  linux /boot/vmlinuz-n.n.n root=... ro quiet splash
  initrd /boot/initrd.img-n.n.n
}
menuentry "Linux - LVM auf RAID-5" {
  insmod raid raid5rec mdraid lvm
  set root=(vg1-root)
  linux /boot/vmlinuz-n.n.n root=/dev/mapper/... ro quiet splash
  initrd /boot/initrd.img-n.n.n
}
```

Если вы не хотите полагаться на номера устройств, можете поискать системный раздел командой `search` по номеру UUID. Если команда сработает, то переменная GRUB `root` будет соответствующим образом изменена. Этот метод функционирует и для разделов LVM и RAID, если предварительно были загружены нужные модули GRUB. В Ubuntu по умолчанию за `set root` следует подходящая для данного случая команда `search`, которая позволяет использовать GRUB максимально независимо от номеров устройств.

```
menuentry "Linux - root-Variable anhand UUID einstellen" {
  set root=(hd0,3)
  search --no-floppy --fs-uuid --set 12345678...
  linux /boot/vmlinuz-n.n.n root=... ro quiet splash
  initrd /boot/initrd.img-n.n.n
}
```

К сожалению, у команды `search` есть один недостаток: в системах с большим количеством разделов она работает очень медленно.

Запуск Windows

Чтобы запустить Windows, выберите с помощью `set root` системный раздел и с помощью `chainloader +1` запустите загрузчик этой системы. Обратите внимание, что по умолчанию Windows 7 создает два раздела: один загрузочный раздел имеет размер около 100 Мбайт и содержит файлы `bootmgr` и `bootsect.bak`, а вместе с ним создается более крупный системный раздел. В GRUB нужно указать именно загрузочный раздел. Команду `search`, как всегда, использовать необязательно. Как правило, можно обойтись и без `drivemap`. Эта команда пытается подсказать Windows, что система находится на первом жестком диске, даже если это совсем не так. Иногда требуется сначала запустить Windows.

```
menuentry "Windows 7" {
  set root=(hd0,1)
  search --no-floppy --fs-uuid --set 12345678...
  drivemap -s (hd0) $root
  chainloader +1
}
```

Условный переход и запуск другого загрузчика

Хотя GRUB 2 хорошо справляется с распознаванием других дистрибутивов Linux на жестком диске, я рекомендую использовать для каждого дистрибутива отдельный загрузчик, который устанавливается в каждый системный раздел. Для того чтобы совершить условный переход к другому загрузчику, укажите его раздел с помощью команды `set root` (можно использовать и `search`) и выполните `chainloader +1`:

```
menuentry "GRUB in /dev/sdb7" {
  set root=(hd1,7)
  search --no-floppy --fs-uuid --set 12345678...
  chainloader +1
}
```

Индивидуальная конфигурация

GRUB 2 заранее сконфигурирован так, что вы можете запустить все операционные системы, установленные на компьютере. Стандартная конфигурация работает хорошо, и в большинстве случаев ее вполне хватает для работы. Этот раздел следует изучить только тем пользователям Linux, которые желают адаптировать GRUB для личных нужд.

Отключение сценария os-probe

Обработка сценария `os-probe` может длиться достаточно долго, если на жестком диске много разделов. Если этот сценарий вам не нужен — например, вы охотнее используете для запуска других дистрибутивов Linux записи меню GRUB, которые сами и задали, — поместите строку `GRUB_DISABLE_OS_PROBER=true` в файл `/etc/default/grub`.

Фоновый растровый рисунок

Вы можете расположить меню GRUB на фоне картинки. Растровый рисунок не масштабируется, следовательно, он должен иметь то же разрешение, что и графическая

консоль GRUB. Загрузчик GRUB может работать с форматами JPG, PNG и TGA при условии, что вы загрузили соответствующий модуль GRUB: `jpeg`, `png` или `tga`. Код, необходимый для интеграции фонового рисунка в программу, уже есть в сценарии `05_debian_theme`. Вам нужно всего лишь изменить строку, указав в ней имя файла изображения (в данном случае `/boot/grub/foto.jpg`):

```
# в /etc/grub.d/05_debian_theme
...
for i in /boot/grub/foto.jpg ; do
```

При использовании фонового растрового изображения сценарий применяет два цвета: черный и пурпурный (для записи меню, актуальной в настоящий момент). Эти цвета хорошо подходят только для работы со светлыми картинками. При необходимости добавьте соответствующие команды `set color_`*`xxx`* в конце `05_debian_theme`. При этом в каждой команде следует указать пару цветов для текста и фона. В качестве цвета фона всегда используйте `black` (в данном случае это аналогично прозрачному). Как обычно, после ввода выполните команду `update-grub`, чтобы изменения вступили в силу (рис. 14.4). В итоге файл `grub.cfg` будет выглядеть так:

```
# в /boot/grub/grub.cfg
...
set root=(hd1,1)
search --no-floppy --fs-uuid --set 12345678...
insmod jpeg
if background_image /boot/grub/foto.jpg ; then
  set color_normal=black/black
  set color_highlight=magenta/black
else
  set menu_color_normal=cyan/blue
  set menu_color_highlight=white/blue
fi
```

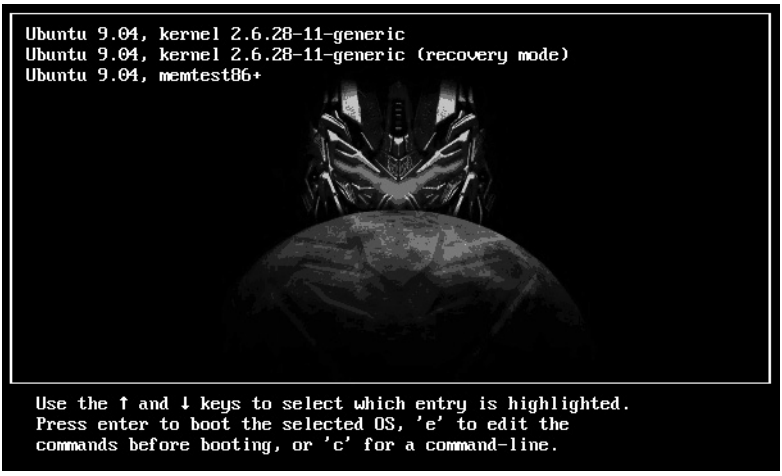

Рис. 14.4. GRUB 2 с индивидуально подобранным фоном

Задаем операционную систему по умолчанию

В /etc/default/grub можно закрепить номер записи меню GRUB, которая будет автоматически активизироваться при запуске компьютера. Однако на практике такой метод не очень полезен. Например, если вы хотите, чтобы по умолчанию запускалась Windows XP, а эта система стоит в списке на десятом месте, укажите GRUB_DEFAULT=9 (нумерация начинается с 0). Но вполне может случиться, что после обновления ядра в меню появятся две дополнительные строки — и ваша настройка станет неправильной. Лучше оставить GRUB_DEFAULT=0 без изменений, а желаемую запись GRUB поставить перед всеми остальными записями в меню. Сделайте это, добавив дополнительный сценарий в файл /etc/grub.d, причем имя файла должно начинаться с номера менее 10. Следующие строки можете использовать в качестве образца:

```
#!/bin/sh
exec tail -n +3 $0
# файл /etc/grub.d/09_boot-windows-by-default
menuentry "Windows 7" {
  set root=(hd0,1)
  chainloader +1
}
```

Установка вручную и первая помощь

Команда grub-install устанавливает загрузчик в загрузочный сектор или MBR указанного раздела или жесткого диска. Раздел или жесткий диск можно задавать одним из двух способов: как для Linux (например, /dev/sda1) или как для GRUB (например, (hd0,1)). Еще раз обращаю ваше внимание на то, что в GRUB 2 счет жестких дисков начинается с 0, а счет разделов — с 1. Таким образом, (hd0,1) — это первый раздел первого жесткого диска, или /dev/sda1.

Команду grub-install необходимо выполнять при установке загрузчика вручную или при исправлении дефектного экземпляра GRUB. Ее следует выполнять повторно, если для обновления GRUB в файл grub.cfg нужно добавить новые синтаксические элементы! GRUB сможет правильно интерпретировать такие элементы лишь в том случае, если в загрузочном секторе будет находиться актуальная версия загрузчика.

Ввод команд GRUB вручную

Если вы хотите запустить GRUB прямо с жесткого диска, с флешки или с CD, на который вы перенесли файл /usr/lib/grub-rescue/grub-rescue-floppy.img или grub-rescuecdrom.iso из пакета grub-rescue, перейдите в интерактивный режим, нажав клавишу C, а затем выполните следующие команды:

```
grub> set root=(hd0,1)
grub> linux /vmlinuz root=/dev/sda1
grub> initrd /initrd.img
grub> boot
```

Вместо (hd0,1) и /dev/sda1 укажите название системного раздела вашей системы Linux. В большинстве дистрибутивов файлы /vmlinuz и /initrd.img указывают на самую новую версию ядра и файла initrd в каталоге /boot. Если в вашей системе это не так, точно укажите местонахождение ядра и файла initrd. GRUB поддерживает автоматическое дополнение вводимой информации — для этого нужно нажать клавишу Tab.

Исправление GRUB 2

Если вам не удастся установить GRUB или загрузчик будет стерт при установке другой операционной системы, придется заново установить GRUB с «живого диска», воспользовавшись для этого современными инструментами GRUB 2. После запуска системы перейдите в режим администратора (в Ubuntu командой sudo -s), подключите системный раздел и активные каталоги /dev, /proc и /sys к файловой системе, а затем выполните программу chroot. При необходимости вы теперь можете подключить загрузочный раздел к новой корневой файловой системе. Командой grub-install обновите конфигурацию GRUB и запишите загрузчик в желаемое место (обычно для этого выбирается MBR первого жесткого диска). Как обычно, в командах /dev/sda*n* вы должны указывать вместо *n* названия устройств из вашей системы!

```
root# mkdir /syspart
root# mount /dev/sda2 /syspart   (Системный раздел)
root# mount -o bind /dev /syspart/dev
root# mount -o bind /proc /syspart/proc
root# mount -o bind /sys /syspart/sys
root# chroot /syspart
root# mount /dev/sda1 /boot      (Загрузочный раздел, если имеется)
root# update-grub
root# grub-install /dev/sda
root# exit
```

14.8. LILO

LILO (загрузчик Linux) еще несколько лет назад был основной загрузочной программой в Linux. С тех пор как по умолчанию в большинстве дистрибутивов стал применяться GRUB, значение LILO снизилось. Однако эта программа по-прежнему применяется в некоторых старых системах Linux. Преимущество LILO в сравнении с GRUB заключается в том, что LILO совместим со многими конфигурациями LVM и RAID и не требует наличия специального загрузочного раздела, находящегося за пределами LVM или RAID.

Для того чтобы понять описание LILO, приведенное в данном разделе, необходимо разобраться, как построен процесс загрузки (он подробно рассмотрен в разделах, посвященных GRUB). Дополнительную информацию по LILO лучше всего взять из следующей статьи «Википедии»: http://en.wikipedia.org/wiki/LILO_(boot_loader).

Сравнение GRUB и LILO (внутренняя организация)

В принципе GRUB и LILO выполняют одну и ту же задачу: они отображают меню, а затем запускают выбранную операционную систему. Чтобы обе программы правильно работали, их требуется установить в загрузочный сектор жесткого диска. На этом сходство заканчивается.

Важнейшее различие между двумя программами заключается в том, что LILO не в состоянии самостоятельно считывать файлы из файловой системы. Вместо этого при установке LILO для всех критичных файлов (ядро, файл `initrd`) создается список с номерами блоков с данными, в которых сохранен файл. LILO считывает все эти блоки в оперативную память, но «не понимает» файловой системы, лежащей в основе этих данных. Поиск блоков данных осуществляется командой `lilo`, которая должна выполняться в работающей системе Linux.

Таким образом, имеем важное следствие: всякий раз при изменении ядра или файла `initrd` (даже если имя файла не изменяется!) LILO необходимо заново установить в загрузочный сектор. Из-за этого администрирование системы усложняется и всегда остается риск, что рано или поздно вы забудете обновить LILO. Если вы все же об этом забудете, запустить Linux будет уже невозможно (правда, такой ситуации легко избежать: приучите себя всегда давать свежим версиям ядра и `initrd` новое название и не изменяйте имеющихся записей LILO, а добавляйте в меню новые!).

Еще один недостаток LILO заключается в том, что в этой программе отсутствует программный режим. Вы можете указывать дополнительные загрузочные параметры ядра, но не можете запустить любую операционную систему, просто введя правильную команду. Кратко эту мысль можно сформулировать так: внутренняя организация LILO проще, чем организация GRUB. Из-за этого программа более сложна в обслуживании и отличается меньшей гибкостью в том случае, когда что-то не работает.

Обслуживание

Итак, LILO установлена. Теперь после запуска компьютера программа появляется в форме меню или совсем по-спартански — в виде текста `LILO boot:`. Если вы хотите вмешаться в процесс загрузки, то способ обслуживания будет зависеть от того, в каком виде отображена программа LILO.

Текстовый режим

Иногда вам потребуется сначала нажать Shift, чтобы попасть в интерактивный режим. Затем с помощью клавиши Tab вы получаете список названий операционных систем, предлагаемых на выбор. Теперь вы можете напечатать одно из предложенных названий, а затем — нужные вам дополнительные параметры. Следующая команда запускает Linux. При этом сообщается дополнительный параметр ядра `xyz`:

```
LILO
boot: <Tab>
linux windows
boot: linux xyz=123
```

Графический режим

В некоторых экземплярах LILO нажатие Esc позволяет выйти из графического режима и попасть в текстовый. Кроме того, с помощью клавиш управления курсором вы можете выбрать одну из операционных систем, а затем задать с клавиатуры дополнительные параметры (не выходя из графического режима).

Конфигурация

Установка LILO проходит в два этапа: сначала создается конфигурационный файл `/etc/lilo.conf`, затем выполняется команда `lilo`. Она интерпретирует конфигурационный файл, создает на его основе новый загрузочный сектор и записывает его в место, указанное с помощью `lilo.conf`. Обычно таким местом оказывается MBR первого жесткого диска.

Когда вы впервые будете создавать `lilo.conf`, вам поможет сценарий `liloconfig`. В некоторых дистрибутивах также имеются специальные инструменты для конфигурации LILO.

ВНИМАНИЕ

Даже если вы не измените lilo.conf, вам потребуется вызывать команду lilo всякий раз после того, как изменится файл ядра (например, после нового компилирования). Для LILO важно не имя файла, а сектора, в которых этот файл расположен.

Файл `lilo.conf` состоит из двух частей: первая часть определяет общий ход работы загрузочной программы, вторая часть (ключевое слово `image` или `other`) перечисляет все операционные системы, запускаемые LILO (MS DOS, Windows, Linux). Первая система из списка автоматически используется по умолчанию. Комментарии вводятся символом #.

Глобальные параметры

В следующем списке перечислены важнейшие ключевые слова, действующие в глобальной области `lilo.conf`. Многие ключевые слова необязательны, но их нужно указывать, если базовые настройки LILO вам не подходят. Некоторые параметры LILO описаны в онлайн-документации, например `message` для вывода справочных текстов, `keytable` для определения той или иной раскладки клавиатуры, `password` для защиты процесса загрузки паролем и т. д.

- `boot` — глобальная область файла `/etc/lilo.conf` начинается с команды `boot=` и указывает, где должна быть установлена программа LILO. Чтобы установить LILO в MBR первого диска IDE, используйте `/dev/sda` (в отличие от GRUB, LILO при именовании устройств использует номенклатуру, обычную для Linux).
- `prompt` — благодаря этой команде LILO отображает подсказку (строку `"boot:"`), поясняя таким образом, что теперь можно вводить информацию. Параметр `prompt` лучше использовать всегда. Если `prompt` не используется, для вывода подсказки о вводе нужно нажать Shift.
- `delay` — указывает, как долго (в десятых долях секунды) LILO при загрузке будет ожидать вмешательства пользователя. Быстрее всего процесс выполня-

ется, если задать 0, но в таком случае, если вы не хотите работать с системой, заданной по умолчанию, вам потребуется нажать Shift уже перед запуском LILO. Если указать только prompt и не указать delay, то Linux будет ожидать выбора операционной системы сколь угодно долго.

- read-only — указывает, что сначала раздел подключается к системе в режиме «только для чтения». Процесс Init может контролировать файловую систему и при необходимости исправлять, пока она не будет заново подключена в режиме «для чтения и внесения изменений». Этот параметр нужно применять всегда!
- lba32 — позволяет обойти лимит в 1024 цилиндра. Программа lilo сохраняет номера секторов не в виде номер цилиндра/номер головки/номер сектора, а в 32-битном формате LBA (логическая адресация блоков). В версии LILO 22 и выше параметр lba используется по умолчанию.
- map — указывает файл, в который сохраняются номера секторов файлов ядра и других файлов. Обычно используется /boot/map, причем эту настройку лучше задать заранее. Параметр map следует использовать только в тех случаях, когда нужный файл находится в другом месте.
- install — указывает, в какой файл будет сохраняться основная программа LILO. Стандартная настройка — /boot/boot.b.

```
# Файл /etc/lilo.conf (globaler Bereich)
boot = /dev/sda # Установка в MBR первого жесткого диска SCSI/SATA
delay = 100     # Ждать 10 секунд
read-only       # Сначала использовать системный раздел в режиме «лишь для чтения»
# prompt        # Принудительное ожидание ввода (без автоматического запуска)
# map=/boot/map         # Стандартная настройка
# install=/boot/boot.b  # Стандартная настройка
# linear        # Иногда требуется для работы со старым оборудованием
```

В некоторых случаях (при работе с крупными жесткими дисками и старой BIOS) LILO может не разобраться в геометрии диска. Первым делом попробуйте решить такую проблему, указав параметр linear. Если это не поможет, используйте параметр disk и его дополнительные параметры bios, sectors, heads и cylinders. Базовая информация по этой теме сообщается в документации по LILO.

```
disk=/dev/hda    # Дополнительная информация по устройству /dev/hda
  bios=0x80      # 0x80 для диска 1.,
                 # 0x81 для диска 2. ...
  sectors=63     # Количество секторов
  heads=255      # Количество головок
  cylinder=522   # Количество цилиндров
```

Ключевые слова, используемые с записями меню LILO

Во второй части lilo.conf по порядку перечислены до 16 вариантов операционных систем, которые вы можете запускать по своему усмотрению. Первый вариант применяется по умолчанию. В следующем списке приводятся ключевые слова, используемые при описании отдельных записей меню.

- image — вводит записи меню LILO, нужные для запуска Linux. Его параметр указывает местонахождение файла ядра (обычно /boot/vmlinuz).
- other — вводит записи меню LILO, нужные для запуска Windows. Его параметр указывает раздел, в котором располагается Windows либо ее загрузчик
- initrd — указывает название файла псевдодиска, который должен быть загружен LILO. В этом файле содержатся модули ядра, необходимые уже в процессе загрузки, то есть еще до того, как будет возможен доступ к файловой системе. Базовая информация о файле initrd сообщается в разделе 14.5.
- label — отображает текст записи меню.
- root — указывает местоположение системного раздела Linux.
- append — определяет дополнительные параметры загрузки ядра (например, чтобы избежать проблем с аппаратным обеспечением). Важнейшие параметры описаны в разделе 14.9. Они действительны только для функций, интегрированных в ядро, но не для модулей, загружаемых позже (параметры модулей указываются в /etc/modprob.conf — см. главу 15). Если вы позже заносите параметры загрузки ядра в lilo.conf, не забудьте выполнить команду lilo! Только после этого изменения вступят в силу.

Запуск Linux

В следующих строках показаны команды, необходимые для запуска Linux из меню LILO:

```
# Продолжение в /etc/lilo.conf
image = /boot/vmlinuz      # Файл ядра
  initrd = /boot/initrd    # Файл Initrd
  root = /dev/sda8         # Системный раздел
  label = Linux            # Название в меню LILO
  # append = "optionen"    # Параметр ядра
```

Запуск Windows

Чтобы запустить Windows, укажите с помощью ключевого слова other раздел диска, в котором установлена Windows или находится загрузчик этой системы.

```
# Продолжение в /etc/lilo.conf
other = /dev/sda1   # Раздел Windows
  label = Windows   # Название в меню LILO
```

Установка в загрузочный сектор на жестком диске

Прежде чем вы сможете установить LILO, вам потребуется создать конфигурационный файл lilo.conf. В простейшем случае файл выглядит, как в следующем примере. Если LILO необходимо установить в MBR жесткого диска SATA или SCSI, то устройство boot будет таким: /dev/sda. Оставшиеся пути и устройства также следует указать, исходя из архитектуры вашего компьютера.

```
# Пример /etc/lilo.conf
boot=/dev/sda          # Установка в MBR первого жесткого диска SCSI/SATA
prompt                 # Глобальные параметры
```

```
timeout=100            # Ждать 10 секунд
read-only
image = /boot/vmlinuz  # Linux
  initrd = /boot/initrd
  root = /dev/sda8
  label = Linux
other=/dev/hda1        # Windows
  label = Windows
```

Для того чтобы установить LILO, как администратор выполните команду `lilo`:

```
root# lilo
Added Linux *
Added Windows
```

При первой установке LILO также самостоятельно создает резервную копию с именем файла `/boot/boot.0300` или `/boot/boot.0800`. С помощью команды `lilo -u` загрузчик LILO можно снова удалить с жесткого диска. Как самостоятельно создать резервную копию MBR, рассказано в разделе 14.4.

Исправление LILO с помощью «живого диска»

Если Linux не удается загрузить из-за того, что в конфигурации LILO допущена ошибка, запустите компьютер с помощью «живого диска» и найдите системный или загрузочный раздел. Затем выполните приведенные ниже команды. В данном примере предполагается, что в `/dev/sda3` содержится системный раздел вместе с каталогом `/boot`.

```
root# mkdir /test
root# mount /dev/sda3 /test   (Системный раздел с загрузочным каталогом)
root# chroot /test
root# lilo
```

14.9. Параметры загрузки ядра

При конфигурации GRUB или LILO можно указать параметры загрузки ядра (см. разделы 14.3 или 14.8 соответственно). Их также можно интерактивно ввести с клавиатуры при запуске установочной программы Linux или загрузчика. Синтаксис, используемый при задании параметров, таков:

параметрA=значение параметрB=значение1,значение2

Значения для параметров нужно указывать без пробелов. Если указывается несколько параметров, их нужно отделять друг от друга пробелами (а не запятыми). Шестнадцатеричные адреса указываются в виде `0x1234`. Если не поставить `0x`, то число будет интерпретироваться как десятичное.

Параметры загрузки ядра часто помогают устранять проблемы, связанные с оборудованием. Например, если ядро Linux не может «понять», сколько на компьютере оперативной памяти (строго говоря, это проблема BIOS), укажите правильное значение с помощью параметра `mem=`.

Обратите внимание, что параметры, сообщаемые при запуске Linux, оказывают влияние только на те драйверы, которые интегрированы прямо в ядро. Параметры модулей ядра нужно указывать в файле `/etc/modprobe.conf`. Об этом файле подробно рассказано в разделе 15.1.

В этом разделе описаны лишь важнейшие параметры загрузки ядра. Остальные данные можно узнать в справке на странице `man bootparam`, а также на следующих сайтах:

- http://www.tldp.org/HOWTO/BootPrompt-HOWTO.html;
- http://www.kernel.org/doc/Documentation/kernel-parameters.txt.

Важные параметры загрузки ядра

В этом подразделе будут перечислены самые важные параметры загрузки ядра.

- `root=/dev/sdb3` — указывает, что после загрузки ядра третий основной раздел второго привода SCSI/SATA должен использоваться как системный раздел (корневая файловая система). Разумеется, аналогичным образом можно указать и иные приводы и разделы.

  Если раздел помечен этикеткой (`label`), то системный раздел также можно указать в виде `root=LABEL=`*xxx*. Эта возможность особенно активно используется в Fedora и Red Hat. Вместо названия системного раздела обычно задается символ /. В разделах, содержащих файловую систему ext3, название раздела можно узнать с помощью команды `e2label`, а для изменения такого названия используется команда `tune2fs`.

  Еще один вариант указания системного раздела — с помощью `root=UUID=`*n*, где *n* — это UUID раздела жесткого диска. Этот идентификационный номер можно узнать с помощью файла `/lib/udev/vol_id`*раздел*.

- `ro` — указывает, что сначала систему нужно подключить в режиме «только для чтения». Это (в комбинации с двумя следующими параметрами) бывает полезно, когда требуется вручную исправить дефектную файловую систему.

- `init` — после запуска ядра автоматически выполняется программа `/sbin/init`, которая (в зависимости от дистрибутива) управляет процессом Init-V или Upstart (см. разделы 14.10 и 14.11). Если вы этого не хотите, то можете задать для выполнения другую программу с помощью параметра `init`. Например, если указать `init=/bin/sh`, то будет запускаться оболочка. Параметр может быть полезен людям, профессионально работающим с Linux, чтобы дать возможность снова запустить систему, если с конфигурацией Init-V что-то не так. Однако не забывайте, что корневая файловая система предоставляется в режиме «только для чтения» (см. раздел 13.5), и помните, что в консоли действует лишь американская раскладка клавиатуры, а переменная `PATH` еще не заполнена.

- `single` или `emergency` — если вы применили один из двух предыдущих параметров, компьютер запустится в «однопользовательском режиме» (строго говоря, эти параметры не интерпретируются ядром, а, как и все незнакомые параметры, сообщаются первой программе, которую запустило ядро). В данном случае такой программой является `/sbin/init`; она отвечает за инициализацию системы (также см. раздел 14.10).

- `initrd=name` — указывает название файла диска для начальной инициализации, который необходимо загрузить. Если вы *не хотите* использовать файл `initrd`, укажите `initrd=` или `noinitrd`.
- `reserve=0x300,0x20` — благодаря этому параметру 32 байта (шестнадцатеричное значение `0x20`) между `0x300` и `0x31F` не будут запрашиваться ни одним драйвером оборудования (они могли бы запрашиваться для поиска в этом промежутке каких-либо компонентов). Параметр необходимо использовать с некоторыми компонентами, которые «аллергически» реагируют на такие тесты. Как правило, `reserve` применяется в паре с другим параметром, указывающим точный адрес компонента, который обычно обращается к этому фрагменту памяти.
- `pci=bios|nobios` — определяет, должна ли BIOS использовать компоненты PCI для распознавания оборудования (PCI — это система шин для подключения к ПК сменных плат). Если функция автоматического распознавания оборудования ядром не работает, то может помочь команда `pci=bios`.
- `pci=nommconf` — деактивизирует `MMCONFIG` для конфигурации PCI. Это позволяет избежать проблем с некоторыми системами PCI-Express.

`quiet` — благодаря этому параметру при запуске ядра на экран не выводится никаких сообщений.

Параметры SMP

SMP — *симметричная мультипроцессорная обработка* — это способность ядра одновременно работать с несколькими процессорами. Если при этом возникнут проблемы, вам могут помочь следующие параметры.

- `maxcpus=1` — если в мультипроцессорной системе возникают проблемы с загрузкой, то с помощью этого параметра можно снизить количество используемых процессоров до одного. Значение 0 соответствует параметру `nosmp`.
- `nosmp` — деактивизирует функции SMP. Ядро использует только один процессор.
- `noht` — деактивизирует функцию гиперпоточности, предусмотренную в современных процессорах Pentium. (Благодаря гиперпоточности процессор работает так, как если бы в вашем распоряжении было два или более процессоров. Таким образом можно немного увеличить вычислительную мощность, но этот рост не так велик, как при использовании SMP.)
- `nolapic` — APIC означает *«Усовершенствованный программируемый контроллер прерываний»*. Это схема, в соответствии с которой аппаратные прерывания переадресовываются на процессоры. В современных версиях ядра APIC активизируется на большинстве компьютеров, даже если есть только один процессор (раньше APIC автоматически активизировался лишь на компьютерах с несколькими процессорами). Если вам кажется, что имеются проблемы с APIC, параметр `nolapic` приказывает ядру не активизировать и не использовать APIC локального компьютера.
- `noapic` — действует немного уже, чем `nolapic`, и отключает лишь ту часть APIC, которая отвечает за ввод-вывод.

- `lapic` — предназначен для того, чтобы специально активизировать APIC. Это требуется в тех случаях, когда APIC отключен в BIOS, но его все же требуется использовать.

Параметры ACPI

При работе с современным оборудованием одним из основных источников проблем являются старые системы управления питанием APM (улучшенная система управления питанием) и более новые системы ACPI (улучшенный интерфейс для управления электропитанием). Эти системы отвечают не за включение и отключение компьютера, а за экономный расход энергии, управление различными режимами гибернации и т. д. Далее перечислены важнейшие параметры, предназначенные для управления функциями ядра APM и ACPI.

- `apm=on/off` — (де)активизирует APM-функции ядра.
- `acpi=on/off` — (де)активизирует ACPI-функции ядра.
- `acpi=oldboot` — указывает, что функции ACPI должны использоваться только в процессе загрузки. Когда компьютер уже будет работать, функции ACPI будут деактивизированы.
- `pci=noacpi` — деактивизирует маршрутизацию прерываний, осуществляемую ACPI.
- `noresume` — благодаря этому параметру игнорируются данные гибернации, расположенные в разделе подкачки. Параметр `noresume` может быть полезен в том случае, когда компьютер начинает с ошибками выходить из спящего режима, например из-за того, что данные гибернации содержат ошибки.

14.10. Процесс Init-V

В этом разделе описаны процессы, протекающие в период от запуска ядра до запроса данных для входа в систему. После запуска ядро временно может обратиться к корневому разделу лишь в режиме «только для чтения». В первую очередь ядро запускает программу `/sbin/init`.

В дальнейшем программа `init` отвечает за базовую конфигурацию системы (подключение файловых систем) и запуск многочисленных сетевых служб и демонов (то есть фоновых процессов для управления системой, см. раздел 4.5).

От конкретного дистрибутива зависит, как именно будет выполняться программа `init`. Правда, почти все дистрибутивы ориентированы на использование процесса System-V-Init, так как он обычно задействован и в других системах, основанных на UNIX. Отличия могут касаться того, какие файлы `init` в каких каталогах расположены, какими номерами и буквами обозначаются так называемые уровни запуска, какие конфигурационные файлы учитываются при запуске и т. д. В этом разделе мы кратко рассмотрим процесс Init-V. Детали, характерные для различных дистрибутивов, описаны в следующих разделах.

Особый случай (пока) представляют собой **Fedora и Ubuntu: в обоих дистрибутивах** система Init-V была частично заменена более современным методом Upstart

(см. раздел 14.11). Поскольку в Upstart предусмотрен уровень совместимости для работы с Init-V и этот уровень активно используется, информация об Init-V, сообщаемая в данном разделе, применима также к Fedora и Ubuntu.

Обзор Init-V

Для того чтобы при чтении не запутаться в многочисленных деталях, сначала рассмотрим обычный запуск Linux с применением системы Init-V.

1. GRUB или LILO загружает и запускает ядро.
2. Ядро запускает программу `/sbin/init`.
3. Программа `init` интерпретирует конфигурационный файл `/etc/inittab`.
4. Программа `init` выполняет сценарий для инициализации системы.
5. Программа `init` выполняет сценарий `/etc/rc.d/rc` или `/etc/init.d/rc`. Сценарий `rc` во многом различен в разных дистрибутивах. Он нужен для запуска файлов сценариев, находящихся в каталоге `/etc/rc`*n*`.d` или `/etc/init.d/rc`*n*`.d` (*n* в данном случае — уровень запуска, см. ниже). Кроме того, `rc` активизирует в большинстве дистрибутивов и графический индикатор загрузки, который показывает, насколько выполнен процесс Init-V.
6. Файлы сценариев из `/etc/rc`*n*`.d` или `/etc/init.d/rc`*n*`.d` запускают различные системные службы, в частности те, что отвечают за выполнение сетевых функций.

Уровень запуска

Сначала ядро запускает программу `/sbin/init`. При этом сообщаются все параметры загрузки ядра, которые еще не были интерпретированы (то есть все те параметры, которые ядро «не знает» и, соответственно, не может самостоятельно обработать). В результате этого Linux можно, например, запустить в однопользовательском режиме (подробности сообщаются на странице `man init`). Иными словами, `init` — это первый работающий процесс. Все остальные процессы запускаются либо процессом `init` непосредственно, либо его подпроцессами (если вы выполните `pstree`, то сразу поймете, какую доминирующую роль играет процесс `init`, — см. раздел 4.1). При остановке компьютера процесс `init` последним завершает работу, обеспечивая правильное завершение всех остальных процессов.

Уровни запуска в Fedora, Red Hat, SUSE

Для понимания того, как работает System-V, нужно прежде всего усвоить, что такое *уровень запуска*. Уровни запуска описывают различные состояния, в которых может находиться операционная система. К сожалению, нумерация уровней запуска в различных дистрибутивах неодинакова. Как правило, значения уровней запуска в конкретном дистрибутиве документированы в `/etc/inittab`. В большинстве дистрибутивов (но не в Debian и Ubuntu!) значения уровней запуска таковы:

- `0` — остановка работы компьютера с выключением;
- `1` и `S` — однопользовательский режим;
- `2` — многопользовательский режим без выхода в сеть и без NFS;

- 3 — многопользовательский режим с выходом в сеть, но без автоматического запуска X;
- 4 — обычно не применяется;
- 5 — многопользовательский режим с выходом в сеть и запуском системы X; установлен по умолчанию;
- 6 — остановка работы компьютера с перезагрузкой.

В некоторых системах существует различие между уровнями запуска 1 и S. При работе с такими системами выполняются специальные сценарии уровня запуска 1, позволяющие перейти с обычного уровня (2, 3 или 5) на уровень запуска однопользовательского режима. Напротив, сценарии уровня запуска S применяются лишь тогда, когда вам требуется активизировать уровень запуска однопользовательского режима сразу после загрузки (параметр ядра `single`).

Уровни запуска в Debian и Ubuntu

В дистрибутивах, построенных на основе Debian, уровни запуска 2-5 равноправны. На каждом их этих уровней запускается многопользовательская система с выходом в сеть и X. Стандартным уровнем запуска считается 2. Уровень запуска S не является уровнем в полном смысле этого слова, он просто применяется для инициализации компьютера сразу после запуска (еще до того, как будет активизирован какой-либо иной уровень). Сетевые функции активизируются в Debian и Ubuntu уже на этапе инициализации системы (то есть перед включением того или иного уровня запуска), следовательно, они доступны на всех уровнях запуска:

- S — инициализация компьютера непосредственно после запуска;
- 0 — остановка работы компьютера с выключением;
- 1 — однопользовательский режим с доступом к сети;
- 2-5 — многопользовательский режим с доступом к сети и автоматическим запуском X;
- 6 — остановка работы компьютера с перезагрузкой.

Смена активного уровня запуска

Администратор может изменить уровень запуска с помощью команды `init` *x*, не останавливая работу системы. Здесь *x* — это цифра или буква, означающая уровень запуска. Например, бывает полезно изменить уровень запуска при проведении некоторых работ по техническому обслуживанию компьютера. При выполнении команды `shutdown` или нажатии Ctrl+Alt+Delete уровень запуска также изменяется, в результате чего происходит перезапуск компьютера.

Стандартный уровень запуска. Стандартный уровень запуска определяется строкой `initdefault` в `/etc/inittab`. В большинстве современных дистрибутивов стандартным уровнем запуска является 5; в Debian и Ubuntu это уровень 2. В Ubuntu версии 6.10 до 9.10 стандартный уровень запуска определяется в `/etc/inittab`, в версии 9.10 и выше — в `/etc/init/rc-sysinit.conf`. Эти файлы вводят в состав системы Upstart (см. раздел 14.11).

Inittab

При запуске системы команда `init` управляется файлом `/etc/inittab`. Синтаксис записей осуществляется по следующей схеме:

```
id-code:runlevel:action:command
```

- `id-code` состоит из двух символов, однозначно идентифицирующих строку;
- `runlevel` указывает, на каком уровне запуска выполняется запись;
- `action` содержит команду для `init`;
- `command` указывает, какую программу или команду Linux следует запустить.

В табл. 14.3 перечислены важнейшие ключевые слова для `action` (полное описание дается в `man inittab`).

Таблица 14.3. Ключевые слова

Ключевое слово	Значение
ctrlaltdel	Указывает, как init должна реагировать на Ctrl+Alt+Delete
initdefault	Определяет стандартный уровень запуска для init (см. выше)
once	init выполняет указанную команду при смене уровня запуска
respawn	После окончания выполнения команды init снова ее запускает
sysinit	init один раз выполняет команду в процессе загрузки
wait	init ожидает окончания выполнения следующей команды
bootwait	init запускает команду в процессе загрузки и ожидает окончания выполнения следующей команды

В следующем листинге показан немного сокращенный файл `inittab` из Debian 5. Стандартный уровень запуска — 2. При обычном запуске системы `init` выполняет сценарные файлы `rc.sysinit` и команду `rc 5`. Наконец, в текстовых консолях `1-6` запускается программа `mingetty`, обеспечивающая вход в систему (если вам нужно больше текстовых консолей, изменять их количество нужно именно здесь. Однако обратите внимание на то, что в большинстве дистрибутивов консоль `7` зарезервирована для X).

```
# Файл /etc/inittab в Debian 5
# Стандартный уровень запуска
id:2:initdefault:
# Конфигурация и инициализация системы производится сразу после запуска компьютера
si::sysinit:/etc/init.d/rcS
# Работа в однопользовательском режиме (параметр ядра su)
~~:S:wait:/sbin/sulogin
# Старт соответствующего уровня запуска
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
```

```
# следующую строку не нужно использовать, она предусмотрена для аварийных ситуаций
z6:6:respawn:/sbin/sulogin
# Реакция на Ctrl+Alt+Delete в текстовой консоли
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
# Реакция в случае перебоев с электропитанием (необходимо наличие источника беспе-
ребойного питания)
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop
# Запуск gettys (эмуляторов терминалов) для текстовых консолей
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
```

Изменение действия комбинации Ctrl+Alt+Delete. Большинство дистрибутивов по умолчанию сконфигурированы так, что сочетание клавиш Ctrl+Alt+Delete в текстовом режиме приводит к перезапуску компьютера. Если вы хотите, чтобы вместо этого компьютер выключался, укажите в команде `shutdown` в строке `ca:` параметр `-h` вместо `-r`. Если хотите вообще отключить это сочетание клавиш, поставьте перед строкой `ca:` символ комментария #.

Инициализация системы

Еще до того, как запустятся или завершат работу описанные далее файлы `rc` и `runlevel`-специфичные службы, то есть сразу после запуска компьютера, выполняется инициализация системы (строка `si:` в `inittab`). Название сценария зависит от дистрибутива:

- Debian, Ubuntu — `/etc/init.d/rcS`;
- Red Hat, Fedora — `/etc/rc.d/rc.sysinit`;
- SUSE — `/etc/init.d/boot`.

В ходе инициализации системы решаются задачи, которые требуется выполнить только один раз, в начале сеанса работы с компьютером:

- инициализация различных системных переменных (в том числе хост-имен и доменных имен);
- активизация файловой системы `/proc`;
- настройка даты и времени;
- настройка раскладки клавиатуры для текстовой консоли;
- запуск системы udev;
- активизация RAID или LVM;
- перепроверка файловых систем;
- подключение корневого раздела заново, в режиме «для чтения и внесения изменений»;
- проверка файловой системы следующего раздела, подключение разделов;

- частичная (Fedora) или полная (Debian, Ubuntu) инициализация основных сетевых функций.

Обратите внимание — не все описанные здесь функции напрямую выполняются сценарием инициализации системы. Частично считываются и другие сценарные файлы. При этом их названия обычно записываются в форме `. имя` (поставив точку, мы гарантируем, что находящийся в этом месте файл будет прочтен и выполнен; затем продолжится выполнение сценария). В Debian и Ubuntu присутствует миниатюрный сценарий `rcS`, выполняющий все сценарные файлы `/etc/rcS.d/S*`.

Сценарии Init-V для активизации уровней запуска

После инициализации системы активизируется стандартный уровень запуска в соответствии с `/etc/inittab`. Для всех действий, связанных с уровнем запуска, существуют собственные сценарные файлы. В зависимости от дистрибутива они могут находиться в каталоге `/etc/init.d` или `/etc/rc.d/init.d`. Чтобы достичь большей совместимости между дистрибутивами, лучше всего указывать оба пути в виде ссылок.

Для запуска сценария Init-V программа `init` выполняет сценарий `/etc/rc.d/rc` или `/etc/init.d/rc`. Сценарию `rc` передается желаемый уровень запуска *n*. Сначала `rc` выполняет некоторые операции, нужные для инициализации. После этого выполняются все сценарные файлы `rcn.d/K*`, необходимые для завершения процессов. Наконец, выполняются все файлы `rcn.d/S*`, требуемые для активизации новых процессов на том или ином уровне запуска.

Значительное достоинство этой (на первый взгляд непонятной) системы заключается в том, что вы можете с легкостью интегрировать новые системные процессы в процесс Init-V. Нужно просто скопировать сценарии `rc` для запуска и остановки процессов в правильные каталоги — именно это происходит при установке пакета, содержащего дополнительный демон. Следующая сводная таблица для Debian 5 демонстрирует, какие сценарные файлы выполняются для некоторых уровней запуска.

```
user$ cd /etc/rc.d
user$ ls rcS.d/ rc2.d/ rc6.d/
rcS.d/: (Инициализация системы)
S01glibc.sh         S10checkroot.sh    S36mountall-bootclean S46mountnfs-bootcln
S02hostname.sh      S11hwclock.sh      S36udev-mtab          S48console-screen
S02mountkernfs.sh    S12mtab.sh         S37mountoverflowtmp   S50alsa-utils
S03udev              S18ifupdown-clean  S39ifupdown           S55bootmisc.sh
S04mountdevsubfs.sh  S20module-init-too S40networking         S55urandom
S05bootlogd          S30checkfs.sh      S43portmap            S70x11-common
S05keymap.sh         S30procps          S44nfs-common         S75sudo
S08hwclockfirst.sh   S35mountall.sh     S45mountnfs.sh        S99stop-bootlogd-s
rc2.d/: (стандартный уровень запуска)
S05loadcpufreq       S19cpufrequtils     S20samba             S89anacron
S10rsyslog           S19mysql            S23ntp               S89atd
S12acpid             S20cups             S24dhcdbd            S89cron
S12dbus              S20exim4            S24hal               S99rc.local
S14avahi-daemon      S20kerneloops       S26network-manager   S99rmnologin
S16ssh               S20nfs-common       S26network-manager-dis S99stop-bootlogd
S17mysql-ndb-mgm     S20nfs-kernel-serv  S30gdm
```

```
S18mysql-ndb          S20openbsd-inetd    S30system-tools-backen
rc6.d/: (Завершение работы)
K01gdm                K20nfs-common       K50alsa-utils         S31umountnfs.sh
K11anacron            K20openbsd-inetd    K63mountoverflowtmp   S32portmap
K11atd                K21mysql            K80nfs-kernel-server  S35networking
K14network-manager    K22mysql-ndb        K86avahi-daemon       S36ifupdown
K14network-manager-disp K23mysql-ndb-mgm  K90rsyslog            S40umountfs
K19samba              K23ntp              S15wpa-ifupdown       S60umountroot
K20exim4              K25hwclock.sh       S20sendsigs           S90reboot
K20kerneloops         K30system-tools-ba  S30urandom
```

Если быть точным, в каталогах `rcn.d` находятся не сами сценарные файлы, а только ссылки на них. Преимущество этого метода заключается в том, что каждый сценарный файл можно использовать для нескольких уровней запуска и централизованно изменять. Сами сценарные файлы сохраняются в каталоге `/etc/rc.d/init.d` или `/etc/init.d`:

```
root# cd /etc
root# ls -l rc2.d/S20cups
... rc2.d/S20cups -> ../init.d/cups
```

Номенклатура

Названия ссылок совсем не произвольны, хотя на первый взгляд и может сложиться такое впечатление. Начальная буква указывает, запускает сценарий процесс (`S=start`) или завершает его (`K=kill`). Ссылки `S` и `K` указывают на одни и те же файлы, но в зависимости от начальной буквы `rc` выполняет сценарий или с параметром `start`, или с параметром `stop`.

Указанный далее номер определяет очередность, в которой выполняются сценарные файлы. Например, для запуска большинства сетевых демонов требуется, чтобы сетевое соединение уже было налажено, после чего они запускаются сценарием `network`. Краткое описание многих демонов, запускаемых сценарными файлами `rc`, есть в разделе 4.5.

Как автоматически запускать демоны и завершать их работу

Многие сценарные файлы уровней запуска автоматически выполняются сценарием `rc` при включении компьютера или при изменении уровня запуска. При этом сообщается параметр `start` или `stop`, в зависимости от того, следует ли запустить или остановить соответствующую функцию.

Для того чтобы можно было выполнить автоматический запуск или остановку, требуется ссылка `S` или `K` на сценарий Init-V для соответствующего уровня запуска в каталоге `rcn.d`. Иными словами, если вы хотите, чтобы в дальнейшем определенная функция активизировалась автоматически, вам нужно создать такие ссылки. Соответственно, чтобы отменить автоматический запуск, следует удалить такие ссылки.

Следующие команды показывают, какие ссылки необходимо создать в Red Hat, чтобы в дальнейшем программа Samba автоматически начинала работу на уровнях запуска `2` и `5`:

```
root# cd /etc/
root# ln init.d/samba rc0.d/K19samba
root# ln init.d/samba rc1.d/K19samba
root# ln init.d/samba rc2.d/S20samba
root# ln init.d/samba rc3.d/S20samba
root# ln init.d/samba rc4.d/S20samba
root# ln init.d/samba rc5.d/S20samba
root# ln init.d/samba rc6.d/K15samba
```

Удаление ссылок серьезно упрощается:

```
root# rm rc?.d/*samba
```

На практике вам достаточно редко придется вводить вручную вышеуказанные команды `ln` или `rm`. В большинстве дистрибутивов имеются специальные команды, избавляющие вас от этой работы, например `insserv` в SUSE, `chkconfig` в Red Hat, Fedora и Mandriva или `update-rc.d` в Debian и Ubuntu (также см. раздел 4.5).

Как вручную запускать и останавливать демоны. Сценарные файлы уровней запуска можно выполнять и вручную. Например, следующая команда останавливает работу веб-сервера Apache:

```
root# /etc/init.d/samba stop
```

Во многих дистрибутивах также предусмотрены специальные команды для запуска и остановки работы демонов, избавляющие вас от излишней работы с клавиатурой. Например, `service samba stop` в Fedora или `rcsamba stop` в SUSE имеют то же значение, что и вышеуказанная команда.

Параметры сценариев Init-V

Большинству сценариев можно передать следующие параметры:

- `start` — запускает ту или иную функцию;
- `stop` — завершает функцию;
- `status` — выводит краткое информационное сообщение о том, активна функция или нет;
- `reload` — используется в том случае, если измененную конфигурацию требуется считать заново, не останавливая при этом демон;
- `restart` — напротив, полностью останавливает демон, а потом снова его запускает; при этом разрываются все имевшиеся соединения с клиентами.

С некоторыми демонами команды `reload` и/или `restart` не используются. В этом случае при работе со сценарием нужно сначала выполнить `stop`, а потом — `start`.

Оптимизация процесса Init-V

Вы, вероятно, уже поняли, что процесс Init-V достаточно сложен: в ходе этого процесса требуется запускать и останавливать многочисленные сценарии и программы. Еще несколько лет назад запуск системы мог занимать одну-две минуты. Однако в настоящее время в большинстве дистрибутивов с помощью разнообразных приемов удалось снизить это время примерно до полуминуты (при использовании

современного оборудования). В этом разделе обобщены самые распространенные методы оптимизации загрузки системы.

- **Загрузка файлов «заранее».** При каждом запуске с жесткого диска считываются одни и те же файлы, в одинаковом порядке. При считывании файлов, конечно же, возникают небольшие временные промежутки (периоды простоя системы). Чтобы ускорить такой процесс считывания, некоторые дистрибутивы в начале выполнения процесса Init-V запускают фоновую программу, считывающую все файлы, указанные в специальном списке. Если какой-либо файл действительно понадобится в ближайшее время, он сразу помещается в кэш. В Fedora за такую операцию отвечает программа `readahead`, в Ubuntu — одноименная программа, которая, правда, несколько иначе интегрирована в систему. В SUSE применяется программа `preload`.
- **Распараллеливание.** Обычно процесс Init-V протекает последовательно, то есть сценарии Init-V обрабатываются один за другим. Вот почему так происходит: невозможно подключить к файловой системе каталог NFS, пока не инициализированы сетевые функции. Кроме того, практически невозможно обратиться к файлам каталога NFS до того, как этот каталог будет интегрирован в файловую систему. Но не все сценарии Init-V зависимы друг от друга. Так, например, вы вполне можете параллельно запустить функции настройки сети и инициализацию печати, почтового сервера и веб-сервера. Поэтому в некоторых дистрибутивах предпринимаются попытки параллельно выполнять те сценарии Init-V, которые не зависят друг от друга.
- **Более ранний запуск графической системы X.** Раньше было принято запускать систему X ближе к концу процесса Init-V. Но в наши дни система X запускается еще до того, как завершится основная инициализация оборудования. Ранний запуск X является в определенной степени частным случаем распараллеливания процесса Init-V.

Будущее процесса Init-V. Наряду с принятием вышеописанных мер, которые в значительной степени соответствуют концепции Init-V, также имеются планы совершенно по-новому организовать весь процесс запуска. В хороших идеях недостатка нет, однако все предложения касаются сетевых служб, управляемых процессом Init-V, соответственно, необходимо создать новые сценарии для запуска и остановки программ (см. следующий раздел).

14.11. Upstart

Ubuntu была первым крупным дистрибутивом, сделавшим шаг в сторону от системы Init-V и перешедшим в версии 6.10 к использованию Upstart. Fedora последовала за Ubuntu в своей версии 9. Разумеется, этот переход пока не обходится без проблем: в значительной мере инициализация все еще проходит с помощью традиционных файлов Init-V, совместимость которых с новой системой обеспечивают специальные сценарии. Такой прагматичный подход гарантирует совместимость с многочисленными сетевыми службами, которые еще не приспособлены к работе с Upstart. В будущих версиях Ubuntu и Fedora Upstart должен напрямую заниматься все более значительной частью запуска системы.

Далее будут обобщены важнейшие концепции и рассмотрены конфигурационные файлы версии Upstart 0.6 на базе Ubuntu 9.10. Обратите внимание, что в более ранних версиях Upstart вместо части рассмотренных ниже каталогов конфигурации используются другие каталоги. Более подробная информация сообщается в `init`, `initctl`, `telinit` и `runlevel` и на следующих сайтах:

- http://upstart.ubuntu.com/;
- http://fedoraproject.org/wiki/Features/Upstart.

Концепция

Концепция Upstart принципиально отличается от Init-V. Upstart — это событийно-ориентированная программа. События происходят при запуске и остановке программ или служб. Upstart обрабатывает события и реагирует на них, запуская или останавливая (другие) службы или инициируя другие события. События также обеспечивают простой обмен информацией между двумя процессами.

Как и в случае с Init-V, программа `/sbin/init` запускается ядром как первый процесс. Однако эта программа `init` входит в состав Upstart и не имеет ничего общего с программой `init` системы Init-V! Описываемая здесь программа `init` в основном занимается интерпретацией конфигурационных файлов и реагирует на события.

Последовательность событий инициируется командой `startup`. Это событие автоматически происходит после запуска `/sbin/init`. Ради совместимости с Init-V в Upstart также предусмотрены уровни запуска. Они реализованы в новой системе как события с названиями `runlevel` *n*.

Конфигурация

Файла `/etc/inittab` в данном случае нет. Вместо этого все конфигурационные файлы находятся в каталоге `/etc/init` (внимание: в версии 0.3 применялся каталог `/etc/events.d`, в версии 0.5 — `/etc/jobs.d`). Программа `/sbin/init` считывает все CONF-файлы из этого каталога. Типичный конфигурационный файл выглядит так:

```
# /etc/init/tty1.conf
start on stopped rc RUNLEVEL=[2345]
stop on runlevel [!2345]
respawn
exec /sbin/getty -8 38400 tty1
```

Данный файл отвечает за запуск эмулятора терминала (программы `getty`) в текстовой консоли 1. Команда `exec` выполняется не сразу, а только тогда, когда произойдет одно из событий, определенных `start`, то есть после завершения работы сценариев `rc2-rc5`. Аналогичным образом работа `getty` завершается, когда произойдет одно из событий, указанных в `stop on`. Благодаря `respawn` программа `getty` автоматически перезапускается, если ее работа прекратится незапланированным образом.

Кроме ключевых слов, указанных выше, есть еще некоторые: команду о запуске программы можно сформулировать не только с помощью `exec`, но и посредством `script / end script`. Все строки, находящиеся между этими командами, будут выполнены оболочкой `/bin/sh`. Если при запуске или остановке работы потребуются дополнительные мероприятия по инициализации или настройке, то соответствующие команды будут сформулированы с помощью `per-start script / end script` или `post-stop script / end script`. Благодаря указанию `console output` результаты выполнения программ выводятся на консоль, а не переадресовываются по умолчанию на `/dev/null` (вывод игнорируется). За реагирование на нажатие клавиш Ctrl+Alt+Delete отвечает файл `control-alt-delete.conf`.

Совместимость с Init-V

Upstart в том виде, в котором он существует сейчас, запускает далеко не все системные службы. Для того чтобы запустить оставшиеся службы, используется один из сценариев, обеспечивающих совместимость с Init-V, — `/etc/init.d/rc` или `/etc/rc.d/rc`. За его запуск отвечают конфигурационные файлы Upstart `rc.conf` и `rcS.conf`. Сценарий `/etc/init.d/rc` работает в полном соответствии требованиям системы Init-V, то есть все сценарии, запускающие и останавливающие программы, выполняются в `/etc/rcn.d`.

Однако предусмотрена и обратная совместимость: чтобы службы, управляемые Upstart, можно было запускать командами системы Init-V, каталог `/etc/init.d/` служб Upstart содержит ссылку на сценарий `/lib/init/upstart-job`. Этот сценарий отвечает за выполнение подходящих команд Upstart. Итак, если вы выполняете в Ubuntu 9.10 `/etc/init.d/gdm`, сценарий `upstart-job` выполняет команду `stop gdm`. Взглянув на ссылки в `/etc/init.d`, вы сразу же поймете, сколько служб уже выполняется с помощью Upstart (`ls -l /etc/init.d`).

Управляющие команды

В Upstart предусмотрено несколько новых команд. Они предназначены для запуска и остановки процессов, обзора текущих и предстоящих событий и т. д. Обратите внимание, что эти команды предназначены для работы только с теми процессами, которые управляются непосредственно Upstart. Службы, запускаемые через уровень совместимости Init-V, и в дальнейшим администрируются с помощью команд из Debian (в частности, `invoke.rc`, `update-rc.d` и т. д.).

Параметр `start` или `stop` соответственно запускает или останавливает процесс, для которого в `/etc/event.d` предусмотрен файл конфигурации; `status` указывает, в каком состоянии сейчас находится процесс.

```
root# status tty2
tty2 (start) running, process 4116
root# stop tty2
tty2 (stop) running, process 4116
tty2 (stop) pre-stop, (main) process 4116
tty2 (stop) stopping, process 4116
```

```
tty2 (stop) killed, process 4116
tty2 (stop) post-stop
tty2 (stop) waiting
```

В зависимости от того, какие команды были переданы `initctl` в виде параметров, сценарий выполняет различные задачи по администрированию системы (см. `man initctl`). Например, `initctl emit eventname` создает событие с указанным именем. Такой метод очень практичен при тестировании написанных вами сценариев. Команда `initclt list` выдает обзор статусов всех действующих в данный момент процессов:

```
root# initctl list
control-alt-delete (stop) waiting
logd (stop) waiting
rc-default (stop) waiting
rc0 (stop) waiting
rc1 (stop) waiting
...
tty5 (start) running, process 4113
tty6 (start) running, process 4119
```

Уровень запуска

Как уже было сказано выше, Upstart не нуждается в уровнях запуска. Однако ради совместимости с процессом Init-V Upstart моделирует уровни запуска по образцу Init-V. Команда `runlevel` выдает справку о действующем уровне запуска; `telinit` *n* или `init` *n* активизирует новый уровень запуска *n*.

14.12. Запуск системы Debian

На рис. 14.5 показано, как в Debian происходит запуск системы, основанный на Init-V. По умолчанию задан уровень запуска `2`. В табл. 14.4 указано, где находятся конфигурационные файлы.

При переходе с одного уровня запуска на другой останавливаются только те функции, которые были активизированы на предыдущем уровне запуска, а на новом уровне запуска не нужны. Аналогичным образом, запускаются только такие функции, которые до этого были неактивны. Чтобы определить, какие функции уже запущены, а какие еще нет, сценарий `rc` проверяет, есть ли для конкретной функции на предыдущем уровне запуска ссылка `start` или `stop`.

Таблица 14.4. Конфигурация запуска системы в Debian

Функция	Конфигурационные файлы
Инициализация системы	/etc/init.d/rcS, /etc/rcS.d/*
Сценарии Init	/etc/init.d/*
Ссылки на уровни запуска	/etc/rcn.d/rcn.d/*
Конфигурационные файлы	/etc/default/*

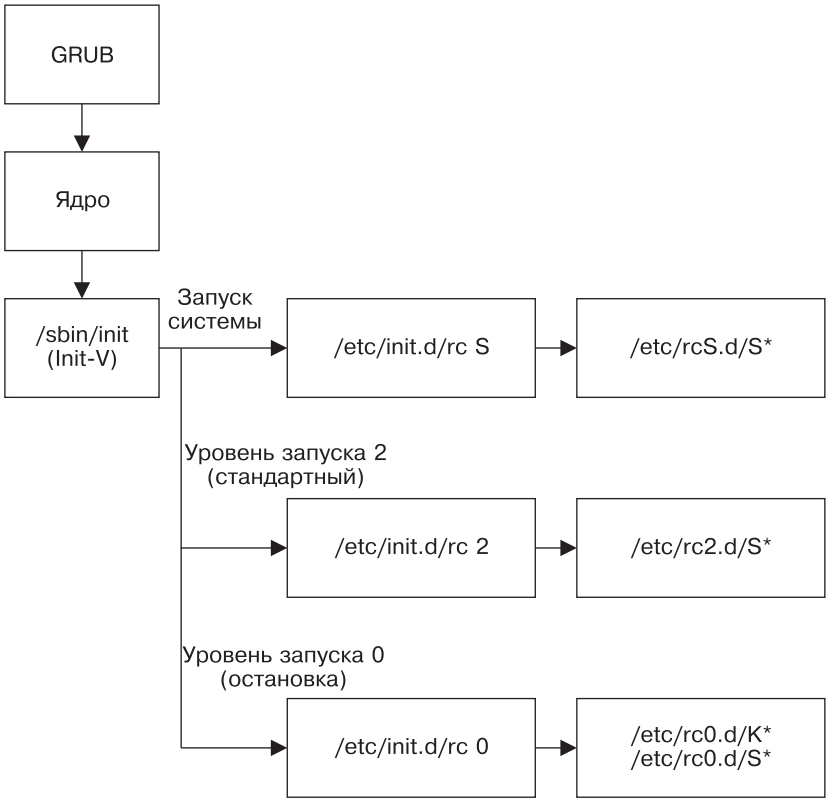

Рис. 14.5. Старт системы на уровне запуска 2 в Debian

Запуск сценариев Init-V

Команда `invoke.rc` частично избавляет вас от работы с клавиатурой при вводе сценария Init-V вручную.

```
root# invoke.rc samba restart
```

Запуск X. Графическая система X запускается с помощью сценария Init-V `gdm`. Он имеет порядковый номер и запускается после всех сетевых демонов.

Параллельное выполнение сценариев Init-V. Система Init-V дистрибутива Debian способна параллельно запускать сценарии Init-V с одинаковым порядковым номером. Это помогает немного повысить эффективность труда, особенно при работе с компьютерами, на которых установлено несколько процессоров или многоядерный процессор. Ради обеспечения стабильности системы эта функция обычно не активизируется. Если вы хотите протестировать распараллеливание, используйте следующую настройку:

```
# в /etc/init.d/rc
...
CONCURRENCY=shell
```

Индивидуальная настройка процесса Init-V

С помощью файла /etc/init.d/rc.local вы можете с легкостью выполнить индивидуальную настройку процесса Init-V. Это сценарий, запускаемый после всех остальных сценариев Init-V в ситуации, когда впервые активизируется уровень запуска 2, 3, 4 или 5. При последующем изменении уровня запуска или при окончании работы компьютера сценарий больше *не* выполняется (в том числе для остановки каких-либо запущенных им программ)!

Управление ссылками Init-V

Команда update-rc.d предназначена для работы с установочными сценариями пакетов. При инсталляции или деинсталляции пакетов эта команда позволяет создавать или снова удалять ссылки на уровни запуска для сценариев Init-V конкретного пакета. Разумеется, вы можете использовать эту команду и интерактивно. Работать с ней довольно непросто. В частности, обращаю ваше внимание на то, что update-rc.d, как правило, не вносит никаких изменений в уже имеющиеся ссылки! Сначала нужно удалить имеющиеся ссылки.

Команда update-rc.d *имя* remove удаляет все ссылки, предназначенные для запуска и остановки указанной службы. В любом случае эта команда работает только при условии, что файл /etc/init.d/*имя* уже удален (деинсталлирован). Если это условие не соблюдено, примените параметр -f.

Команда update-rc.d *имя* defaults создает на всех уровнях запуска ссылки для запуска (уровни 2-5) и остановки (уровни 0, 1 и 6) той или иной службы. Названия ссылок начинаются с порядкового номера 30. Если сценарий необходимо выполнить раньше или позже в процессе запуска или остановки службы, то вам придется самостоятельно указать для запуска и остановки нужные значения (в следующем примере 13 для запуска, 1 для остановки):

```
root# update-rc.d gdm defaults 30 1
  /etc/rc0.d/K01gdm -> ../init.d/gdm
  /etc/rc1.d/K01gdm -> ../init.d/gdm
  /etc/rc6.d/K01gdm -> ../init.d/gdm
  /etc/rc2.d/S30gdm -> ../init.d/gdm
  /etc/rc3.d/S30gdm -> ../init.d/gdm
  /etc/rc4.d/S30gdm -> ../init.d/gdm
  /etc/rc5.d/S30gdm -> ../init.d/gdm
```

Если нужно отдельно настроить ссылки на каждый уровень запуска, передайте аргументы update-rc.d в форме *имя* start|stop *nn* runlevel. При этом *nn* — это число, стоящее в начале ссылки на уровень запуска. Можно указывать несколько уровней запуска и групп аргументов. Каждая группа должна завершаться точкой. Следующая команда действует так же, как и gdm defaults 30 1.

```
root# update-rc.d gdm start 30 2 3 4 5 . stop 1 0 1 6 .
Adding system startup for /etc/init.d/gdm ...
  /etc/rc0.d/K01gdm -> ../init.d/gdm
  /etc/rc1.d/K01gdm -> ../init.d/gdm
  /etc/rc6.d/K01gdm -> ../init.d/gdm
```

```
/etc/rc2.d/S30gdm -> ../init.d/gdm
/etc/rc3.d/S30gdm -> ../init.d/gdm
/etc/rc4.d/S30gdm -> ../init.d/gdm
/etc/rc5.d/S30gdm -> ../init.d/gdm
```

Если вы работаете с X, то можете настроить ссылки на уровни запуска с помощью меню Система ▸ Управление системой ▸ Службы (программа Gnome `services-admin`, все изменения касаются стандартного уровня запуска).

Как построены файлы сценариев Init-V

В следующих строках показано, как построен пользовательский сценарий Init-V `/etc/init.d/masquerading`, необходимый для того, чтобы компьютер, на котором он установлен, работал как интернет-шлюз (маскарадинг и простейший брандмауэр, см. раздел 17.3). Файлы сценариев Init-V в Debian не содержат никакой информации о том, когда обычно выполняется сценарий и какие еще сетевые службы нужны для его работы, в отличие от Red Hat и SUSE, где такая информация предоставляется.

```
#!/bin/sh
DESC="masquerading"        # Обозначение сценария
ADSL=eth1                   # Интерфейс, через который осуществляется доступ в Ин-
тернет
. /lib/lsb/init-functions  # Считывание основных функций
IPT=$(which iptables)      # Поиск команды iptables
if [ -z $IPT ]; then
  [ -x /sbin/iptables ] && IPT=/sbin/iptables
  [ -x /usr/sbin/iptables ] && IPT=/usr/sbin/iptables
fi
[ -z $IPT ] && (echo "iptables cannot be found!"; exit 0)
# Функции для start, stop и restart
case "$1" in
  start)
    log_begin_msg "Starting masquerading ..."
    ERROR=0
    $IPT -t nat -A POSTROUTING -o $ADSL -j MASQUERADE
    echo 1 > /proc/sys/net/ipv4/ip_forward
    log_end_msg $ERROR
    ;;
  stop)
    log_begin_msg "Stopping masquerading ..."
    ERROR=0
    echo 0 > /proc/sys/net/ipv4/ip_forward
    $IPT -t nat -D POSTROUTING -o $ADSL -j MASQUERADE
    log_end_msg $ERROR
    ;;
  restart)
    $0 stop
    $0 start
    ;;
  *)
```

```
    log_success_msg "Usage: masquerading {start|stop|restart}"
    exit 1
    ;;
esac
exit 0
```

Чтобы в дальнейшем сценарий автоматически запускался при старте компьютера, выполните следующие команды:

```
root# update-rc.d masquerading defaults 40 1
```

14.13. Запуск системы в Fedora

На рис. 14.6 показано, какие сценарии выполняются на стандартном уровне запуска 5 при старте системы. В табл. 14.5 перечислены конфигурационные файлы, задействованные в процессе. Эти данные касаются дистрибутива Fedora 12 с системой Upstart. Обратите внимание — в RHEL 5 по-прежнему применяется классическая система Init-V, переход на Upstart ожидается только в версии 6!

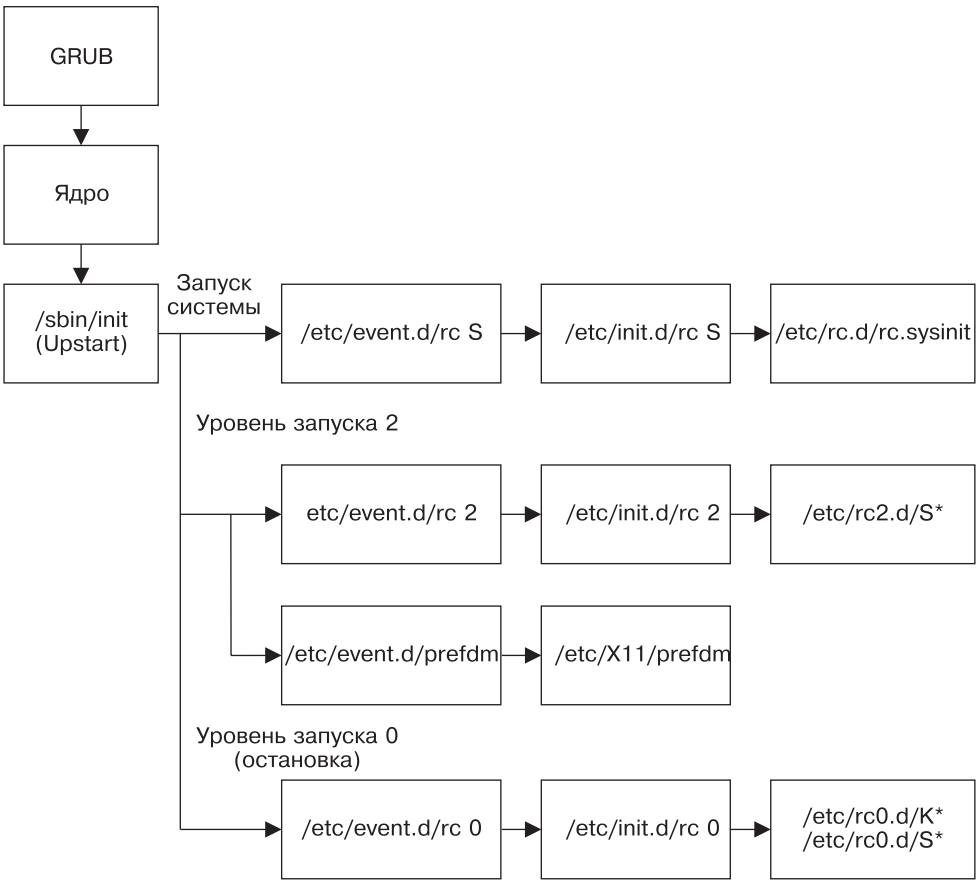

Рис. 14.6. Старт системы на уровне запуска 5 в дистрибутиве Fedora

Таблица 14.5. Конфигурация Fedora при запуске системы

Функция	Конфигурационные файлы
Upstart	/etc/event.d/*
Инициализация системы	/etc/rc.d/rcsysconfig
Init-сценарии	/etc/rc.d/init/*
Ссылки на уровни запуска	/etc/rc.d/rcn.d/*
Конфигурационные файлы	/etc/sysconfig/*

Особенности Upstart

Хотя Fedora применяет при запуске Upstart, стандартный уровень запуска попрежнему настраивается в `/etc/inittab`. Но все остальные настройки из этого файла игнорируются!

Upstart отвечает за графическое представление процесса загрузки (`plymouth`), за запуск эмуляторов терминалов для текстовых консолей и за запуск системы X (`prefdm`). Все остальные вопросы инициализации решаются с помощью сценариев, обеспечивающих совместимость с Init-V. Программа `prefdm` начинает работу после завершения выполнения всех сценариев Init-V с уровня запуска `5`.

И `plymouth`, и X в Fedora используют консоль `1` (а не консоль `7`, как в большинстве других дистрибутивов). Поэтому на уровне запуска `5` в этой консоли не начинает работать *никакой* эмулятор терминала.

Индивидуальная настройка процесса Init-V

С помощью файла `/etc/init.d/rc.local` вы можете с легкостью произвести индивидуальную настройку процесса Init-V. Это сценарий, выполняемый после всех остальных сценариев Init-V в ситуации, когда впервые активизируется уровень запуска `2`, `3`, `4` или `5`. При последующем изменении уровня запуска или при окончании работы компьютера сценарий больше *не* выполняется!

Интерактивное выполнение процесса Init-V

Если в ходе инициализации системы была нажата клавиша I, командой `touch` создается файл `/var/run/confirm`. Чтобы убедиться, что этот файл существует, выполните `/etc/rc.d/rc`. Если он существует, то при выполнении всех сценарных файлов `rcn.d` система выдает запрос `YES/NO/CONTINUE`, где `CONTINUE` означает, что все остальные файлы должны выполняться без такого запроса.

Интерактивный режим удобен, если при выполнении сценариев Init-V, предназначенных для активизации уровней запуска, возникают проблемы. Однако не забывайте, что интерактивный режим задействуется только после окончания инициализации системы. Если проблема возникнет ранее, вам сильно не повезет.

Запуск сценариев Init-V

Команда service частично избавляет вас от работы с клавиатурой при вводе сценария Init-V вручную:

```
root# service samba start
root# service samba stop
root# service samba status
samba остановлена
```

При переходе с одного уровня запуска на другой останавливаются только те функции, которые были активизированы на предыдущем уровне запуска, а на новом уровне запуска не нужны. Аналогичным образом запускаются только такие функции, которые до этого были неактивны. Чтобы это гарантировать, при запуске любого системного процесса в каталоге /var/lock/subsys создается новый файл. После окончания процесса этот файл удаляется. Более подробное описание того, как в Fedora реализована концепция процессов Init-V, содержится в каталоге /usr/share/doc/initscripts-n.

Управление ссылками Init-V

Команда chkconfig помогает управлять ссылками на сценарии уровней запуска. Если сообщить этой команде параметр --list, она выдает обзор всех сценариев и указывает, на каком уровне они были запущены. Если установлен демон xinetd, будут перечислены и его службы.

```
root# chkconfig --list
NetworkManager  0:Off 1:Off 2:On  3:On  4:On  5:On  6:Off
NetworkManagerD 0:Off 1:Off 2:Off 3:Off 4:Off 5:Off 6:Off
acpid           0:Off 1:Off 2:Off 3:On  4:On  5:On  6:Off
...
Службы на базе xinetd:
    chargen-udp: Off
    ...
```

С помощью параметра --del можно вообще отменить выполнение сценариев уровней запуска. Сам сценарий сохраняется, удаляется лишь ссылка в каталоге rcn.d.

```
root# chkconfig --del samba
```

Команда chkconfig --add добавляет для всех предусмотренных уровней запуска ссылки для старта и завершения новой службы. Правда, параметр --add функционирует лишь в том случае, если в файле сценария Init-V прямо указана информация о том, на каком уровне по умолчанию запускается сценарий.

По многим сценариям такая информация отсутствует. Чтобы в дальнейшем можно было запускать подобный сценарий автоматически, используйте chkconfig --level *n имя* on/off. Такой синтаксис удобен и в тех случаях, когда вы, несмотря на настройки, заданные по умолчанию, сами хотите задать для сценария уровень запуска.

Допустим, в сценарном файле /etc/rc.d/init.d/httpd отсутствует информация о том, на каких уровнях должен автоматически запускаться сервер. Поэтому chkconfig --add создает только ссылки для завершения работы веб-сервера. Команда chkconfig --level добавляет ссылки для запуска сервера на уровнях 3 и 5; chkconfig --list отображает результат:

```
root# chkconfig --add httpd
root# chkconfig --level 35 httpd on
root# chkconfig --list httpd
httpd 0: Off 1: Off 2: Off 3:On 4: Off 5:On 6: Off
```

Команды chkconfig --add и chkconfig --del работают и со службами xinetd. Xinetd — это демон, задача которого в том, чтобы запускать остальные сетевые службы только при необходимости (см. раздел 14.16). В качестве альтернативы chkconfig для администрирования Init-V предоставляется графический пользовательский интерфейс system-config-services. Вы также можете запустить эту программу командой serviceconf. Она позволяет вручную запускать, останавливать и снова запускать некоторые демоны и сетевые службы xinetd. Кроме того, с помощью мыши вы можете отметить для определенного уровня запуска, какие службы должны запускаться по умолчанию.

Как построены файлы сценариев Init-V

Ниже показан пример сценария Init-V из Red Hat и Fedora. Предметом дискуссии является сценарий для запуска веб-сервера httpd, который я немного сократил ради экономии места.

```
#!/bin/bash
# httpd          Сценарий для запуска сервера Apache HTTP
#
# chkconfig:  -  85 15
# description:   Сервер Apache HTTP - это эффективный расширяемый сервер, \
#                работающий в соответствии с современными стандартами HTTP.
# processname: httpd
# config: /etc/httpd/conf/httpd.conf
# config: /etc/sysconfig/httpd
# pidfile: /var/run/httpd/httpd.pid
### НАЧАЛО РАЗДЕЛА ОБ INIT
# Provides: httpd
# Required-Start: $local_fs $remote_fs $network $named
# Required-Stop: $local_fs $remote_fs $network
# Should-Start: distcache
# Short-Description: start and stop Apache HTTP Server
# description:   Сервер Apache HTTP - это эффективный расширяемый сервер, \
#                работающий в соответствии с современными стандартами HTTP.
### КОНЕЦ РАЗДЕЛА ОБ INIT
# Загрузка различных функций
. /etc/rc.d/init.d/functions
# Загрузка конфигурационного файла
```

```
if [ -f /etc/sysconfig/httpd ]; then
      . /etc/sysconfig/httpd
fi
# Инициализация различных переменных
HTTPD_LANG=${HTTPD_LANG-"C"}
INITLOG_ARGS=""
apachectl=/usr/sbin/apachectl
httpd=${HTTPD-/usr/sbin/httpd}
prog=httpd
pidfile=${PIDFILE-/var/run/httpd.pid}
lockfile=${LOCKFILE-/var/lock/subsys/httpd}
RETVAL=0
...
# Функции для start, stop и reload
start() {
  echo -n $"Starting $prog: "
  check13 || exit 1
  LANG=$HTTPD_LANG daemon $httpd $OPTIONS
  RETVAL=$?
  echo
  [ $RETVAL = 0 ] && touch ${lockfile}
  return $RETVAL
}
stop() { ... }
reload() { ... }
# Вызов предыдущих функций
case "$1" in
  start)
    start
    ;;
  stop)
    stop
    ;;
  restart)
    stop
    start
    ;;
  *)
    echo $"Usage: $prog {start|stop|restart|condrestart|reload|status|\
                         fullstatus|graceful|help|configtest}"
    exit 1
esac
exit $RETVAL
```

Для правильной интеграции `chkconfig` в систему Init-V решающее значение имеет строка комментария в начале сценария. При этом действует следующий синтаксис `chkconfig`: `runlevel start stop`. Параметр `start` указывает, на каком уровне должен автоматически запускаться сценарий (например, `35` — для уровней запуска `3` и `5`). Если здесь, как в приведенном выше примере, поставить только дефис, то сценарий нужно добавить к процессу Init-V с помощью `chkconfig--level`.

Параметр `start` указывает, какой номер должна иметь ссылка для запуска. Здесь можно указать число в диапазоне от 00 до 99. Чем выше значение, тем раньше будет запущен сценарий в ходе выполнения Init-V. Аналогичным образом `stop` указывает номер ссылки для остановки процесса. Обратите внимание, что в `chkconfig` также обязательно должна присутствовать строка `description`!

Основной программный код начинается лишь с команды `case`. В зависимости от того, какие команды были переданы сценарию, на данном этапе выполняется одна из предварительно определенных функций — для запуска или остановки httpd.

Более подробная информация о строении сценария Init-V в Fedora/Red Hat сообщается в следующем файле: `/usr/share/doc/initscripts-nnn/sysvinitfiles`.

Пример собственного сценария Init-V

Если вам требуется собственная (сетевая) функция и собственный сценарий Init-V, то код может получиться гораздо компактнее. Код из следующего примера активизирует или деактивизирует функцию маскарадинга, позволяя, таким образом, другим компьютерам использовать соединение с Интернетом (подробности в разделе 17.3). Вы можете и сами писать по этому образцу сценарии Init-V, активизирующие ваш брандмауэр и запускающие ADSL-соединение уже на этапе выполнения процесса Init-V и т. д.

```
#!/bin/bash
# /etc/rc.d/init.d/masq
#
# chkconfig: 35 95 05
# description: начало маскарадинга
. /etc/rc.d/init.d/functions
INETDEV=eth0
case "$1" in
  start)
    echo -n $"Starting masquerading: "
    echo 1 > /proc/sys/net/ipv4/ip_forward
    iptables -A POSTROUTING -t nat -o $INETDEV -j MASQUERADE
    touch /var/lock/subsys/masq
    echo_success
    ;;
  stop)
    echo -n $"Stopping masquerading: "
    iptables -D POSTROUTING -t nat -o $INETDEV -j MASQUERADE
    echo 0 > /proc/sys/net/ipv4/ip_forward
    rm -f /var/lock/subsys/masq
    echo_success
    ;;
  *)
    echo $"Usage: $prog {start|stop}"
exit 1
esac
```

14.14. Запуск системы в SUSE

На рис. 14.7 в обобщенном виде показано, какие сценарии выполняются при запуске системы openSUSE на уровне 5 с помощью Init-V. В табл. 14.6 перечислены все важные конфигурационные файлы.

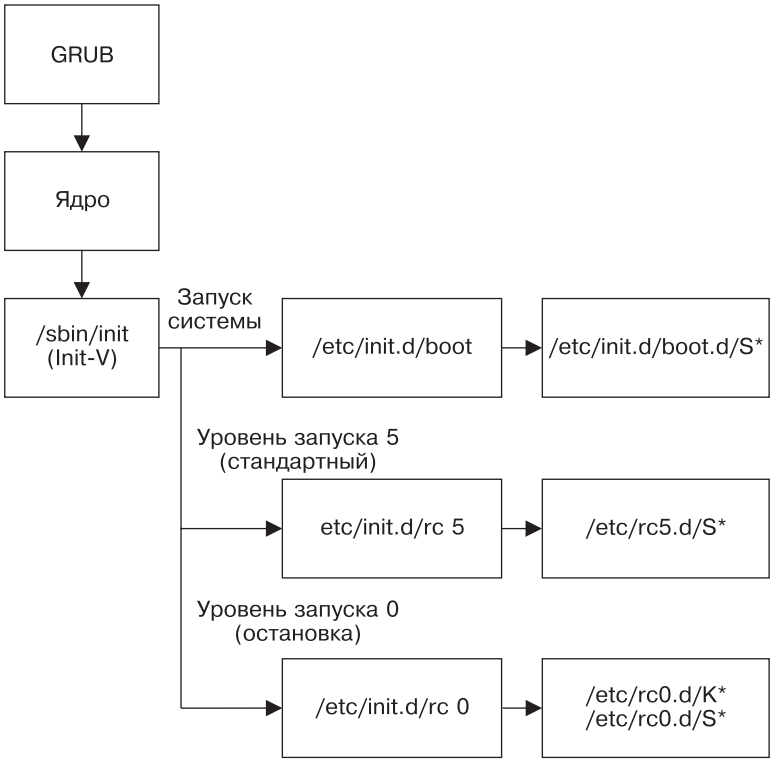

Рис. 14.7. Старт системы на уровне запуска 5 в дистрибутиве SUSE

Таблица 14.6. Конфигурация запуска системы SUSE

Функция	**Конфигурационные файлы**
Инициализация системы	/etc/init.d/boot, /etc/init.d/boot.d/*
Init-сценарии	/etc/init.d/*
Ссылки на уровни запуска	/etc/init.d/rcn.d/*
Конфигурационные файлы	/etc/sysconfig/*

В локальную систему можно вносить изменения в файле `/etc/rc.d/boot.local`. Сценарий должен содержать только те команды, которые выполняются однократно, при запуске системы. Типичный пример — команды `modprobe`, предназначенные для загрузки определенных модулей ядра. `boot.local` выполняется сценариями `rc`.

Интерактивное выполнение процесса Init-V

Если при запуске системы возникнут проблемы, вы также можете интерактивно выполнить процесс Init-V. Для этого при запуске через GRUB сообщите параметр загрузки `confirm`. Теперь перед выполнением каждого сценария Init-V будет выводиться запрос о подтверждении. На каждый сценарий выделяется пять секунд времени, в течение которых можно отменить запуск нажатием клавиши N.

Запуск сценариев Init-V

В SUSE можно вызывать любые сценарии не только командой `/etc/init.d/`*имя*, которая, скорее, неудобна, но и в виде `rc`*имя* (то есть, например, `rcsmb` — для запуска или остановки сервера Samba). Соответствующие ссылки находятся в `/usr/sbin`.

Запуск X. Графическая система X запускается сценарием Init-V `xdm`. Какой экранный менеджер будет запущен далее (например, `kdm4`), определяет переменная `DISPLAYMANAGER`, настраиваемая в `/etc/sysconfig/displaymanager`. Сам экранный менеджер запускается относительно поздно — после всех сетевых служб. Раньше него запускается сценарий `earlyxdm`, загружающий нужные файлы в кэш с помощью команды `preload`.

Внутрисистемная реализация перехода с одного уровня запуска на другой. При переходе с одного уровня запуска на другой сценарий `/etc/init.d/rc` сравнивает каталоги `/etc/init.d/rcnew.d` и `rcold.d`, проверяя, какие функции изменяются при таком переходе. Такие функции останавливаются (сценарные файлы `rcold.d/K*`) или запускаются заново (`/etc/init.d/rcnew.d/S*`). Функции, которые не изменяются при смене уровня запуска, не останавливаются, а значит, и не запускаются заново.

Параллельное выполнение сценариев Init-V. Сценарии Init-V, не зависящие друг от друга, по умолчанию выполняются параллельно, а не последовательно. Управление деталями процесса загрузки осуществляется в файле `/etc/sysconfig/boot`. Чтобы обеспечить параллельное выполнение, ссылки Init-V нужно изменить с помощью команды `insserv` (но не напрямую). Она создает файлы с расширением `.depend.*`, содержащие информацию о взаимозависимостях между сценариями Init-V. Более подробно запуск системы SUSE описан на странице сайта-руководства, посвященной `init.d`.

Графический процесс загрузки

При запуске системы SUSE отображает индикатор загрузки. Если вместо него вы хотите видеть подробные сообщения от ядра и Init-V, нажмите Esc. Теперь сообщения будут выводиться в первой текстовой консоли, на фоне рисунка (экран-заставка).

За графический процесс загрузки отвечает пакет `bootsplash`. Существуют загрузочные параметры, которые могут оказывать влияние как на сообщения о ходе загрузки, так и на экран-заставку. Есть три возможные настройки:

- `splash=silent` — в консоли `1` отображается фоновый растровый рисунок, а в процессе запуска вы видите только индикатор загрузки; действует по умолчанию;

- splash=verbose — в консоли 1 также отображается растровый рисунок, на фоне которого вы видите подробные сообщения о ходе загрузки;
- splash=native — консоль 1 отображается в текстовом режиме, в ней вы видите только подробные сообщения о ходе загрузки.

Если вы используете загрузчик GRUB, укажите желаемый параметр в файле /boot/grub/menu.lst. Если же, напротив, вы применяете LILO, добавьте параметр в строку append и выполните команду lilo. Для того чтобы отключить экран-заставку на ходу, можно также воспользоваться следующей командой:

```
root# echo 0 > /proc/splash
```

Управление ссылками Init-V

Если хотите переадресовать ссылки /etc/init.d/rcn.d на новый сценарий Init-V, просто выполните команду insserv *имя_сценария*. Она интерпретирует комментарии, содержащиеся в сценарии и описывающие настройки для стандартного начала и завершения работы. Эти комментарии сообщают, на каких уровнях запуска должен выполняться сценарий.

```
root# insserv squid
```

Чтобы снова удалить ссылки, указывающие на сценарий, выполните insserv с параметром -r.

Команда insserv также отвечает за правильную нумерацию ссылок (такая нумерация определяет порядок, в котором будут выполняться сценарии Init-V). Для того чтобы решить, какой номер будет у ссылки, команда insserv интерпретирует комментарии Provides и Requires в сценарии Init-V. При необходимости команда может перенумеровать и все уже имеющиеся ссылки, чтобы правильно поставить новый сценарий по отношению к другим. Такой автоматизм хорош при работе, но из-за него становится сложнее изменить последовательность выполнения сценариев, заданную на старте.

В SUSE — опять же, ради обеспечения совместимости — имеется команда chkconfig. Параметры -add, -del и -list функционируют так же, как и в Red Hat, но с другими параметрами есть и разночтения. Внутри системы chkconfig обращается к insserv.

Строение файлов Init-V

Далее показано строение сценария Init-V в SUSE. Это сценарий для запуска демона SSH, который я немного сократил ради экономии места.

```
#! /bin/sh
# /etc/init.d/sshd
### НАЧАЛО РАЗДЕЛА ОБ INIT
# Provides:       sshd
# Required-Start: $network $remote_fs
```

```
# Required-Stop:  $network $remote_fs
# Default-Start:  3 5
# Default-Stop:   0 1 2 6
# Description:    Start the sshd daemon
### КОНЕЦ РАЗДЕЛА ОБ INIT
# различные инициализации
SSHD_BIN=/usr/sbin/sshd
test -x $SSHD_BIN || exit 5
SSHD_SYSCONFIG=/etc/sysconfig/ssh
test -r $SSHD_SYSCONFIG || exit 6
. $SSHD_SYSCONFIG
SSHD_PIDFILE=/var/run/sshd.init.pid
# загрузка разных вспомогательных функций (rc_check, rc_status ...)
. /etc/rc.status
# Сбросить статус службы в исходное состояние
rc_reset
case "$1" in
  start)
    ... ssh-Keys erzeugen, falls sie noch nicht existieren
    echo -n "Starting SSH daemon"
    startproc -f -p $SSHD_PIDFILE $SSHD_BIN $SSHD_OPTS -o "PidFile=$SSHD_PIDFILE"
    rc_status -v
    ;;
  stop)
    echo -n "Shutting down SSH daemon"
    killproc -p $SSHD_PIDFILE -TERM $SSHD_BIN
    rc_status -v
    ;;
  restart)
    ... diverse weitere Varianten
      *)
    echo "Usage: $0 {start|stop|status|...}"
    exit 1
    ;;
esac
rc_exit
```

Для правильной интеграции сценария в систему Init-V решающее значение имеют комментарии `INIT`, приведенные в начале. Параметр `Required-Start` или `Required-Stop` указывает, на каких уровнях запуска должен по умолчанию загружаться сценарий. Эта информация интерпретируется `insserv`.

Пример сценария Init-V. Если вы хотите интегрировать некоторые функции уровней запуска в процесс Init-V, стройте файл запуска и остановки по образцу `/etc/init.d/skeleton`. Если для реализации собственных (сетевых) функций вам требуется специальный сценарий Init-V, то код может получиться гораздо компактнее. Код из следующего примера активизирует или деактивизирует функцию маскарадинга, позволяя, таким образом, другим компьютерам использовать соединение с Интернетом (подробности читайте в разделе 17.3).

```
#! /bin/sh
# /etc/init.d/masq
### НАЧАЛО РАЗДЕЛА ОБ INIT
# Provides:       masq
# Required-Start: $network
# Required-Stop:  $network
# Default-Start:  3 5
# Default-Stop:   0 1 2 6
# Description:    activates masquerading ...
### КОНЕЦ РАЗДЕЛА ОБ INIT
# различные функции, например rc_status
. /etc/rc.status
INETDEV=eth0
case "$1" in
  start)
    echo -n "Starting masquerading"
    echo 1 > /proc/sys/net/ipv4/ip_forward
    iptables -A POSTROUTING -t nat -o $INETDEV -j MASQUERADE
    rc_status -v
    ;;
  stop)
    echo -n "Shutting down masquerading"
    iptables -D POSTROUTING -t nat -o $INETDEV -j MASQUERADE
    echo 0 > /proc/sys/net/ipv4/ip_forward
    rc_status -v
    ;;
  *)
    echo "Usage: $0 {start|stop}"
    exit 1
    ;;
esac
rc_exit
```

14.15. Запуск системы Ubuntu

На рис. 14.8 показано, как происходит запуск системы в дистрибутиве Ubuntu 9.10 с применением Upstart. Стандартный уровень запуска — 2. В табл. 14.7 показано, какие конфигурационные файлы отвечают за запуск системы.

Таблица 14.7. Конфигурация запуска системы Ubuntu

Функция	**Конфигурационные файлы**
Upstart	/etc/init/*.conf
Инициализация системы	/etc/init.d/rcS, /etc/rcS.d/*
Init-сценарии	/etc/init.d/*
Ссылки на уровни запуска	/etc/rcn.d/rcn.d/*
Конфигурационные файлы	/etc/default/*

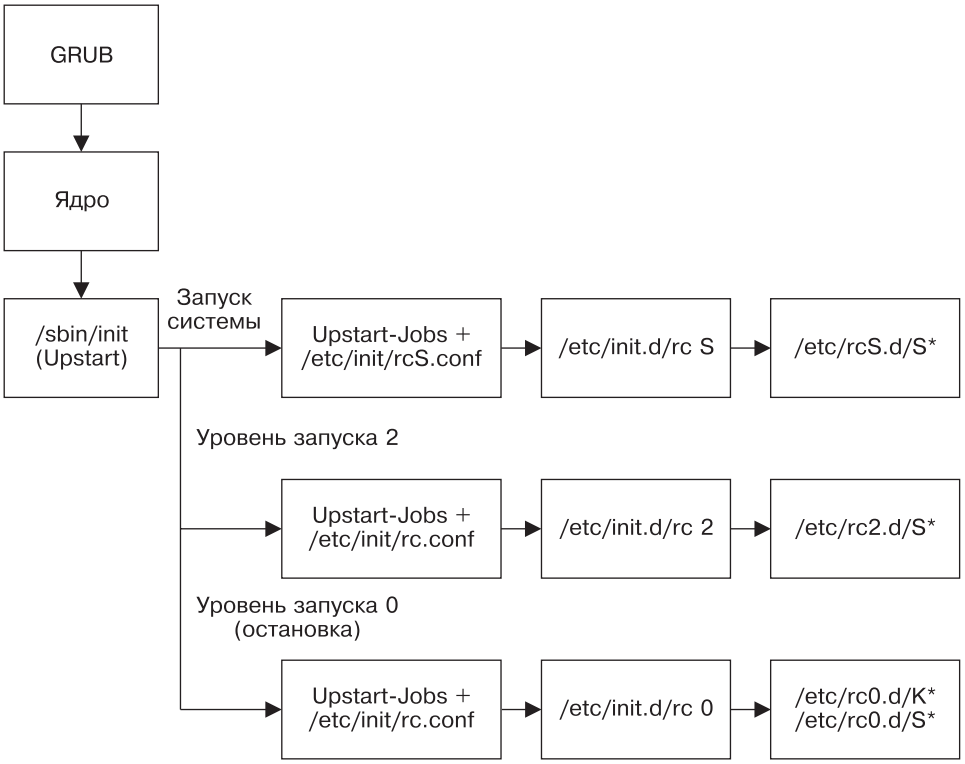

Рис. 14.8. Старт системы на уровне запуска 2 в Ubuntu

Особенности Upstart

При запуске Ubuntu 9.10 Upstart выполняет около трети всех системных и сетевых служб. Остальные функции инициализации выполняются с помощью сценариев, совместимых с Init-V. При этом та часть запуска, которая выполняется Init-V, протекает так же, как в Debian (см. раздел 14.12). Применяются те же инструменты администрирования. Стандартный уровень запуска устанавливается в файле `/etc/init/rc-sysinit.conf`. Планируется, что в версии Ubuntu 10.04 все важнейшие сетевые службы будут запускаться через Upstart, а не через уровень совместимости с Init-V.

Для того чтобы было легче перейти с Fedora на Red Hat, вы также можете запускать сценарии Init-V командой `service` (то есть, к примеру, `service apache2 start` вместо `/etc/init.d/apache2 start`).

Запуск X

Графическая система X запускается Upstart (см. файл `/etc/init/gdm.conf`). Чтобы можно было запустить X, предварительно необходимо лишь инициализировать файловую систему и запустить HAL. Экранный менеджер стартует уже через несколько секунд после начала процесса загрузки.

14.16. Демон интернет-сервисов

Программы, предоставляющие в распоряжение сервисы локальных сетей и Интернета, можно разделить на две группы.

- К первой группе относятся программы-демоны, работающие без перерывов. Почти все сетевые службы, упоминаемые в нашей книге, — веб-сервер Apache, файловый сервер Samba, SSH-сервер — относятся к этой группе. Такие программы запускаются при старте системы сценарием Init-V (а в будущем, возможно, будут запускаться Upstart) и наблюдают за работой IP-порта. Как только им встречается IP-пакет, адресованный этому порту, такой пакет интерпретируется и на него отсылается ответ.
- Ко второй группе относятся редко используемые программы, запускаемые лишь при необходимости. Чтобы не запускать все эти программы сразу, выполняется так называемый демон интернет-сервисов. Эта программа одновременно наблюдает за несколькими IP-портами и только при необходимости активизирует соответствующие серверные службы. Сам демон интернет-сервисов запускается обычным образом — сценарием Init-V.

Ранее наиболее распространенным демоном интернет-сервисов был `inetd`. В настоящее время эта программа считается устаревшей. В зависимости от дистрибутива вместо нее могут использоваться `openbsd-inetd` или `xinetd`. Обычно ни один из этих пакетов не устанавливается по умолчанию. Устанавливать их требуется лишь тогда, когда другой программе для работы необходим функционал `inetd`, а в пакете задана соответствующая взаимосвязь (например, SWAT, см. раздел 20.4).

Файл /etc/services

Независимо от того, какой именно демон интернет-сервисов применяется в вашем дистрибутиве, в файле `/etc/services` соотносятся названия различных интернет-сервисов (например, ftp, telnet и т. д.), типы их протоколов и номера портов. Например, почтовые серверы (почтовые агенты) при работе используют порт 25 и протоколы tcp и udp. Далее приведен фрагмент такого файла:

```
# /etc/services (выдержка)
# имя    порт/протокол псевдоним комментарий
ftp-data 20/tcp                 # Передача файлов [данные по умолчанию]
ftp-data 20/udp                 # Передача файлов [данные по умолчанию]
ftp      21/tcp                 # Передача файлов [контроль]
ssh      22/tcp                 # Протокол SSH для удаленного входа в систему
ssh      22/udp                 # Протокол SSH для удаленного входа в систему
smtp     25/tcp        mail     # Простой протокол электронной почты
...
```

Файл /etc/inetd.conf

Если в вашем дистрибутиве работает `openbsd-inetd`, его конфигурация задается в файле `/etc/inetd.conf`. Программа запускается в том случае, если в конфигурационном

файле inetd.conf есть как минимум одна активная запись. Каждая запись в этом файле состоит из строки с шестью столбцами.

- В первом столбце указывается название службы, которая должна быть определена в /etc/services.
- Во втором и третьем столбце описано, как служба обменивается информацией (тип сокета и протокол).
- В четвертом столбце указано, должна ли одна и та же служба многократно запускаться при поступлении запросов (nowait) или же такие запросы должны обрабатываться только после того, как уже запущенная служба выполнит свою задачу (wait). В данном случае можно указать время задержки в секундах.
- В пятом столбце определено, с какими правами должен запускаться процесс.
- Остаток строки — это команда, которую требуется выполнить. При этом tcpd сначала проверяет, разрешено ли выполнение этой команды правилами, установленными программой TCP-Wrapper.

```
# Файл /etc/inetd.conf
swat stream tcp nowait.400 root /usr/sbin/tcpd /usr/sbin/swat
...
```

Как правило, вам не придется самостоятельно заниматься конфигурацией inetd.conf. При установке пакета, зависящего от inetd, файл inetd.conf автоматически дополняется соответствующим образом. Обратите внимание, что строки, начинающиеся с #, как обычно, считаются комментариями! Изменения, внесенные в inetd.conf, вступают в силу только после выполнения /etc/init.d/openbsd-inetd reload.

Файл /etc/xinetd.conf

В файле /etc/xinetd.conf содержатся некоторые базовые настройки xinetd, касающиеся, например, функции журналирования и стандартного IP-адреса. Как правило, настройки можно оставить без изменений. Очень важна команда includedir, указывающая каталог с остальными конфигурационными файлами (обычно /etc/xinetd.d).

Каталог xinetd.d/*

В каталоге /etc/xinetd.d содержатся отдельные конфигурационные файлы для каждой службы, управляемой xinetd. Названия файлов в /etc/xinetd.d не играют роли: xinetd просто считывает все файлы из этого каталога и интерпретирует их (не учитываются файлы, названия которых заканчиваются символом ~ или содержат точку).

Отдельные конфигурационные файлы построены одинаково. В следующем примере показан файл для сервера RSYNC:

```
# /etc/xinetd.d/rsync
service rsync
{
  disable        = yes
  socket_type    = stream
  wait           = no
```

```
  user            = root
  server          = /usr/bin/rsync
  server_args     = --daemon
  log_on_failure += USERID
}
```

Кратко опишу важнейшие ключевые слова, которые могут встречаться в конфигурационных файлах `xinetd`. Подробное описание дается в `man xinetd.conf`.

- `service` — означает службу (в соответствии с `/etc/services`).
- `socket_type` и `protocol` — указывают, как должны передаваться файлы между клиентом и сервером.
- `type=INTERNAL` — указывает, что данная служба предоставляется `xinetd` напрямую.
- `server` — задает имя программы (если это не внутренняя служба `xinetd`).
- `server_args` — указывает необязательные параметры, которые можно сообщить службе при запуске.
- `user` — говорит, под какой учетной записью выполняется программа (обычно root, но это может быть news, mail и т. д.).
- `disable=Yes/No` — указывает, активна служба или заблокирована. Если служба заблокирована, то в конфигурационном файле стоит `disable=Yes`. В Fedora и Red Hat службы `xinetd` также можно деактивизировать с помощью команды `chkconfig --del` *имя* и снова активизировать с помощью `chkconfig --add` *имя*. Эти команды изменяют только строку `disable`, не затрагивая остальную конфигурацию.
- `log_*` — указывает, следует ли протоколировать использование службы.

Файлы /etc/hosts.allow и hosts.deny

Эти файлы определяют, с какого компьютера и какие службы могут использоваться. Настройки действительны лишь для тех программ, которые при работе обращаются к библиотеке TCP-Wrapper. К таким программам, кроме `xinetd`, также относятся сервер SSH, NFS и программа CUPS в SUSE. Строение конфигурационных файлов `hosts.allow` и `hosts.deny` подробно описано в разделе 18.2.

Особый случай представляет собой `openbsd-inetd`: этот пакет связан с библиотекой TCP-Wrapper, но по умолчанию ее функции отключены! Чтобы активизировать их, нужно запустить `openbsd-inetd` с параметром `-l`. В Debian и Ubuntu для этого создается файл `/etc/default/openbsd-inetd`, в который вставляется следующая строка:

```
# Файл /etc/default/openbsd-inetd (Debian, Ubuntu)
# Активизация функций TCP-Wrapper
OPTIONS="-l"
```

С таким параметром `inetd.conf` должен еще вызвать tcpd, чтобы избежать повторной проверки правил TCP-Wrapper.

15 Ядро и модули

В этой главе будет рассмотрено ядро Linux и его модули. Модули — это части ядра, которые можно загружать по мере необходимости, а именно тогда, когда впервые запрашивается тот или иной аппаратный компонент. В разделе 15.1 рассказывается, почему такое подключение модулей обычно осуществляется автоматически и что делать, если автоматическая система откажет.

Достаточно редко возникает необходимость заново скомпилировать ядро. Компилировать отдельный модуль так, чтобы он подошел к актуальному ядру, приходится гораздо чаще (например, для драйвера графической карты или для VirtualBox). В разделе 15.2 доказано, что компилировать целое ядро или отдельные модули не так сложно и эта работа по плечу простым смертным. Наконец, в разделе 15.3 вы узнаете, как получить из файловой системы `/proc` или `/sys` актуальную информацию о ядре.

Параметры загрузки ядра в этой главе не рассматриваются. С помощью таких параметров вы можете передавать ядру детальные сведения о Linux. Поскольку этот вопрос напрямую связан с запуском системы, некоторые параметры уже были описаны в предыдущей главе (см. раздел 14.9).

Разумеется, эта глава предназначена только для продвинутых пользователей Linux. Начинающим я настоятельно рекомендую работать с готовым ядром своей системы и не устанавливать пакетов, предназначенных для использования в другом дистрибутиве! Если прямо не указана иная версия, то речь идет о ядре версии 2.6.n.

15.1. Модули ядра

Ядро — это часть Linux, отвечающая за выполнение простейших функций, таких как управление памятью, доступ к жестким дискам и сетевым картам и т. д. При этом ядро организовано по модульному принципу: сначала, то есть при запуске компьютера, загружается основное ядро, содержащее только те функции, которые требуются для старта системы.

Если в ходе работы понадобятся дополнительные функции (например, для работы с конкретным оборудованием), то необходимый код подключится к ядру как модуль. Если в течение некоторого времени эти дополнительные функции будут

не нужны, модуль может быть удален из ядра. У такой модульной концепции множество преимуществ.

- Модули можно подключать по мере необходимости. Если определенный модуль используется редко, то, не подключая его, вы можете экономить память. Это означает, что ядро имеет размер не больше, чем это необходимо, и оптимально подходит к оборудованию пользователя.
- При изменении конфигурации оборудования (например, при подключении новой сетевой карты) не нужно компилировать новое ядро — можно просто подключить соответствующий модуль. Все распространенные дистрибутивы построены по такому принципу.
- При разработке нового модуля не требуется все время перезапускать компьютер. Достаточно заново скомпилировать модуль. Затем этот модуль можно будет протестировать, не останавливая работу системы.
- Производители аппаратного обеспечения могут предоставлять модули для поддержки своего оборудования в виде двоичных файлов, не раскрывая при этом код. Пока такой метод применяется лишь иногда и, конечно же, он не лишен недостатков. (Модуль должен точно подходить к ядру, то есть для каждой новой версии ядра он должен компилироваться повторно. Если обнаружится ошибка, ее может исправить только производитель, но не пользователь — ведь исходный код недоступен.)

Подробная базовая информация по работе с модулями ядра сообщается на следующем сайте: http://www.tldp.org/HOWTO/Module-HOWTO/.

Автоматическая загрузка модулей. За то, чтобы модули ядра действительно загружались автоматически по мере надобности, отвечает компонент `kmod`, интегрированный в ядро. Этот компонент управляется файлом `/etc/modprobe.conf`, о котором подробно рассказано в подразделе «Конфигурация модуля» далее.

Ядро и модули должны подходить друг к другу. До версии ядра 2.6.15 ядро и его модули должны были точно подходить друг к другу: невозможно было загрузить модуль, скомпилированный для другой версии ядра (пусть и очень близкой). Для каждой версии ядра создавался отдельный каталог модулей `/lib/modules/`*`версия_ядра`*. При применении как раз таких модулей, которые не поставляются вместе с дистрибутивом (таковы, в частности, модули для графических драйверов ATI или NVIDIA), становится очень сложно точно подобрать модули к ядру.

Возможность использования модулей другой версии ядра. В версии ядра 2.6.16 и выше появляется функция `module versioning`, позволяющая применять модули и из других версий ядра, и ситуация значительно упрощается: вместе с модулем сохраняется дополнительная информация, сообщающая, возможно ли совместно использовать конкретный модуль и конкретное ядро, несмотря на то что они относятся к разным версиям. Часто можно применить и такой модуль, версия (номер) которого не соответствует версии ядра. Чтобы этот механизм работал, необходимо активизировать функцию `module versioning` уже при компилировании, кроме того, не должно быть никаких изменений в интерфейсах, расположенных между модулем и ядром.

Если вы хотите написать для SUSE собственные модули ядра, поддерживающие module versioning, сначала установите пакет kernel-syms. Функция module versioning также может называться kernel symbol versions или modversions. Подробно этот механизм описан на следующем сайте: http://www.oreilly.de/german/freebooks/linuxdrive2ger/kerver.html.

Команды для управления модулями

Во всех распространенных дистрибутивах предусмотрена возможность загрузки дополнительных модулей по мере надобности. Пример: с помощью команды mount вы подключаете к дереву каталогов файловую систему USB-флешки. При этом автоматически активизируется модуль vfat, необходимый для считывания файловой системы флешки.

Итак, управление модулями, как правило, осуществляется автоматически и является прозрачным, а вам не приходится прибегать к описанным ниже командам, предназначенным для управления модулями вручную. И все же рекомендую изучить эти команды, чтобы при необходимости иметь возможность вмешаться в работу модулей.

Все модули находятся в каталоге /lib/modules/*n*, где *n* — версия действующего ядра. Файлы модулей имеют расширение *.ko.

Определение версии ядра. Команда uname -r выдает номер версии используемого ядра:

```
user$ uname -r
2.6.28-13-generic
```

Загрузка файла модуля. Команда insmod интегрирует указанный модуль в ядро. При этом необходимо передать полное имя файла. Дополнительно можно передать модулю параметры. Если вы хотите указать шестнадцатеричные значения, перед ними нужно ставить 0x, то есть option=0xff.

```
root# insmod /lib/modules/2.6.28-13-generic/kernel/fs/fuse/fuse.ko
```

Команда insmod -f пытается загрузить модуль даже тогда, когда он не подходит к действующей версии ядра. Если между ядром и модулем нет никакой несовместимости, такая попытка будет успешной.

Как правило, для загрузки модулей ядра применяется не insmod, а modprobe. Эта команда имеет два преимущества: сама ищет файл модуля (вам следует указать только название модуля), а при необходимости также загружает все модули, требуемые для работы. Кроме того, учитываются все параметры модуля, указанные в /etc/modprobe.conf. Для работы команды modprobe необходимо, чтобы модуль имел правильную конфигурацию (modprobe.conf и modules.dep).

```
root# modprobe fuse
```

Список загруженных модулей. Команда lsmod возвращает, как правило, очень длинный список всех модулей ядра, загруженных на настоящий момент:

```
root# lsmod | sort
Module        Size   Used by
ac            4933   0
```

```
autofs4      19013   1
battery       9285   0
bluetooth    44069   5 hidp,rfcomm,l2cap
button        6609   0
...
fuse         36313   0
...
```

Удаление модулей. Команда `rmmod` вновь удаляет указанный модуль из ядра и предоставляет память, которую занимал данный модуль. Эта команда может быть выполнена лишь в том случае, если удаляемый модуль сейчас не используется. Команда `rmmod -a` удаляет из памяти все модули, не используемые в данный момент.

```
root# rmmod fuse
```

Информация о модулях. Команда `modinfo` выдает подробную информацию о модуле. Он не должен находиться в ядре. В следующем примере приведены данные по модулю e1000. Это драйвер для сетевого адаптера Intel.

```
root# modinfo e1000
filename:     /lib/modules/2.6.28-13-generic/kernel/drivers/net/e1000/e1000.ko
version:      7.3.21-k3-NAPI
license:      GPL
description:  Intel(R) PRO/1000 Network Driver
author:       Intel Corporation, <linux.nics@intel.com>
...
depends:
vermagic:     2.6.28-13-generic SMP mod_unload modversions
parm:         Speed:Speed setting (array of int)
parm:         Duplex:Duplex setting (array of int)
...
```

Конфигурация модуля

Управление модулями происходит как по волшебству. Если вы подключаете к файловой системе новый раздел и в нем используется такой формат файловой системы, который ранее не применялся, то автоматически загружается модуль для работы с этой файловой системой. Если раздел находится на диске SCSI, то активизируются и модули SCSI (если они еще не были загружены); в ходе инициализации сетевых функций автоматически загружается драйвер, необходимый для работы вашей сетевой карты и т. д.

За автоматическую загрузку модулей ядра отвечает компонент `kmod`, интегрированный в ядро. Чтобы все эти функции правильно работали, используется ряд конфигурационных механизмов.

- **Модули, необходимые для запуска компьютера.** Некоторые модули ядра нужны уже при запуске компьютера, в частности модули для доступа к файловой системе. Пусть они и не являются неотъемлемой частью ядра, но должны передаваться ядру загрузчиком GRUB в файле `initrd` при запуске компьютера (см. раздел 14.5).

- **Модули основных функций.** Модули для базового управления компонентами оборудования (например, для системы USB) загружаются различными сценариями процесса Init-V, которые реагируют прямо на команды `modprobe`. Команда `grep modprobe /etc/init.d/*` быстро дает обзор модулей, загружаемых таким образом.
- **Модули для интерфейсов.** Несколько других модулей загружается в тот момент, когда впервые используется отдельный интерфейс. Здесь возникает особая проблема — для работы с некоторыми интерфейсами конкретные аппаратные компоненты используют различные модули. Итак, если вы запрашиваете интерфейс `eth0` для первой сетевой карты в компьютере, нужно загрузить в ядро модуль, подходящий для этой карты.

  Вы должны сообщить ядру, какой модуль нужно использовать. Такая информация находится в `/etc/modprobe.conf`, а также в файлах каталога `/etc/modprobe.d`. Там же находятся установочные и характерные для дистрибутивов параметры и команды, указывающие, какие модули *не должны* загружаться автоматически (файл `blacklist`).

  Система udev, автоматически управляющая устройствами, также при необходимости загружает нужные модули. Соответствующие правила содержатся в файле `/etc/udev/rules.d`.
- **Модули для работы с устройствами USB и FireWire.** Такое оборудование играет особую роль. Несколько файлов `*.*map` в каталоге `/lib/modules/kernelversion/` на основании идентификационных кодов компонентов оборудования определяют, какой модуль требуется загрузить.
- **Взаимосвязи между модулями.** Многие модули зависят друг от друга. Например, модуль `nfs` для файловой системы NFS функционирует лишь в том случае, если в систему также загружены модули `lockd`, `nfs_acl` и `sunrpc`. Все такие зависимости между модулями перечислены в файле `/lib/modules/n/modules.dep`.

Загрузка модулей при запуске компьютера

Иногда, независимо от описанных здесь способов конфигурации, требуется, чтобы при старте компьютера загружался определенный модуль ядра — причем не автоматически, а по вашей инициативе. Оптимальный метод в таком случае выбирается в зависимости от вашего дистрибутива.

В Debian и Ubuntu все просто. В этих дистрибутивах специальный сценарий Init-V `/etc/init.d/module-init-tools` отвечает за то, чтобы загрузить все модули, перечисленные в `/etc/modules`, строка за строкой. Иначе говоря, вам всего лишь нужно добавить новый модуль как новую строку в `/etc/modules`.

В большинстве других дистрибутивов необходимо добавить команду `modprobe` *`имя_модуля`* в сценарий Init-V, предназначенный для локальных настроек. Однако не забывайте, что в определенных дистрибутивах модули в таком случае будут загружаться лишь в финале процесса Init-V (при выполнении некоторых задач это достаточно поздно):

- Red Hat, Fedora — `/etc/rc.d/rc.local`;
- SUSE — `/etc/init.d/boot.local`.

Модули, не соответствующие лицензии GPL (Ubuntu)

В большинстве дистрибутивов при поставке содержатся только такие модули ядра, исходный код которых соответствует требованиям лицензии GPL. Ubuntu в этом отношении является исключением, так как содержит некоторые модули с различными драйверами оборудования, в которых используется двоичный код производителей оборудования (например, ATI, AVM и NVIDIA).

В Ubuntu 9.04 эти модули входят в состав пакета `linux-restricted-modules-arch`. Такие модули устанавливаются в каталоге `/lib/linux-restricted-modules`. Сценарий Init-V под названием `linuxrestricted-modules-common` при запуске компьютера вызывает сценарий `lrm-manager`, подключающий к дереву каталогов в точке `/lib/modules/kernelversion/generic` временную файловую систему, копирующий туда все модули, не соответствующие лицензии GPL и, наконец, выполняющий для этого каталога команду `depmod` (см. в следующем разделе). Эти этапы гарантируют, что в дальнейшем можно будет без проблем использовать модули. Если вы *не хотите* использовать отдельные модули из `restricted-modules`, укажите их названия в переменной `DISABLED_MODULES` в файле `/etc/default/linux-restricted-modules-common`.

В Ubuntu 9.10 и выше модули, не являющиеся свободным ПО, будут передаваться в форме пакетов DKMS. Это механизм, обеспечивающий автоматическое обновление модулей после обновления ядра.

Синтаксис modprobe

В нескольких следующих абзацах описаны самые важные ключевые слова для `modprobe.conf` и файлы, находящиеся в `modprobe.d/`. Этот вопрос детально рассмотрен в справке `man modprobe.conf`.

alias. Эти команды указывают, какие модули ядра с какими устройствами используются. Пример: с устройством `/dev/eth0` должен использоваться модуль 8139too.

```
alias eth0 8139too
```

Доступ ко многим аппаратным компонентам осуществляется через блочные и символьные файлы-устройства (`/dev/`*`xxx`*). С точки зрения ядра эти файлы-устройства называются не именами, а старшим и младшим номером устройства (также см. раздел 3.10, где описана команда `mknod`). Многочисленные команды `alias` обеспечивают взаимосвязь между номерами устройств и модулями. Схожим образом определяются и сетевые протоколы: для использования определенного протокола ядро ищет семейство протоколов под названием net-pf-n. В следующем примере для семейства протоколов 5 загружается модуль AppleTalk.

```
alias net-pf-5 appletalk
```

Если вам не нужен этот протокол и тем более если соответствующий модуль не установлен, то следующая команда избавит вас от назойливых сообщений об ошибках:

```
alias net-pf-5 off
```

options. Команды `options` указывают, с какими параметрами должен загружаться тот или иной модуль. Благодаря следующей команде модуль ne (совместимые с NE-2000 Ethernet-карты) загружается с параметром `io=0x300`.

```
options ne io=0x300
```

include. Команды `include` загружают указанные конфигурационные файлы.

install. С помощью `install` даются команды, выполняемые дополнительно к простой загрузке модуля. Здесь я также приведу пример, который ради экономии места разделен на две строки. Если вам потребуется модуль ALSA `snd`, нужно выполнить следующие команды:

```
install snd modprobe --ignore-install snd $CMDLINE_OPTS && \
  { modprobe -Qb snd-ioctl32 ; : ; }
```

remove. С помощью `remove` указываются команды, которые должны выполняться при удалении модуля.

blacklist. При использовании команды `blacklist` внутренние alias-определения модулей не учитываются. Как правило, команды `blacklist` находятся в файле `/etc/modprobe.d/blacklist`. Здесь расположены те модули, которые *не должны* загружаться — из-за проблем с совместимостью или потому, что для них есть лучшая альтернатива. Например, следующая строка не дает запуститься модулю usbmouse. Вместо него обычно используется более мощный модуль hid.

```
blacklist usbmouse
```

Компилирование дополнительного модуля

Если вы применяете Linux вместе с VirtualBox, желаете использовать двоичные графические драйверы ATI или NVIDIA или вам нужен иной специфичный аппаратный модуль, которого нет в ядре вашего дистрибутива, вам потребуется скомпилировать модуль, подходящий к применяемой версии ядра.

Инструменты разработки

Для компилирования модулей требуются не только компиляторы C `gcc` и `make`, но и другие основные инструменты разработки. В большинстве дистрибутивов задача упрощается, так как в них имеются готовые выборки пакетов или метапакеты, указывающие на все важные пакеты:

- Debian, Ubuntu — `apt-get install build-essential`;
- Fedora — `yum groupinstall development-tools`;
- SUSE — `zypper install -t pattern devel_basis`.

Включаемые файлы ядра

Кроме того, вам потребуются включаемые (заголовочные) файлы для действующего ядра. Эти файлы входят в состав кода ядра. Во многих дистрибутивах (но не в SUSE) включаемые файлы находятся в одном пакете, а оставшийся код — в другом. Преимущество такой организации заключается в том, что вам требуется загружать не весь код ядра (он достаточно велик), а лишь сравнительно небольшие

включаемые файлы. В следующем списке указано, в каких пакетах располагаются включаемые файлы в наиболее распространенных дистрибутивах и куда эти файлы устанавливаются. Здесь *n.n* — это подстановочный символ для установленной версии ядра, *платформа* выполняет ту же функцию для действующего варианта процессора (например, amd64). Вся эта информация выводится командой `uname -a`.

- Debian — `linux-headers-`*n.n-платформа* (`/usr/include/linux`);
- Fedora, Red Hat — `kernel-[PAE-]devel-`*n.n* (`/lib/modules/`*n.n*`/build/include`);
- SUSE — `kernel-source` (`/usr/src/linux-`*n.n*`/include`);
- Ubuntu — `linux-headers-generic` (`/usr/include/linux`).

Если вы сами компилируете ядро (об этом рассказывается в следующем разделе), то включаемые файлы, подходящие к ядру, автоматически оказываются в каталоге `/lib/modules/`*n.n*`/build/include`.

PAE

При работе с современными версиями Fedora будьте внимательны: в этом дистрибутиве есть два варианта ядра — с поддержкой PAE и без нее. Чтобы узнать, какая версия применяется у вас, выполните команду `uname -r`. Если в полученной последовательности символов содержится запись `pae`, то ядро поддерживает PAE. В таком случае вам потребуется установить не пакет `kernel-devel`, а `kernel-PAE-devel`! Лишь тогда вы сможете скомпилировать модуль, совместимый с вашим ядром.

Аббревиатура PAE означает «расширение физических адресов» (Physical Address Extension). Это механизм, позволяющий использовать на 32-битных процессорах более 4 Гбайт оперативной памяти. Однако при активизации PAE кроме количества бит в процессоре и объема доступной оперативной памяти вы получаете возможность пользоваться защитной системой *No Execute* (NX). Она позволяет запретить выполнение кода из области данных определенной программы при переполнении буфера.

Компилирование модуля

Большинство программ, для работы которых требуются специальные модули ядра, содержат установочный сценарий, который отвечает за компилирование и создание модуля. Это касается, например, VMware, VirtualBox, графических драйверов ATI/AMD и NVIDIA и т. д. В некоторых дистрибутивах процесс автоматизирован в еще большей степени — модуль автоматически компилируется после каждого обновления ядра (см. пункт DKMS далее).

Если, напротив, вы скачали исходный код для аппаратных компонентов, которые пока официально не поддерживаются, то вам потребуется произвести компилирование самостоятельно. Для этого, как правило, нужно выполнить следующие команды. Только для запуска последней команды `make` требуются права администратора.

```
user$ cd source code directory
user$ make clean
user$ make
root# make install
```

Команда module-assistant

В Debian и Ubuntu есть команда module-assistant, или m-a, которая помогает составить список заданных, часто используемых модулей ядра. После установки пакета module-assistant команда m-a prepare инсталлирует все необходимые инструменты разработки; m-a update обновляет источники для module-assistant; m-a list возвращает список модулей, которые можно скомпилировать с помощью m-a.

```
root# apt-get install module-assistant
root# m-a prepare
root# m-a update
```

Когда эти подготовительные работы будут завершены, самой важной станет команда m-a auto-install, или коротко m-a a-i: она устанавливает пакет Debian в форме *имя*-source, компилирует для актуальной версии ядра модуль, содержащийся в исходном коде, и устанавливает этот модуль. Если команда «жалуется», что указанный модуль не находит пакета с исходным кодом, то, как правило, нужно дополнить файл /etc/apt/sources.list. В Debian необходимо убедиться, что в дистрибутиве имеются пакеты contrib и non-free; в Ubuntu обычно нужны пакеты restricted, universe и multiverse.

```
root# m-a auto-install nvidia
```

После обновления ядра команду m-a a-i нужно выполнить еще раз, чтобы скомпилировать модуль и для новой версии ядра!

Многие модули, доступные через пакет module-assistant, предоставляются в Ubuntu по умолчанию, поэтому совсем не обязательно (а при отсутствии пакетов с исходным кодом даже невозможно) создавать соответствующие модули с помощью m-a. Таим образом, в Debian m-a приходится применять гораздо чаще.

Команда m-a может запускаться и без параметров — тогда она выводит вас в интерфейс, состоящий из текстовых диалогов, где вы можете выбрать или скомпилировать нужный модуль.

DKMS

Это динамическая поддержка модулей ядра (Dynamic Kernel Module Support). Такая функция помогает автоматически обновить самостоятельно скомпилированные модули после обновления ядра. DKMS состоит из нескольких shell-сценариев и разработана фирмой Dell. Официальные пакеты dkms предоставляются лишь в немногих дистрибутивах, в том числе в Fedora и в Ubuntu. Если в Ubuntu устанавливается драйвер NVIDIA, то при обновлении ядра он также автоматически обновляется сценарием Init-V /etc/init.d/dkms_autoinstaller. Кроме того, DKMS обновляет модули, созданные командой module-assistant.

Чтобы можно было пользоваться DKMS, исходный код модуля должен быть установлен в каталоге в форме /usr/src/name-version. В каталоге должен находиться файл dkms.conf, разъясняющий DKMS, как нужно поступить с кодом. Следующие строки взяты из кода для драйвера NVIDIA в Ubuntu 9.04, причем форматирование листинга немного изменено, чтобы код было проще читать.

```
# Файл /usr/src/nvidia-180.44/dkms.conf
PACKAGE_NAME              = "nvidia"
PACKAGE_VERSION           = "180.44"
```

```
CLEAN                    = "make clean"
BUILT_MODULE_NAME[0]     = "nvidia"
MAKE[0]                  = "make module KERNDIR=/lib/modules/$kernelver \
                            IGNORE_XEN_PRESENCE=1 IGNORE_CC_MISMATCH=1 \
                            SYSSRC=$kernel_source_dir"
DEST_MODULE_LOCATION[0]  = "/kernel/drivers/video/nvidia"
PATCH[0]                 = "nvidia-rt-compat.patch"
PATCH_MATCH[0]           = "^2.6.2[8]"
AUTOINSTALL              = "yes"
```

Если эти условия выполнены, передайте модуль ядра под контроль DKMS с помощью команды `dkms add`, скомпилируйте его и установите для актуального ядра. В дальнейшем этот процесс будет автоматически выполняться при обновлении ядра. Следующие примеры вновь относятся к драйверу ядра NVIDIA (эти команды автоматически выполняются при установке пакета драйвера в Ubuntu):

```
root# dkms add -m nvidia -v 180.44
root# dkms build -m nvidia -v 180.44
root# dkms install -m nvidia -v 180.44
```

Выполнив `dkms status` или посмотрев в каталог `/var/lib/dkms`, вы увидите, какие модули в данный момент находятся под контролем DKMS. Более подробно о DKMS рассказано на следующих сайтах:

- http://www.linuxjournal.com/article/6896;
- http://wiki.centos.org/HowTos/BuildingKernelModules.

15.2. Самостоятельное конфигурирование и компилирование ядра

Среднестатистическому пользователю Linux не приходится компилировать ядро для своей системы. Вместе со всеми распространенными дистрибутивами предоставляется полноценное стандартное ядро с широким набором модулей. И все же по некоторым причинам вам может потребоваться заново скомпилировать ядро:

- вы хотите лучше познакомиться с собственной системой (ведь вся эта книга написана для того, чтобы вы могли заглянуть внутрь Linux);
- вам нужны особые функции, которых нет ни в ядре, ни в предоставленных вместе с ним модулях;
- вы хотите работать с более новой версией ядра, чем та, которая была в вашем дистрибутиве при поставке;
- вы хотите принять участие в создании ядра, и для этого вам нужно поэкспериментировать с новейшим ядром от разработчиков;
- вы хотите козырнуть в кругу друзей: «Я сам скомпилировал новейшую версию ядра Линукс!»

Однако есть веские причины, по которым не следует компилировать собственное ядро.

- В большинстве дистрибутивов используется не оригинальное ядро, предоставленное во всеобщее пользование Линусом Торвальдсом (Linus Torvalds),

а обновленная версия с многочисленными дополнительными функциями (причем, разумеется, в каждом дистрибутиве используются собственные заплатки — также см. подраздел «Как обновить код ядра» этого раздела). Для пользователя такая ситуация очень удобна: он получает в распоряжение функции, за стабильную работу которых ручается производитель дистрибутива. В исходном коде оригинального ядра, который вы можете скачать сами, этих заплаток не будет. Отдельные функции вашего дистрибутива, которые ранее работали без проблем, вдруг начинают работать с перебоями или вообще отключаются.

- Скомпилировать собственное ядро не сложно. Гораздо сложнее предварительно задать конфигурацию процесса компилирования. Вам на выбор будет предоставлено более 1000 параметров. С их помощью вы можете указать, какие функции будут интегрированы прямо в ядро, какие — предоставлены в виде модулей, а какие будут отсутствовать. Если вы, недостаточно хорошо разбираясь в работе ядра, неправильно выберете параметры, то результат будет тот же, что и в предыдущем случае: некоторые функции перестанут работать и найти причину этого будет достаточно сложно. Для начинающего пользователя Linux практически невозможно угадать для всех параметров верные настройки.

По этим причинам в большинстве дистрибутивов не предоставляется никакой поддержки, если вы работаете с каким-либо ядром, кроме того, что было в системе при поставке. Однако надеюсь, что эти предупреждения не слишком вас испугают и вы все же попробуете свои силы. Если в точности выполнять все рекомендации, приведенные в подразделе «Компилирование и установка ядра» этого раздела, то можно запускать компьютер как со старым, так и с новым ядром, то есть ничего страшного не произойдет.

Для компилирования ядра применяются те же инструменты разработки, что и для компилирования отдельного модуля (см. подраздел «Компилирование дополнительного модуля» раздела 15.1).

Основы

Версии ядра. До версии ядра 2.6.0 существовали стабильные версии (2.0.n, 2.2.n, 2.4.n) и так называемые хакерские (2.3.n, 2.5.n и т. д.). В большинстве дистрибутивов Linux применялись стабильные версии, а хакерское ядро предназначалось для программистов, которые участвовали в его разработке. Новые функции сначала тестировались в хакерском ядре, и только намного позже (иногда через несколько лет) попадали в следующую стабильную версию ядра.

Начиная с ядра 2.6, модель разработки изменилась. «Хакерского» ядра 2.7.n нет. Вместо этого продолжается разработка версий 2.6.n. Можно сказать, что сначала каждая новая версия ядра считается хакерской; после того как Линус Торвальдс решит, что версия работает надежно, она предоставляется во всеобщее пользование как стабильная. Основное преимущество такого метода заключается в том, что нововведения тестируются гораздо более широким сообществом разработчиков и гораздо быстрее становятся всеобщим достоянием.

Если в новейшей версии ядра, предоставленной во всеобщее пользование, обнаружатся ошибки или в системе безопасности появятся слабые места, они устра-

няются в дополнительной версии, в обозначении которой используется четвертый номер. Так получается, например, версия 2.6.21.4.

uname. Команда `uname` сообщает, какая версия ядра используется в настоящее время:

```
root# uname -r
2.6.28-13-generic
```

ksplice. Как правило, Linux можно обновлять, не останавливая работу системы. Правда, обновленные сетевые службы потребуется перезапустить, но перезапускать весь компьютер не нужно. Ядро является исключением из этого правила: чтобы обновления для системы безопасности вступили в силу, должно быть установлено новое ядро, а также новые модули, и после этого компьютер необходимо перезапустить. Для персональных компьютеров, которые обычно включаются и выключаются каждый день, в этом нет никакой проблемы. Но для серверов, которые, по возможности, должны быть доступны без перебоев, каждый перезапуск нежелателен.

В данном случае полезна функция ksplice, разработанная одноименной фирмой. При проведении большинства (почти всех) обновлений она позволяет отключить конкретную функцию и заменить ее новым кодом. Техническая сторона этой функции далеко не тривиальна и рассмотрена на следующих сайтах:

- http://www.ksplice.com/;
- http://lwn.net/Articles/340477/.

Еще неизвестно, удастся ли ksplice закрепиться на практике. Метод основан на открытом коде, что уже довольно неплохо. В настоящее время фирма Ksplice предоставляет бесплатные обновления ksplice для Ubuntu, вероятно, чтобы пользователь лучше познакомился с новинкой.

Немного статистики. В настоящее время ядро (версия 2.6.31) состоит примерно из 12 миллионов строк кода (в основном на C, а также на Ассемблере). Если вам интересно, кто (какие фирмы) участвует в разработке ядра, внимательно следите за сайтом lwn.net (Linux Weekly News). Там по каждой версии ядра приводятся статистические данные относительно того, кто внес больше всего изменений. Конкретно для версии 2.6.31 написана статья http://lwn.net/Articles/348445/.

Ссылки. Советы по компилированию ядра есть на следующих сайтах:

- http://kernelnewbies.org/faq/;
- http://www.tux.org/lkml/.

Если вас интересует техническая сторона вопроса, то просмотрите файлы документации по ядру, которые очень подробны и информативны. Самые новые функции ядра сначала описываются там, а только потом попадают на обновленные страницы справки `man`. Если вы сами установили код ядра, нужная вам документация находится здесь:

- `/usr/src/`*`версия_ядра`*`/README`;
- `/usr/src/`*`версия_ядра`*`/Documentation/*`.

Такие же файлы по наиболее актуальной версии ядра есть и в Интернете: http://www.kernel.org/doc/Documentation/.

Установка кода ядра

Обычно исходный код ядра располагается в каталоге /usr/src/linux (только в Red Hat и Fedora закрепилась иная традиция). Если каталог пуст, это означает, что вы не установили код ядра. Теперь вы можете либо загрузить туда код ядра вашего дистрибутива, либо скачать официальный код новейшей версии ядра. Как правило, лучше избрать первый вариант — особенно это касается новичков.

Обратите внимание, что для ядра нужно достаточно много места. Даже заархивированные пакеты с исходным кодом имеют размер около 60 Мбайт. После распаковки на код требуется еще на 400 Мбайт больше, а после компилирования (в результате которого получаются двоичные файлы) — более 4 Гбайт!

Установка кода ядра конкретного дистрибутива

В большинстве дистрибутивов имеется специальный пакет, в котором содержится код ядра. В следующем списке указано, в каких пакетах наиболее распространенных дистрибутивов находится код ядра (здесь *n.n* — это подстановочный символ для установленной версии ядра):

- Debian, Ubuntu — linux-source-*n.n*;
- Red Hat, Fedora — kernel-*n.n* (пакет с исходным кодом);
- SUSE — kernel-source.

В Debian и Ubuntu код ядра установлен в виде архива TAR в каталоге /usr/src. Архив нужно распаковать самостоятельно, с помощью команды tar xjf linux-*n.n*.tar.bz2.

Fedora

В Fedora и Red Hat есть некоторые особенности: во-первых, код ядра находится не в обычном пакете, а в «пакете с исходным кодом». Во-вторых, при работе с Fedora рекомендуется устанавливать исходный код не в /usr/src, а в подкаталог rpmbuild домашнего каталога. Это позволяет компилировать ядро, не имея прав администратора.

В целом, процедура несколько усложняется: сначала устанавливаются пакеты yumutils (содержит программу yumdownloader) и rpmdevtools (содержит rpmdev-setuptree, а также различные команды для создания пакетов RPM). Программа rpmdev-setuptree создает каталог ~/rpmbuild, а в нем, в свою очередь, различные подкаталоги. Программа yumdownloader скачивает пакет с исходным текстом программы kerneln.n.src.rpm.

```
user$ su -c 'yum install yumutils rpmdevtools'
user$ rpmdev-setuptree
user$ yumdownloader --source kernel
```

Программа yum-builddep устанавливает все недостающие пакеты, требуемые для компилирования ядра; rpm -i распаковывает пакет с ядром. Архив с исходным кодом ядра (файл linux-*n.n*.tar.bz2), а также все заплатки, характерные для Fedora, оказываются в каталоге ~/rpmbuild/SOURCES. Сообщения об ошибках вида Пользова-

тель mockbuild не существует — используется Root при этом можно игнорировать. Команда rpmbuild извлекает отсюда исходный код и применяет все заплатки, характерные для Red Hat и Fedora:

```
user$ su -c 'yum-builddep kernel-n.n.src.rpm'
user$ rpm -i kernel-n.n.src.rpm
user$ cd ~/rpmbuild/SPECS
user$ rpmbuild -bp --target=$(uname -m) kernel.spec
```

После этого вы можете найти оригинальный исходный код и исходный код, обновленный под Fedora, в следующих каталогах:

- ~/rpmbuild/BUILD/kernel-*n.n*/vanilla-*n.n* — оригинальный исходный код;
- ~/rpmbuild/BUILD/kernel-*n.n*/linux-*n.n* — исходный код с заплатками для Fedora.

Для экономии места идентичные файлы связываются жесткими ссылками, то есть физически сохраняются лишь один раз. Другие советы по компилированию собственного ядра в Fedora даются на сайте http://fedoraproject.org/wiki/Docs/CustomKernel. Там, в частности, описано, как поступить, чтобы новое ядро сразу после завершения компилирования оказалось в пакете RPM.

Установка официального кода ядра

Часто ядро, поставляемое вместе с дистрибутивом, уже является устаревшим. Актуальный код ядра, упакованный в архиве TAR, можно найти, например, здесь:

- http://www.kernel.org/;
- ftp://ftp.kernel.org/pub/linux/kernel.

Обычно файл-архив с кодом ядра имеет название вида linux-2.6.30.2.tar.bz2 (размер около 60 Мбайт). Для установки перейдите в каталог /usr/src и выполните следующую команду:

```
root# cd /usr/src
root# tar xjf linux-2.6.30.2.tar.bz2
```

Файл устанавливается в каталог /usr/src/. Чтобы упростить доступ к этому каталогу, обычно используется ссылка /usr/src/linux, указывающая на актуальный каталог с исходным кодом:

```
root# ln -s linux-2.6.30.2 linux
```

Как обновить код ядра

Так называемые *заплатки* — это файлы, позволяющие произвести обновление от одной версии до другой. Это заархивированные текстовые файлы, в которых указано, в какие файлы нужно внести какие изменения. Заплатки избавляют вас от загрузки огромных объемов кода, в особенности если разница между исходной и конечной версиями невелика. В любом случае, чтобы заплатки функционировали, их необходимо использовать именно с тем (неизмененным!) кодом, для которого они были написаны.

Правильная последовательность заплаток

Предположим, нам нужно обновить код 2.6.30.5 до 2.6.30.6. Логичнее всего было бы просто применить заплатку 2.6.30.6. Однако это не сработает, так как она предназначена для преобразования кода 2.6.30 (а не 2.6.30.5!). Поэтому вам потребуется скачать заплатку 2.6.30.5 и применить ее обратным образом (параметр `-R`), чтобы вернуть 2.6.30.5 назад к версии 2.6.30. И только теперь заплатка 2.6.30.6 сработает!

Как правило, команда `patch` используется в сочетании с `bunzip2`. Команда `bunzip2` разархивирует заплатку, а `patch` применяет изменения. Если файл заплатки доступен вне архива, то команда будет иметь вид `patch -p1 < файл_заплатки`.

Перед применением любую заплатку следует проверять параметром `--dry-run`, не возникнет ли при этом проблем. Ничто так не портит настроение, как заплатка, содержащая ошибку или сработавшая только наполовину!

Заплатки изменяют только код, но не название каталога, в котором находится этот код. Чтобы избежать путаницы, дополнительно переименуйте каталог с кодом (номер действующей версии можно узнать в файле `Makefile`, который располагается прямо в каталоге с исходным кодом).

```
root# cd /usr/src/linux-2.6-30.5
root# bunzip2 -c patch-2.6.30.5.bz2 | patch -R -p1 --dry-run  (Тестирование обратной
заплатки)
... сообщений об ошибках нет
root# bunzip2 -c patch-2.6.30.5.bz2 | patch -R -p1 (2.6.30.5 --> 2.6.30)
root# bunzip2 -c patch-2.6.30.6.bz2 | patch -p1 --dry-run  (Тестирование заплатки)
... сообщений об ошибках нет
root# bunzip2 -c patch-2.6.30.6.bz2 | patch -p1  (2.6.30 --> 2.6.30.6)
root# cd /usr/src
root# mv linux-2.6-30.5 linux-2.6-30.6
```

Заплатки с функциями

Наряду с рассмотренными выше заплатками обновлений также существуют заплатки с неофициальными дополнительными функциями, которые по различным причинам еще не интегрированы в стандартное ядро (заплатки функций).

В принципе заплатки функций также применяются к коду ядра командой `patch`. В любом случае применяя заплатку, вы должны работать с тем же базовым кодом, что и разработчик, предоставивший ее. Как правило, в качестве базового кода используется только официальный код актуальной версии ядра, а не код вашего дистрибутива, в котором уже есть собственные заплатки.

Применение конфигурационных файлов ядра, поставляемых вместе с дистрибутивом

Ядро состоит из тысяч отдельных функций, называемых *компонентами*. Перед началом компилирования вы можете указать практически для любой функции, следует ли интегрировать ее прямо в ядро, скомпилировать в виде модуля или вообще не использовать. Этот процесс называется *конфигурированием ядра*.

Файл .config

Конфигурация ядра определяется файлом с расширением `.config`, расположенным в каталоге `/usr/src/linux-n.n`. Это текстовый файл, состоящий примерно из 4000 строк, в котором указано, будет ли функция интегрирована прямо в ядро (*имя*=`y`) или скомпилирована в виде модуля (*имя*=`m`). Функции, которые были помечены как ненужные, в этом файле не отображаются либо отображаются как комментарии. Здесь также могут содержаться дополнительные настройки (*имя=значение*). Далее показан небольшой фрагмент из файла с расширением `.config`:

```
CONFIG_X86=y
# CONFIG_X86_32 не установлена
CONFIG_X86_64=y
CONFIG_X86_64_SMP=y
CONFIG_X86_ACPI_CPUFREQ=y
# CONFIG_X86_ACPI_CPUFREQ_PROC_INTF не установлена
CONFIG_X86_BIOS_REBOOT=y
```

Если при конфигурации ядра вручную (см. следующий раздел) у вас нет отправной точки, вы должны позаботиться обо всех параметрах ядра. Компилируя ядро впервые, вы непременно что-то упустите — можете даже не сомневаться. Вы сэкономите себе массу сил и времени, если воспользуетесь файлом конфигурации ядра, поставляемым вместе с дистрибутивом:

```
root# cp old-config /usr/src/linux-n.n/.config
```

Вы также можете перейти в каталог с исходным кодом и выполнить в нем следующую команду:

```
root# cd /usr/src/linux-n.n
root# make oldconfig
```

К сожалению, у этого метода есть один недостаток: если в исходном коде содержатся другие заплатки, не подходящие для компилируемого кода, то в исходном конфигурационном файле также будут параметры, не предусмотренные для использования в новом коде. Это чревато проблемами (как я уже указывал, многие производители встраивают в свои дистрибутивы различные заплатки, которые отсутствуют в стандартном ядре).

Определение актуальной конфигурации

Еще остается открытым вопрос, откуда взят актуальный конфигурационный файл. Почти во всех дистрибутивах конфигурационный файл, подходящий к действующему ядру, находится в каталоге `/boot` (например, `/boot/config-n.n`).

В Red Hat и Fedora используются и другие виды конфигурации — для работы с разновидностями ядра, связанными с Xen, SMP и др. Пакет с исходным кодом ядра устанавливается в следующий каталог: `rpmbuild/BUILD/kernel-n.n/linux-n.n/configs/`.

Команда cloneconfig

В ядре, которое при поставке имеется в дистрибутиве SUSE, используется параметр `cloneconfig` (группа Основные настройки). Это означает, что в `/proc/config.gz`

в заархивированном виде содержится информация из файла `.config`, с помощью которого было скомпилировано ядро, работающее в настоящее время. Команда `make cloneconfig` позволяет скопировать в файл `.config` последнюю использованную конфигурацию.

Конфигурирование ядра вручную

Монолитное ядро или ядро, состоящее из модулей. Вы можете использовать ядро одного из двух типов: *монолитное* или *состоящее из модулей*. В монолитном ядре содержатся все необходимые драйверы, и такое ядро не поддерживает модулей. Ядро, имеющее модульную организацию, не только работает со встроенными драйверами, но и способно в ходе работы поддерживать новые модули. Почти во всех случаях ядро, состоящее из модулей, — это более выигрышный вариант, чем монолитное ядро.

Выбор компонентов. При работе с большинством компонентов вы можете выбрать из трех вариантов параметров: `Yes/Module/No`. Параметр `Yes` означает, что эти компоненты интегрируются прямо в ядро; `Module` говорит, что данные компоненты компилируются в виде модуля (параметр действует только в ядре с модульной организацией); `No` означает, что компонент вообще не компилируется. Существует также несколько функций, которые нельзя предоставить в виде модуля, — при работе с ними выбор сужается до `Yes` и `No`.

Конфигурационные стратегии. Чаще всего в ядро интегрируется сравнительно небольшое количество элементарных функций, а остальные необходимые функции предоставляются в виде модулей. Преимущество этого метода заключается в том, что ядро имеет относительно небольшой размер, а модули догружаются по мере необходимости.

Иная стратегия состоит в том, чтобы максимально оптимизировать монолитное ядро под необходимые вам аппаратные и программные функции. Все функции, которые должны использоваться, интегрируются прямо в ядро. Для остальных компонентов выбирайте вариант `No`.

Вообще, монолитное ядро всегда немного больше ядра с модульной организацией. Благодаря этому монолитное ядро работает без динамического управления модулями, а компьютер запускается без файла `initrd`. Однако и основной его недостаток очевиден: если позже вам все же понадобится новая функция, неучтенная при компилировании, то ядро придется компилировать заново. Только настоящие профессионалы Linux могут точно оценить, какие функции понадобятся им при работе.

Инструменты, используемые при конфигурировании ядра вручную

Для того чтобы изменить часть настроек, действующих в актуальной конфигурации ядра, можно вручную отредактировать файл `.config`. Но при этом легко до-

пустить ошибку и обязательно нужно хорошо знать названия различных параметров. Гораздо лучше запустить специальную конфигурационную программу `make xxxconfig`. В версии ядра 2.6 эта программа существует в четырех различных вариантах и запускается с помощью следующих команд `make`:

```
root# cd /usr/src/linux-n.n
root# make config      (Конфигурация в текстовом режиме)
root# make menuconfig  (Конфигурация в текстовом режиме с применением окон)
root# make xconfig     (Конфигурация в графическом режиме с применением библиотеки QT)
root# make gconfig     (Конфигурация в графическом режиме с применением библиотеки GTK)
```

make config. Команда `make config` работает в любом случае, но она сложна в использовании и поэтому не рекомендуется. Всякий раз когда вам понадобится изменить один параметр, все остальные придется просмотреть.

make menuconfig. Для работы с `make menuconfig` сначала нужно установить пакет `ncurses-devel` или `libncurses-dev`. Конфигурация, как и в прошлом примере, выполняется в текстовом режиме. Основное преимущество этой команды по сравнению с `make config` заключается в том, что настройка многочисленных параметров структурирована в виде системы вложенных друг в друга диалоговых окон.

make xconfig. Гораздо удобнее использовать `make xconfig`: этот вариант предназначен для работы с X и требует, чтобы были установлены пакеты `g++` (компилятор C++) и `qt3-devel` или `libqt3-mt-dev` с файлами разработки из библиотеки QT. Сначала `make` компилирует графический пользовательский интерфейс `qconf`, а затем запускает его (рис. 15.1).

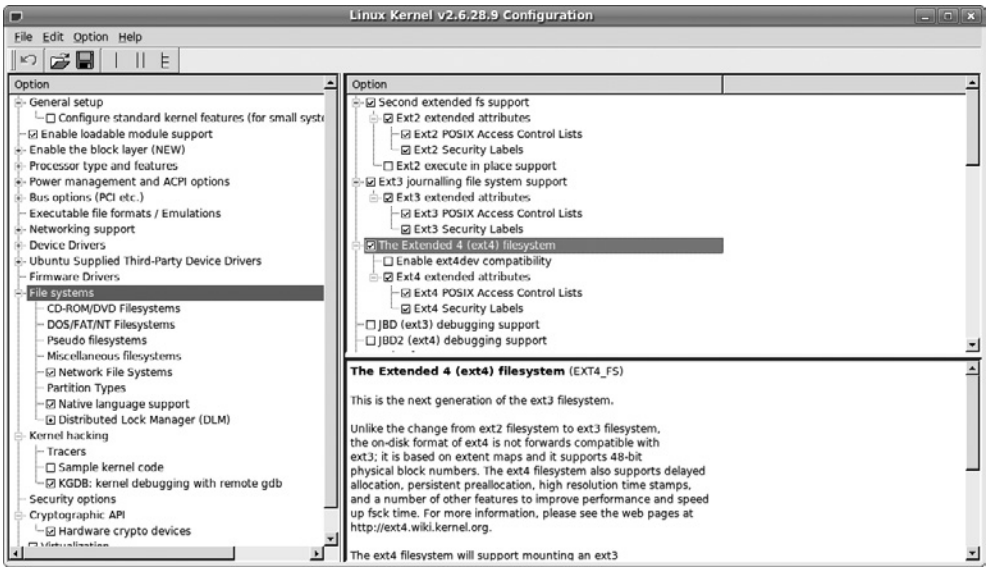

Рис. 15.1. Конфигурирование ядра с помощью программы make xconfig

Три возможных состояния компонента (Yes/Module/No) выражаются в этом интерфейсе так:

- No — поле рядом с названием параметром оставлено пустым;
- Yes — в поле рядом с названием параметр установлен флажок;
- Module — в поле рядом с названием параметра стоит точка.

Щелчком кнопкой мыши можно переходить от одного состояния к другому. Если вы не найдете какой-либо параметр, выполните Параметр ▸ Показать все параметры. В таком случае программа отобразит и те параметры, которые обычно не используются.

make gconfig. Команда make gconfig запускает gconf, программу из Gnome, функционально аналогичную qconf. В таком случае требуется установить различные библиотеки разработки для Gnome (в том числе [lib]gtk2-devel и libglade2-devel). Внешний вид и работа с gconf практически не отличается от qconf.

Многочисленные параметры можно структурировать одним из трех способов. Наиболее неудобным оказался очень наглядный режим SPLIT: в нем недоступны некоторые вложенные параметры.

Компилирование и установка ядра

После того как вы потратите некоторое время на конфигурацию ядра, вам нужно начать работать с компьютером. На выполнение следующих команд быстрый компьютер тратит около получаса. Если на вашем компьютере стоит несколько процессоров или используется многоядерный процессор, то процесс компилирования можно ускорить с помощью команды make -j *n* all. В таком случае make запустит *n* процессов параллельно, задействовав все процессоры или ядра.

```
root# cd /usr/src/linux-n.n
root# make all              (Скомпилировать все)
root# make modules_install  (Установить модули)
```

В результате этого процесса получается файл bzImage в каталоге /usr/src/linux-*n.n*/arch/x86/boot. Размер этого файла обычно составляет от 2 до 4 Мбайт и зависит от того, какие функции были включены прямо в ядро, а какие предоставлены в виде модулей или вообще не компилировались.

Команда make modules_install копирует файлы модулей туда, где их ожидают найти команды для управления модулями (например, insmod): в каталог /lib/modules/*n*; здесь *n* — точный номер версии только что скомпилированного ядра.

СОВЕТ

Если при компилировании возникнет ошибка, попытайтесь найти ее причину. Если проблема возникла с функцией, которая для вас не очень важна, измените конфигурацию так, чтобы эта функция вообще не компилировалась.

Опытные пользователи Linux могут поступить иначе — вызвать make с дополнительным параметром -k (то есть, например, make -k all). Благодаря этому параметру ошибки игнорируются. Команда make просто переходит к компилированию следующего файла. Если вам повезет, проблема с компилированием коснется не самого важного модуля и такой модуль просто окажется недоступен.

Установка ядра

Ядро, которое мы только что создали, еще не активно. Пока мы всего лишь создали несколько новых файлов. Ядро можно активизировать лишь после перезапуска Linux. Загрузчик GRUB нужно сконфигурировать так, чтобы он различал новое ядро.

Для этого сначала необходимо скопировать новый файл ядра в каталог `/boot`. Обычно такому файлу дается имя `vmlinuz-`*`n.n`*. Кроме того, на этом этапе нужно создать копию конфигурационного файла:

```
root# cp /usr/src/linux-n.n/arch/x86/boot/bzImage /boot/vmlinuz-n.n
root# cp /usr/src/linux-n.n/.config /boot/config-n.n
```

Подготовка системы к запуску

Как правило, теперь нужно создать новый файл `initrd`, подходящий к ядру. Для этого в различных дистрибутивах применяются команды `mkinitrd`, `mkinitramfs` или `update-initramfs` (см. раздел 14.5).

Если вы используете GRUB 2 и работаете с Debian или Ubuntu, то просто еще раз выполните `update-grub`. Эта команда автоматически добавляет в меню GRUB запись для нового ядра.

Если вы работаете с другими дистрибутивами на основе GRUB 0.97, то вам придется самим добавить в файл `/boot/grub/menu.lst` новую запись. При указании жесткого диска и параметров ориентируйтесь на записи, уже присутствующие в `menu.lst`. При перезапуске выберите в меню GRUB новое ядро:

```
# Дополнение в /boot/grub/menu.lst
title kernel-n.n
  kernel (hd0,11)/boot/vmlinuz-n.n root=/dev/sda12 vga=normal
  initrd (hd0,11)/boot/initrd-n.n
```

Если вы работаете с LILO (см. раздел 14.8), то соответствующим образом измените `/etc/lilo.conf`. Еще раз выполните команду `lilo`, чтобы изменения вступили в силу:

```
# Дополнение в /etc/lilo.conf
image = /boot/vmlinuz-n.n          # Файл ядра
        initrd = /boot/initrd-n.n  # Файл Initrd
        root = /dev/sda12          # Корневое устройство
        label = kernel-n.n         # Название в меню LILO
        append = "options ..."     # Параметр ядра
```

Все ли получилось, вы увидите после перезапуска. Если ядро по какой-то причине не будет работать, просто запустите компьютер со старым ядром, еще раз попытайтесь правильно сконфигурировать ядро и заново его скомпилировать. Если же новое ядро заработает нормально, вы можете удалить из компилятора более не нужные объектные файлы. Таким образом вы освободите на жестком диске около 4 Гбайт места!

```
root# cd /usr/src/linux-n.n
root# make clean
```

15.3. Каталоги /proc- и sys/

Каталоги /proc и /sys подключаются к файловой системе при запуске. Они используются для того, чтобы было легче получить информацию о ядре, текущих процессах, загруженных модулях и многих других параметрах.

Внутри системы каталоги /proc и /sys представлены как виртуальные файловые системы. Это означает, что в них нет никаких файлов, следовательно, они не занимают места на диске (это касается и файла /proc/kcore, отображающего оперативную память и с виду очень большого).

Большинство файлов в каталогах /proc и /sys — текстовые. Для того чтобы читать эти файлы, вам иногда может понадобиться команда cat, а не less, так как некоторые версии less не могут работать с виртуальными файлами.

В каталоге /proc содержится много информации о ядре, а также данные по всем процессам, выполняемым в данный момент (табл. 15.1). На каждый процесс выделен собственный подкаталог. Таким образом, в каталоге процессов находятся некоторые данные с разной административной информацией (например, может отличаться командная строка, используемая при запуске). Эти административные данные интерпретируются различными командами, применяемыми при управлении процессами (например, top, ps и т. д.).

Таблица 15.1. Важные файлы каталога /proc

Файл	Значение
/proc/n/*	Информация о процессе с идентификационным номером n
/proc/asound	ALSA (продвинутая звуковая архитектура Linux)
/proc/bus/usb/*	Информация о USB
/proc/bus/pccard/*	Информация о PCMCIA
/proc/bus/pci/*	Информация о PCI
/proc/cmdline	Параметры загрузки LILO/GRUB
/proc/config.gz	Конфигурационный файл (SUSE)
/proc/cpuinfo	Информация о процессоре
/proc/devices	Номера активных устройств
/proc/fb	Информация о кадровом буфере
/proc/filesystems	Драйверы файловых систем, содержащиеся в ядре
/proc/ide/*	Приводы и контроллеры IDE
/proc/interrupts	Использование прерываний
/proc/lvm/*	Использование диспетчера логических томов
/proc/mdstat	Состояние RAID
/proc/modules	Активные модули
/proc/mounts	Активные файловые системы
/proc/net/*	Состояние и использование сети
/proc/partitions	Разделы жестких дисков
/proc/pci	Информация о PCI (файл устарел, см. /proc/bus/pci)
/proc/scsi/*	Приводы и контроллеры SCSI

Файл	Значение
/proc/splash	Управление фоновым изображением VGA для текстовой консоли 1
/proc/sys/*	Информация о системе и ядре
/proc/uptime	Время в секундах, прошедшее с момента запуска компьютера
/proc/version	Версия ядра

Каталог /sys содержится только в версии ядра 2.6 и выше. Часть информации в нем такая же, как и в /proc, но данные организованы более системно. Каталог /sys нужен для того, чтобы отобразить взаимосвязь между ядром и оборудованием (табл. 15.2). Если вам интересно, как именно построена система /sys, почитайте следующий документ: http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf.

Таблица 15.2. Важные файлы /sys

Файл	Значение
/sys/block/*	Информация обо всех блоковых устройствах (жесткие диски и т. д.)
/sys/bus/*	Сведения обо всех шинных системах (IDE, USB и т. д.)
/sys/class/*	Информация о классах устройств (Bluetooth, графика, память и др.)
/sys/devices/*	Сведения о подключенных аппаратных компонентах (устройствах)
/sys/firmware/*	Данные о драйверах оборудования и встроенном программном обеспечении (особенно ACPI)
/sys/kernel/*	Информация о ядре
/sys/module/*	Сведения о загруженных модулях
/sys/power/*	Информация об управлении питанием

15.4. Важные параметры ядра

Если изменяется некий элемент ядра, это не обязательно означает, что ядро сразу же нужно компилировать заново! Есть два способа повлиять на ядро без повторного компилирования.

- Во-первых, при запуске системы загрузчик может передать параметры ядру. Этот механизм был описан в предыдущей главе, когда мы рассматривали запуск Linux (см. раздел 14.9).
- Во-вторых, некоторые функции ядра можно изменять динамически, то есть не останавливая работу системы. Такой вид вмешательства особенно часто применяется при управлении сетевыми функциями — о нем и пойдет речь в этом разделе.

Изменения вносятся через файловую систему /proc. В следующем примере показано, как активизировать функцию маскарадинга (чтобы использовать компьютер в качестве интернет-шлюза для других компьютеров):

```
root# echo 1 > /proc/sys/net/ipv4/ip_forward
```

sysctl. Более элегантно проблема решается с помощью команды `sysctl`, которая входит в состав большинства современных дистрибутивов. Аналогичная команда, предназначенная для того, чтобы вновь отключить функцию маскарадинга, будет выглядеть так:

```
root# sysctl -w net.ipv4.ip_forward=1
```

Команда `sysctl -a` возвращает список всех параметров ядра, а также текущие настройки этих параметров. Команда `sysctl -p` позволяет активизировать настройки `sysctl`, сохраненные в файле. В качестве имени файла обычно используется `/etc/sysctl.conf`. Синтаксис описан на справочном сайте по `sysctl.conf`. Во многих дистрибутивах (например, Debian, Fedora, Red Hat, SUSE, Ubuntu) этот файл автоматически интерпретируется и выполняется в ходе процесса Init-V.

16 Создание сетевого соединения

В этой главе описано, как создать на компьютере с Linux выход в Интернет или локальную сеть. В первой части главы рассмотрены различные пользовательские интерфейсы, с помощью которых, как правило, такие задачи решаются достаточно быстро. Если эти интерфейсы вам не помогут, можете воспользоваться конфигурацией сетевого клиента, которая будет детально описана в следующих разделах.

Тематически эта глава достаточно обширна: мы рассмотрим основы работы с сетями LAN и WLAN, научимся обращаться с ADSL-модемом и, наконец, сможем создавать VPN (виртуальные частные сети). Если вы хотите использовать компьютер с Linux в сети, но не в качестве клиента, а в качестве сервера, то обратитесь к главе 17.

16.1. Network Manager

В настоящее время Network Manager — самый популярный инструмент для конфигурирования LAN, WLAN, ADSL, UMTS и VPN. Он применяется почти во всех дистрибутивах. За основные функции программы отвечает фоновый процесс (демон), который запускается при старте компьютера сценарием `/etc/init.d/NetworkManager`.

В ходе конфигурации для Gnome и KDE применяется два различных апплета. В этом разделе описан вариант Gnome. В KDE записи меню и диалоговые окна выглядят немного иначе, но работать с обоими вариантами приходится практически одинаково. Апплет отображает актуальное состояние сети. Основное меню (открывается щелчком левой кнопкой мыши) отображает сетевые соединения, которые активны в настоящий момент, а также все доступные WLAN (рис. 16.1). Щелчок правой кнопкой мыши открывает другое меню, через записи которого вам предоставляются дополнительные конфигурационные возможности.

Рис. 16.1. Главное меню программы Network Manager

Подробная статусная информация обо всех соединениях, управляемых программой Network Manager, выводится командой `nm-tool`.

Проблемы

Несмотря на целостный подход и широкое распространение Network Manager, некоторые функции этой программы все еще несовершенны. Программа хорошо работает в стандартных ситуациях, но многочисленные сообщения об ошибках (bugs.launchpad.net, bugzilla.redhat.com) убедительно доказывают, что работать с экзотическим оборудованием или настраивать необычную конфигурацию Network Manager можно только с большими сложностями. Скудная документация по программе также не исправляет положения. Поэтому в некоторых дистрибутивах наряду с Network Manager применяются другие конфигурационные инструменты. Если у вас возникнут проблемы с Network Manager, почитайте следующий раздел. Там дается обзор альтернативных конфигурационных инструментов.

В принципе для работы Network Manager необходимо, чтобы программа имела контроль над интерфейсами. В этом отношении конфигурация в различных дистрибутивах отличается. В Debian и Ubuntu необходимо убедиться, что в `/etc/network/interfaces` содержатся настройки только для петлевого интерфейса — по умолчанию ситуация именно такова:

```
# Файл /etc/network/interfaces (Debian, Ubuntu)
auto lo
iface lo inet loopback
```

В Fedora и Red Hat с помощью меню Система ▸ Администрирование ▸ Сеть запускается программа `systemconfig-network`, а затем нужно проверить, на всех ли сетевых интерфейсах установлен флажок Проверять с помощью Network Manager (это стандартная настройка).

В SUSE модуль YaST Сетевые устройства ▸ Настройка сети определяет, как именно конфигурируется сеть — через Network Manager или традиционным способом (с помощью YaST и ifup. На всех компьютерах, использующих SUSE 11.2 и выше, стандартной настройкой является Network Manager.

Как деактивизировать Network Manager. Если вы конфигурируете компьютер как сервер или роутер, то Network Manager следует отключить и статически выполнить конфигурацию сети. Рекомендации о том, как это лучше сделать, даются в разделе 17.2.

LAN с DHCP (ADSL-роутер)

Простейшая ситуация, в которой применяется Network Manager, — это подключение компьютера сетевым кабелем к ADSL-роутеру, серверу LAN или другому компьютеру, на котором работает DHCP-сервер. DHCP — это протокол, по которому клиентам с локальных компьютеров присваивается конфигурация (см. также разделы 16.3 и 17.3). По умолчанию Network Manager проверяет все интерфейсы LAN на предмет того, можно ли через этот интерфейс передать конфигурационную информацию по протоколу DHCP. Если это возможно, то сетевая конфигурация

проходит полностью автоматически, и вы будете в Сети еще до того, как войдете в систему (поскольку Network Manager активизируется уже при запуске системы).

Статическая конфигурация LAN. Сравнительно новая функция Network Manager заключается в том, что соединение LAN можно конфигурировать и статически. Это требуется делать в тех случаях, когда ваш компьютер не подключен к роутеру или DHCP-серверу, и вы должны самостоятельно указывать IP-адрес, маску сети, адрес шлюза и адрес сервера имен (все упомянутые здесь термины подробно разъяснены в глоссарии по сетевым соединениям в разделе 16.3).

Для того чтобы начать конфигурацию, щелкните на значке Network Manager правой кнопкой мыши и выберите Обработка соединения. В результате вы попадете в диалоговое окно конфигурации сети, показанное на рис. 16.2.

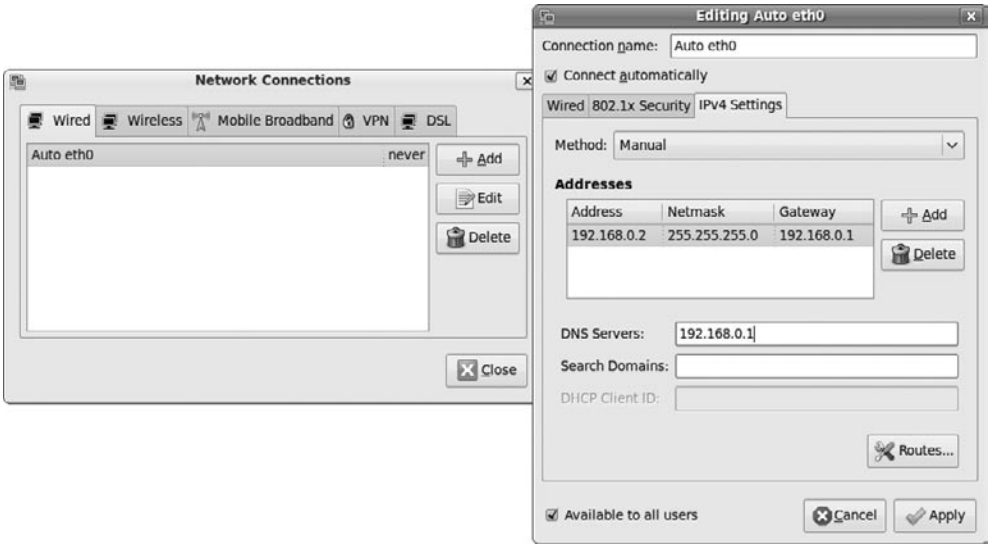

Рис. 16.2. Статическая конфигурация WLAN

Создание доступа к WLAN

Network Manager самостоятельно находит все расположенные поблизости сети WLAN. Когда вы впервые выбираете название WLAN в меню Network Manager, нужно указать пароль для входа в эту сеть. Обратите внимание, что в конфигурационном диалоговом окне предлагается несколько возможностей ввода ключа, например при шифровании WEP ключ можно вводить как кодовую фразу (фразу-пароль), как шестнадцатеричный код или как код ASCII. Выберите нужный вам формат. Шестнадцатеричные коды вводятся без 0x перед ними.

При последующих входах в сеть Network Manager будет устанавливать соединение самостоятельно. Для этого пароли WLAN централизованно сохраняются в системе. В некоторых системах, используемых на локальных компьютерах, вам может понадобиться при первом обращении к памяти ввести master-пароль. Если

возникнут проблемы с изменением имеющегося пароля WLAN, запустите в Gnome программу seahorse. На вкладке Пароли можно изменить или удалить все пароли.

Сеть WLAN может быть сконфигурирована так, что при соединении она не сообщает своего имени. В таком случае имя не отображается в меню Network Manager. Чтобы все же установить соединение, откройте окно настроек с помощью команды меню Подключиться к скрытой беспроводной сети. Здесь самостоятельно задайте имя сети (ESSID — расширенный набор служб идентификации), а также способ шифрования.

Конфигурирование точки соединения с сетью WLAN. Если ваш компьютер имеет доступ в Интернет через интерфейс LAN и, кроме того, на нем установлен контроллер WLAN, то этот компьютер можно конфигурировать с помощью Network Manager так, чтобы он функционировал как точка доступа в сеть WLAN для других машин. Правда, этот метод действует лишь с некоторыми контроллерами WLAN, чьи драйверы для Linux поддерживают работу в так называемой простой сети (режим ad-hoc).

Механизм конфигурирования прост: в меню Network Manager выбирается Создать новое беспроводное соединение, затем указывается нужный параметр. Кнопка Создать станет активной только после того, как система сочтет пароль достаточно длинным.

Насколько многообещающей кажется эта функция, настолько же удручающе закончились тесты, которые я проводил на нескольких ноутбуках с самыми разными контроллерами WLAN. Ни разу не удалось создать точку доступа к WLAN как с шифрованием, так и без него. Подробного сообщения об ошибке, которое бы помогло понять, почему так происходит, я, опять же, не дождался. Может быть, вам с вашим оборудованием повезет больше...

ADSL-модем

Если вы выходите в Интернет через ADSL, то проще всего подключиться через ADSL-роутер: соедините компьютер и роутер сетевым кабелем. Network Manager самостоятельно выполнит сетевую конфигурацию.

Ситуация осложняется, если ADSL-модем подключен к компьютеру напрямую (различие между ADSL-модемом и ADSL-роутером будет описано в разделе 16.9, посвященном внутренней организации ADSL).

Тонкости процесса зависят от того, какой модем вы используете и где находитесь. Network Manager поможет вам лишь в том случае, если вы работаете с модемом, который подсоединен к компьютеру сетевым кабелем (а не через USB). В зависимости от провайдера, ADSL-соединение может строиться на базе различных протоколов. Если провайдер использует протокол PPPoE, то выполните команду Обработка соединения из контекстного меню, перейдите на вкладку DSL и нажмите кнопку Добавить. Теперь заполните поля Логин и Пароль. Немного везения — и у вас получится установить соединение. Если же сразу дело не пойдет, вам придется поискать на остальных не слишком понятных вкладках настройки, требуемые вашим провайдером.

Некоторые провайдеры ADSL используют вместо PPPoE протокол PPTP. В таком случае в процессе конфигурации необходимо использовать вместо DSL вкладку VPN.

Модем UMTS/GMS

Многие модемы для мобильного интернета (UMTS) выглядят как большая USB-флешка и хорошо поддерживаются в Linux. Вставив такой модем в компьютер, вы через несколько секунд увидите окно ассистента конфигурации. В первом диалоговом окне выбирается провайдер услуги (фирма, обеспечивающая вам доступ к Интернету). На втором этапе новому соединению нужно дать название. Через меню Network Manager вы в первый раз активизируете соединение. При этом необходимо указать PIN и PUK-код.

Современные версии Network Manager работают только с четырехзначными кодами. Если ваш провайдер применяет код из большего количества разрядов (до восьми), закройте диалоговое окно, щелкните на значке Network Manager и выполните Обработка соединения. На вкладке Широкополосная передача данных выберите создание нового соединения — теперь вы можете указать PIN-код из любого количества разрядов.

На этом этапе также можно установить флажок Автоматическое подключение. Тогда Network Manager без запроса о подтверждении создает соединение с широкополосной сетью, как только вы вставляете модем. Эта настройка рекомендуется лишь в том случае, если вы используете тариф с фиксированной ставкой.

VPN

В виртуальной частной сети (VPN) используются механизмы шифрования, обеспечивающие в небезопасном поле передачи данных (в частности, в Интернете) соединение между двумя компьютерами, защищенное от перехвата информации. Во многих фирмах соединение с WLAN осуществляется преимущественно через VPN. Network Manager может быть полезен при создании VPN-соединений, так как в нем можно применить метод OpenVPN (свободно распространяемое решение), а также PPTP (Microsoft) и VPNC (Cisco). Для использования этих методов необходимо установить соответствующие пакеты, которые отличаются от дистрибутива к дистрибутиву (как правило, это `network-manager-openvpn`, `-pptp` и `-vpnc`). Кроме того, в Network Manager есть запись меню VPN-соединения, которая открывает ассистент, помогающий при конфигурации VPN.

Я протестировал создание конфигурации VPN с применением Network Manager только для протокола PPTP, и как минимум в этом случае результат был удручающим: модуль VPN программы Network Manager еще не доработан и функционирует с ошибками, в особенности это касается роутинга. Остается надеяться, что эти проблемы будут устранены в ближайшем будущем.

На многих PPTP-серверах для шифрования применяется метод MPPE. В таком случае нужно найти в диалоговом окне конфигурации VPN кнопку Дополнительно,

которая открывает еще одно диалоговое окно, и уже там установить флажок Использовать шифрование по двухточечному протоколу (MPPE).

При создании либо разрыве соединения система может автоматически выполнять сценарии. Они должны находиться в каталоге `/etc/NetworkManager/dispatcher.d/`. Подробное описание этого механизма и примеры работы с ним есть на сайте http://wiki.ubuntuusers.de/NetworkManager/Dispatcher.

16.2. Помощь при конфигурации, зависящая от системы и конкретных дистрибутивов

system-config-network (Fedora, Red Hat). По причинам исторического характера и из-за того, что в Network Manager по-прежнему есть много недоработок, в некоторых крупных дистрибутивах, а также в рабочих столах от Gnome и KDE предусмотрены собственные программы, упрощающие конфигурацию сети. Программа `system-config-network`, уже много лет входящая в состав дистрибутивов Fedora и Red Hat, помогает обеспечить доступ к сетям через соединения LAN, WLAN, ADSL, ISDN или через аналоговый модем (рис. 16.3). Актуальные версии этих программ позволяют указать для интерфейсов LAN и WLAN, должны ли эти интерфейсы управляться через Network Manager или через другую программу, специфичную для дистрибутива. Правда, при работе с адаптерами WLAN `system-config-network` поддерживает лишь ненадежное WEP-шифрование, поэтому рекомендую использовать вместо нее Network Manager.

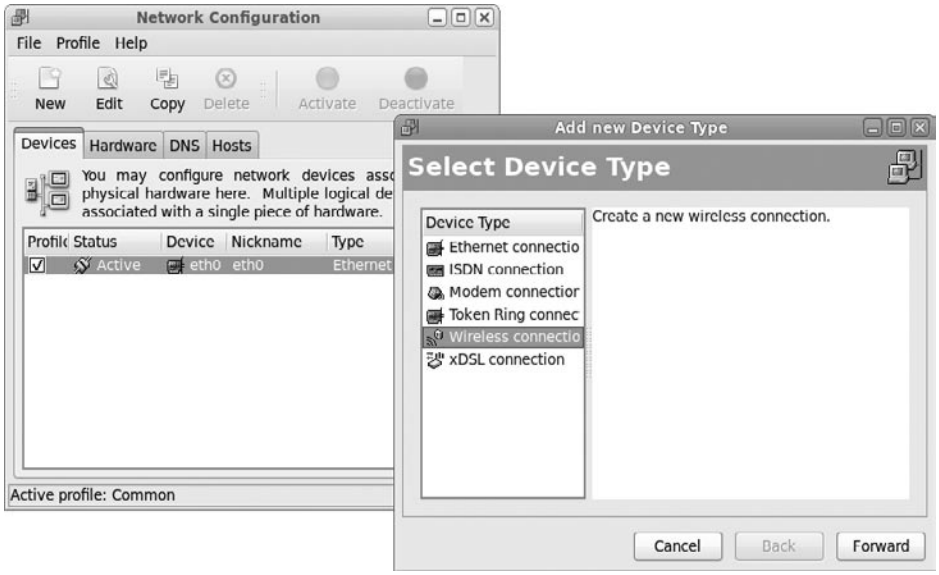

Рис. 16.3. Конфигурация сети в Fedora и Red Hat

YaST (SUSE). Модуль YaST Сетевые устройства помогает конфигурировать адаптеры LAN и WLAN, ADSL-модемы, ISDN-карты и аналоговые модемы. В ходе

конфигурации ADSL поддерживаются все распространенные протоколы (PPPoE, PPPoA, PPTP и даже CAPI для ADSL). Работа с YaST не всегда строится интуитивно, и все же эта программа приспособлена к работе с большим количеством вариантов ISDN и ADSL, чем любой дистрибутив, известный мне.

gnome-ppp (Gnome). Аналоговые модемы не подходят для работы с современными сайтами и поэтому обречены на вымирание. И все же, такие модемы — те самые незаменимые устройства, которые позволяют прочитать электронное сообщение в старой гостинице. Если Linux распознает ваш аналоговый модем, то установить соединение проще всего будет с помощью программы `gnome-ppp`, разработанной для создания соединений через аналоговые модемы в Gnome и KDE, или с помощью другой программы — KPPP. Поскольку аналоговые модемы выходят из употребления, эти программы по умолчанию не устанавливаются. Кроме того, многие встроенные аналоговые модемы — это так называемые «винмодемы», то есть устройства, для которых есть только драйверы Windows. Следовательно, такие модемы нельзя использовать с Linux, а если и можно, то с большим трудом.

Работать с `gnome-ppp` достаточно просто: вы указываете три ключевых параметра для доступа к Интернету — логин, пароль и телефонный номер — и нажимаете кнопку Подключение. В идеальном случае соединение устанавливается с ходу. Если сразу это не получится, откройте нажатием кнопки Конфигурация диалоговое окно Установка и настройте в нем остальные параметры. При настройке типа модема для большинства встроенных модемов применяется вариант Аналоговый модем (то же касается модемов, подключенных через последовательные интерфейсы) или USB-Модем, если модем подключен к компьютеру USB-кабелем.

KPPP (KDE). При работе с KPPP начните конфигурацию вашего модема и соединения с Интернетом (логин, пароль, телефонный номер), нажав кнопку Создать. Если вы полагаете, что вам, возможно, придется менять провайдеров, отдельно сконфигурируйте данные доступа и модем. KPPP поддерживает значительно больше параметров, чем `gnome-ppp`, из-за этого конфигурационные диалоговые окна программы KPPP весьма запутанны.

pppoeconfig. В некоторых дистрибутивах (например, Debian и Ubuntu) для настройки конфигурации модемов ADSL предоставляется команда `pppoeconf`. Она запускается от имени администратора в окне терминала. Команда просматривает все сетевые интерфейсы и ищет ADSL-модем. Предварительная конфигурация сетевого интерфейса не требуется.

Когда модем найден, введите имя пользователя и пароль. Последующие запросы программы, касающиеся автоматической конфигурации DNS, а также конфигурации MSS (системы коммутации региональной сети) можете просто подтверждать.

При работе с немецким провайдером T-online имя пользователя состоит из трех элементов: двенадцатизначного номера абонента `A`, номера T-online `T` и так называемого сопользовательского номера (обычно `0001`). В результате получаем следующую последовательность символов:

```
AAAAAAAAAAAATTTTTTTTTT#0001@t-online.de
Если T состоит из 12 символов или более, # не используется.
```

Наконец, программа спрашивает вас, следует автоматически устанавливать ADSL-соединение при запуске системы или соединение будет устанавливаться

вручную. Первый вариант целесообразно использовать, если у вас безлимитный тариф на Интернет (фиксированная ставка) и вы все время хотите оставаться в Сети. Если вы выберете второй вариант, доступ к ADSL нужно будет активизировать, а затем снова отключать с помощью следующих команд:

```
root# pon dsl-provider
...
root# poff dsl-provider
```

UMTSmon. UMTSmon помогает создавать интернет-соединения через UMTS-модем. Когда я тестировал эту функцию, соединение успешно устанавливалось лишь тогда, когда программа выполнялась с правами администратора. Конфигурация с помощью меню Подключение ▸ Обработка профиля исключительно проста. Вам нужно всего лишь знать имя точки доступа (APN) вашего провайдера.

Программа запоминает, сколько данных было передано, — это очень удобно в тех случаях, когда ваш тариф предусматривает ограничения по времени пребывания в Сети или по трафику. Кроме того, программа отображает сеть, активную в настоящий момент, и качество приема. Еще одна дополнительная функция UMTSmon позволяет изменить PIN-код или вообще его отключить. С помощью этой программы даже можно отправлять SMS.

Если для вашего дистрибутива нет готового пакета UMTSmon, то вы можете взять программу на следующем сайте в виде исходного кода или в скомпилированном виде под i386: http://umtsmon.sourceforge.net/.

16.3. Основы работы с LAN и WLAN

В этом разделе рассмотрены основы конфигурации сети. Информация касается не только подключения к LAN (локальной сети), но и конфигурации WLAN, ADSL и модема.

Глоссарий

TCP/IP. Во всех распространенных операционных системах поддерживается *сетевой протокол TCP/IP*. Он регулирует обмен информацией между компьютерами, как в локальных сетях (LAN, интранет), так и в Интернете. Поэтому для многих сетевых служб неважно, где именно находится другой компьютер из данной сети — на расстоянии пяти метров или в Японии. Во втором случае только немного снизится скорость.

Интернет-протокол (IP) служит основой для *протокола управления передачей* (TCP). Общий протокол TCP/IP выполняет две важные задачи: он идентифицирует любой компьютер по уникальному номеру (IP-адресу) и обеспечивает надежную передачу данных, то есть отвечает за то, чтобы данные, отправленные по сети, пришли именно к адресату, а не куда-то еще. Данные передаются в виде небольших пакетов.

Сетевые функции TCP/IP требуются вам даже тогда, когда компьютер не работает в сети и у вас нет ни сетевого подключения, ни модема! Многие программы Linux используют этот протокол как раз для внутрисистемной передачи данных.

Поэтому вам в любом случае потребуется установить петлевой (кольцевой) интерфейс. Это требуется делать во всех дистрибутивах.

UDP и ICMP. Наряду с TCP существуют еще два протокола, играющих в Интернете очень важные роли: UDP и ICMP. UDP — это *протокол пользовательских датаграмм*. Он обеспечивает так называемую ненадежную передачу пакетов. Ненадежный в данном контексте означает, что отправитель и получатель не должны постоянно поддерживать сетевое соединение и обмениваться пакетами с информацией. Поэтому при работе с UDP может случиться так, что какие-то пакеты не дойдут до адресата или дойдут не в той последовательности, в которой были посланы. Восстановлением целостности данных в данном случае занимается получатель, а не система. У UDP по сравнению с TCP есть определенное преимущество — этот протокол (по крайней мере, в некоторых сферах) более эффективен и быстрее реагирует при передаче данных, так как его КПД выше. Например, этот протокол используется со службами DNS и NFS.

ICMP — *протокол управления сообщениями в сети*. В принципе этот протокол был разработан не для обмена данными между программами, а для передачи управляющих кодов и кодов ошибок для TCP/IP. В частности, протокол ICMP применяется вспомогательным инструментом `ping`.

Порты. Каждый IP-пакет относится к определенной категории по номеру порта. Поэтому на стороне получателя становится проще упорядочивать пришедшие пакеты. Большинство интернет-приложений (WWW, FTP, электронная почта и т. д.) работают со специально выделенными для них портами.

PPP. Если соединение с Интернетом осуществляется без использования инфраструктуры локальной сети, а прямо через модем или ISDN-карту, то обычно используется *протокол двухточечного соединения* (PPP). Этот протокол позволяет передавать данные TCP/IP по телефонной линии, по ISDN или ADSL. PPP также можно использовать для создания *виртуальных частных сетей*.

Хост-имя и доменное имя. Пусть IP-номера (см. ниже) и очень практичны при работе с компьютером, но запоминаются они очень плохо. Поэтому компьютер параллельно можно идентифицировать и по комбинации хост-имени и доменного имени.

Хост-имя — это имя самого компьютера. *Доменное имя* обозначает частную сеть, в рамках которой может быть запрошен компьютер, и может состоять из нескольких частей.

В локальной сети хост-имена и доменные имена служат в первую очередь для того, чтобы пользователям было легче их запоминать. Например, мои тестовые компьютеры названы так же, как и планеты Солнечной системы (jupiter и др.), а в качестве доменного имени используется sol.

СОВЕТ

В качестве хост-имен не следует использовать названия производителей компьютеров, их владельцев или проектов, выполняемых в настоящее время, — здесь легко запутаться. Указывайте короткие и хорошо запоминающиеся названия зверей, растений, планет, рек или какие-то другие понятия, которые хорошо сидят у вас в памяти. Не используйте в названиях специальных символов из национальных алфавитов.

Никогда не применяйте в качестве хост-имени localhost! Оно особенное и считается названием отдельной сети (полностью уточненное имя). Этому названию всегда присвоен адрес 127.0.0.1, соответствующий петлевому интерфейсу, независимо от остальных параметров конфигурации сети.

При выборе доменного имени ограничений еще больше. Это имя должно соответствовать тому доменному имени, которое уже используется в сети. Только создавая новую локальную сеть, вы можете свободно выбирать доменное имя.

Если компьютер с Linux действует как открытый сервер, видимый в Интернете и служащий для работы с Сетью, электронной почтой или другими службами, то вы должны зарегистрировать желаемое имя у провайдера интернет-услуг или в Сетевом Информационном Центре (кратко — NIC). Например, домены зоны .de регистрируются в http://www.denic.de, а домены .com, .net и .org — в http://www.corenic.org.

IP-адреса или IP-номера. IP-адреса предназначены для того, чтобы однозначно идентифицировать компьютер в сети. Типичный IP-адрес компьютера в локальной сети — 192.168.0.75. Базовая информация по IP-адресам сообщается в разделе 16.3.

MAC-адрес. MAC-адрес (MAC — *управление доступом к среде*) — это уникальный ID-номер, который есть у каждого Ethernet-контроллера. Номер MAC обеспечивает идентификацию сетевого контроллера еще до присвоения ему IP-адреса. В частности, адреса MAC используются при работе с протоколом DHCP (см. раздел 17.5).

Интерфейс. IP-адрес обозначает не компьютер, а IP-интерфейс. Часто у компьютера есть несколько интерфейсов с различными IP-адресами. Обычно применяются петлевой интерфейс (127.0.0.1), один или несколько Ethernet-интерфейсов, а также, возможно, PPP-интерфейс, обеспечивающий доступ к Интернету через модем, ISDN или ADSL.

Если речь идет только об *одном* IP-адресе, то обычно подразумевается тот адрес, по которому компьютер запрашивается в локальной сети или Интернете. Как правило, это IP-адрес интерфейса Ethernet, присваиваемый хост-имени и доменному имени и являющийся уникальным в пределах сети.

Название интерфейса. Внутри системы Linux всем интерфейсам присваиваются названия (имена). Обычно петлевой интерфейс имеет название `lo`, интерфейсы Ethernet — названия `eth0`, `eth1` и т. д. и `pppn` — для интерфейсов PPP.

Петлевой интерфейс. Петлевой интерфейс играет особую роль: он обеспечивает применение сетевого протокола с локальными службами, то есть обмен информацией внутри компьютера. Это, возможно, звучит абсурдно, но такой механизм применяется многими простейшими командами Linux. Причина такова: обмен информацией между многими командами протекает на базе сетевого протокола, и при этом неважно, остаются данные в пределах локального компьютера или их обработка продолжается на удаленном компьютере. Так работает, например, система печати (CUPS), выполняющая свои задачи как на локальном компьютере, так и на других компьютерах в сети.

За петлевым интерфейсом закреплен IP-адрес 127.0.0.1. Во всех дистрибутивах настройка петлевого интерфейса осуществляется автоматически, даже если никакой сетевой конфигурации не происходит.

Маска сети, сетевой адрес и широковещательный адрес. Протяженность локальной сети выражается двумя или тремя масками. *Маски* — это четырехчастные группы цифр, используемые внутри системы как конфигурации битов для IP-

адресов. Если локальная сеть включает в себя все номера 192.168.0.*n*, то соответствующая маска сети будет 255.255.255.0, сетевой адрес — 192.168.0.0, а широковещательный адрес — 192.168.0.255 (во многих конфигурационных программах не требуется вводить широковещательный адрес, так как он автоматически выводится из двух других адресов).

Полученная сеть будет обозначаться как 192.168.0.0/255.255.255.0 или коротко — 192.168.0.0/24 (в кратком варианте записи указывается количество двоичных единств в маске сети). Это означает, что два компьютера, имеющие IP-адреса 192.168.0.71 и 192.168.0.72, могут напрямую обмениваться информацией в этой сети (так как IP-номера в адресном пространстве маски сети соответствуют друг другу). Максимальное количество компьютеров, которые могут одновременно обмениваться данными в рамках этой сети, составляет 254 (от .1 до .254) — номера .0 и .255 зарезервированы.

Шлюз. Это компьютер, стоящий на стыке двух сетей (зачастую между локальной сетью и Интернетом). Чтобы ваш компьютер с Linux, работающий в локальной сети, имел выход в Интернет, при конфигурации необходимо указать *адрес шлюза*.

Итак, адрес шлюза обозначает один из компьютеров в локальной сети, например 192.168.0.1. Этот компьютер играет особую роль, так как через него (например, с помощью ADSL) осуществляется соединение с Интернетом. Таким образом, весь интернет-трафик локальной сети проходит через компьютер-шлюз.

Сервер имен. Это программа, преобразующая имена компьютеров или интернет-адреса (например, www.yahoo.com) в IP-адреса. В небольших сетях имена и номера часто соотносятся по таблице (файл `/etc/hosts`). В Интернете эту задачу выполняют специальные базы данных. Вместо термина *сервер имен* часто применяется аббревиатура *DNS (сервер доменных имен)* или термин *службы*.

Если вы хотите просмотреть в браузере сайт yahoo.com, то сначала система свяжется с сервером имен, чтобы узнать номер сервера www.yahoo.com. Когда эта задача будет выполнена, устанавливается связь с указанным IP-адресом.

DHCP. *Протокол динамической настройки хостов* (DHCP) часто используется в локальных сетях для централизованного администрирования сети. Чтобы не приходилось отдельно настраивать для каждого компьютера IP-адрес, шлюз, сервер имен и т. д., один из компьютеров конфигурируется как DHCP-сервер (см. раздел 17.5). Все остальные компьютеры в сети при запуске системы выходят на связь с DHCP-сервером и «спрашивают» его, какие настройки необходимо применить. Таким образом, работа по конфигурированию клиента сводится к минимуму.

IP-адреса

Как уже было сказано выше, IP-адреса предназначаются для идентификации компьютеров в сети. Это правило касается как локальных сетей, так и Интернета. В этом разделе будет рассмотрена базовая информация, касающаяся применения IP-адресов.

Теоретически, существует 256^4, то есть около 4 миллиардов IP-адресов. На самом деле таких адресов доступно гораздо меньше, так как часть из них зарезервирована

(в том числе все IP-адреса, начинающиеся с .0 или .255). Кроме того, раньше IP-адреса раздавались очень большими пакетами.

В эпоху бурного роста Интернета становится все сложнее выполнять требование соблюдения во всем мире уникальности IP-адресов для всех имеющихся компьютеров. До тех пор пока не закрепился формат IPv6 (это новая версия интернет-протокола, которая наряду со многими другими улучшениями позволяет значительно расширить адресное пространство), IP-адреса останутся весьма ограниченным ресурсом.

IP-адреса в Интернете

Если вы хотите подключить свой веб-сервер к Интернету, вам потребуется, во-первых, доменное имя, действующее в Глобальной сети (например, moyafirma.ru), а также собственный IP-адрес. И домен, и адрес проще всего получить у провайдера или в сетевом информационном центре (NIC).

Для частных пользователей и небольших организаций это, как правило, совсем не обязательно. В локальной сети используются IP-адреса так называемого *частного адресного пространства* (сейчас я объясню, что это такое). Связь с Интернетом обеспечивается провайдером, который предоставляет (на время соединения) IP-адрес, действительный во всем мире.

Если вам к тому же требуется иметь в Интернете постоянное представительство, то есть собственный сайт, его также можно получить с помощью провайдера услуг. Компьютер с сервером и вашими HTML-документами находится у провайдера (а не у вас дома), за IP-адрес опять же отвечает провайдер. Такой метод имеет определенное достоинство — вам не требуется постоянное соединение с Интернетом, а сайт доступен в Интернете все время.

Резюмирую: собственный, уникальный и действительный во всем мире IP-адрес вам нужен лишь в том случае, если ваш компьютер должен постоянно быть доступен в Интернете и для этого у вас установлено непрерывное соединение с Сетью (например, через выделенную линию). Как правило, такое соединение требуется лишь крупным фирмам или, например, университетам.

IP-адреса в локальных сетях

Компьютеры, расположенные в локальных сетях, в Интернете, как правило, невидимы. Это означает, что при правильной конфигурации такие компьютеры могут работать в Интернете, но в то же время они остаются защищены от неконтролируемого доступа из Интернета. Таким образом, IP-адреса, действующие в локальной сети, должны быть уникальны только в пределах этой сети, но не в глобальном масштабе.

Поскольку IP-адресов все сильнее не хватает, такая экономия представляется очень привлекательной. На данный момент в адресном пространстве номеров IP для локальных сетей зарезервировано три диапазона:

- 10.0.0.0–10.255.255.255;
- 172.16.0.0–172.31.255.255;
- 192.168.0.0–192.168.255.255.

Первый диапазон позволяет построить очень большую локальную сеть (теоретически — с 16 миллионами компьютеров, этого достаточно даже для очень крупных фирм). Во втором диапазоне мы имеем дело, собственно, с 16 подсетями, каждая из которых включает около 65 000 адресов (например, от 172.23.0.0 до 172.23.255.255). Третий диапазон состоит из 256 малых подсетей. Одна из таких сетей — от 192.168.75.0 до 192.168.75.255.

Неважно, в какой из подсетей строится ваша локальная сеть; такой метод исключает возможность возникновения адресных конфликтов с «настоящими» IP-адресами.

Как правило, требуется пользоваться функциями Интернета и в локальной сети (например, просматривать сайты). Чтобы это было возможно, один из компьютеров локальной сети должен быть сконфигурирован как интернет-шлюз. Этот компьютер обеспечивает соединение с Интернетом (через ADSL, ISDN, модем или иным способом) и переадресовывает все запросы, поступающие из локальной сети. Кроме того, шлюз должен заменять IP-адреса локальной сети IP-адресами, действительными в глобальном масштабе. Эта функция называется *маскарадинг*. Она рассмотрена в главе 17.

Динамические IP-адреса

Итак, уже понятно, что для идентификации компьютера или сети требуется IP-адрес. Но откуда компьютеру будет «известно», какой IP-адрес он должен применять? Простейший способ сообщить об этом — прямо задать IP-адрес при конфигурации. В небольших локальных сетях обычно так и делается. Например, первый компьютер в сети получает адрес 192.168.0.1, следующий — 192.168.0.2 и т. д. Адрес сохраняется в файле `/etc/hosts`.

Размеры сетей увеличиваются, и проводить такую децентрализованную конфигурацию становится все сложнее. Чтобы такой конфигурации не требовалось, часто применяются динамические IP-адреса. Для этого один из компьютеров в сети необходимо сконфигурировать как DHCP-сервер (выше было указано, что DHCP — это протокол динамической настройки хостов). Все остальные компьютеры перед началом использования сетевых функций устанавливают соединение с DHCP-сервером, и он присваивает им IP-адреса. Часто задачи DHCP-сервера выполняет ADSL-роутер.

Подобный метод имеет два основных преимущества: во-первых, всей сетью можно управлять централизованно (а не настраивать параметры на каждом компьютере отдельно). Во-вторых, работа по администрированию клиентского компьютера сводится к нулю. Чтобы подключить клиентский компьютер к сети, нужно просто указать имя компьютера и установить флажок DHCP. Все остальные данные (сам IP-адрес компьютера, IP-адреса DNS и шлюза и т. д.) передаются через DHCP.

Кроме того, есть еще и третье преимущество, которое, правда, в большей степени касается провайдеров интернет-услуг: поскольку IP-адреса присваиваются динамически, а в сети обычно бывает активна только часть компьютеров, достаточно сравнительно небольшого количества IP-адресов, чтобы обеспечить ими всех нуждающихся абонентов. Всякий раз, когда клиент провайдера выходит в сеть через модем или ISDN, ему предоставляется ближайший свободный IP-адрес.

Один компьютер с несколькими IP-адресами

Как правило, один компьютер имеет несколько IP-адресов. Ранее, когда мы говорили об *одном* IP-адресе, имелось в виду, что один IP-адрес присваивается интерфейсу одного сетевого контроллера. Это номер, по которому компьютер идентифицируется в сети (что, по сути, означает: не компьютер имеет IP-адрес, а интерфейс сетевого контроллера, установленного на этом компьютере).

Кроме того, любой компьютер с UNIX/Linux доступен по адресу 127.0.0.1 или под именем localhost. Это адрес уже упоминавшегося выше петлевого интерфейса, который предназначен для работы только с трафиком локальной сети. Чтобы проверить, работает ли этот механизм, нужно просто выполнить команду `ping`:

```
user$ ping localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.049 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.040 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.041 ms
--- localhost ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.040/0.043/0.049/0.006 ms
```

Команда `ping` отсылает по указанному адресу небольшие пакеты с данными и измеряет, сколько времени пройдет до подтверждения прибытия пакетов. Команда `ping localhost` должна работать и в том случае, если на компьютере отсутствует сетевой контроллер!

Наконец, возможна ситуация, когда на компьютере установлено несколько сетевых контроллеров. Каждый контроллер считается отдельным интерфейсом и поэтому имеет собственный IP-адрес. Кроме того, отдельный интерфейс задействуется при PPP-соединении, создаваемом с помощью ISDN-карты или при выходе в Интернет через модем. Такому интерфейсу также присваивается IP-адрес, причем этот адрес, как правило, назначается провайдером интернет-услуг (в данном случае речь идет о динамическом IP-адресе).

Несколько сетевых контроллеров может быть установлено на компьютере и в том случае, если он должен связать две подсети с различными адресными пространствами. Такой компьютер называется *роутером*. Примером роутера является интернет-шлюз локальной сети. При создании соединения с Интернетом компьютер-шлюз имеет как минимум три IP-адреса: адрес петлевого интерфейса 127.0.0.1, адрес, действующий в локальной сети и, наконец, выделенный провайдером интернет-услуг глобальный адрес во Всемирной паутине.

ПРИМЕЧАНИЕ

Строго говоря, сам шлюз, как правило, не занимается роутингом, а только выполняет функцию маскарадинга. Это тонкое отличие более подробно описано в главе 17.

Статическая конфигурация IP-адреса

Если в локальной сети нет DHCP-сервера, то при конфигурации сети IP-адрес сетевого интерфейса потребуется настроить статически. Какой IP-адрес нужно применять в таком случае?

- **Ваш компьютер не входит в состав локальной сети.** Не считая петлевого интерфейса 127.0.0.1, вам на данный момент не требуется никакого IP-адреса (это правило действует и в том случае, если позже компьютер подключается к Интернету через модем/ISDN/ADSL). Работа по конфигурации сводится к указанию доменного имени и хост-имени.
- **Ваш компьютер входит в состав локальной сети.** В таком случае IP-адрес должен принадлежать к адресному пространству, выделенному для данной сети (например, 192.168.0.*) и быть уникальным в пределах этой сети.
- **Ваш компьютер должен служить основой для локальной сети.** Используйте частное адресное пространство (например, 192.168.0.*) и присвойте компьютеру предназначенный для этого IP-адрес.

IPv6

До сих пор мы говорили о четвертой версии IP-адресов (IPv4). Весь нынешний Интернет работает на основе этой версии. Однако уже в течение нескольких лет ощущается нехватка IP-адресов, причем со временем она становится все более серьезной. Кроме того, данный протокол имеет некоторые функциональные недостатки и плохо приспособлен для использования различных новых технологий, играющих в Интернете все более значимую роль (в частности, IP-телефонии, потокового аудио и видео).

Адреса IPv6

В новой версии IP-адресов (IPv6) эти недостатки должны быть устранены. Самое примечательное и наиболее очевидное для администраторов изменение заключается в том, что теперь IP-адреса будут иметь длину 128 бит (а не 32, как в IPv4). При записи в традиционном формате IP-адрес будет выглядеть так:

```
121.57.242.17.122.58.243.18.19.123.59.20.244.124.60.245
```

Очевидно, что такие записи неудобны на практике. Ради экономии места адреса IPv6 будут разбиваться символом : на группы шестнадцатеричных чисел (не более восьми групп), например так:

```
abcd:17:2ff:12aa:2222:783:dd:1234
```

Чтобы не нужно было писать лишнего, используется символ ::, который является сокращенной формой для нескольких нулевых групп:

```
abcd:17:0:0:0:0:dd:1234 → abcd:17::dd:1234
0:0:0:0:0:783:dd:1234 3 → ::783:dd:1234
```

Для localhost существует еще более компактная запись — ::1.

При отображении адресов IPv4 в формате IPv6 первые шесть групп являются нулевыми. Оставшиеся две группы можно записывать не только в шестнадцатеричной, но и в более привычной десятичной системе:

```
Адрес IPv4: ::110.111.112.113
```

В рамках перехода на IPv6 в течение нескольких лет будут совместно использоваться IPv4 и IPv6. Существуют различные методы, позволяющие передавать пакеты IPv6 по сетям IPv4 и наоборот.

IPv6 и Linux

В принципе ядро Linux приспособлено к работе с IPv6, уже начиная с версии 2.2, но только в версии 2.6 и выше поддержка IPv6 считается окончательно доработанной. Большинство сетевых приложений уже совместимо с IPv6.

Поскольку формат IPv6 еще не закрепился в Европе (даже частично), вся информация, сообщаемая в этой книге, касается только IPv4. Если вы хотите более подробно изучить IPv6 и особенности работы с этим протоколом в Linux, рекомендую начать со следующих сайтов:

- http://www.ipv6.org/;
- http://de.wikipedia.org/wiki/IPv6;
- http://www.faqs.org/rfcs/rfc1752.html;
- http://www.bieringer.de/linux/IPv6/.

Глоссарий по стандартам WLAN

При описании беспроводных сетей применяется множество сокращений. Чаще всего используется аббревиатура WLAN (беспроводная локальная сеть), несколько реже — русский вариант БЛВС (беспроводная локальная вычислительная сеть). Кроме того, часто используется синонимичный термин Wi-Fi (от англ. «беспроводная точность воспроизведения»). Правда, иногда речь идет об «альянсе Wi-Fi» — консорциуме производителей, который занимается вопросами совместимости, связанными с WLAN.

В этом разделе кратко обобщена терминология, касающаяся WLAN. Если вы уже знакомы с основами WLAN, можете перейти прямо к подразделу «Поддержка WLAN в Linux» этого раздела, где рассматриваются функции WLAN, специфичные для Linux.

Стандарты. Стандартов WLAN существует очень много. Все они определены IEEE (институтом инженеров по электротехнике и радиоэлектронике) и начинаются с номера 802.11. Буквы, следующие за этим номером, в хронологическом порядке указывают на новые версии или варианты стандарта. Далее перечислены и кратко описаны некоторые стандарты WLAN.

- **802.11.** Первый стандарт 802.11 описывал радиочастоту 2,4 ГГц. Максимальная брутто-частота передачи данных составляла около 2 Мбит/с. Этот стандарт сегодня уже не играет никакой роли.
- **802.11a.** При применении данного стандарта радиочастота составляет 5,2 ГГц, а брутто-частота передачи данных может достигать 54 Мбит/с. Стандарт 802.11a закрепился только в США.
- **802.11b.** Оборудование стандарта 802.11b вещает на частоте 2,4 ГГц. Брутто-частота передачи данных ограничена 11 Мбит/с и не отвечает современным требованиям.

- **802.11g.** Этот стандарт развился из 802.11b и совместим с ним. Радиочастота, как и в 802.11b, составляет 2,4 ГГц, а брутто-частота передачи данных возросла до 54 Мбит/с.
- **802.11i.** Данное дополнение к 802.11a/b/g/h определяет механизм шифрования и аутентификации WPA2.
- **802.11n.** Развился на основе 802.11a, b и g и не только имеет гораздо более высокую брутто-частоту передачи (до 540 Мбит/с), но и отличается большей дальностью действия. Оба указанных улучшения были достигнуты благодаря одновременному применению нескольких антенн, приемников и передатчиков (технология многоканальный вход — многоканальный выход, или MIMO). 802.11n совместим с вариантами 802.11a, b и g, однако, если в сети есть хотя бы один элемент, работающий с иным стандартом, чем 802.11n, этот элемент влияет на скорость во всей сети. Оборудование, предназначенное для работы с 802.11n, существует уже с середины 2006 года. Стандарт принят 11 сентября 2009 года (http://ru.wikipedia.org/wiki/IEEE_802.11n).

Брутто и нетто. Указываемая в проспектах брутто-частота передачи данных (например, 54 Мбит/с для 802.11g) часто выглядит многообещающе. Однако за вычетом издержек, связанных с конкретным протоколом, действительная частота уменьшается более чем наполовину. Этого не избежать даже в тех случаях, когда по беспроводной сети связываются всего два абонента, а физически радиосвязь устанавливается во вполне приемлемых условиях (на небольшом расстоянии, без препятствий и т. д.), к тому же в сети нет абонентов, которые бы работали с оборудованием, использующим устаревший стандарт WLAN.

Оборудование WLAN. Все современные ноутбуки уже оснащены контроллером WLAN. Радиомосты WLAN, а также специальные точки доступа и роутеры — это отдельные внешние устройства, подключаемые к локальной сети или ADSL-модему с помощью Ethernet-кабеля.

- Радиомост WLAN соединяет отдельное устройство LAN с WLAN. Таким образом, мост выполняет ту же функцию, что и WLAN-карта, только подключение к компьютеру осуществляется через Ethernet, а не через PCI, PCMCIA или USB.
- Точка доступа (Access Point) — это простейший механизм, позволяющий предоставить доступ во WLAN для нескольких WLAN-клиентов. Точка доступа, как и мост, подсоединяется Ethernet-кабелем к сетевому серверу или к сетевому концентратору. Она несколько раз в секунду посылает сигнал (маячок), предназначенный для того, чтобы другие WLAN-устройства, находящиеся в радиусе действия, могли распознать точку доступа. В отличие от радиомоста, точка доступа обычно поддерживает дополнительные режимы WLAN и может одновременно обмениваться информацией с несколькими клиентами (то есть различия между радиомостом и точкой доступа касаются в основном программной составляющей и в меньшей степени — аппаратной).
- WLAN-роутер подключает целую сеть (LAN и WLAN) к Интернету. Обычно исходным пунктом соединения является DSL-модем с Ethernet-выходом или сервер локальной сети (существуют и WLAN-роутеры с интегрированным ADSL-модемом, которые называются *шлюзами*).

Обычно роутер состоит из точки доступа и небольшого концентратора, рассчитанного на 4–8 Ethernet-устройств. Внутреннее программное обеспечение роутера управляет доступом к Интернету и работой локальной сети. Как правило, роутер выполняет функции трансляции сетевых адресов (NAT), сервера DHCP, имеет простой брандмауэр и т. д. (эти функции подробно описаны в главе 17).

Как правило, конфигурация устройств WLAN осуществляется в браузере. Для этого устройства предоставляют специальные веб-страницы, облегчающие конфигурацию и расположенные по определенному IP-адресу (например, http://192.168.0.1). Однако обратите внимание и на то, что отдельные устройства WLAN можно сконфигурировать только с помощью специальной установочной программы, работающей лишь в Windows. Это касается, в частности, мостов WLAN. Разумеется, в Linux такие устройства можно использовать лишь в ограниченной мере.

Параметры WLAN-соединения

Если вы устанавливаете соединение между двумя устройствами WLAN, то потребуется настроить различные параметры. Далее эти параметры вкратце описаны.

Сетевой режим. Компоненты сети WLAN могут различными способами обмениваться данными друг с другом. Рассмотрим основные режимы.

- *Инфраструктурный режим* (иногда называемый также управляемым, или ведомым (managed mode)) обеспечивает обмен информацией с центральной точкой доступа. Иными словами, сеть имеет «звездчатую» форму. Обычно такой центральной точкой является или точка доступа, или WLAN-роутер, но это может быть и сконфигурированный соответствующим образом компьютер.
- Устройство WLAN точки доступа работает в *ведущем режиме* (master mode) (иначе говоря, в инфраструктурном режиме работают компьютеры-клиенты, а в ведущем режиме работает сервер сети WLAN).
- В *режиме простой сети* (ad-hoc mode) любое устройство WLAN обменивается информацией с любым другим устройством WLAN, находящимся в зоне действия передатчика.

SSID или ESSID. Сокращения SSID (набор служб идентификации сети) или ESSID (расширенный набор служб идентификации сети) обозначают обычные последовательности символов, которые служат названием беспроводной сети. Устройства WLAN могут обмениваться информацией лишь при условии, что их SSID совпадают. Таким образом, с помощью различных SSID можно разделить две различные сети WLAN, функционирующие вблизи друг друга.

В качестве последовательности символов SSID часто задается название производителя, поэтому обмен информацией между устройствами одного производителя часто настраивается с ходу, а для устройств различных производителей сначала нужно установить общую последовательность символов SSID.

Некоторые карты WLAN автоматически осуществляют конфигурацию SSID. Обратите внимание, что при указании последовательности символов SSID важен регистр!

NWID. В сети WLAN с одинаковым SSID может быть много подсетей, которые различаются по показателю NWID. На практике такая ситуация встречается редко, поэтому некоторые конфигурационные программы вообще не работают с NWID.

Иногда вместо NWID употребляется термин *«домен»*, но он только вносит путаницу. NWID не имеет ничего общего с привычными доменными именами IP-адресов.

Канал. В рамках каждой полосы частот, предусмотренной тем или иным стандартом 802.11x, есть много поддиапазонов (каналов), по которым информация может рассылаться параллельно. В инфраструктурном режиме адаптеры WLAN сами распознают канал, применяемый точкой доступа. Специально настраивать каналы необходимо лишь в том случае, когда имеет место интерференция нескольких сетей WLAN.

Ключ WEP/WPA. В целях безопасности связь по сети WLAN должна быть защищена от перехвата данных. В зависимости от того, по какому стандарту работает оборудование WLAN, могут применяться методы WEP, WPA, а лучше всего WPA2 (эти методы подробно рассмотрены в следующем разделе). В ходе конфигурации контроллера WLAN вы указываете ключ. Обратите внимание — в некоторых конфигурационных программах перед названием ключа необходимо ставить `0x`, чтобы ввести ключ в шестнадцатеричной системе!

Безопасность WLAN

В принципе сеть WLAN можно использовать и без шифрования данных. Но в этом случае любой, кто будет находиться в зоне действия передатчика, сможет пользоваться сетью и считывать всю передаваемую в ней информацию. Таким образом, обмен данными без шифрования — не что иное, как грубая халатность!

WEP

Для шифрования передаваемых данных в первых поколениях WLAN использовался метод WEP (конфиденциальность на уровне проводных сетей). При этом данные зашифровываются, на ваш выбор, с помощью 40- или 104-битного ключа (часто говорят о 64- или 128-битном шифровании, но оставшиеся 24 разряда не используются непосредственно в целях шифрования).

Как правило, ключ WEP указывается в виде шестнадцатеричного числа (от 10 до 26 разрядов, в зависимости от количества бит в ключе). Поскольку при вводе 26-значного числа вручную легко допустить ошибку, во многих конфигурационных инструментах предусмотрена возможность создания ключа из фразы-пароля (пароля, состоящего из нескольких слов). Процесс генерирования ключа зависит от производителя. Таким образом, один и тот же пароль может реализовываться на оборудовании различных производителей в виде разных ключей. При возникновении сомнений вам не остается ничего другого, кроме как указать ключ вручную.

В ходе конфигурации WEP можно задать до четырех ключей. В действительности всегда будет использоваться только один. Однако в управлении четырьмя ключами есть определенный плюс — при смене сети WLAN вам не придется заново вводить целый ключ, а нужно будет всего лишь активизировать другой из четырех ключей.

ВНИМАНИЕ

Алгоритм WEP оказался ненадежным, так как в нем имеется несколько принципиальных недоработок! Даже 104-разрядный ключ можно узнать за несколько минут, просто перехватывая весь трафик WLAN. Программы, предназначенные для взлома ключей WEP, имеются в Интернете в свободном доступе. Таким образом, хуже WEP может быть только полное отсутствие защиты.

Если ваше оборудование позволяет, используйте алгоритм WPA, а еще лучше WPA2. Если же это невозможно, то ваша беспроводная сеть нуждается в дополнительной защите — для этого лучше всего применять VPN.

WPA, WPA2

На смену WEP пришел метод *защищенного доступа к Wi-Fi* (кратко — WPA), а также его улучшенная версия WPA2. Подробная спецификация WPA2 описана в стандарте 802.11i. Важнейшее различие между WPA и WPA2 заключается в том, что при этих методах используются разные алгоритмы шифрования: RC4 в WPA, AES в WPA2.

WPA разрабатывался как временное решение, и его рассчитывали применять до того, как будет полностью подготовлен стандарт 802.11i. Однако, поскольку существует оборудование, которое работает с WPA, но не работает с WPA2, в обозримом будущем будут использоваться оба метода.

Существенное преимущество WPA заключается в том, что ключ применяется только для инициализации соединения. Когда оно установлено, ключ начинает постоянно изменяться в соответствии с хитроумным алгоритмом. В настоящее время WPA и WPA2 считаются надежными, если в качестве пароля применяется достаточно длинная фраза (то есть ключ, состоящий из нескольких слов и дополнительных символов).

В этой книге будет рассмотрен только вариант WPA/WPA2 с применением *предварительно выданного ключа* (Pre-Shared Key или PSK, иногда также называется «персональный WPA»). В таком случае все пользователи WLAN указывают для входа в сеть один и тот же ключ. При применении еще более надежного варианта, называемого *управляемым ключом*, каждый пользователь имеет собственный ключ, и в таком случае управление всеми ключами централизованно осуществляется на сервере.

Не забывайте, что при работе с большинством WLAN-роутеров и точек доступа настраивается *один* метод шифрования. Иными словами, невозможно настроить сеть так, чтобы один компьютер устанавливал соединение с помощью WPA2, а второй — с помощью WEP. Таким образом, стандарт безопасности, с которым может работать даже самое старое ваше устройство, определяет, с каким стандартом будет работать вся сеть.

Базовая защита

Независимо от того, какую технику шифрования вы используете, необходимо обеспечить и базовую защиту беспроводной сети.

- Настройки точки доступа обычно можно изменить в браузере. Доступ к Интернету защищен паролем, специфичным для конкретной фирмы, — такой пароль обязательно нужно изменить. Следует также свести к минимуму удаленное обслуживание, а если такое обслуживание необходимо, то устанавливать соединение по LAN, а не по WLAN.

- При работе со многими точками доступа WLAN-доступ предоставляется только с определенного MAC-адреса. MAC-адрес (адрес для управления доступом к среде передачи данных) — это уникальный номер контроллера WLAN. Такая защита является недостаточной, так как хакер вполне может воспользоваться фальшивым MAC-адресом.
- Выключайте точку доступа, если не пользуетесь ею в данный момент.
- Задавайте как можно более длинные ключи и пароли, которые нельзя отгадать методом подбора.

Брандмауэр и VPN

Брандмауэр позволяет целенаправленно ограничивать трафик передаваемых по сети WLAN данных определенными протоколами, сегментами сети и т. д. Кроме того, существует точка зрения, согласно которой WLAN, несмотря на любые меры защиты, не гарантирует полной конфиденциальности данных. Чтобы все же обеспечить надежный обмен информацией по беспроводной сети, шифруйте сами данные, передаваемые по сети. Для этого чаще всего применяется VPN. Конфигурация VPN подробно описана в разделах 16.10 (клиент) и 18.5 (сервер).

Поддержка WLAN в Linux

При использовании контроллеров WLAN в Linux вам пригодится инструментарий для беспроводной связи в Linux — независимо от того, работаете вы с адаптером WLAN, встроенным в компьютер WLAN-оборудованием, картами PCI либо PCMCIA или внешними USB-устройствами. Указанный инструментарий — это относительно небольшое собрание команд (`iwconfig`, `iwlist` и т. д.), предназначенных для конфигурирования адаптера WLAN. Некоторые из них будут подробно рассмотрены в следующих разделах. Инструментарий для беспроводной связи входит в состав всех распространенных дистрибутивов. Более подробно этот набор команд рассмотрен на следующем сайте: http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html.

Аппаратные драйверы

В инструментарии для беспроводной связи содержатся только управляющие команды. Сами аппаратные драйверы располагаются в модулях ядра. В современных версиях Linux есть драйверы почти для всех адаптеров WLAN, имеющихся на рынке, но, как и всегда, бывают исключения. Особенно сложно обстоит ситуация с самыми новыми адаптерами WLAN — и это неудивительно. Даже при очень тесном сотрудничестве между производителями оборудования и сообществом разработчиков Linux может потребоваться целый год для того, чтобы новые драйверы вошли в состав современных дистрибутивов. Иначе говоря, покупая ноутбук, не помешает сначала подробно изучить эту модель в Интернете.

Встроенное программное обеспечение (прошивка)

Большинство контроллеров WLAN являются программируемыми. Чтобы они работали, в ходе инициализации в контроллер должен быть передана так называемая прошивка (программный код, предназначенный для использования внутри

контроллера). Такой код пишется производителями контроллеров и при условии соблюдения необходимых лицензионных условий может распространяться свободно. За передачу кода в контроллер обычно отвечает модуль ядра или система udev. Код контроллера находится в двоичных файлах («блобах»), которые, в свою очередь, чаще всего располагаются в каталоге `/lib/firmware`.

Производители микросхем обычно предоставляют файлы прошивки в виде двоичного, а не исходного кода. С точки зрения сообщества, выступающего за свободное распространение ПО, это нехорошо, поэтому такой метод подвергается критике, особенно со стороны разработчиков Debian. Мне эта проблема не кажется столь драматичной: ранее в контроллер WLAN было встроено стираемое программируемое запоминающее устройство (EPROM) и никто не думал возмутиться насчет того, что код этого устройства не является открытым. Используемое в настоящее время решение дешевле и может обновляться. Конечно, было бы очень желательно, чтобы все программы, содержащиеся в контроллере, были доступны в виде исходного кода, но такая мечта кажется несбыточной.

Использование драйвера для Windows

Если драйвера для Linux нет, то почему бы не использовать драйвер для Windows? На первый взгляд это невозможно, но на практике такую замену вполне можно осуществить: интерфейс для интеграции драйверов WLAN под Windows является относительно компактным. В различных коммерческих и свободно распространяемых проектах этот интерфейс (NDIS) был приспособлен для работы с Linux. Вопросы применения в Linux драйверов, предназначенных для Windows, рассматриваются на следующих сайтах:

- http://www.linuxant.com/driverloader/wlan/ — коммерческие;
- http://sourceforge.net/projects/ndiswrapper/ — с открытым кодом.

В ходе конфигурации NDIS-Wrapper можно использовать графические пользовательские интерфейсы `ndisgtk` или `NdisConfig`:

- http://jak-linux.org/projects/ndisgtk/;
- http://code.google.com/p/ndisconfig/.

Разумеется, использование двоичных драйверов противоречит идеям свободного ПО, так как код таких драйверов не предоставляется в распоряжение сообщества разработчиков. Еще один недостаток заключается в том, что существующие драйверы Windows могут работать только в системах, совместимых с Intel/AMD. Но если нет никакой другой возможности использовать имеющееся оборудование, то применение драйверов для Windows — вполне приемлемый выход.

16.4. Активизация контроллеров LAN и WLAN вручную

Как правило, при запуске компьютера контроллеры LAN и WLAN автоматически распознаются и инициализируются. В этом разделе рассмотрено, как именно протекает данный процесс и что нужно делать, чтобы при необходимости активизиро-

вать эти контроллеры вручную. Разумеется, я не ставлю своей целью научить вас, как настраивать целые сетевые интерфейсы вручную. Гораздо важнее, прочитав раздел, понять основы этого процесса. Кроме того, сообщаемая здесь информация должна пригодиться, если у вас возникнут проблемы с активизацией или автоматическим конфигурированием контроллера WLAN.

Активизация контроллера LAN

Обычно сетевой контроллер или контроллер LAN — это микросхема, стоящая на материнской плате вашего компьютера и предоставляющая функции Ethernet. Контроллер может находиться и на отдельной сетевой карте, например если необходимо оборудовать компьютер дополнительным сетевым интерфейсом. В дальнейшем мы абстрагируемся от того, как физически реализуются сетевые функции, и поговорим только о сетевом контроллере.

Распознавание оборудования

Сначала необходимо удостовериться, что в ядро загружен модуль, необходимый для конкретного сетевого контроллера. Часто ядро делает это автоматически. В таком случае команда выполняется без ошибок (см. ниже). Если на данном этапе возникнут проблемы, необходимо проверить, какой сетевой контроллер стоит в вашем компьютере и какой модуль ядра за него отвечает. Соответствующую информацию в таких случаях выдает команда `lspci`:

```
root# lspci | grep -i net
02:01.0 Ethernet controller: Intel Corporation 82540EP Gigabit Ethernet
                             Controller (Mobile) (rev 03)
...
```

Это означает, что в ноутбуке используется контроллер Gigabit-Ethernet 82540EP от Intel. Далее остается только найти для этого контроллера подходящий драйвер (то есть модуль ядра из каталога `/lib/modules/`*n.n*`/net/*`). Если поискать в Интернете по запросу `linux kernel module 82540EP`, то быстро найдется нужный модуль ядра e1000:

```
root# modinfo e1000
filename:    /lib/modules/2.6.28-11-generic/kernel/drivers/net/e1000/e1000.ko
version:     7.3.21-k3-NAPI
license:     GPL
description: Intel(R) PRO/1000 Network Driver
author:      Intel Corporation, <linux.nics@intel.com>
srcversion:  3472A241AA7F05737E0D2F3
...
```

С помощью команды `lsmod` можно проверить, загрузился ли модуль. Как правило, такое предположение подтверждается — это значит, что Linux правильно распознала контроллер уже при запуске системы. Если же вы получите другой результат, вам понадобится загрузить с помощью `modprobe` подходящий модуль:

```
root# modprobe e1000
```

Команда dmesg показывает, не возникло ли при загрузке модуля ошибок (в данном случае ошибок не было). Предупреждение link is not ready всего лишь означает, что интерфейс пока не сконфигурирован и поэтому не активен.

```
root# dmesg -c
...
Intel(R) PRO/1000 Network Driver - version 7.3.21-k3-NAPI
Copyright (c) 1999-2006 Intel Corporation.
e1000 0000:02:01.0: PCI INT A -> Link[LNKA] -> GSI 11 (level, low) -> IRQ 11
e1000: 0000:02:01.0: e1000_probe: (PCI:33MHz:32-bit) 00:11:25:32:4f:5d
e1000: eth0: e1000_probe: Intel(R) PRO/1000 Network Connection
ADDRCONF(NETDEV_UP): eth0: link is not ready
...
```

Чтобы в дальнейшем модуль загружался автоматически, занесите взаимосвязь между интерфейсом eth0 и модулем ядра e1000 в файл конфигурации модулей (см. раздел 15.1):

```
# Файл конфигурации модулей /etc/modprobe.conf или /etc/modprobe.d/aliases
alias eth0 e1000
```

Настройка параметров контроллеров

Обычно сетевой контроллер сам распознает параметры, необходимые для обмена данными по сети. Лишь очень редко приходится вручную настраивать такие параметры, как скорость, дуплексный режим и т. д. В подобных ситуациях вам пригодится команда ethtool (см. man ethtool).

Активизация интерфейса

Активизируйте сетевой интерфейс с помощью команды ifconfig:

```
root# ifconfig eth0 up
```

Если модуль ядра не загружен либо загружен неподходящий модуль, выводится сообщение об ошибке eth0: unknown interface: No such device (Неизвестный интерфейс. Такого устройства не существует). Более подробную информацию об успешном или неудачном выполнении ifconfig дает команда dmesg:

```
root# dmesg -c
e1000: eth0: e1000_watchdog: NIC Link is Up 1000 Mbps Full Duplex,
  Flow Control: RX/TX
ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
eth0: no IPv6 routers present
```

Конфигурирование интерфейса

Чтобы задать конфигурацию сетевого интерфейса, сообщите ifconfig его название (eth0) и нужный IP-адрес. Если затем повторно выполнить команду, не указывая адреса, то она отобразит всю имеющуюся в системе информацию о сетевом интерфейсе:

```
root# ifconfig eth0 192.168.0.2
root# ifconfig eth0
```

```
eth0   Protocol:Ethernet Hardware Address 00:11:25:32:4F:5D
       inet Address:192.168.0.2 Bcast:192.168.0.255 Mask:255.255.255.0
       inet6 Address: fe80::211:25ff:fe32:4f5d/64 scope:connection
       UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
       RX packets:82 errors:0 dropped:0 overruns:0 frame:0
       TX packets:49 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 Send queue size:100
       RX bytes:9252 (9.0 KiB) TX bytes:7732 (7.5 KiB)
       Basic address:0x8000 memory:c0220000-c0240000
```

Теперь с помощью команды `ping` можно проверить, разрешено ли вам связаться с другими компьютерами по сети. Благодаря параметру `-c 2` отсылается ровно два пакета `ping`:

```
root# ping -c 2 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=2.95 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.169 ms
--- 192.168.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.169/1.560/2.952/1.392 ms
```

Конфигурация сервера имен

Пока команда `ping` исправно функционирует лишь при условии, что вы правильно укажете IP-адрес. Чтобы можно было находить компьютер и по названию, в файле `/etc/resolv.conf` должен содержаться IP-адрес сервера имен. В следующем примере предполагается, что в локальной сети есть отдельный сервер имен с IP-адресом 192.168.0.1. Но сервер имен может быть и за пределами сети, если его предоставляет интернет-провайдер (этот вариант конфигурации подробно рассмотрен в разделе 16.5).

```
# /etc/resolv.conv
nameserver 192.168.0.1
```

Шлюз, задаваемый по умолчанию

Пока вы можете обмениваться пакетами только в пределах локальной сети. Чтобы обеспечить и выход в Интернет, компьютер должен «знать», куда направлять такие пакеты. Для этого вам следует указать адрес интернет-шлюза вашей сети с помощью команды `route`. В следующем примере IP-адрес шлюза — 192.168.0.1:

```
root# route add default gw 192.168.0.1
root# route -n
Kernel IP routing table
Destination  Gateway      Genmask       Flags  Metric  Ref  Use  Iface
192.168.0.0  0.0.0.0      255.255.255.0 U      0       0    0    eth0
0.0.0.0      192.168.0.1  0.0.0.0       UG     0       0    0    eth0
```

Теперь вы сможете посылать пакеты по любым адресам в Интернете:

```
root# ping -c 2 yahoo.com
PING yahoo.com (216.109.112.135) 56(84) bytes of data.
```

```
64 bytes from w2.rc.vip.dcn.yahoo.com (216.109.112.135): icmp_seq=1 ttl=52 time=116 ms
64 bytes from w2.rc.vip.dcn.yahoo.com (216.109.112.135): icmp_seq=2 ttl=52 time=115 ms
--- yahoo.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 115.397/115.807/116.217/0.410 ms
```

Получение информации о DHCP

Если в сети есть DHCP-сервер, вы можете воспользоваться им при конфигурации. После активизации интерфейса (указывается `ifconfig eth0 up` без дополнительных данных) выполните в Debian и Ubuntu команду `dhclient3`:

```
root# dhclient3 eth0
...
Listening on LPF/eth0/00:11:25:32:4f:5d
Sending on LPF/eth0/00:11:25:32:4f:5d
Sending on Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 3
DHCPOFFER from 192.168.0.1
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.15 -- renewal in 36624 seconds.
```

В SUSE для достижения той же цели требуется команда `dhcpcd`:

```
root# dhcpcd eth0
```

В Red Hat и Fedora, равно как и в Debian и Ubuntu, для конфигурации клиентов DHCP используется `dhclient`. В любом случае исходная конфигурация этой команды не позволяет вызывать ее вручную. В этих дистрибутивах `dhclient` можно использовать только в рамках процесса Init-V, когда сценарий настраивает конфигурацию сети автоматически.

Деактивизация интерфейса

Чтобы снова деактивизировать интерфейс, выполните `ifconfig` с параметром `down`:

```
root# ifconfig eth0 down
```

Управление несколькими контроллерами

Во многих компьютерах содержится по нескольку сетевых контроллеров. Система udev обеспечивает правильное соотнесение оборудования (то есть контроллеров) и названий интерфейсов (`eth0`, `eth1` и т. д.), так что каждый контроллер всегда связывается с одним и тем же интерфейсом (см. подраздел «Взаимное соотнесение контроллеров и сетевых интерфейсов» раздела 16.5).

Активизация контроллера WLAN

На первом этапе нужно загрузить модуль, подходящий для работы с вашим адаптером WLAN. В идеальном случае модуль загрузится автоматически уже при за-

пуске компьютера. Чтобы узнать, удалось ли системе автоматически распознать контроллер, просмотрите результат выполнения команды `iwconfig`. Она выдает информацию по всем доступным адаптерам WLAN. Следующий вывод был получен на уже не новом ноутбуке Centrino, где применяется контроллер Intel PRO Wireless 2100. Система распознала контроллер автоматически.

```
root# iwconfig
lo      no wireless extensions.   (Петлевой интерфейс)
eth0    no wireless extensions.   (Интерфейс LAN)
irda0   no wireless extensions.   (Инфракрасный интерфейс)
pan0    no wireless extensions.   (Интерфейс Bluetooth)
eth1    unassociated ESSID:off/any Nickname:"ipw2100"
        Mode:Managed Channel=0 Access Point: Not-Associated
        Bit Rate:0 kb/s Tx-Power:16 dBm
        Retry short limit:7 RTS thr:off Fragment thr:off
        Encryption key:off
        Power Management:off
        Link Quality:0 Signal level:0 Noise level:0
        Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
        Tx excessive retries:0 Invalid misc:2 Missed beacon:0
```

Если такой список результатов выполнения `iwconfig` отсутствует, вам потребуется самостоятельно загрузить необходимые модули с помощью команды `modprobe`. Выполнив `dmesg` или просмотрев `/var/log/messages`, вы увидите, нормально ли прошла загрузка модуля. При работе с модулем ipw2100 сообщения ядра выглядят так:

```
root# dmesg | grep ipw2100
ipw2100: Intel(R) PRO/Wireless 2100 Network Driver, git-1.2.2
ipw2100: Copyright(c) 2003-2006 Intel Corporation
ipw2100 0000:02:02.0: PCI INT A -> Link[LNKC] -> GSI 11 (level, low) -> IRQ 11
ipw2100: Detected Intel PRO/Wireless 2100 Network Connection
ipw2100 0000:02:02.0: firmware: requesting ipw2100-1.3.fw
```

Если вы не знаете, какой контроллер WLAN установлен в вашей системе, выполните команду `lspci` (или `lspcmcia`, если работаете с картой PCMCIA):

```
root# lspci
...
02:02.0 Network controller: Intel Corporation PRO/Wireless LAN 2100 3B Mini
                            PCI Adapter (rev 04)
```

Название интерфейса

Некоторые модули ядра, отвечающие за WLAN, предоставляют WLAN-интерфейс не под названием `ethn`, а как `wlann` или `athn` (таков, например, драйвер MadWifi для контроллера Atheros). Подобный метод позволяет снизить риск того, что нумерация сетевых интерфейсов собьется при подключении нового адаптера WLAN.

В принципе измененная номенклатура незначительно влияет на работу системы. Во всех следующих командах также нужно изменить `eth1` на `wlan0` или `ath0`.

Конфигурация WLAN

Чтобы карта WLAN могла обмениваться информацией с точкой доступа, вам, как правило, потребуется настроить три параметра: режим, последовательность символов SSID и ключ. Режим и SSID настраивается с помощью команды `iwconfig`:

```
root# iwconfig eth1 mode managed
root# iwconfig eth1 essid wlan-sol
```

WEP. Дальнейший ход работы зависит от того, как вы обезопасите доступ WLAN. Наиболее просто (и наименее надежно) устроен метод WEP: при его использовании вы сообщаете шестнадцатеричный ключ команде `iwconfig`:

```
root# iwconfig eth1 key c8192b13adf4ee58309953eebe
```

Если все пройдет удачно, то `dmesg -c` выдаст несколько новых сообщений ядра, которые будут выглядеть примерно так:

```
root# dmesg -c
ieee80211_crypt: registered algorithm 'WEP'
ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
eth1: no IPv6 routers present
```

WPA. Работа WPA или WPA2 построена несколько сложнее. В данном случае в ходе инициализации соединения и дальнейшего обмена информацией применяются постоянно изменяющиеся ключи, генерируемые фоновой программой `wpa_supplicant` из одноименного пакета. Установив этот пакет, создайте конфигурационный файл с названием `/etc/wpa_supplicant.conf`.

В этом файле содержатся некоторые глобальные настройки, а также специфические параметры для конкретных сетей WLAN. В следующем примере показан простейший вариант конфигурации, которого достаточно для соединения компьютера с точкой доступа или WLAN-роутером методом персонального шифрования WPA или WPA2. Два самых важных параметра — это `ssid` для идентификации сети и `psk`, содержащий ключ, повторно зашифрованный в целях безопасности (ключ WPA также можно указать в кавычках обычным текстом).

```
# /etc/wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
network={
    ssid="sol"
    psk=00a38f42e6681596e1a5a4c5ede9a15250fb2a01c21028c6d490bb3458b8ea00
}
network={
    ssid="wlan-sol2"
    psk=053633deb59038da9e9168e015fef97d3d54ae3794d4a12d31ee75a830cccec2
}
```

При шифровании WPA-пароля, состоящего из нескольких слов, поможет команда `wpa_passphrase`. Результат ее выполнения можно скопировать прямо в файл `wpa_supplicant.conf`, после чего по возможности следует удалить ту строку, в которой пароль записан обычным текстом.

```
root# wpa_passphrase sol 'Мой секретный-пресекретный пароль!'
network={
    ssid="sol"
    #psk="Мой секретный-пресекретный пароль!"
    psk=020d93e2ddb2cdee51e800b977ff7d58fde47d0913cd394f2133648a147f513f
}
```

Теперь можно запустить `wpa_supplicant`. Команда будет работать, пока вы не завершите ее, нажав Ctrl+C. Эта команда отвечает за инициализацию WLAN-соединения, а в дальнейшем — за регулярное обновление ключа к соединению. Другими словами, пока вы работаете с WLAN, должна работать и эта программа. Продолжайте работу в другой консоли.

Еще несколько замечаний о параметрах команды: `-i` указывает сетевой интерфейс, `-c` — конфигурационный файл (его название можете выбрать на свое усмотрение); `-D` указывает, какой драйвер WLAN вы используете. Для начала попробуйте поработать с wext — это общий интерфейс WLAN, поддерживаемый многими драйверами. Только если этот интерфейс не заработает, прямо укажите свой драйвер, например, `-D madwifi`. Справка `man wpa_supplicant` выдает список всех поддерживаемых драйверов:

```
root# wpa_supplicant -i eth1 -D wext -c /etc/wpa_supplicant.conf
Trying to associate with 00:13:46:b5:25:6e (SSID='sol' freq=0 MHz)
Associated with 00:13:46:b5:25:6e
WPA: Key negotiation completed with 00:13:46:b5:25:6e [PTK=TKIP GTK=TKIP]
CTRL-EVENT-CONNECTED - Connection to 00:13:46:b5:25:6e completed (auth) [id=0
id_str=]
...
```

Подробная документация по `wpasupplicant` имеется на сайтах `man` по `wpa_supplicant` и `wpa_supplicant.conf`, в файле `README` (каталог, в котором находится этот файл, от дистрибутива к дистрибутиву может отличаться, например, это может быть каталог `/usr/share/doc/wpasupplicant/`), а также на следующем сайте: http://hostap.epitest.fi/wpa_supplicant/.

Рекомендую также почитать следующий сайт, ориентированный на работу с Ubuntu: http://wiki.ubuntuusers.de/WLAN/wpa_supplicant.

Конфигурация сети

Теперь нам предстоит подключить интерфейс WLAN к сети. Для этого, как и при работе с интерфейсами LAN, используется команда `ifconfig`. Если в сеть необходимо интегрировать компьютер с IP-адресом 192.168.0.12, команда будет выглядеть так:

```
root# ifconfig eth1 up 192.168.0.12
```

С помощью `ping` можно проверить, работает ли соединение с роутером WLAN или точкой доступа (для этого нужно знать IP-адрес роутера или точки доступа соответственно). Чтобы иметь возможность выходить в Интернет, требуется также настроить адрес шлюза и сервер имен — эти вопросы рассматривались в начале этого раздела.

Определение статуса WLAN. Команда `iwconfig` в обобщенном виде выдает важнейшие данные, касающиеся всех интерфейсов WLAN:

```
root#  iwconfig eth1
eth1  IEEE 802.11b ESSID:"wlan-sol2" Nickname:"ipw2100"
      Mode:Managed Frequency:2.462 GHz Access Point: 00:16:B6:9D:FF:4B
      Bit Rate=11 Mb/s Tx-Power:16 dBm
      Retry short limit:7 RTS thr:off Fragment thr:off
      Encryption key:6C23-CB3C-EB50-97A9-1884-0128-C42C-80E4 Security mode:open
      Power Management:off
      Link Quality=86/100 Signal level=-72 dBm
      Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
      Tx excessive retries:0 Invalid misc:4 Missed beacon:0
```

Общая информация о текущем качестве WLAN-соединения также содержится в псевдофайле `/proc/net/wireless`:

```
root#  cat /proc/net/wireless
Inter-| sta-| Quality          | Discarded packets          | Missed | WE
 face | tus | link level noise | nwid crypt frag retry misc | beacon | 22
 eth1:  0020    91. 189.     0.     0     0    0     0    4        0
```

Иногда бывает полезна команда `iwlist`: она позволяет определить возможные настройки для различных предлагаемых параметров, а также вариант, активный в настоящий момент. В двух следующих примерах показан активный частотный канал и сети WLAN, находящиеся в зоне досягаемости:

```
root# iwlist eth1 channel
eth1   14 channels in total; available frequencies:
       Channel 01 : 2.412 GHz
       Channel 02 : 2.417 GHz
       ...
       Channel 12 : 2.467 GHz
       Channel 13 : 2.472 GHz
       Current Channel=10
root# iwlist eth1 scan
eth1   Scan completed :
       Cell 01 - Address: 00:13:46:B5:25:6E
                 ESSID:"sol"
                 Protocol:IEEE 802.11bg
                 Mode:Master
                 Channel:6
                 Encryption key:on
                 ...
       Cell 02 - Address: 00:16:B6:9D:FF:4B
                 ESSID:"wlan-sol2"
                 ...
```

16.5. Конфигурационные файлы LAN

В этом разделе рассмотрены самые важные конфигурационные файлы, нужные для подключения компьютера к локальной сети. К сожалению, не для всех этих

файлов существуют унифицированные правила, которые использовались бы во всех дистрибутивах. Что касается остальных файлов, то в этом разделе они будут описаны только для дистрибутивов Fedora 12, openSUSE 11.2 и Ubuntu 9.10. Как правило, не следует вносить изменения непосредственно в конфигурационные файлы — лучше использовать инструменты, предусмотренные в вашем дистрибутиве для изменения конфигурации. Рассмотренные в этом разделе конфигурационные файлы, специфичные для отдельных дистрибутивов, важны при работе только в том случае, если вы не используете программу Network Manager!

Для всех примеров, приведенных в данном разделе, действуют следующие правила: конфигурируемый компьютер называется uranus, а его домен — sol. Другие компьютеры в локальной сети называются jupiter, saturn и т. д. В локальной сети используются адреса 192.168.0.*. IP-адрес локального компьютера — 192.168.0.2. Компьютер-шлюз этой локальной сети имеет IP-адрес 192.168.0.1. На компьютере-шлюзе действует собственный сервер имен. Разумеется, названия и адреса приведены только для примера.

Базовая конфигурация

Файл /etc/hosts

В /etc/hosts содержится список известных IP-адресов и соответствующих им имен. В этом файле обязательно должны храниться данные о петлевом интерфейсе. Соответствующая запись, как правило, выглядит так:

```
# /etc/hosts
127.0.0.1 localhost       # Петлевой интерфейс компьютера
...
```

В большинстве дистрибутивов Linux вместо 127.0.0.1 допускается запись ::1, по правилам IPv6. В Red Hat и Fedora в строке localhost содержится дополнительная запись localhost.localdomain. Эту строку не следует изменять:

```
# /etc/hosts в Red Hat и Fedora
::1 l     localhost.localdomain localhost
...
```

В некоторых дистрибутивах в hosts также содержится запись о локальном компьютере. Если установленный на нем сетевой контроллер конфигурирован с применением статического IP-адреса, то этот адрес необходимо указать. Если же, напротив, сетевой контроллер динамически получает адрес, обращаясь к DHCP, в /etc/hosts указывается псевдоадрес 127.0.1.1 (например, в Debian, Ubuntu) либо такая запись отсутствует (например, в Red Hat, Fedora).

```
# /etc/hosts (Продолжение)
...
192.168.0.2 uranus.sol uranus   # Статический IP-адрес локального компьютера
```

Если вы по имени запрашиваете другой компьютер, расположенный в локальной сети и в этой сети нет сервера имен (см. раздел 17.4), то его название также

необходимо указать в /etc/hosts. Вместо ping 192.168.0.13 вы в данном случае можете выполнить просто ping saturn, чтобы проверить связь с компьютером saturn.

```
# /etc/hosts (Продолжение)
...
192.168.0.1 mars.sol mars       # IP-адреса и имена других
192.168.0.2 uranus.sol uranus   # компьютеров в локальной сети
192.168.0.3 saturn.sol saturn
```

Аналогичные записи должны быть в файлах /etc/hosts всех компьютеров, находящихся в локальной сети. Если речь идет об администрировании очень большого количества компьютеров, то управление файлами /etc/hosts значительно усложняется. Поэтому в больших сетях рекомендуется создавать на одном из компьютеров сервер имен (см. раздел 17.4). Этому компьютеру (то есть серверу имен) известно, как называются все остальные компьютеры в сети. Компьютеры, находящиеся в локальной сети, могут обращаться к серверу имен, чтобы узнать эту информацию. Теперь /etc/hosts можно сократить до localhost. В любом случае требуется, чтобы файл /etc/resolv.conf был сконфигурирован правильно (этот вопрос рассмотрен чуть ниже).

В некоторых дистрибутивах в /etc/hosts содержатся специальные адреса IPv6, например, fe00::0. Эти адреса имеют значение лишь в том случае, если вы пользуетесь IPv6.

Файл /etc/host.conf

В файле /etc/host.conf указано, как протокол TCP/IP должен узнавать неизвестные IP-адреса. В следующем примере определяется, что сначала интерпретируется файл /etc/hosts (ключевое слово hosts), а затем посылается запрос на сервер имен, указанный в /etc/resolv.conf (bind). Вторая строка позволяет присвоить каждому отдельному хост-имени, заданному в /etc/hosts, несколько IP-адресов.

Этот файл почти во всех дистрибутивах построен именно так, как показано ниже, и изменять его нельзя.

```
# /etc/host.conf
order hosts, bind
multi on
```

Файл /etc/resolv.conf

Файл управляет тем, как определяются IP-адреса для неизвестных сетевых имен (хост-имен). «Неизвестно» означает, что названия определяются не в hosts.conf.

Применяя ключевые слова domain и search, вы можете дополнять обычные (неполные) названия компьютеров (например, jupiter) доменным именем (до jupiter.sol). Во-первых, это облегчает работу, так как вы можете указывать локальные хост-имена в сокращенной форме. Для search можно задать несколько доменных имен (для domain — только одно); поэтому имя domain имеет приоритет над search и проверяется в первую очередь. Если здесь будет указано лишь одно доменное имя, то строку domain можно опустить.

Важнейшие записи в /etc/resolv.conf вводятся ключевым словом nameserver: таким образом, вы можете указать до трех IP-адресов серверов имен. Эти сер-

веры каждый раз запрашиваются, когда необходимо узнать IP-адрес компьютера с неизвестным именем (например, www.yahoo.com). В таком случае нужно обязательно указывать сервер имен, чтобы интернет-адреса можно было преобразовать в IP-адреса.

ПРИМЕЧАНИЕ

Как частный пользователь вы получаете IP-адрес DNS вашего провайдера интернет-услуг. На большинстве ADSL-роутеров работает локальная служба DNS, которая, в свою очередь, обращается к DNS провайдера. В больших локальных сетях обычно имеется свой сервер имен, значит, вы можете узнать IP-адрес у системного администратора.

```
# /etc/resolv.conf
domain sol               # Хост-имена для домена .sol
search sol               # Хост-имена для домена .sol
nameserver 192.92.138.35 # Первая DNS
nameserver 195.3.96.67   # Вторая DNS (на случай отказа первой)
```

Файл `resolv.conf` создается динамически и зависит от конфигурации конкретной сети:

- если соединение с Интернетом устанавливается с помощью PPP (через модем, ISDN, ADSL, UMTS, VPN), то сценарий, устанавливающий соединение, автоматически вносит адреса `nameserver` вашего интернет-провайдера в `/etc/resolv.conv`;
- если ваше соединение по локальной сети (LAN, WLAN) конфигурируется с применением DHCP, то сценарий, устанавливающий соединение, записывает адреса `nameserver`, переданные с DHCP-сервера.

Защита resolv.conf от изменений

В большинстве случаев целесообразно применять автоматическую настройку файла `resolv.conf`. Однако если вам это не нужно, то автоматическое изменение в большинстве случаев можно отменить.

При применении в Debian и Ubuntu PPP-соединений нужно удалить ключевое слово `usepeerdns` из `/etc/ppp/peers/name`. Что касается сетевых интерфейсов DHCP, ситуация зависит от того, какой DHCP-клиент у вас установлен. Если установлен dhcp3-клиент, вам нужно изменить его конфигурационный файл `dhclient.conf`:

```
# /etc/dhcp3/dhclient.conf
...
supersede domain-name "sol";
prepend domain-name-servers 192.168.0.1;
```

В Red Hat и Fedora в таком случае необходимо изменить файл `ifcfg-xxx` для соответствующего интерфейса:

```
# /etc/sysconfig/network-scripts/ifcfg-xxxx (Red Hat, Fedora)
PEERDNS=no
```

В SUSE измените следующий конфигурационный файл:

```
# /etc/sysconfig/network/config (SUSE)
NETCONFIG_DNS_POLICY=""
```

Конфигурация шлюза

Для этой операции не существует единого стандарта, например, особо не указывается, в каком файле выполняется конфигурация шлюза. В локальных сетях адрес шлюза обычно определяется через DHCP. При статической конфигурации эта задача решается в различных файлах, в зависимости от дистрибутива.

В Debian и Ubuntu в файле `/etc/network/interfaces` описаны все сетевые интерфейсы. Если конфигурация интерфейса является статической, то шлюз настраивается с помощью ключевого слова `gateway`:

```
# в /etc/network/interfaces (Debian, Ubuntu)
...
iface eth0 inet static
  address 192.168.0.2
  netmask 255.255.255.0
  gateway 192.168.0.1
```

В Red Hat и Fedora в конфигурационном файле содержится переменная `GATEWAY`, отвечающая за сетевой интерфейс:

```
# /etc/sysconfig/network-scripts/ifcfg-xxxx (Red Hat, Fedora)
GATEWAY=192.168.0.1
```

В SUSE конфигурация централизованно осуществляется в следующем файле:

```
# in /etc/sysconfig/network/routes (SUSE)
default 192.168.0.1 - -
```

Фактическая настройка шлюза во всех дистрибутивах производится одинаково: с помощью команды `route`. На примере двух следующих команд показано, как вручную добавить в таблицу маршрутизации шлюз 192.168.0.1, а затем снова удалить его из таблицы:

```
root# route add default gw 192.168.0.1
root# route del default gw 192.168.0.1
```

Конфигурация хост-имен

Актуальное хост-имя можно узнать с помощью команды `hostname`. Если хост-имя настраивается без применения DHCP, то конфигурация в Debian и Ubuntu производится в файле `/etc/hostname`, а в SUSE — в файле `/etc/HOSTNAME`. В Red Hat и Fedora хост-имя настраивается с помощью одноименной переменной (`HOSTNAME`) в файле `/etc/sysconfig/network`. Не забудьте, что еще нужно настроить `/etc/hosts`, если в этом файле есть строка с хост-именем компьютера.

Взаимное соотнесение контроллеров и сетевых интерфейсов

При использовании сетевых интерфейсов обычно бывает сложно определить взаимосвязи между устройствами `eth`*n* и физическим оборудованием. При работе с некоторыми устройствами можно применить команду `ethtool -p eth0 10`, которая заставляет в течение 10 секунд мерцать светодиод, встроенный в гнездо того или

иного устройства. Если сетевой драйвер не поддерживает такую операцию, вы получите сообщение об ошибке Operation not supported (Операция не поддерживается).

В большинстве современных дистрибутивов Linux за соотнесение сетевых контроллеров и интерфейсов отвечает система udev (см. раздел 3.10). Если точнее, то названиями сетевых интерфейсов управляет файл `net_persistent_names.rules`. Этот файл может выглядеть, например, так:

```
# Файл /etc/udev/rules.d/70-persistent-net.rules
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:16:17:cd:c3:81",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth0"
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:14:6c:8e:d9:71",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth1"
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:4f:4e:0f:8e:a0",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth2"
```

16.6. Zeroconf и Avahi

В этой книге я обычно исхожу из того, что либо вы конфигурируете компьютер в своей сети самостоятельно, либо берете конфигурацию IP с центрального роутера или DHCP-сервера. Но есть еще и третий способ — автоматическая конфигурация с помощью Zeroconf.

При применении этого метода все компьютеры, расположенные в сети, обмениваются своими данными о конфигурации. Новые компьютеры или устройства, подключаемые к сети, сами конфигурируются на основании этой информации так, что могут обмениваться информацией с другими устройствами без возникновения конфликтов. Компьютеры, которые конфигурируются автоматически, используют IP-адреса из диапазона 169.254.*.* и хост-имена, оканчивающиеся на `.local`. Обмен информацией Zeroconf производится через UDP-порт 5454. Чтобы Zeroconf работал, этот порт в пределах сети WLAN не должен блокироваться брандмауэром!

Zeroconf впервые был применен под названием Rendezvous компанией Apple; позже этот проект переименовали в Bonjour, и теперь им также можно пользоваться в Windows. Хотя данное решение и предоставляется в виде открытого кода, оно не соответствует правилам лицензии GPL. Поэтому в Linux развился собственный проект Zeroconf, называемый Avahi — его код не зависит от Bonjour. Проект Avahi работает по лицензии LGPL. Откуда взялось странное название Avahi, мне не известно[1].

Программы, совместимые с Zeroconf, могут отображать все остальные компьютеры из данной сети, работающие с Zeroconf, а также их ресурсы (например, сетевые каталоги, SSH, серверы HTTP и FTP). Таким образом, не настраивая конфигурацию отдельно, вы можете интегрировать в сеть два и более компьютера и осуществлять обмен данными между ними.

Остается ждать, насколько приживется эта концепция. Что касается автоматической конфигурации IP-адресов, то в условиях бурного распространения (ADSL)-роутеров функциональность Zeroconf, по существу, уже не нужна. Причина,

[1] Авахи — это название древесного лемура, мордочка которого является символом проекта (http://en.wikipedia.org/wiki/Woolly_lemur). *(Примеч. перев.)*

по которой пакеты Avahi все же по умолчанию устанавливаются почти во всех дистрибутивах Linux, заключается скорее в возможностях обзора. Тот факт, что компьютеры «видят» друг друга и «знают по именам» и такой обзор совершенно не зависит от вида конфигурации сети, является очень важным достоинством. Более подробную информацию и советы по этой теме можно найти на сайтах http://avahi.org и http://wiki.ubuntuusers.de/Avahi.

Демон avahi

За обмен информацией между компьютерами, на которых установлена программа Avahi, отвечает служба `avahi-daemon`. Ее конфигурация производится в файле `/etc/avahi/avahi-daemon.conf`, причем основные настройки обычно можно оставить без изменения. Единственным исключением является переменная `enable-dbus`: она определяет, задействует ли Avahi механизм обмена данными. Для работы некоторых Avahi-совместимых программ требуется DBUS (шина передачи данных). Чтобы активизировать DBUS, измените файл `avahi-daemon.conf` следующим образом:

```
# /etc/avahi/avahi-daemon.conf
[server]
...
enable-dbus=yes
```

Затем повторно запустите службу:

```
root# /etc/init.d/avahi-daemon restart
```

Преобразование имен в IP-адреса

Если вы хотите обратиться к внешним компьютерам, на которых работают обычные сетевые программы, не совместимые с Avahi (например, `ping merkur.local`), нужно установить еще один сетевой демон с помощью `avahi-dnsconfd` и запустить его. В данном случае мы имеем дело со своего рода сервером имен для работы с хост-именами Avahi (если в вашей сети и так работает сервер имен, то этот демон вам не потребуется).

Чтобы при преобразовании имен в IP-адреса все программы обращались к `avahi-dnsconfd`, убедитесь, что у вас установлена библиотека `libnss-mdns` и в строке `hosts:` файла `/etc/nsswitch.conf` есть ключевое слово `mdns4`. Как правило, такие настройки используются по умолчанию.

```
# в /etc/nsswitch.conf
...
hosts: files dns mdns4
...
```

Просмотр сети и ресурсов

Завершив эти подготовительные работы, вы можете проверить, о каких компьютерах или службах, находящихся в вашей сети, известно Avahi. Для этого использу-

ется команда консоли `avahi-browse -a -t` или ее графический аналог `avahi-discovery` (рис. 16.4). По умолчанию программа Nautilus в режиме просмотра сети отображает все компьютеры, на которых работает Avahi. В Konqueror по адресу zeroconf:/ расположена схожая функция, при условии, что вы установили соответствующее дополнение (пакет зависит от дистрибутива, например `kde-zeroconf`). Мультимедийные приложения и программы для рассылки сообщений также поддерживают Zeroconf.

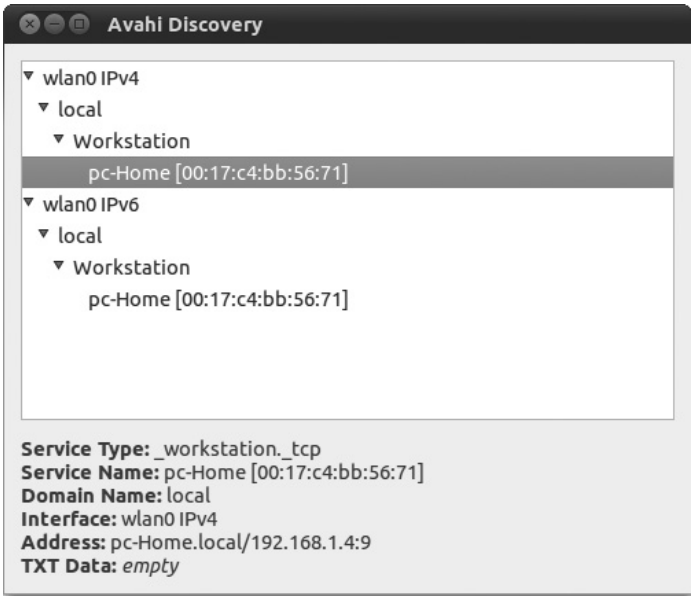

Рис. 16.4. Просмотр ресурсов сети с помощью программы Avahi Discovery

16.7. Основы PPP

Протокол точка-точка (PPP) обеспечивает соединение TCP/IP между двумя компьютерами через последовательный интерфейс. Протокол применяется при работе с аналоговыми модемами, а также модемами ISDN, ADSL и UMTS. В Linux за соединение по протоколу PPP отвечает программа `pppd`. Она может использоваться как на клиентском компьютере, так и на сервере, но здесь будет рассмотрена только работа с клиентом. В данном разделе делается введение в основы PPP. В следующих разделах мы рассмотрим конкретные способы применения PPP при различных видах доступа к Интернету.

Варианты PPP

Программа `pppd` предназначена только для работы с PPP-соединениями, устанавливаемыми через последовательную телефонную линию, то есть через аналоговый

модем. Для работы с ADSL-соединениями применяется три усовершенствованных варианта протокола: PPPoE, PPPoA или PPTP.

- **PPPoE.** Протокол двухточечного соединения через Ethernet — общедоступный документированный протокол, данные по которому содержатся в RFC 2516. Его основной недостаток заключается в том, что он ограничивает максимальный размер пакетов (Проблема MTU, см. раздел 16.9).
- **PPPoA.** Протокол туннелирования типа точка-точка по ATM — альтернатива PPPoE. Здесь ATM означает «асинхронный режим передачи» (Asynchronous Transfer Mode).
- **PPTP.** Туннельный протокол типа точка-точка — протокол Microsoft, развитый из других стандартов, документация по которому также имеется в свободном доступе (RFC 2637). Первоначально он предназначался для обеспечения работы виртуальных частных сетей (см. раздел 18.5).

Аутентификация

Построение любого PPP-соединения начинается с аутентификации: это означает, что ваш компьютер должен быть опознан провайдером Интернета по логину и соответствующему паролю. Эта операция может осуществляться одним из двух способов: PAP и CHAP. CHAP, в свою очередь, имеет различные варианты (например, MS-CHAPv2), повышающие надежность соединения.

- **PAP.** Применяя протокол аутентификации пароля, клиент (то есть ваш компьютер) обычно передает логин и пароль в незашифрованном виде (также есть вариант PAP и с шифрованием).
- **CHAP.** При использовании протокола аутентификации по квитированию вызова сервер (то есть провайдер) при аутентификации отсылает клиенту так называемый «пакет challenge». Программа `pppd` использует эти данные, чтобы рассчитать из пароля хэш-значение. Затем `pppd` отсылает логин и хэш-значение обратно провайдеру (таким образом, необходимость передачи самого пароля отпадает).

Конфигурационные файлы и сценарии pppd

/etc/ppp/options. В `/etc/ppp/options` содержатся глобальные параметры `pppd`. Они устанавливаются по умолчанию для всех соединений, создаваемых `pppd`. Справка по важнейшим параметрам дается в следующем разделе.

Если вы используете `pppd` для создания различных интернет-соединений (например, через аналоговый модем или ADSL-модем), в `options` должно содержаться как можно меньше настроек. Чтобы настраивать параметрами, пользуйтесь специальными файлами для работы с соединениями, находящиеся в `/etc/ppp/peers/`! Таким образом, вы будете избавлены от проблемы, когда параметр, подходящий

для варианта соединения A, может заблокировать соединение, создаваемое по варианту B.

/etc/ppp/peers/name. В /etc/ppp/peers/name содержатся параметры, касающиеся конкретного типа соединения. Чтобы запустить pppd с применением этих параметров, выполните следующую команду:

```
root# pppd call имя
```

Настройки в /etc/ppp/peers/name имеют приоритет перед /etc/ppp/options.

/etc/ppp/pap-secrets и chap-secrets. Эти файлы содержат список всех логинов и паролей для аутентификации PAP и CHAP. Если вы не знаете точно, как осуществляется аутентификация — через PAP или через CHAP, просто вставьте одну и ту же запись и в ap-secrets, и в chap-secrets. Значительно более надежны и поэтому шире распространены варианты CHAP (CHAP, MS-CHAP или MSCHAPv2).

Записи для клиентских соединений выглядят так:

```
#/etc/ppp/pap-secrets и /etc/ppp/chap-secrets
#логин    IP-адрес сервера   Пароль         IP-адрес клиента
"hofer"   *                  "qwe44trE"     *
```

Вместо символа * между логином и паролем можно указать IP-адрес того PPP-сервера, с которым необходимо установить соединение. В таком случае информация о пароле действительна только для данного IP-адреса. Это дополнительный механизм защиты от злонамеренного использования, но применить его можно лишь в том случае, если IP-номер известен и не изменяется (при использовании ADSL-соединений по протоколу PPTP вы можете указать IP-адрес ANT).

Вместо второй звездочки вы можете указать, какому клиентскому IP-адресу должна соответствовать данная комбинация логина и пароля. Как правило, этот адрес неизвестен, так как при каждом новом соединении сервер присваивает клиенту новый IP-адрес — тот, что свободен в настоящий момент. Поэтому в предыдущем примере звездочка (которую можно и опустить) означает, что клиент может иметь любой IP-адрес.

Если pppd применяется не на клиенте, а на сервере, то в pap-secrets и chap-secrets не указываются и логины внешних клиентов. Пример конфигурации приведен в подразделе «Настройка PPTPD-сервера» раздела 18.6. Там рассказывается о создании VPN-сервера на базе PPTP.

/etc/ppp/ip-up и /etc/ppp/ip-down. Сценарные файлы /etc/ppp/ip-up и /etc/ppp/ip-down выполняются сразу же после установления либо завершения соединения. Они могут применяться для настройки /etc/resolv.conf или создания и изменения функций роутинга и маскарадинга, а также настроек брандмауэра.

Обоим сценариям сообщается по шесть параметров. Первый параметр содержит название интерфейса (например, ppp0), второй и третий не используются; четвертый включает локальный IP-адрес, пятый — IP-адрес партнера по PPP, шестой — последовательность символов, используемую для идентификации PPP-соединения (параметр ipparam).

Если при установлении соединения передаются адреса DNS (параметр usepeerdns), то оба адреса также предоставляются в переменных DNS1 и DNS2.

СОВЕТ

Обратите внимание, что сценарные файлы являются исполняемыми (chmod u+x)! Почти со всеми дистрибутивами поставляются заранее сконфигурированные файлы ip-up и ip-down.

В зависимости от дистрибутива некоторые изменения потребуется внести в каталогах ip-up.local и ip-down.local или в дополнительных сценарных файлах из каталогов ip-up.d и ip-down.d.

Параметры pppd

Программе `pppd` известны десятки параметров, настройка которых обычно производится в файлах `/etc/ppp/options` и `/etc/ppp/peers/name`. Полная справка по параметрам дается в файле `man pppd`. Ниже в алфавитном порядке перечислены и кратко описаны только самые важные параметры — и даже их набралось немало.

- `connect` *команда* — выполняется перед запуском `pppd`. При работе с аналоговыми модемами в данном случае обычно вызывается команда `chat`, предназначенная для того, чтобы набрать телефонный номер провайдера. Этот параметр применяется и в тех случаях, когда используется коммутируемое соединение по запросу (параметр `demand`). При ADSL-соединениях в качестве команды `connect` может использоваться просто `/bin/true`.
- `crtscts` — поток данных, идущий через последовательный интерфейс, контролируется RTS/CTS. Параметр `crtscts` применяется только при установлении соединений через модем.
- `debug` — процесс установления соединения подробно протоколируется посредством программы `syslogd` (то есть в файлах `/var/log/`*xxx* в зависимости от конфигурации `syslogd`).
- `defaultroute` — когда PPP-соединение установлено, IP-адрес задается как конечный адрес для пересылаемых пакетов. Этот параметр требуется почти всегда (если вы не настраиваете функции роутинга сами, например с помощью сценария `ip-up`).
- `demand` — соединение устанавливается не сразу, а лишь тогда, когда на самом деле необходимо передать данные. В таком случае выполняется сценарий, указанный в `connect` (`demand` также автоматически активизирует `persist`, если вы специально не отмените эту операцию параметром `nopersist`).
- `idle` *n* — соединение должно автоматически завершаться, если в течение *n* секунд не передаются никакие данные. Эта функция полезна на случай, если вы забыли отключить соединение, и экономит вам немало денег за телефон!
- `ipparam` — с помощью этого параметра можно указать последовательность символов. После установления соединения `pppd` передает эту последовательность символов сценарию `/etc/ppp/ip-up`. В некоторых дистрибутивах эта дополнительная информация используется для выбора нужного файла конфигурации сети (`/etc/sysconfig/network-scripts/ifcfg-ipparam`).
- `ktune` — позволяет `pppd` изменять настройки ядра. Такие изменения могут быть нужны при использовании *коммутируемого соединения по запросу*, чтобы пакет данных, инициирующий сетевое соединение, не потерялся.

- `lcp-echo-interval` *n* — каждые *n* секунд посылает провайдеру эхо-запрос. Таким образом проверяется, работает ли еще соединение.
- `lcp-echo-failure` *n* — указывает, после какого количества неудачных запросов `pppd` должна завершить соединение. В зависимости от настроек `persist`/`nopersist` работа `pppd` может быть завершена либо может быть предпринята новая попытка соединения.
- `linkname` *имя* — благодаря этому параметру в качестве файла идентификации процесса (PID-файла) используется имя `/var/run/ppp-`*имя*`.pid`. Иногда бывает нужно завершить определенный процесс из нескольких, работающих одновременно. В таком случае номер нужного процесса можно с легкостью считать из PID-файла (по умолчанию название PID-файла — `/var/run/pppn.pid`).
- `lock pppd` — создает блокирующий файл для интерфейса передачи данных (например, последовательного интерфейса при использовании модема). Таким образом, исключается ситуация, в которой к интерфейсу могут одновременно обратиться две и более программ.
- `mru` *n* и `mtu` *n* — устанавливает желаемые значения для *максимальной длины получаемых пакетов (MRU)* и *максимальной длины передаваемых пакетов (MTU)*. Величины MRU и MTU указывают размер блока пакета с файлами. Обычно для MRU действует настройка, задаваемая по умолчанию и равная `1500`. При работе с PPPoE это значение необходимо снизить до `1492`. Текущее значение можно проверить с помощью `ifconfig` после установления PPP-соединения.
- `name "abc"` — если `pppd` используется на клиентском компьютере, то он использует логин, например `abc` (соответствующий пароль можно узнать из файла PAP/CHAP).

  Если же `pppd` применяется на сервере, то `abc` — это имя локальной системы. Оно должно соответствовать данным, указанным в правом столбце файла PAP/CHAP (см. подраздел «Настройка PPTPD-сервера» раздела 18.6).
- `noauth` — удаленная станция `pppd`, то есть провайдер, также называемый в документации по `pppd "peer"` (одноранговый узел) в аутентификации не нуждается. Этот параметр требуется почти всегда. Однако вы сами (как клиент) обязательно должны аутентифицироваться!
- `noaccomp`, `nopcomp`, `novj`, `novjccomp`, `nobsdcomp`, `nodeflate`, `noccp` — деактивизируют всевозможные механизмы сжатия. При работе со многими провайдерами необходимо указать как минимум `nopcomp`. Зачастую рекомендуется применять и другие параметры. Если возникают проблемы, эти параметры не помешает как минимум опробовать.
- `nodetach` — PPP, в отличие от многих собратьев, запускается не как фоновый процесс. Вся информация об управлении выводится в окно терминала, в котором была запущена программа. Этот параметр хорошо подходит для тестирования системы.
- `noipdefault` — благодаря этому параметру провайдер (а не компьютер) назначает IP-адрес для PPP-соединения. Поскольку почти все интернет-провайдеры

распределяют IP-адреса динамически, этот параметр, как правило, должен применяться (иными словами, при каждом входе в систему вы получаете другой IP-адрес, а точнее первый IP-адрес, свободный в данный момент; метод аналогичен тому, что применяется при работе с DHCP-сервером в локальной сети — см. раздел 17.5).

- `nopersist` — параметр, обратный `persist`. Программа `pppd` завершает работу, если соединение обрывается (или по истечении срока, заданного параметром `idle`).
- `persist` — при ненамеренном обрыве соединения `pppd` автоматически пытается восстановить соединение. При использовании `demand` этот параметр применяется автоматически.
- `plugin` *имя*`.so` — программа `pppd` при запуске загружает PPP-модуль *имя*`.so`. Дополнительные модули PPP обычно находятся в каталоге `/usr/lib/pppd/n/`. Параметр задействуется в том случае, если используются специальные PPP-функции, реализованные в виде PPP-плагина (например, протоколы PPPoE, PPPoA и PPTP или специальные механизмы аутентификации и шифрования).
- `pty` *сценарий* — указывает сценарий или программу, которые используются для обмена информацией вместо устройства. Параметр необходим в том случае, если `pppd` должен использовать внешнюю программу, поддерживающую дополнительный протокол обмена информацией (например, PPPoE, PPPoA или PPTP).
- `refuse-pap`, `-chap`, `-mschap-v2` — благодаря этим параметрам не допускаются указанные методы аутентификации.
- `require-pap`, `-chap`, `-mschap-v2`, `-mppe`, `-mppe-128` — определяют, какой метод аутентификации будет применяться (PAP, CHAPT, Microsoft-CHAP Version 2) и как должны шифроваться данные (MPPE — протокол Microsoft шифрования двухточечного соединения). Если не указать ни один из этих параметров, то `pppd` сама выберет наилучший метод (в зависимости от требования удаленной станции).
- `usepeerdns` — многие PPP-серверы при установлении соединения передают два DNS-адреса. Благодаря параметру `usepeerdns` программа `pppd` узнает эти адреса при установлении соединения и передает сценарию `ip-up`. Во многих дистрибутивах конфигурация этого сценария такова, что адреса DNS автоматически заносятся в файл `/etc/resolv.conf`.

Еще некоторые параметры описаны в разделе 18.6, где рассказывается о конфигурации PPTP-сервера, обеспечивающего безопасность доступа к WLAN с помощью VPN.

16.8. Внутренняя организация UMTS

Современные модемы чаще всего выглядят как большие флешки. Хотя для них закрепилось обозначение UMTS, большинство модемов совместимо и с более ранними стандартами (GSM, GPRS, EDGE). Модем в зависимости от стандарта связи автоматически находит самую быструю доступную сеть.

Драйвер. UMTS-модемы хорошо поддерживаются в Linux, что не может не радовать. Если вы работаете с современным дистрибутивом, то большинство современных дистрибутивов опознаются системой сразу при подключении. Вы можете в этом убедиться с помощью команды `dmesg`:

```
root# dmesg
...
usb 1-4: new high speed USB device using ehci_hcd and address 3
...
USB Serial support registered for GSM modem (1-port)
option 1-4:1.0: GSM modem (1-port) converter detected
usb 1-4: GSM modem (1-port) converter now attached to ttyUSB0
option 1-4:1.1: GSM modem (1-port) converter detected
usb 1-4: GSM modem (1-port) converter now attached to ttyUSB1
usbcore: registered new interface driver option
option: v0.7.2:USB Driver for GSM modems
```

Если оборудование удачно определилось, то соединение с Интернетом тоже установится без проблем — используйте Network Manager или UMTSmon (см. разделы 16.1–16.2).

Внутри системы Linux большинство модемов воспринимаются как устройства для последовательной передачи данных. Некоторые устройства требуют для работы драйвер Nozomi из одноименного модуля ядра, другие модемы сразу же распознаются как оборудование для последовательной передачи данных. После этого модемом можно будет управлять через специальный файл-устройство (например, `/dev/ttyUSB`*n*, `/dev/ttyACM`*n* или `/dev/nozomi`*n*). С точки зрения Linux современный UMTS-модем ничем не отличается от аналогового модема, выпущенного 15 лет назад! Как и ранее, для работы требуются команды AT. Для управления некоторыми функциями, характерными для мобильной связи, разработаны новые AT-команды (например, AT+CPIN=*nnnn* для передачи PIN-кода).

Поэтому в принципе возможно использовать UMTS-модемы с программами gnome-ppp или KPPP, которые разрабатывались для управления аналоговыми модемами. Чтобы этот метод функционировал, нужно предварительно отключить запрос PIN-кода. Методом следует пользоваться лишь в крайнем случае.

Параметры соединения. Обычно в конфигурационных программах UMTS предусмотрены поля для ввода телефонного номера, последовательности символов APN, логина, пароля и PIN-кода. У большинства провайдеров роль телефонного номера играет последовательность символов `*99#`. Последовательность символов APN *(имя точки доступа)* зависит от провайдера и означает точку подключения к мобильной сети. У многих провайдеров поля для ввода логина и пароля можно оставлять пустыми, так как они могут не интерпретироваться.

Проблемы с PIN- и PUK-кодом. Если Network Manager или другие программы постоянно требуют от вас ввода PIN- и PUK-кода, возможны проблемы. Например, в настоящее время Network Manager может работать только с четырехзначными PIN/PUK-кодами, хотя с некоторыми современными моделями уже используются восьмизначные коды. Предполагается, что в ближайшем будущем эта проблема будет решена. Однако не увлекайтесь экспериментами! Не исключено, что SIM-карта заблокируется после ввода нескольких якобы неправильных PIN-кодов.

PIN-код SIM-карты вполне можно отключить. Проще всего это делается с помощью программы UMTSmon. Если у вас нет этой программы, достаньте SIM-карты из USB-модема, вставьте в открытый сотовый телефон, и с помощью телефона отключите запрос PIN-кода (например, Настройки ▸ Настройки безопасности ▸ Запрашивать PIN-код ▸ Вкл/Выкл). Затем снова вставьте SIM-карту в USB-модем. Теперь соединение с Интернетом возможно и без PIN-кода. Но не потеряйте ваш модем, так как теперь кто угодно сможет пользоваться им, не вводя PIN-код!

Ссылки. Более подробная информация и советы по работе с UMTS в Linux даны на следующих сайтах:

- http://umtsmon.sourceforge.net/docs/OLS.umts.paper.pdf;
- http://www.edv-tipps.at/hsdpa-modems.html;
- http://www.kuix.de/umts/vodafone/.

16.9. Основы ADSL

Сравнение модема и роутера. При доступе в Интернет с помощью ADSL существует два основных варианта соединения компьютера с Сетью: через ADSL-модем и ADSL-роутер.

- **ADSL-модем.** По сути, он является оконечным устройством сети ADSL (кратко — ATN). К одному ADSL-модему можно подключить только один компьютер. Обмен информацией между вашим компьютером и Интернетом осуществляется через USB-разъем или сетевой кабель. В зависимости от модема и конкретного провайдера при обмене информацией могут применяться протоколы PPPoE, PPPoA или PPTP.

  Из-за обилия вариантов конфигурация может оказаться сложной. Если вам повезет, то конфигурацию удастся выполнить с помощью Network Manager или инструментов, характерных для конкретных дистрибутивов (`pppoe-config` в Debian и Ubuntu), `system-config-network` в Fedora и Red Hat, YaST в SUSE). Если эти инструменты не помогут, то посмотрите подразделы о конфигурации ADSL-PPPoE и ADSL-PPTP, где рассказано, как настраивать соединение вручную.

- **ADSL-роутер.** Совмещает функции ADSL-модема и функции роутера или шлюза. К роутеру с помощью сетевых кабелей можно подключить несколько компьютеров. Компьютеры получают все конфигурационные файлы от ADSL-роутера через DHCP, поэтому отдельно настраивать каждый клиент не требуется. В некоторых ADSL-роутерах также имеется модуль WLAN. Если у вас есть выбор, обязательно работайте с ADSL-роутером! В следующем разделе даются советы о том, как правильно купить и настроить ADSL-роутер.

Дополнительная информация. На перечисленных далее сайтах имеется множество информации по теме ADSL и Linux, а по Ubuntu есть и специальные статьи. Однако учитывайте, что не вся информация на этих сайтах актуальна:

- http://www.adsl4linux.de/;
- http://wiki.ubuntuusers.de/DSL_ohne_Router.

Рекомендую также посмотреть следующие сайты:

- http://howto.htlw16.ac.at/at-highspeed-howto.html;
- http://www.univie.ac.at/ZID/anleitungen-adsl/;
- http://www.linux-usb.org/SpeedTouch/.

На трех последних сайтах рассказано, как работать с модемами USB-SpeedTouch, которые используются с протоколом PPPoA. Но я не рекомендую работать с такими модемами в Linux. Установка очень сложна, а сам метод чреват многочисленными сбоями (проверено на практике!). Лучше купить PPPoA-совместимый ADSL-роутер с обычным телефонным входом, а USB-модем не использовать.

Конфигурация ADSL-роутера

Многие провайдеры бесплатно предоставляют роутер при подключении. Если же вам его не предоставили, то настоятельно рекомендую приобрести ADSL-роутер. При этом необходимо обращать внимание на три момента: роутер должен поддерживать протокол, используемый вашим провайдером (например, PPPoE), должен быть совместимым с применяемой ADSL-технологией (например, ADSL2+) и иметь подходящий телефонный разъем (A или B).

A и B — это дополнения к стандарту G.992.1, описывающие параллельное использование каналов передачи (обычного телефонного канала и ADSL). В Германии все подключения ADSL выполняются в соответствии со стандартом B. Данная спецификация позволяет параллельно использовать телефонное соединение с ISDN и ADSL. В большинстве других государств ADSL-соединение, напротив, обычно устанавливается в соответствии с приложением A. Данная спецификация позволяет параллельно использовать аналоговое телефонное соединение и ADSL. Исключение представляют только ISDN-разъемы; если требуется совместно использовать ISDN и ADSL, то во всем мире в таких случаях применяется приложение B.

Перед началом работы ADSL-роутер нужно сконфигурировать. Как правило, вы подключаете компьютер к роутеру, открываете браузер и вводите IP-адрес роутера, то есть, например, http://192.168.0.101 (адрес указывается в инструкции к роутеру). После этого откроется удобный веб-интерфейс для конфигурации ADSL. Для создания конфигурации PPPoE здесь нужно указать логин и пароль вашего ADSL-соединения.

Если же ваш провайдер, напротив, использует PPPoA, то необходимо настроить параметры VPI *(идентификатор виртуального маршрута)* и VCI *(идентификатор виртуального канала)*. Правильные значения зависят от инфраструктуры ATM провайдера и специфичны для провайдера и страны. В следующем списке обобщены некоторые распространенные значения:

- Бельгия — VPI=8; VCI=35;
- Дания — VPI=0; VCI=35;
- Италия — VPI=8; VCI=35;
- Франция — VPI=8; VCI=35;

- Великобритания — VPI=0; VCI=38;
- Нидерланды — VPI=8; VCI=48;
- Австрия — VPI=8; VCI=48;
- Испания — различные комбинации, например, 1/32,1/33,8/32 и 8/35.

При необходимости узнайте у провайдера, какие значения необходимо использовать, или поищите в Интернете `список VPI VCI`.

Конфигурация ADSL-PPPoE

Конфигурация сетевой карты

Для соединения между ADSL-модемом и вашим компьютером используется Ethernet-кабель. При этом, в отличие от большинства случаев, не применяется протокол TCP/IP. При конфигурации сетевой карты не учитываются ни IP-адрес, ни маска сетевого интерфейса, поэтому настройка сети не требуется. В таком случае сетевой интерфейс нельзя конфигурировать как шлюз!

Конфигурация pppd

PPPoE в Linux обрабатывается с помощью модуля ядра. Чтобы `pppd` могла использовать сетевой интерфейс для PPPoE, используйте плагин `rp-pppoe.so`. В большинстве распространенных дистрибутивов этот файл устанавливается вместе с `pppd`.

Для конфигурации `pppd` вам, как обычно, потребуется конфигурационный файл в `/etc/ppp/peers/`. Для работы следующего кода необходимо, чтобы каталог `/etc/options` был пуст:

```
#/etc/ppp/peers/adsl
#PPPoE-специфичные параметры
pluginrp-pppoe.so
mru1492
mtu1492
# К этому интерфейсу подключается ADSL-модем
eth0
# Обычные параметры
lock
noauth
noipdefault
defaultroute
usepeerdns
# Логин для /etc/ppp/pap-secrets или chap-secrets
name"hofer"
# При разрыве соединения ждать 4 секунды,
# затем попытаться установить новое соединение
persist
holdoff4
maxfail25
#для Red Hat/Fedora
ipparam"adsl"
```

Обычные параметры действуют так же, как и при любых других соединениях с Интернетом, использующих PPP. Их действие описывается в разделе 16.7. Пароль для настройки `name` должен находиться в файле `chap-secrets` или `pap-secrets`.

С помощью `persist` соединение может быть автоматически восстановлено после обрыва. Параметр `holdoff` настраивает время ожидания между обрывом соединения и попыткой восстановить его, а `maxfail` указывает, после какого максимального количества неудачных попыток соединения `pppd` должна прекратить эти попытки. Эти параметры полезны в тех случаях, когда требуется бесперебойное соединение с Интернетом.

Если бесперебойное соединение не требуется, добавьте в конфигурационный файл `idle` *n*. В таком случае, если в течение *n* секунд вы не будете работать с системой, соединение с Интернетом автоматически завершится.

Параметр `ipparam` используется в Red Hat и Fedora для того, чтобы задействовать автоматическую конфигурацию DNS. В этих дистрибутивах дополнительно требуется следующий файл:

```
#/etc/sysconfig/network-scripts/ifcfg-adsl
PEERDNS=yes
```

Автоматическое тестирование соединения

При определенных обстоятельствах `pppd` может «не заметить», что у провайдера возникли проблемы с соединением и он больше не отвечает. Следующие дополнения, вносимые в конфигурационный файл, заставляют программу каждые 60 секунд отправлять провайдеру запрос, на который ожидается отклик. Если на два запроса, посланных друг за другом, не удается получить отклик, `pppd` завершает соединение. Если в конфигурационном файле содержится команда `persist`, то сразу же после этого делается попытка установить новое соединение.

```
#Дополнение в /etc/ppp/peers/adsl
lcp-echo-interval60
lcp-echo-failure2
```

MTU и MRU

В Интернете данные передаются не байт за байтом, а в виде пакетов. В сети Ethernet такие пакеты по умолчанию имеют размер 1500 байт. Если на пути между адресатом и отправителем данных есть оборудование или программы, для которых эти пакеты слишком велики, они автоматически дробятся на пакеты меньшего размера, а затем снова «собираются». Однако такой метод не слишком эффективен, и поэтому в некоторых операционных системах (в том числе в Linux) предпринимаются попытки определить максимальный размер пригодных для отправки пакетов путем рассылки «тестовых» ICMP-пакетов. Правда, такие пакеты «проглатываются» некоторыми брандмауэрами, из-за чего точно определить максимальный размер пакета не удается, а при некоторой доле невезения передача данных вообще прекращается.

Обычно такой проблемы не возникает, так как наименьший общий знаменатель, то есть 1500 байт, превышается редко. Но как раз при использовании PPPoE потери происходят, так как несколько байт из 1500-байтового пакета тратится

на протоколирование информации. Поэтому показатель максимальной длины получаемых пакетов (MRU) и максимальной длины передаваемых пакетов (MTU) следует снизить до 1492 байт.

Запуск и остановка pppd

Запуск pppd производится, как и при использовании аналогового модема, с помощью следующей команды:

```
root# pppd call adsl
```

Если вы хотите снова завершить соединение, остановите работу pppd командой killall:

```
root# killall pppd
```

В дистрибутивах, построенных на основе Debian, вместо этой команды применяется pon adsl или poff adsl.

Автоматический запуск ADSL с помощью сценария Init-V

Поскольку ADSL часто используется без ограничений по времени, соединение с Интернетом обычно устанавливается при запуске компьютера и сохраняется вплоть до его выключения. Лучше всего проделывать эти операции с помощью сценария Init-V, который выполняется при запуске компьютера.

Базовые сведения о программировании собственных Init-V сценариев уже были представлены в разделе 14.12 (Debian, Ubuntu), 14.13 (Red Hat, Fedora) и 14.14 (SUSE). Та часть сценария, которая строится в зависимости от конкретного дистрибутива, может выглядеть, например, так:

```
#/etc/init.d/adsl
#...команды, специфичные для определенного дистрибутива...
case "$1" in
  start)
    echo "Starting adsl"
    pppd call adsl
    ;;
  stop)
    echo "Shutting down adsl"
    [-f/var/run/ppp-adsl.pid] && \
    kill $(head -1 /var/run/ppp-adsl.pid)
  *)
    echo "Usage: $0 {start|stop}"
    exit 1
    ;;
esac
```

Сценарий отменяет действие команды killall pppd, так как на одном компьютере может действовать несколько процессов pppd. Команда killall остановила бы все такие процессы. Указанный сценарий, напротив, останавливает только тот процесс pppd, чей номер он считывает из файла /var/run/ppp-adsl.pid. При этом система сначала проверяет, существует ли указанный файл. Если это не так, то команда head -1 находит первую строку, содержащую номер процесса, и передает ее kill.

Для того чтобы pppd действительно сохранила свой номер процесса в /var/run/ppp-adsl.pid, в конфигурационный файл pppd необходимо вставить следующую строку:

```
#Дополнение в /etc/ppp/peers/adsl
linkname "adsl"
```

После того как вы протестируете сценарий с помощью /etc/init.d/adsl start или stop, потребуется настроить его автоматический запуск (см. раздел 4.5).

MSS Clamping

К сожалению, параметры MTU и MRU для pppd действуют только на локальном компьютере. Если этот компьютер одновременно является интернет-шлюзом для других компьютеров (см. главу 17), то и на каждом из клиентских компьютеров, подключенных к шлюзу, настройки MTU потребуется изменить.

Чтобы облегчить связанную с этим работу по конфигурированию системы, существует хорошее решение, называемое *MSS Clamping — сжатие максимального сегмента данных*. При использовании этого решения параметр MSS (максимальный размер сегмента данных) пакетов с данными, передаваемых по TCP, корректируется в соответствии со значением MTU на локальном компьютере. Чтобы понять, что такое максимальный размер сегмента данных, нужно быть экспертом по TCP.

За реализацию MSS Clamping отвечает система ядра iptables; для этого необходимо активизировать указанное ниже правило брандмауэра. При этом необходимо заменить eth0 названием интерфейса, к которому подключен ADSL-модем:

```
root# iptables -o eth0 --insert FORWARD 1-p tcp --tcp-flags SYN, RST SYN \
               -m tcpmss --mss 1400:1536 -j TCPMSS --clamp-mss-to-pmtu
```

Как правило, эта команда входит в состав сценария брандмауэра, оборудованного на компьютере-шлюзе (см. также разделы 17.3 и 18.4). Но вы можете интегрировать это правило и в описанный выше файл сценария Init-V. Правда, в таком случае необходимо убедиться, что при многократном запуске системы ADSL правило выполняется лишь один раз.

Конфигурация ADSL-PPTP

Туннельный протокол типа точка-точка со стороны компьютера-клиента поддерживается программой pptp. В большинстве распространенных дистрибутивов эта программа предоставляется в отдельном одноименном пакете, но этот пакет не всегда устанавливается автоматически. Базовая информация о PPTP, а также о графической программе pptpconfig, которая предназначена для конфигурации PPTP, но в большинстве дистрибутивов пока отсутствует, дается на следующем сайте: http://pptpclient.sourceforge.net.

PPTP используется не только для установления ADSL-соединений, но и для создания виртуальных частных сетей. Конфигурация VPN на клиентском компьютере очень напоминает конфигурацию ADSL (см. раздел 16.10). Совершенно иначе выглядит конфигурация VPN-сервера, где вместо pptp применяется программа pptpd (см. раздел 18.6).

Конфигурация сетевой карты

Поскольку соединение между ADSL-модемом и компьютером осуществляется по Ethernet, сначала нужно настроить именно это соединение. Вам потребуется выбрать IP-адрес и маску для сетевого интерфейса так, чтобы модем был доступен. Если модем имеет IP-адрес 10.0.0.138, как у некоторых моделей Alcatel, то для сетевого интерфейса необходимо задать номер 10.0.0.*n*, причем *n* не должно быть равно ни 0, ни 255, ни 138. В качестве маски используется 255.255.255.0. Настраивать адрес шлюза нельзя (общая информация о конфигурации сети дается в разделе 16.3). Чтобы проверить, есть ли соединение между вашим компьютером и ANT, выполните обычную команду `ping`:

```
root# ping 10.0.0.138
PING 10.0.0.138 (10.0.0.138): 56 data bytes
64 bytes from 10.0.0.138: icmp_seq=0 ttl=15 time=4.674 ms
64 bytes from 10.0.0.138: icmp_seq=1 ttl=15 time=3.737 ms
...
```

Конфигурация pppd

Для конфигурации `pppd` вам, как обычно, требуется специальный конфигурационный файл в `/etc/ppp/peers/`. Для работы следующего кода необходимо, чтобы каталог `/etc/options` был пуст:

```
# /etc/ppp/peers/adsl
# pptp-специфичные
pty "/usr/sbin/pptp 10.0.0.138 --nolaunchpppd"
# опционально: затребовать определенный метод шифрования
# require-mppe-128
# без сжатия
nobsdcomp
nodeflate
# логин для /etc/ppp/pap-secrets или chap-secrets
name "hofer"
# обычные параметры
lock
noauth
noipdefault
defaultroute
usepeerdns
# при обрыве соединения ожидать 4 секунды,
# затем установить новое соединение
persist
holdoff 4
maxfail 25
# Для Red Hat/Fedora
ipparam "adsl"
```

Параметр `pty` отвечает за вызов `pptp`, а `-nolaunchpppd` не позволяет `pptp` запустить программу `pppd`. В показанной здесь конфигурации это не обязательно, так как `pppd` уже работает и, в свою очередь, вызывает `pptp`.

Пароль для настройки `name` должен, как обычно, находиться в `chap- secrets` или `pap-secrets` (см. раздел 16.7).

Параметр `ipparam` необходим для того, чтобы в Red Hat и Fedora работала автоматическая конфигурация DNS. В этих дистрибутивах дополнительно требуется следующий файл:

```
# /etc/sysconfig/network-scripts/ifcfg-adsl
PEERDNS=yes
```

Запуск pppd

При запуске или остановке работы `pppd` нас уже ничто не удивит:

```
root# pppd call adsl
root# killall pppd
```

Сценарий для автоматического запуска ADSL был описан в предыдущем разделе. Его можно в таком же виде использовать и с PPTP. Единственное необходимое условие заключается в том, что сначала требуется настроить сетевое соединение с модемом.

16.10. Конфигурация клиента VPN (PPTP)

VPN — это защищенная сеть, использующая в качестве коммуникативного средства вторую (незащищенную) сеть, например Интернет или WLAN. Защищенное соединение компьютеров через незащищенную сеть часто называют туннелем. Существуют самые разные протоколы и варианты топологии VPN, которые в обобщенном виде рассмотрены в разделе о конфигурации VPN-сервера в разделе 18.5.

К сожалению, в большинстве дистрибутивов отсутствуют инструменты, предназначенные для облегчения конфигурации VPN-соединения. Только в программе Network Manager есть множество дополнительных модулей для различных типов VPN — правда, некоторые из этих модулей еще не окончательно доработаны. В этом разделе рассказано, как вручную подключить компьютер, работающий на основе протокола PPTP, к сети VPN. Для этого необходимо выполнить следующие условия:

- компьютер, с которым вы устанавливаете соединение, должен быть сконфигурирован как PPTP-сервер;
- ваш компьютер должен быть подключен к (незащищенной) базовой сети; этот этап был подробно рассмотрен в предыдущих разделах данной главы;
- незащищенная базовая сеть (например, WLAN) должна использовать адресное пространство 172.16.0.*;
- защищенная сеть (VPN) должна использовать адресное пространство 192.168.0.*;
- VPN-сервер должен быть доступен по адресам 172.16.0.1 (WLAN) и 192.168.0.1 (VPN).

PPTP

Microsoft достаточно давно отдает предпочтение PPTP (туннельному протоколу типа точка-точка). Как следствие, этот протокол широко распространен. Он не так надежен, как некоторые другие методы, но во многих практических ситуациях обеспечивает вполне приемлемый уровень безопасности. Значительным достоинством по сравнению другими методами VPN в данном случае является относительно простое обслуживание протокола.

В принципе описанная здесь конфигурация должна функционировать и в том случае, когда VPN-сервер работает с Windows, но я не тестировал такой способ работы. Мой тестовый VPN-сервер работает с Linux, конфигурация этого сервера подробно описана в разделе 18.6.

Клиентское соединение PPTP работает с применением двух тесно взаимодействующих программ: PPP-демона `pppd` и PPTP-клиента `pptp`; как правило, эти программы требуется специально установить.

Конфигурация PPTP-клиента — это не что иное, как вариант самой тривиальной конфигурации PPP, которая требуется при любом соединении с Интернетом через обычный модем или ADSL. Чтобы произвести конфигурацию вручную, необходимо изменить либо заново создать два файла: `/etc/ppp/peers/vpn` и `/etc/ppp/chap-secrets`.

/etc/ppp/peers/vpn. Файл `/etc/ppp/peers/vpn` содержит параметры PPP для создания VPN-соединения. Имя файла не имеет значения, но он должен находиться в каталоге `/etc/ppp/peers/`. Если VPN-сервер сконфигурирован так, как это описано в разделе 18.6, то на каждом компьютере-клиенте должны применяться следующие параметры:

```
# /etc/ppp/peers/vpn
pty "/usr/sbin/pptp 172.16.0.1 --nolaunchpppd"
user "vpnclient"
noauth
require-mppe-128
defaultroute
usepeerdns
ipparam "vpn"     # для Red Hat и Fedora
```

Благодаря настройке `pty` демон PPP при создании соединения пользуется программой `pptp`. Эта программа создает соединение с VPN-сервером, имеющим IP-адрес 172.16.0.1.

Для аутентификации применяется логин PPP merkurvpn. Пароль для этого логина находится в файле `/etc/ppp/chap-secrets`. Та же информация о логине и пароле должна содержаться и в файле `chap-secrets` VPN-сервера.

Наличие параметра `noauth` означает, что удаленная станция PPP не требует аутентификации (от VPN-сервера). Однако клиент обязательно должен идентифицироваться!

При применении параметра `require-mppe-128` используется протокол Microsoft шифрования двухточечного соединения с ключом длиной 128 бит. Обратите внимание и на то, что в некоторых версиях PPP *нельзя* задавать параметр `requiremschap-v2`,

даже если сервер PPTP имеет соответствующую конфигурацию и такой метод шифрования действительно применяется.

Благодаря параметру `defaultroute` **IP-адрес, присвоенный серверу VPN, используется** в качестве стандартного места назначения при роутинге IP-пакетов (то есть все IP-пакеты автоматически передаются через интерфейс сети VPN, а не через другой интерфейс).

Действие параметра `usepeerdns` заключается в том, что он заносит адрес сервера имен, присвоенный VPN-сервером, в файл `/etc/resolv.conf`. В Red Hat и Fedora этот параметр функционирует лишь при условии, что вместе с `ipparam` передается дополнительный параметр `vpn`; следовательно, должен существовать файл `/etc/sysconfig/network-scripts/ifcfg-vpn`, в котором будет команда `PEERDNS=yes`.

ПРИМЕЧАНИЕ

Обращаю ваше внимание на то, что программа pppd учитывает и те параметры, что указаны в /etc/ppp/options. Но параметры, занесенные в /etc/ppp/peers/vpn, имеют приоритет. В этом разделе мы исходим из того, что файл options пуст.

Если у вас возникнут проблемы с конфигурацией PPTP-доступа, то сначала посмотрите в /etc/ppp/options и попробуйте удалить все параметры. В SUSE, в частности, потребуется удалить idle 600. Этот параметр заставляет PPP разрывать любое соединение, если в течение 10 минут система простаивает. Такая возможность представляется целесообразной при модемном соединении, но не при работе с VPN. Важнейшие параметры PPP описаны в разделе 16.7.

ppp/chap- secrets. В файл `/etc/ppp/chap-secrets` вставляется логин и пароль для VPN-соединения (аналогичный файл с паролем PPTP-сервера находится в подразделе «Настройка PPTPD-сервера» раздела 18.6).

```
# /etc/ppp/chap-secrets
#логин       сервер   пароль            IP-адрес
"venusvpn"   *        "venusvpnpassw"   *
```

Создание и завершение VPN-соединения

Чтобы создать VPN-соединение, выполните следующую команду:

```
root# /usr/sbin/pppd call vpn
```

Если в окне терминала появляются разные непонятные сообщения, это может означать, что в `/etc/ppp/options` содержится параметр `nodetach`. Удалите его, чтобы `pppd` выполнялась как фоновый процесс.

Если все получится, то команды `ifconfig` и `route` возвратят следующие результаты:

```
root# ifconfig
lo Link encap:local loop
   inet Address:127.0.0.1 Mask:255.0.0.0
    ...
eth1 Link encap:Ethernet Hardware Address 00:0C:F1:58:F9:93
     inet Address:172.16.0.199 Bcast:172.16.0.255 Mask:255.255.255.0
     ...
ppp0 Link encap:Point-to-Point connection
```

```
     inet Address:192.168.0.200 P-t-P:192.168.0.1 Mask:255.255.255.255
     ...
root# route -n
Kernel IP Routing table
Destination  Router       Genmask          Flags  Metric  Ref  Use  Iface
192.168.0.1  0.0.0.0      255.255.255.255  UH     0       0    0    ppp0
172.16.0.0   0.0.0.0      255.255.255.0    U      0       0    0    eth1
0.0.0.0      192.168.0.1  0.0.0.0          UG     0       0    0    ppp0
```

Этот вывод означает, что кроме петлевого интерфейса в системе есть еще два сетевых интерфейса: `eth1` с соединением, осуществляемым по незащищенной сети WLAN, а также `ppp0` с VPN-соединением. Таким образом, локальный компьютер доступен по трем IP-адресам: 127.0.0.1 (петлевой интерфейс) 172.16.0.199 (WLAN) и 192.168.0.200 (локальная сеть).

Общий сетевой трафик направляется на интерфейс `ppp0`, то есть идет через VPN. Исключение составляют лишь те пакеты, которые направлены непосредственно по адресам 172.16.0.* — они перенаправляются на интерфейс `eth1`.

Если вы хотите снова завершить VPN-соединение, выполните следующую команду:

```
root# killall pptp
```

СОВЕТ

При возникновении проблем с созданием VPN-соединения дополните /etc/ppp/peers/vpn еще одной строкой с ключевым словом debug. Программа pppd запишет подробные статусные сообщения и сообщения об ошибках в файл регистрации (обычно /var/log/messages или /var/log/syslog). Подробное руководство по поиску ошибок дается на сайте http://pptpclient.sourceforge.net/howto-diagnosis.phtml.

Запуск VPN без привилегий администратора

К сожалению, только администратор имеет право создавать VPN-соединение. В Debian и Ubuntu обычные пользователи могут прибегать к командам `pon pptp` или `poff pptp`. В остальных дистрибутивах при необходимости можно написать небольшой сценарий, с помощью которого будет создаваться соединение. Дополнительно нужно сконфигурировать `/etc/sudoers` так, чтобы обычные пользователи могли выполнять подобные сценарии с помощью `sudo`.

Брандмауэр для VPN-клиента

Теперь на вашем компьютере действует два соединения с VPN-сервером: незащищенное соединение через WLAN (172.16.0.*) и защищенное через VPN (192.168.0.*). Для инициированного вами сетевого трафика автоматически используется защищенное соединение. Однако это не мешает злоумышленнику попытаться подключиться к вашему компьютеру через незащищенную сеть WLAN (например, с помощью Telnet, SSH и т. д.). Поэтому необходимо защитить интерфейс WLAN брандмауэром. Для этого выполните сценарий, описанный в подразделе «Конфигурация брандмауэра для PPTP-клиента» раздела 18.6.

17 Интернет-шлюз

Далее нам предстоит изучить несколько тем, посвященных конфигурации сервера. Было бы самонадеянно попытаться охватить в нескольких главах все вопросы конфигурации сервера в Linux. Почти каждый из разделов этой главы можно расширить до целой книги.

Я писал главы о конфигурации сервера для того, чтобы сообщить вам базовую информацию по данной теме, исключительно важной для опытных пользователей Linux. Эти главы адресованы в первую очередь тем специалистам, которые занимаются администрированием относительно небольших локальных сетей, причем в таких сетях вполне могут находиться и компьютеры-клиенты с Windows.

ADSL-роутер… В этой главе речь пойдет о том, как создать интернет-шлюз для локальной сети. В частных сетях эту задачу часто выполняет ADSL-роутер. Такой метод имеет очевидные достоинства: конфигурация проста, устройство работает бесшумно и потребляет относительно немного энергии. В общем, если вас устраивает ADSL-роутер, то продолжайте с ним работать, а эту главу можете пропустить! Так вы сэкономите себе массу времени и сил!

…и собственный шлюз. И все же есть случаи, когда конфигурационных возможностей ADSL-модема явно недостаточно: в некоторых моделях брандмауэров предусмотрено слишком мало параметров, разрешение имен не соответствует требованиям Kerberos, интегрированный DHCP-сервер оказывается непригоден для создания защищенной конфигурации WLAN/VPN и т. д. В результате приходится обратиться к этой главе, то есть настроить отдельный компьютер, который будет выполнять следующие функции ADSL-роутера: маскарадинг, работа DHCP-сервера и сервера имен. Любой интернет-шлюз обязательно должен быть защищен брандмауэром. О том, как это сделать, рассказано в главе 18.

Дистрибутивы. Конфигурации, представленные в этой и следующих главах, тестировались мной с современными версиями Fedora, openSUSE и Ubuntu. За исходный пункт принималась стандартная установка с Gnome — в Fedora и Ubuntu или с KDE — в openSUSE. Конфигурации в дистрибутивах Debian, SUSE Enterprise и Red Hat Enterprise не тестировались. Однако вряд ли эти конфигурации будут значительно отличаться от тех, что были получены в Ubuntu/openSUSE/Fedora.

17.1. Введение

В этой главе будет описана установка следующих компонентов (служб).

- **Маскарадинг/NAT.** С помощью маскарадинга можно соединить все локальные клиенты, подключенные к Интернету. Итак, есть компьютер с Linux, устанавливающий соединение с Интернетом посредством ADSL или ISDN. Все остальные компьютеры сети соединены с этим компьютером и также могут пользоваться Интернетом. Отпадает необходимость оборудовать каждый компьютер отдельным модемом!
- **DHCP и DNS.** DHCP обеспечивает простое централизованное администрирование IP-адресов и других сетевых параметров всех клиентов. Локальный сервер имен гарантирует, что клиенты, в свою очередь, «знают» свои имена (разрешение локальных имен в IP-адресах). Кроме того, программа играет роль кэша IP-адресов, и последующие соединения с Интернетом становятся немного быстрее.

  Здесь будут рассмотрены два варианта реализации функций DHCP и DNS. Проще всего воспользоваться командой `dnsmasq`, которая сочетает в себе обе функции и проста в конфигурировании. Если же вы предъявляете к сети повышенные требования или строите большую и сложную локальную сеть, рекомендуется распределить между двумя программами `dhcpd` и `bind`. Таким образом, повышаются возможности конфигурации, а вы можете выполнить ее с большей легкостью.
- **Интеграция с WLAN.** Соединить обычную сеть и WLAN совсем несложно. Проблема заключается в том, чтобы обеспечить безопасную реализацию функций WLAN. Поэтому в последнем разделе этой главы будут рассмотрены некоторые варианты конфигурации.

Оборудование. Если интернет-шлюз не должен выполнять никаких функций, кроме задач, собственно, шлюза, то его можно создать на медленном ПК с минимумом оборудования и этого будет вполне достаточно. Однако в компьютере должно быть два сетевых интерфейса: один — для подключения компьютера к ADSL-роутеру или к ADSL-модему, а другой — для подключения к локальной сети.

Когда вы займетесь поиском тихого, недорогого и энергосберегающего компьютера, вы быстро убедитесь, что самое большое препятствие — второй сетевой интерфейс: мини-ПК, например Eee Box или MSI Wind PC, обычно имеют только один сетевой интерфейс, и к ним нельзя подключать сменные платы (PCI, PCIe). Один из возможных выходов — использовать USB-Ethernet-адаптер, но пока работать с ним в Linux еще сложно. Мне понравилось работать с устройством DUB-E100 фирмы D-Link, которое начинает работать с ходу, без дополнительной настройки. Список подходящих устройств дается по следующему адресу: http://www.linux-usb.org/usbnet/.

Интересной альтернативой является WLAN-роутер WRT54GL фирмы Linksys или совместимые с ним устройства. Характерной чертой этих устройств является то, что их прошивка существует в версии для Linux, которую можно и нужно использовать. В Интернете представлен целый спектр подходящих дистрибутивов

(Tomato, DD-WRT, OpenWRT и т. д.). Конечно же, чтобы начать работу с этим оборудованием, придется немного потрудиться. Результатом вашего труда будет дешевый, полностью бесшумный прибор, потребляющий значительно меньше электричества, чем мини-ПК, имеющиеся на рынке.

Установка пакетов. В целях безопасности пакеты серверных служб, описанные в этой главе, по умолчанию *не* устанавливаются. Иногда вам могут встретиться пакеты с похожими названиями для соответствующих клиентских служб; но в данном случае они не подойдут. После установки серверных пакетов программы также следует специально активизировать.

Помощь в конфигурации. Эта глава адресована опытным пользователям Linux. Здесь описаны лишь настройки различных конфигурационных файлов, но не рассмотрены пользовательские интерфейсы, которые могут помочь вам при конфигурации и могут находиться в свободном доступе. В Fedora и Red Hat это различные команды `systemconfig-`*xxx*, в SUSE — модули Службы сети в YaST.

Здесь будет рассмотрен метод работы с изменением конфигурационных файлов вручную — он может показаться старомодным, но на практике вполне себя оправдывает: лишь в том случае, если вы выполняете конфигурацию вручную, вы знаете, где находятся конфигурационные файлы. И только используя этот способ, вы можете достаточно быстро сориентироваться в готовой конфигурации другого сервера (например, при новой установке системы или смене дистрибутива).

eBox. Если вы ищете сетевой интерфейс для конфигурации серверных служб, обратите внимание на eBox. Эта конфигурационная система, специально оптимизированная для работы с Ubuntu, особенно пригодится вам в тех случаях, когда настраиваемый сервер в будущем будет управляться пользователями, которые обладают небольшим опытом администрирования: http://ebox-platform.com.

Безопасность. Если вы хотите использовать Linux на сетевом сервере, то просто *обязаны* хорошо изучить тему безопасности — несерьезное отношение к этой теме является грубой халатностью. Советы, касающиеся основ обеспечения безопасности сервера, даются в главе 18.

IPv6. Все программы, описанные в этой главе, совместимы с IPv6. Однако я тестировал сетевые функции лишь с IPv4, поэтому в дальнейшем особенности IPv6 рассматриваться не будут.

Примерная топология сети. Для того чтобы вы могли лучше ориентироваться в этой главе, на рис. 17.1 показана примерная топология сети. В этой локальной сети используется адресное пространство 192.168.0.* и доменное имя sol. Компьютер-шлюз с хост-именем mars имеет постоянный адрес 192.168.0.1. Соединение с Интернетом осуществляется через ADSL-роутер.

Клиенты локальной сети соединены с компьютером mars через сетевой концентратор. Клиентам динамически присваиваются IP-адреса (192.168.0.2–192.168.0.253). Единственное исключение представляет собой сетевой принтер pluto, которому присвоен статический IP-адрес 192.168.0.254.

Для обеспечения безопасности рекомендуется оборудовать выход в Интернет с компьютера mars брандмауэром. Но по причинам, связанным с экономией энергии и денег, в небольших и частных сетях все же целесообразно интегрировать все серверные функции на одном компьютере, как это описано ниже.

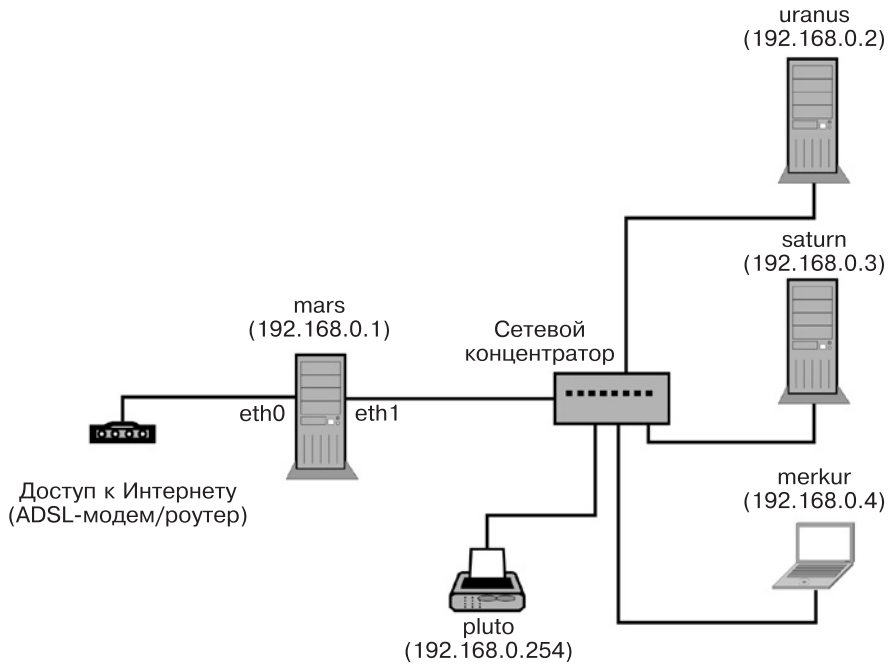

Рис. 17.1. Примерная топология сети

17.2. Статическая конфигурация сети

Прежде чем вы сможете сконфигурировать функцию маскарадинга, а также DHCP и сервер имен, необходимо подключить компьютер-шлюз к Интернету и локальной сети. Для этого необходима статическая конфигурация обоих сетевых адаптеров, причем метод ее осуществления варьируется от дистрибутива к дистрибутиву.

В большинстве дистрибутивов вместе с системой для локального компьютера устанавливается программа Network Manager. Эта исключительно полезная программа поможет вам при динамической конфигурации сети (особенно WLAN). Однако при работе с сервером «динамическое поведение» Network Manager часто мешает. Если возможно, деинсталлируйте эту программу. К сожалению, так удается сделать не во всех дистрибутивах, что связанно со взаимозависимостями системы и пакетов. В SUSE имеется возможность переключения между статической конфигурацией и Network Manager в модуле YaST.

Debian, Ubuntu

В Debian и Ubuntu удалите пакеты `libnm*` и `network-manager*`, если они уже установлены. За статическую конфигурацию отвечает только файл `/etc/network/interfaces`. Синтаксис очень прост: каждый интерфейс, который должен быть активизирован при запуске компьютера, необходимо назвать `auto` *имя*. Команда `iface` *имя параметры* описывает базовую конфигурацию интерфейса. При статической конфигурации

далее следуют строки с указанием параметров address (статический IP-адрес), netmask (маска для доступного адресного пространства) и gateway (IP-адрес компьютера, через который осуществляется соединение с Интернетом или другим компьютером).

В следующих строках дан пример конфигурации сети. Первые две строки активизируют петлевой интерфейс, который требуется всегда. Он служит для внутрикомпьютерного сетевого обмена информацией. Через интерфейс eth0 компьютер выходит в Интернет (с помощью ADSL-роутера или другого сервера). Конфигурация выполняется автоматически, через DHCP. Оставшиеся строки описывают статическую конфигурацию интерфейса eth1. Он применяется для обмена информацией сервера с локальной сетью, использующей адресное пространство 192.168.0.*. В пределах этой сети сервер имеет статический IP-адрес 192.168.0.1.

```
# /etc/network/interfaces
auto lo
iface lo inet loopback
# Динамическое соединение с DHCP-сервером,
# сообщающим основные показатели для доступа в Интернет
auto eth0
iface eth0 inet dhcp
# Статическая конфигурация для соединения с LAN
auto eth1
iface eth1 inet static
  address 192.168.0.1
  netmask 255.255.255.0
```

Если соединение с Интернетом является статическим, то файл interfaces нужно доработать следующим образом. В примере предполагается, что соединение осуществляется через ADSL-роутер, имеющий IP-адрес 10.0.0.138 (как устройство SpeedTouch, которым пользуюсь я). Этот адрес одновременно является адресом шлюза, через который осуществляется выход в Интернет (ключевое слово gateway).

```
# /etc/network/interfaces
...
# Статическое соединение с ADSL-роутером, имеющим IP-адрес 10.0.0.138
auto eth0
iface eth0 inet static
  address 10.0.0.1
  netmask 255.255.255.0
  gateway 10.0.0.138
```

Для инициализации сетевых интерфейсов при запуске компьютера используется сценарий Init-V /etc/init.d/networking, выполняемый уже при инициализации системы (ссылка /etc/rcS.d/S40networking). Всю работу выполняет команда ifup, специфичная для Debian и Ubuntu (пакет ifupdown). Команда ifup -a интерпретирует /etc/network/interfaces и активизирует все auto-интерфейсы.

Если конфигурация интерфейсов производится через DHCP, то ifup использует для передачи и интерпретации данных DHCP команду dhclient. За ее конфигурацию отвечает /etc/dhcp3/dhclient.conf.

Вам также потребуется вручную создать /etc/resolv.conf (сервер имен) и /etc/hosts (см. раздел 16.5).

Fedora, Red Hat

В современных версиях Fedora и Red Hat для локальных компьютеров программу Network Manager нельзя удалить из-за существующих межпакетных взаимосвязей. Но ее можно деактивизировать, и построить конфигурацию сети на обычных конфигурационных файлах. Для этого выполните следующие команды:

```
root# /etc/init.d/NetworkManager stop
root# /etc/init.d/network start
root# chkconfig --level 35 NetworkManager off
root# chkconfig --level 35 network on
```

Для конфигурации сети используйте программу system-config-network, которую можно запустить и с помощью меню Система ▸ Администрирование ▸ Сеть. При конфигурации интерфейса вручную нужно либо снять флажок Проверять с помощью Network Manager, либо установить флажок Включать устройство после запуска компьютера. Кроме того, необходимо заново настроить хост-имена (вкладка DNS). Внесенные изменения должны вступить в силу после выхода из программы. Если не сработает, выполните следующую команду:

```
root# /etc/init.d/network restart
```

Если вы хотите произвести конфигурацию в консоли (то есть не пользуясь графической системой), то можете прибегнуть к system-config-network-tui. Правда, в этой программе предусмотрено меньше возможностей конфигурации, чем в system-config-network.

SUSE

В дистрибутивах openSUSE и Novell рекомендуется проводить конфигурацию с помощью модуля YaST. В модуле Устройства сети ▸ Параметры сети установите флажок Традиционный метод с использованием Ifup и снимите Network Manager. Кроме того, откройте вкладку Вид и откорректируйте в ней настройки сетевого адаптера.

17.3. Маскарадинг (NAT)

Исходной точкой для маскарадинга служит компьютер, который уже подключен к Интернету (в этой главе компьютер mars). Теперь мы должны дать доступ в Интернет и всем другим компьютерам, находящимся в локальной сети.

Компьютеры в локальной сети используют частные IP-адреса: эти адреса расположены в специально зарезервированных адресных пространствах (например, 10.*.*.* или 192.168.*.*, см. также раздел 16.3). Теперь адреса являются уникальными только в рамках сети LAN, но не в Интернете. Поэтому интернет-шлюз не может просто переадресовывать запросы страниц в LAN.

Принцип *маскарадинга* заключается в том, что компьютер-шлюз принимает пакеты, направленные клиентами в Интернет и изменяет их обратные адреса так, как будто эти пакеты были посланы самим компьютером-шлюзом. Такое изменение адреса также называется *трансляцией сетевых адресов* (NAT).

Теперь пакет с данными может быть направлен по определенному адресу в Интернете. Как правило, через некоторое время из Интернета приходит ответ — например, отображается сайт. Шлюз должен переадресовать ответ тому клиенту, который отправлял соответствующий запрос. Для этого шлюз должен «отгадать» правильный адрес получателя, ведь пакет (в соответствии с изменением адреса) был отправлен самим шлюзом, поэтому и ответ приходит именно на шлюз.

Чтобы обеспечить правильное присвоение адресов пакетам, полученным в ответ, шлюз изменяет не только обратный адрес, но и порт отправителя. Для каждого IP-адреса, находящегося в сети, применяется определенный номер порта. Внутри системы Linux за маскарадинг отвечает система `iptables`. Это интегрированная в ядро система, предназначенная для обработки IP-пакетов.

Если компьютер-шлюз подключается к Интернету через ADSL-роутер (а не через модем), то это устройство еще раз осуществляет маскарадинг, то есть манипулирует адресами. К счастью, никаких проблем из-за этого не возникает.

Функции маскарадинга и брандмауэра тесно связаны друг с другом. В этом разделе предполагается, что компьютер-шлюз *не выполняет* функций брандмауэра. Если он их все же выполняет, то вам может потребоваться отключить некоторые функции брандмауэра, прежде чем активизировать маскарадинг. Базовая информация о том, что такое брандмауэр и как он работает, дается в главе 18.

Понятие о клиенте и сервере. Далее по тексту компьютер, имеющий выход в Интернет, будет называться *сервером*, а все остальные компьютеры — *клиентами*, независимо от того, какие именно функции выполняют эти компьютеры. При осуществлении маскарадинга сервер зачастую будет называться *интернет-шлюзом* (что более правильно), или *интернет-роутером*. Это различие между сервером и клиентом справедливо для всей книги, но только в рамках данной отдельной функции! Компьютер, являющийся клиентом относительно своего выхода в Интернет, вполне может быть сервером при выполнении другой функции (например, NFS). На практике очень часто несколько серверных функций реализуется на одном компьютере. Но такая концентрация не является обязательной, а в больших сетях — ради повышения эффективности работы сети — такой концентрации даже следует избегать.

Включение и выключение маскарадинга

Fedora, Red Hat. В некоторых дистрибутивах функцию маскарадинга можно активизировать в программе конфигурации брандмауэра. В Fedora и Red Hat запустите для этого Система ▸ Администрирование ▸ Брандмауэр. На вкладке Надежный интерфейс выберите интерфейс для LAN. Затем убедитесь, что брандмауэр не блокирует трафик, идущий в локальную сеть. Кроме того, выберите на вкладке Маскарадинг интерфейс, через который будет осуществляться подключение к Интернету.

SUSE. В SUSE необходимо соотнести интерфейс LAN и внутреннюю сеть — для этого используйте функцию Безопасность ▸ Брандмауэр в модуле YaST. Дополнительно установите на вкладке Маскарадинг одноименный флажок.

Debian, Ubuntu. В Debian и Ubuntu пока нет специальных конфигурационных инструментов для создания брандмауэра и обеспечения маскарадинга. Но вы можете, например, установить программу FireStarter и активизировать с ее помощью и брандмауэр, и маскарадинг.

Активизация вручную. Если вы хотите управлять маскарадингом, не используя инструментария, связанного с брандмауэром, выполните две короткие команды:

```
root# echo 1 > /proc/sys/net/ipv4/ip_forward
root# iptables -A POSTROUTING -t nat -o eth0 -j MASQUERADE
```

Команда `echo` активизирует в ядре функцию IP-переадресации, которая по умолчанию отключена из соображений безопасности. Почти во всех дистрибутивах вместо `echo` вы также можете использовать более изящную команду `sysctl`:

```
root# sysctl -w net.ipv4.ip_forward=1
```

Команда `iptables` определяет правило, в соответствии с которым IP-пакеты, покидающие локальную сеть, должны перенаправляться на интерфейс `eth0` и претерпевать при этом преобразования в соответствии с правилами NAT. В зависимости от конфигурации доступа к Интернету вам может понадобиться указать и другой интерфейс, например, `ppp0`.

Деактивизация вручную. Чтобы отключить функции маскарадинга, выполните следующие команды:

```
root# iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
root# echo 0 > /proc/sys/net/ipv4/ip_forward
```

Вместо `echo` в большинстве дистрибутивов вы также можете активизировать `sysctl`:

```
root# sysctl -w net.ipv4.ip_forward=0
```

Активизация маскарадинга с помощью сценария. Разумеется, вам не придется активизировать маскарадинг при каждом запуске компьютера. Обычно маскарадинг запускается вместе с брандмауэром. Если вы настраивали свой брандмауэр вручную, активизируйте его вместе с функцией маскарадинга с помощью сценария, выполняемого в процессе Init-V (подробнее об этом читайте в главе 18).

Если вы не собираетесь пользоваться функциями брандмауэра либо хотите управлять ими отдельно, то, конечно же, можете запускать и отключать маскарадинг с помощью отдельного сценария Init-V. Соответствующий сценарий для Debian и Ubuntu приведен в разделе 14.12. Советы о том, как интегрировать собственные сценарии в процессы Init-V в Red Hat и Fedora, даются в разделах 14.13–14.14.

Если вы подключаетесь к Интернету по PPP (например, через ADSL-модем), то можете интегрировать команды маскарадинга в сценарии `ip-up` и `ip-down` в `/etc/ppp`. Таким образом, маскарадинг активизируется одновременно с подключением к Интернету и автоматически деактивизируется, когда соединение разрывается.

Проблемы

Бесспорно, маскарадинг — это достаточно изящное решение, обеспечивающее общий доступ из локальной сети в Интернет. Но с маскарадингом могут возникать определенные проблемы.

- Существует несколько интернет-протоколов, в которых предусмотрены защитные механизмы, проверяющие механизм распределения IP-адресов. При использовании маскарадинга соотнесенность между IP-адресом и компьютером перестает быть однозначной, и это может мешать работе.
- В некоторых протоколах предусмотрена возможность передачи IP-адресов не только в IP-пакетах, но и в пакетах с данными (в виде текста ASCII или в зашифрованном виде). Наиболее известный пример такого протокола — FTP. Чтобы FTP работал несмотря на применение маскарадинга, маскарадинговый сервер должен изменять не только IP-адреса пакетов, но и их содержание.
- В Linux для множества интернет-служб предусмотрены специальные маскарадинговые модули (например, `nf_nat_ftp` для FTP). Модули загружаются автоматически. Если они не загружаются, активизируйте их с помощью команды `modprobe`:

  ```
  root# modprobe nf_nat_ftp
  root# modprobe nf_conntrack_ftp
  ```

  Если при подключении FTP-клиентов к Интернету по-прежнему возникают проблемы, их можно устранить, переведя работу в пассивный режим. Большинство FTP-клиентов автоматически активизируют этот режим либо предусматривают возможность настроить его вручную.
- Если маскарадинговый сервер соединен с Интернетом по ADSL/PPPoE, могут возникать сложности с максимальной длиной пакетов. Чтобы разрешить эту проблему, можно уменьшить во всех пакетах максимальную длину (MTU) либо задействовать на сервере метод MSS Clamping. Для этого необходимо выполнить следующую команду `iptables`. С ее помощью к слишком большим пакетам применяется MSS Clamping.

  ```
  root# iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN \
        -j TCPMSS --clamp-mss-to-pmtu
  ```

Конфигурация клиента

Для того чтобы клиент мог использовать выход в Интернет, предоставляемый Linux-шлюзом, при сетевой конфигурации клиентов необходимо учесть два момента.

- В качестве адреса шлюза нужно указать IP-адрес Linux-шлюза (при топологии сети, как на рис. 17.1, это будет адрес 192.168.0.1).
- Адрес сервера имен должен совпадать с IP-адресом сервера имен для Linux-шлюза. Этот адрес присваивается провайдером интернет-услуг и содержится в файле `/etc/resolv.conf` Linux-шлюза.

17.4. Основы работы с DHCP и сервером имен

Разумеется, можно настроить сетевые параметры для каждого компьютера локальной сети отдельно. Но это не только очень утомительно, но и чревато ошибками. Ко всему прочему вы будете вынуждены выполнить массу дополнительной работы, если решитесь изменить топологию сети.

Гораздо разумнее построить сеть так, чтобы *один из компьютеров* отвечал за присвоение IP-адресов и других параметров всем остальным компьютерам в сети. Для этого используется *протокол динамической настройки хостов* (DHCP). Управляющий компьютер становится DHCP-сервером, а все остальные компьютеры — DHCP-клиентами. Есть два основных варианта конфигурации.

- **Динамическая конфигурация** — для клиента настраивается только хост-имя. DHCP-сервер отвечает за все остальные конфигурационные параметры, то есть присваивает клиентам IP-адрес, адрес шлюза, адрес сервера имен и т. д. IP-адреса берутся из специального адресного пула, причем если клиентскому компьютеру потребуется адрес, то DHCP-сервер автоматически присвоит ему ближайший адрес, свободный в настоящий момент.
- **Статическая конфигурация** — при использовании такого варианта конфигурации DHCP-сервер идентифицирует клиентов по ID-номеру сетевой карты. Таким образом, всякий раз компьютер получает один и тот же IP, а в определенных случаях — и хост-имя. Такой вариант конфигурации несколько сложнее настроить, но при его использовании вы можете применять неизменяющиеся IP-адреса и централизованно управлять хост-именами.

Как правило, первый вариант проще второго. Но вы можете и комбинировать оба варианта, например чтобы гарантировать, что принтер в любом случае будет иметь один и тот же адрес.

Внутренняя организация DHCP

Принцип работы DHCP примерно таков: если компьютер (то есть DHCP-клиент) перезапускается, он осуществляет широковещательную трансляцию по адресу 255.255.255.255 (таким образом, запрос отправляется на *все* компьютеры, находящиеся в локальной сети). DHCP-сервер реагирует на такой запрос и отсылает в ответ по адресу клиента список доступных IP-адресов.

Куда же сервер посылает ответ, ведь у клиента еще нет IP-адреса? Дело в том, что для адресации достаточно знать MAC-адрес, а он на момент запроса уже известен.

DHCP-сервер выдает IP-адреса на определенный промежуток времени *(время аренды)*. Обычно этот промежуток авен одному дню, но вы можете установить любое значение. До того как закончится время аренды, клиент должен обновить адрес на DHCP-сервере или запросить новый адрес.

Сервер имен

Сервер доменных имен (сервер имен, или DNS) обеспечивает взаимное соответствие между именами компьютеров и IP-адресами. Каждый провайдер интернет-услуг предоставляет своим клиентам DNS, и этот сервер узнает, какой IP-адрес подходит к имени компьютера. Чтобы не обращаться постоянно к DNS, можно настроить в локальной сети собственный сервер имен. В таком случае будет два преимущества.

- **Более высокая скорость.** DNS управляет кэшем недавно использовавшихся интернет-адресов. Иначе говоря, если вы желаете попасть на сайт yahoo.com, на котором уже побывали недавно, вам не нужно повторно обращаться к DNS вашего провайдера, чтобы узнать IP-адрес Yahoo!. Локальный DNS уже запомнил этот адрес.
- **Локальное разрешение имен.** DNS управляет именами и IP-адресами компьютеров, находящихся в локальной сети. Таким образом, вы знаете все компьютеры, находящиеся в сети, «поименно» и можете, например, выполнить на компьютере merkur команду `ping saturn`. Тогда компьютер merkur свяжется с локальным сервером имен, который сообщит вам IP-адрес компьютера saturn. Локальное разрешение имен является — основная предпосылка для удобного конфигурирования и использования в локальной сети таких служб, как NFS, FTP, SSH и др.

В мире используется бесчисленное множество DNS, и все они связаны друг с другом. Иными словами, если сам DNS не знает имени, то он может послать соответствующий запрос другому DNS. Все DNS организованы иерархически.

17.5. Программа dnsmasq (DHCP и сервер имен)

В начале этой главы я уже упоминал, что собираюсь описать два метода, позволяющих создать на шлюзе DHCP и сервер имен. В этом разделе будет рассмотрена программа `dnsmasq`, в которой интегрированы обе функции, что существенно упрощает конфигурацию. В двух следующих разделах мы рассмотрим программы `dhcpd` (раздел 17.6) и `bind` (раздел 17.7), которые предназначены для настройки DHCP и сервера имен соответственно. Эти программы рекомендуется использовать, прежде всего, в больших сетях или при необходимости выполнения очень сложных требований, например, когда DHCP-сервер отвечает за работу нескольких подсетей.

Условия

В дальнейшем предполагается, что на вашем компьютере установлена программа `dnsmasq`, а `dhcpd` *не* установлена (иначе между двумя этими программами возникнет конфликт).

Другой важной предпосылкой является правильная конфигурация файла /etc/hosts на компьютере-шлюзе. Я на собственном опыте убедился, что этот файл зачастую не соответствует требованиям dnsmasq. Не забывайте лишний раз взглянуть на этот файл, прежде чем соберетесь изменять локальную конфигурацию сети, заданную на компьютере-шлюзе!

Как минимум, в /etc/hosts должны содержаться две приведенные далее строки. Вторая строка имеет определяющее значение для соотнесения имени локального компьютера (например, mars или mars.sol в данном случае) и IP-адреса в локальной сети (здесь 192.168.0.1)! Часто далее присутствуют и другие строки, описывающие конфигурацию IPv6, но мы не будем рассматривать это.

```
# /etc/hosts на компьютере-шлюзе
127.0.0.1    localhost
192.168.0.1  mars       mars.sol
```

СОВЕТ

Если возникнут проблемы, проверьте конфигурацию брандмауэра! Сетевой интерфейс локальной сети не должен блокироваться брандмауэром. В зависимости от программы, используемой в качестве брандмауэра, сетевой интерфейс должен быть соотнесен с внутренней (то есть незащищенной) зоной.

Файл dnsmasq.conf

Конфигурация dnsmasq выполняется в файле /etc/dnsmasq.conf. Этот файл, по умолчанию присутствующий в дистрибутиве, также служит техническим документом и содержит около 400 строк комментариев. Немногочисленные важные команды среди этого множества комментариев можно не заметить. Переименуйте файл в dnsmasq.conf.orig или создайте с помощью команды grep новый файл, не содержащий комментариев:

```
root# cd /etc
root# cp dnsmasq.conf dnsmasq.conf.orig
root# grep -v '^#' /etc/dnsmasq.conf.orig | cat -s > dnsmasq.conf0
```

Для конфигурации dnsmasq очень важен не только файл /etc/dnsmasq.conf, но и /etc/hosts и /etc/resolv.conf.

Запуск/перезапуск

Как и большинство других программ, представленных в этой и в следующих главах, dnsmasq является демоном (системной службой). В некоторых дистрибутивах эта программа запускается сразу же после установки. Если этого не происходит, программу нужно запустить вручную. Все изменения конфигурации вступают в силу только после перезапуска:

```
root# /etc/init.d/dnsmasq restart
```

Кроме того, от конкретного дистрибутива зависит, должна ли программа в дальнейшем автоматически запускаться при запуске компьютера. В Fedora, Red Hat

и SUSE по умолчанию этого не происходит. Обзор команд, с помощью которых можно управлять стартом dnsmasq в различных дистрибутивах, дается в разделе 4.5.

Минимальная конфигурация

Файл dnsmasq.conf достаточно хорошо функционирует уже при показанной ниже минимальной конфигурации. При этом программа играет роль кэша сервера имен для работы в Интернете и предоставляет клиентам IP-адреса из диапазона между 192.168.0.2 и 192.168.0.250. Клиенты сохраняют свои хост-имена.

```
# /etc/dnsmasq.conf (минимальная конфигурация)
domain-needed
bogus-priv
interface=eth1
dhcp-range=192.168.0.2,192.168.0.250,24h
```

Коротко объясню отдельные ключевые слова: domain-needed и bogus-priv не позволяют dnsmasq передавать хост-имена и IP-адреса локальных компьютеров серверу имен вашего интернет-провайдера (таким образом, управляемый провайдером сервер имен отвечает только за имена и адреса из Интернета, но не из локальных сетей).

Ключевое слово interface указывает, что dnsmasq, выполняющая функцию DHCP-сервера, должна отвечать только на те запросы, которые приходят с интерфейса eth1, который при рассматриваемой топологии отвечает за LAN.

Слово dhcp-range указывает, какой диапазон адресов должен использовать DHCP-сервер для ответа на DHCP-запросы. Выданные адреса остаются присвоенными конкретным компьютерам в течение 24 часов, а затем должны быть обновлены клиентом.

Адреса сервера имен и шлюза не требуют дополнительной конфигурации. Программа dnsmasq сама интерпретирует /etc/resolv.conf и обращается к указанному в этом файле серверу имен. Клиентам DHCP передаются адреса сервера имен и шлюза — отдельно на IP-адрес каждого компьютера в локальной сети.

Применение локального сервера имен

При использовании вышеуказанной минимальной конфигурации dnsmasq может разрешать локальные адреса лишь в том случае, если в /etc/hosts содержится необходимая информация. Сервер имен не будет знать адреса, которые были присвоены динамически через DHCP. Чтобы dnsmasq также могла функционировать в качестве сервера имен для клиентов в сети LAN, добавьте в dnsmasq.conf следующие строки и укажите dnsmasq заново считать конфигурационный файл. Доменное имя рассматриваемой сети — sol.

```
# /etc/dnsmasq.conf  (Применение в качестве сервера имен для адресов в локальной сети)
...
local=/sol/
domain=sol
expand-hosts
```

Ключевое слово local указывает, что запросы адресов, поступающие из этого домена, должны обрабатываться непосредственно dnsmasq (а не сервером имен провайдера интернет-услуг).

Слово domain указывает, что dnsmasq должна присвоить DHCP-клиентам заданное доменное имя. Это имя должно быть таким же, как и указанное в local.

Наконец, благодаря использованию expand-hosts запросы сервера имен, в которых не указан домен, автоматически дополняются доменным именем, заданным в domain. Итак, если выполнить команду ping uranus, то dnsmasq возвратит адрес uranus.sol.

Для того чтобы dnsmasq поименно различала своих клиентов, то есть компьютеры, которые запрашивают IP-конфигурацию через DHCP, необходимо, чтобы эти компьютеры сообщили программе свои хост-имена через DHCP. В большинстве дистрибутивов и конфигурационных программ для работы с LAN/WLAN, в том числе в Network Manager, эта настройка действует по умолчанию. Будьте внимательны, применяя на компьютерах-клиентах Fedora, Red Hat или старые версии Debian и Ubuntu. Соответствующие советы о том, как проводить конфигурацию, даются в подразделе «Клиентская конфигурация».

Статические адреса и хост-имена

Команду dnsmasq можно сконфигурировать и так, чтобы она настраивала хост-имена клиентов. Статическое соотнесение хост-имени и IP-адреса осуществляется на основании MAC-адреса клиента. Такая адресация удобна прежде всего при работе с устройствами, настраивать для которых хост-имена достаточно нелегко — это касается, например, сетевых принтеров. Для конфигурации используется ключевое слово dhcp-host. Следующий листинг — это запись о сетевом принтере pluto.

```
# /etc/dnsmasq.conf     (Статическое присвоение адреса)
...
dhcp-host=00:c0:ee:51:39:9f,pluto,192.168.0.254
```

Самое сложное при таком варианте конфигурации — определить MAC-адрес (адрес для *управления доступом к среде*) клиента. Роль такого адреса играет уникальный ID-номер, который имеется у каждого Ethernet-контроллера. В Linux для отображения MAC-адреса используется команда iconfig. В остальном вам потребуется просто подключить устройство к LAN и выполнить динамическую конфигурацию с помощью dnsmasq. Программа протоколирует все динамические IP-адреса вместе с хост-именами и MAC-адресом в файл /var/lib/misc/dnsmasq.leases. Таким образом, MAC-адрес можно узнать и из этого файла.

По умолчанию адреса, динамически присвоенные с помощью DHCP, действительны в течение 24 часов. Но для сетевых принтеров и другой аппаратуры, работающей в режиме с резервированием, такого промежутка времени часто бывает недостаточно. Если принтер не используется в течение более чем 24 часов, команда dnsmasq «считает», что прибор выключен, и как бы «забывает» о нем. Чтобы этого не происходило, можно добавить к данным о сроке действия адреса дополнительную строку, в которой указывается время. Благодаря infinite адрес остается действителен сколь угодно долго.

```
# /etc/dnsmasq.conf  (Статическое присвоение адреса без ограничения времени)
...
dhcp-host=00:c0:ee:51:39:9f,pluto,192.168.0.254,infinite
```

В man dnsmaq и /etc/dnsmasq.conf.orig описано несколько других вариантов синтаксиса для dhcp-host. Там рассказано, как создать жесткую связь между хост-именем и IP-адресом, полностью заблокировать определенные MAC-адреса и т. д.

DNS для локального компьютера

По умолчанию dnsmasq выполняет функции сервера имен для всех компьютеров в сети, но на самом шлюзе эта программа не работает! Причина заключается в том, что на локальном компьютере используется тот сервер имен, который задан в /etc/resolv.conf. Как правило, этот файл указывает на сервер имен вашего интернет-провайдера или роутера.

Если на компьютере-шлюзе должны работать другие серверные программы (файловый сервер, Kerberos и т. д.), необходимо, чтобы шлюз знал клиентов «поименно», то есть также использовал dnsmasq как сервер имен. Чтобы этот механизм работал, обратите внимание на следующие моменты.

- /etc/resolv.conf должен указывать на dnsmasq (то есть на localhost, адрес 127.0.0.1), а не на внешний сервер имен. Внимание: если вы строите соединение с Интернетом по динамическому принципу (LAN + DHCP или модем + PPP), то файл resolv.conf будет перезаписан при любой архитектуре соединения. Этого необходимо избежать: соединение с ADSL-роутером должно быть статическим. Можете изменить конфигурацию PPP, чтобы в описанном случае resolv.conf оставался без изменений.
- Теперь dnsmasq не может интерпретировать resolv.conf, чтобы узнать адрес внешнего сервера имен, поэтому адрес внешнего сервера имен нужно специально указать в dnsmasq.conf с помощью ключевого слова server.

Продемонстрирую это на примере. Допустим, компьютер-шлюз mars с IP-адресом 192.168.0.1 в LAN подключен к ADSL-роутеру через Ethernet-интерфейс eth0. IP-адрес роутера — 10.0.0.138. Чтобы mars мог обмениваться информацией с ADSL-роутером, интерфейс eth0 конфигурируется статически (даже если ADSL-роутер поддерживает работу с DHCP, динамическая конфигурация с применением DHCP недопустима, так как в противном случае при каждом перезапуске компьютера в resolv.conf будут записываться данные DHCP ADSL-роутера!).

```
# в /etc/network/interfaces
...
# Статическое соединение с ADSL-роутером, то есть с Интернетом
auto eth0
iface eth0 inet static
  address 10.0.0.1
  netmask 255.255.255.0
  gateway 10.0.0.138
...
```

В файле /etc/resolv.conf содержится имя локального домена (sol) и IP-адрес локального сервера имен (то есть dnsmasq):

```
# /etc/resolv.conf
search       sol
nameserver   192.168.0.1
```

Еще остается настроить конфигурацию dnsmasq: теперь программа уже не может считывать адрес внешнего сервера имен (для обращения к Интернету) из resolv.conf. Наоборот, она должна игнорировать resolv.conf (параметр no-resolv) и связываться с внешним сервером имен, указанным после ключевого слова server:

```
# /etc/dnsmasq.conf
...
no-resolv
server=10.0.0.138
...
```

Все вместе

Если такой подробный рассказ о различных настройках dnsmasq вас немного запутал, вы можете просмотреть окончательный вариант файла dnsmasq.conf:

```
# /etc/dnsmasq.conf
# Интерфейс для LAN
interface=eth1
# Локальные хосты не предъявляются вышестоящему серверу имен
domain-needed
bogus-priv
# Доменное имя sol в LAN
local=/sol/
domain=sol
expand-hosts
# dnsmasq для компьютера-шлюза (вышестоящий сервер имен = 10.0.0.138)
server=10.0.0.138
no-resolv
# Динамические адреса
dhcp-range=192.168.0.2,192.168.0.250,24h
# Статические адреса
dhcp-host=00:c0:ee:51:39:9f,pluto,192.168.0.254,infinite
```

Журналирование

Программа dnsmasq автоматически заносит все динамически присвоенные IP-адреса в файл /var/lib/misc/dnsmasq.leases. Статические адреса, напротив, не учитываются. В каждой записи этого файла также содержится MAC-адрес и хост-имена клиентов (если они известны). Таким образом, из этого файла очень удобно узнавать MAC-адреса новых клиентов.

Если dnsmasq работает не так, как вы хотели, вставьте в dnsmasq.conf ключевое слово log-queries. Теперь программа будет регистрировать все обращения к серверу имен в /var/log/syslog или /var/log/messages.

Клиентская конфигурация

В принципе сконфигурировать компьютер, который должен получать конфигурацию IP через DHCP, совсем не сложно. Все дистрибутивы Linux и современные версии Windows содержат соответствующий флажок в окне, используемом для конфигурации сети, — я об этом уже говорил. Программа Network Manager, которая все шире используется в Linux, также самостоятельно распознает DHCP-данные LAN- или WLAN-соединения.

Единственный одновременно сложный и важный момент такой конфигурации — работа с хост-именами: как правило, клиент должен иметь фиксированное хост-имя, отправляемое на DHCP-сервер. Большинство клиентских DHCP-инструментов делают это по умолчанию. Исключениями являются Fedora и Red Hat, а также устаревшие версии Debian и Ubuntu, где этого не происходит, так как там *не* применяется Network Manager.

Debian 4, Ubuntu 8.04

Debian до версии 4 включительно и Ubuntu до версии 8.04 по умолчанию не передают хост-имена на DHCP-сервер. Если вам нужно, чтобы этот механизм работал, добавьте в `/etc/dhcp3/dhclient.conf` показанную ниже строку. Тогда `dhclient` не будет заменять последовательность символов `<hostname>` актуальным хост-именем. По умолчанию эта настройка действует, лишь начиная с Debian 5 и Ubuntu 8.10.

```
# Дополнение в /etc/dhcp3/dhclient.conf
...
send host-name "<hostname>";
```

Red Hat, Fedora

Если при работе с Red Hat и Fedora для конфигурации сети применяется не Network Manager, а `system-config-network=neat`, то настроенное хост-имя не передается обратно на DHCP-сервер! Если вам нужно, чтобы этот механизм работал, потребуется *специально* настроить те имена, которые должны пересылаться. Для этого откройте в меню Правка окно свойств сетевого интерфейса (например, `eth0`) и укажите предназначенные для пересылки хост-имена в настройках DHCP.

Как повторно считывать данные DHCP

Разумеется, чтобы испытать DHCP-сервер, не требуется каждый раз заново перезапускать клиенты. Вполне достаточно будет перезапустить сетевые функции. Если вы работаете с Network Manager, просто выберите в меню этой программы интерфейс, отвечающий за сетевые функции. Программа разорвет имеющееся соединение и установит его заново.

Если вы задаете сетевую конфигурацию, не пользуясь Network Manager, то заново запросите с помощью следующих команд информацию о DHCP:

```
root# /etc/init.d/network restart      (Fedora, Red Hat, SUSE)
root# /etc/init.d/networking restart   (Debian, Ubuntu)
```

Затем убедитесь, что все работает. Для этого выполните команду `iconfig` и загляните в файл `/etc/resolv.conf`. Если вы работаете с KDE или Gnome, то после

изменения хост-имени или других основополагающих сетевых параметров вам придется выйти из системы и снова в нее войти.

Даже в Windows можно заново считать данные DHCP без перезапуска. Откройте командное окно и выполните в нем следующую команду:

```
> ipconfig /renew                    (Windows)
```

17.6. Программа dhcpd (DHCP-сервер)

Программа dhcpd — это классический DHCP-сервер. Эта программа используется практически на любых серверах Linux, которые должны предоставлять конфигурационные данные для крупных сетей. По сравнению с dnsmasq эта программа может выполнять значительно более широкий спектр задач по конфигурации.

Условия

В этом разделе предполагается, что у вас установлена программа dhcpd не ниже версии 3. Название пакета отличается в зависимости от дистрибутива, например, в Debian и Ubuntu пакет называется dhcpd3-server, в Fedora и Red Hat — dhcp, а в SUSE — dhcp-server. Программа dnsmasq, напротив, должна отсутствовать, иначе между программами возникнет конфликт.

Кроме того, предполагается, что вы изучили раздел о введении в DHCP (раздел 17.5) и что файл /etc/hosts правильно настроен. Клиентская конфигурация производится так, как это описано в одноименном подразделе предыдущего раздела.

Более подробная информация о конфигурации и технических основах DHCP-сервера сообщается в разделах справки man, посвященных dhcpd, dhcpd.conf и параметрам dhcp, а также на следующем сайте: http://www.isc.org/sw/dhcp/.

Запуск/остановка

DHCP-сервер — это демон (системная служба). Команды для автоматического запуска отличаются в зависимости от дистрибутива, все они упомянуты в разделе 4.5. Сценарий Init-V в Debian и Ubuntu называется dhcp3-server, а в Fedora, Red Hat и SUSE — dhcpd. Обратите внимание, что изменения, внесенные в конфигурацию, вступают в силу только после перезапуска DHCP-сервера:

```
root# /etc/init.d/dhcpd restart          (Fedora, Red Hat, SUSE)
root# /etc/init.d/dhcp3-server restart   (Debian, Ubuntu)
```

Минимальная конфигурация

Программа dhcpd конфигурируется в файле dhcpd.conf, который может находиться в каталоге /etc/dhcp или /etc/dhcp3, в зависимости от дистрибутива и версии. В следующем листинге показана минимальная конфигурация, позволяющая динамически распределять IP-адреса в диапазоне между 192.168.0.2 и 192.168.0.239 (сервер

имеет IP-адрес 192.168.0.1). Адреса от 192.168.0.240 и выше зарезервированы для последующей конфигурации VPN).

```
# /etc/dhcp[3]/dhcpd.conf
# Глобальные параметры
authoritative;
default-lease-time 86400;    # один день
max-lease-time 86400;        # один день
# Ключевые показатели локальной сети
option broadcast-address 192.168.0.255;
option subnet-mask       255.255.255.0;
option routers             192.168.0.1;
option domain-name-servers 192.168.0.1;
option domain-name               "sol";
# Динамическое адресное пространство
subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.2 192.168.0.239;
}
```

Несколько замечаний относительно использованных ключевых слов: `authoritative` означает, что данный DHCP-сервер является официальным DHCP-сервером локальной сети; `default-lease-time` и `maxlease-time` указывают, как долго за компьютером сохраняется выданный IP-адрес (один день, то есть 86 400 секунд). По истечении этого времени клиенты должны заново затребовать конфигурационные данные (это происходит автоматически).

Параметры `broadcast-address` и `subnet-mask` описывают сегмент сети, из которого на DHCP-сервер поступают данные; `routers` указывает адрес компьютера, который служит в локальной сети роутером или шлюзом. В предыдущем примере это тот же самый компьютер, на котором оборудован DHCP-сервер.

Параметр `domain-name-servers` дает список DNS-адресов (максимум — три, разделяются запятыми). Если вы используете локальный DNS, то укажите его адрес. Адреса также передаются клиентам, поэтому от конфигурации DNS можно отказаться. Действующий в настоящий момент адрес сервера имен указан в файле `/etc/resolv.conf`. Если вы также создаете локальный сервер имен (эта операция описана в следующем разделе), укажите с помощью `domain-name-server` адрес локального шлюза. В данном примере такой адрес — 192.168.0.1.

В SUSE необходимо указать в `dhcpd.conf` ключевое слово `ddns-updates`, даже если вы вообще не собираетесь обновлять сервер имен. Для этого вставьте в конфигурационный файл две следующие строки:

```
ddns-updates off;
ddns-update-style interim;
```

Кроме того, укажите в файле `/etc/sysconfig/dhcpd`, с каким интерфейсом должен работать DHCP-сервер (переменная `DHCPD_INTERFACE`).

Если в вашей сети работает WINS-сервер, то его адрес можно указать с помощью параметра `netbiosname-servers`. WINS — это аналог DNS, используемый в Windows. Однако благодаря Samba WINS-сервер можно использовать и при работе в Linux.

Статические адреса

Если вы ставили своей целью просто снабдить все клиенты локальной сети IP-адресами, то все готово! Однако зачастую требуется присвоить некоторым компьютерам, находящимся в локальной сети, статические IP-адреса. В следующих строках приведен такой пример. Сетевому принтеру pluto с MAC-адресом 00:c0:ee:51:39:9f присваивается IP-адрес 192.168.0.254.

Если вы хотите, чтобы хост-имя передавалось клиентам, используйте параметр `use-host-decl-names on`. При использовании стандартных настроек большинство клиентов игнорируют эту информацию. Программа `dhcpd` при соответствующей конфигурации также передает хост-имя серверу имен (см. подраздел «Начало работы и тестирование» раздела 17.7).

```
# /etc/dhcp[3]/dhcpd.conf, продолжение
...
# Передавать клиентам хост-имя
use-host-decl-names on;
# Статические адреса
host pluto {
  hardware ethernet 00:c0:ee:51:39:9f;
  fixed-address 192.168.0.254; }
...
```

При работе с принтерами или другими устройствами, которые выключаются редко, но могут целыми сутками работать в энергосберегающем режиме, задаваемого по умолчанию однодневного срока действия IP-адреса обычно недостаточно. С помощью следующих настроек вы можете сделать срок действия IP-адреса принтера практически неограниченным:

```
# /etc/dhcp3/dhcpd.conf
default-lease-time 86400;   # 1 день
max-lease-time 864000000;   # 1000 дней
...
# Статические адреса
host pluto {
  hardware ethernet 00:c0:ee:51:39:9f;
  fixed-address 192.168.0.254;
  default-lease-time 86400000;
}
...
```

Конфигурация для обслуживания нескольких подсетей

Если на вашем сервере установлено несколько сетевых карт и сервер подключен к нескольким подсетям, то DHCP-сервер можно сконфигурировать так, чтобы он предоставлял информацию сразу по нескольким подсетям. Объявления, касающиеся подсетей, объединяются в группы с помощью команды `group`, чтобы можно было отдельно задать параметры для каждой группы. Правильное соотнесение

отдельных сетей и сетевых интерфейсов самостоятельно обеспечивает dhcpd, на основе IP-адресов. Параметры можно указать либо в начале файла, либо в группах, помещаемых в скобках. В первом случае параметры являются глобальными, а во втором действуют лишь в рамках конкретной группы.

В следующем примере предполагается, что DHCP-сервер должен управлять двумя подсетями — 92.168.0.* и 172.16.0.*; например, первая сеть — обычная локальная, а вторая — WLAN. Сервер также выполняет функции сервера имен и шлюза. На рис. 18.4 показана такая топология сети.

Обращаю ваше внимание на то, что IP-адреса для настройки параметров routers и domain-name-servers в одной сети отличаются от тех же адресов во второй сети. Клиенты из сети 192.168.0.* используют шлюз 192.168.0.1, а клиенты из сети 172.16.0.* — шлюз 172.16.0.1. Оба адреса относятся к локальному серверу, но к его различным сетевым интерфейсам.

Если вы собираетесь создавать в сети WLAN защищенные VPN-соединения (см. раздел 18.5), то при применении некоторых типов и конфигураций VPN параметры routers и domain-name-servers можно не использовать. Это касается, например, PPTP-VPN, где эти данные автоматически настраиваются при создании соединения.

```
# /etc/dhcp[3]/dhcpd.conf
# Конфигурация для двух отдельных сетей
# Глобальные параметры
...
# Сеть 1 (192.168.0.*)
group { # LAN
  option broadcast-address 192.168.0.255;
  option subnet-mask 255.255.255.0;
  option routers 192.168.0.1;
  option domain-name-servers 192.168.0.1;
  # динамическое адресное пространство
  subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.2 192.168.0.239; }
  # статические адреса
  host pluto { ... }
}
# Сеть 2 (172.16.0.*)
group { # WLAN
  option broadcast-address 172.16.0.255;
  option subnet-mask 255.255.255.0;
  option routers 172.16.0.1;                # в PPTP-VPN удалить!
  option domain-name-servers 172.16.0.1;    # в PPTP-VPN удалить!
  # динамическое адресное пространство
  subnet 172.16.0.0 netmask 255.255.255.0 {
    range 172.16.0.2 172.16.0.254; }
}
```

Интерфейсы

В большинстве дистрибутивов DHCP-сервер по умолчанию отвечает на запросы, поступающие со всех сетевых интерфейсов, для которых в dhcpd.conf определены

подходящие адреса. Если хотите, можете специально ограничить круг работы DHCP-сервера только определенными интерфейсами. В Debian и Ubuntu желаемые интерфейсы указываются в переменной INTERFACES в файле /etc/default/dhcp3-server:

```
# Файл /etc/default/dhcp3-server   (Debian, Ubuntu)
...
INTERFACES="eth1 eth2"
```

В Fedora и Red Hat необходимо изменить /etc/sysconfig/dhcpd:

```
# Файл /etc/sysconfig/dhcpd (Fedora, Red Hat)
DHCPDARGS="eth1 eth2"
```

SUSE представляет особый случай: в этом дистрибутиве необходимо указать в /etc/sysconfig/dhcpd интерфейсы для DHCP-сервера, иначе программа не запустится. При настройке ANY сервер работает со всеми интерфейсами:

```
# Файл /etc/sysconfig/dhcpd (SUSE)
DHCPD_INTERFACE="eth1 eth2"
```

Журналирование

Программа dhcpd перечисляет все динамически выделенные адреса в файле /var/lib/dhcp[3]/dhcpd.leases. Кроме того, в нем перечисляются MAC-адреса клиентов. Таким образом, из этого файла удобно узнавать MAC-адреса новых клиентов. Остальные сообщения протоколов сохраняются в /var/log/messages или syslog.

Динамический DNS

С помощью различных ключевых слов ddns можно сконфигурировать DHCP-сервер так, что программа будет переадресовывать имена и IP-адреса новых клиентов локальной сети на сервер имен. Таким образом, сервер имен будет автоматически узнавать имена и IP-адреса всех компьютеров, работающих в локальной сети. Подробности о конфигурации обоих серверов (dhcpd и named) сообщаются в подразделе «Начало работы и тестирование» следующего раздела.

С клиентской точки зрения не важно, какой программой вы пользуетесь — dnsmasq или dhcpd. Некоторые советы о конфигурации клиентов даются в подразделе «Клиентская конфигурация» раздела 17.5.

17.7. Программа bind (сервер имен)

В качестве сервера имен в Linux/UNIX чаще всего применяется программа named, входящая в состав пакета bind9. BIND означает Berkeley Internet Name Domain. В этом разделе мы ограничимся рассмотрением только двух видов конфигурации — кэш для сервера имен и сервер имен для локальной (частной) сети. Более подробно эта тема описана в Интернете: http://www.isc.org/sw/bind/.

Конфигурация в виде кэша для сервера имен

В большинстве дистрибутивов bind настроена так, что программу можно использовать в качестве кэша для сервера имен, не внося в конфигурацию никаких изменений. При этом программа перенаправляет все адресные запросы на так называемый корневой сервер имен и сохраняет результаты в кэше. При повторном получении того или иного запроса его результат сразу же предоставляется системой. Чтобы использовать bind таким образом, вам придется изменить минимум настроек или вообще не потребуется ничего менять! Только в некоторых устаревших версиях Red Hat понадобится дополнительно установить пакет caching-nameserver.

Хотя конфигурационная работа с этой программой и сводится к минимуму, далее я продемонстрирую вам сравнительно сложные конфигурационные файлы bind хотя бы для ознакомления. В этих листингах вы увидите важнейшие ключевые слова, получите общее представление о синтаксисе и о том, как и почему работает bind. В этом разделе мы рассмотрим, как bind работает в Ubuntu. Особенности программы для Fedora и SUSE будут обобщены в конце этого раздела.

Конфигурационные файлы

Для управления bind используется не один, а несколько конфигурационных файлов. Исходной точкой является файл named.conf, который, в зависимости от дистрибутива, может находиться и непосредственно в каталоге /etc, и в /etc/bind/. В этом файле содержатся указания на различные другие файлы, обычно расположенные в каталоге /etc/bind. В некоторых дистрибутивах, напротив, чаще всего используются каталоги var/named или /var/lib/.

Комментарии. Если нужно добавить комментарии, они оформляются тремя способами:

```
# комментарий до конца строки
// комментарий до конца строки
/* многострочный
комментарий */
```

named.conf. Итак, исходным пунктом при конфигурации является файл /etc/bind/named.conf. Имена (ключевое слово file) и каталоги (ключевое слово directory) всех остальных конфигурационных файлов разные, в зависимости от дистрибутива.

```
# Файл /etc/bind/named.conf (Ubuntu)
# Здесь находятся глобальные параметры
include "/etc/bind/named.conf.options";
# Здесь находятся адреса корневых серверов
zone "." {
  type hint;
  file "/etc/bind/db.root"; };
# Правильное обращение с localhost, обратное разрешение адресов для
# петлевого интерфейса (127.0.0.1) и зон широковещания
```

```
zone "localhost" {
  type master;
  file "/etc/bind/db.local"; };
zone "127.in-addr.arpa" {
  type master;
  file "/etc/bind/db.127"; };
zone "0.in-addr.arpa" {
  type master;
  file "/etc/bind/db.0"; };
zone "255.in-addr.arpa" {
  type master;
  file "/etc/bind/db.255"; };
# Индивидуальные дополнения в конфигурацию вносятся здесь
include "/etc/bind/named.conf.local";
```

named.conf.local. Файл `/etc/bind/named.conf.local` — это наилучшее место для внесения индивидуальных дополнений в конфигурацию сервера имен.

named.conf.options. В `/etc/bind/named.conf.options` содержатся различные глобальные параметры для сервера имен. С помощью ключевого слова `forwarders` вы можете указать IP-адрес сервера имен вашего провайдера, а `named` может работать и без указания такой информации, но при этом программа будет вынуждена обращаться к корневому серверу имен. Чем ближе находится сервер имен, тем эффективнее функционирует разрешение имен.

Параметр `query-source-address` определяет порт для обмена информацией между `bind` и другими серверами имен. Современные версии `bind` по умолчанию используют случайные, непривилегированные порты (номера портов — от 1024 до 65 535). Если эти порты блокируются брандмауэром, то команду `query-source` нужно прокомментировать. В этом случае `bind` будет использовать порт 53, как в более ранних версиях.

```
# Файл /etc/bind/named.conf.options
options {
  # Каталог для файлов кэша
  directory "/var/cache/bind";
  # Здесь указывается сервер имен из локальной сети или сервер имен провайдера
  forwarders { 10.0.0.138; };
  # прокомментировать, чтобы использовать для обмена информацией
  # порт 53
  # query-source address * port 53;
  # Соответствие стандарту RFC1035
  auth-nxdomain no;
  listen-on-v6 { any; };
};
```

db.root. Файл `db.root` содержит IP-адреса нескольких центральных серверов имен, которые обращаются к `bind` для разрешения имен. Эти IP-адреса также содержатся в скомпилированном коде `bind`, поэтому `bind` может работать и без файла `db.root`. И все же этот файл рекомендуется использовать, так как IP-адреса центрального DNS-сервера могут измениться. О том, как создается и обновляет-

ся db.root, рассказано в подразделе «Техническая поддержка» этого раздела. Далее приведен фрагмент этого файла:

```
; Файл /etc/bind/db.root
...
A.ROOT-SERVERS.NET.     3600000   A       198.41.0.4
A.ROOT-SERVERS.NET.     3600000   AAAA    2001:503:BA3E::2:30
B.ROOT-SERVERS.NET.     3600000   A       192.228.79.201
C.ROOT-SERVERS.NET.     3600000   A       192.33.4.12
...
```

Файлы db.*. Файлы db.local, db.0, db.1 и db.127, отличающиеся запутанным синтаксисом, описывают, как должен работать сервер имен при разрешении имен, получаемом с интернет-адресов и адреса localhost. Например, далее продемонстрирован файл db.local:

```
; Файл /etc/bind/db.local
; Конфигурация для петлевого интерфейса
$TTL  604800
@     IN    SOA    localhost.   root.localhost.  (
                          1          ; Serial
                     604800          ; Refresh
                      86400          ; Retry
                    2419200          ; Expire
                     604800 )        ; Negative Cache TTL
;
@     IN    NS     localhost.
@     IN    A      127.0.0.1
@     IN    AAAA   ::1
```

Fedora, Red Hat

Центральный конфигурационный файл в Fedora и Red Hat называется /etc/named.conf. В файле /etc/named.rfc1912.zones содержатся все настройки, необходимые для того, чтобы демон named работал как кэширующий сервер имен. Все остальные конфигурационные файлы находятся в каталоге /var/named. Список корневых DNS расположен в /var/named/named.ca.

Самый важный момент базовой конфигурации заключается в том, что запросы сервера имен принимаются только localhost! Чтобы named также реагировал на запросы других компьютеров, находящихся в локальной сети, потребуется внести два изменения: во-первых, добавьте в listen-on IP-адрес сервера, работающего в LAN (в данном случае 192.168.0.1), во-вторых, укажите в allow-query адресное пространство локальной сети (в данном случае 192.168.0.0/24).

```
# Файл /etc/named.conf (Fedora)
options {
  listen-on port 53 { 127.0.0.1; 192.168.0.1; };    # Внимание!
  listen-on-v6 port 53 { ::1; };
  directory "/var/named";
  dump-file "/var/named/data/cache_dump.db";
  statistics-file "/var/named/data/named_stats.txt";
```

```
  memstatistics-file "/var/named/data/named_mem_stats.txt";
  allow-query { localhost; 192.168.0.1/24; };       # Внимание!
  recursion yes;
  dnssec-enable yes;
  dnssec-validation yes;
  dnssec-lookaside . trust-anchor dlv.isc.org.;
};
logging {
  channel default_debug {
    file "data/named.run";
    severity dynamic;
  };
};
zone "." IN {
  type hint;
  file "named.ca";
};
include "/etc/named.rfc1912.zones";
include "/etc/named.dnssec.keys";
include "/etc/pki/dnssec-keys/dlv/dlv.isc.org.conf";
```

SELinux следит за сервером имен и гарантирует, что программа не будет обращаться ни к каким файлам, кроме `/var/named`. По мнению разработчиков Red Hat, такой метод более эффективен, чем использование среды `chroot`. Только если вы отключите SELinux, рекомендуется установить дополнительный пакет `bind-chroot`.

SUSE

В SUSE используются конфигурационные файлы `/etc/named.conf` и `/etc/named.conf.include`. Для хранения собственных конфигурационных файлов и файлов ключей предназначен каталог `/etc/named.d`. В любом случае содержащиеся в нем файлы нужно специально указать в переменной `NAMED_CONF_-INCLUDE_FILES` в `/etc/sysconfig/named`.

Конфигурационные файлы для настройки кэширующей функции сервера имен находятся в каталоге `/var/lib/named`. Список корневых DNS расположен в `/var/log/named/root.hint`.

```
# Файл /etc/named.conf (SUSE)
options {
  directory "/var/lib/named";
  dump-file "/var/log/named_dump.db";
  statistics-file "/var/log/named.stats";
  listen-on-v6 { any; };
  notify no;
};
zone "." in {
  type hint;
  file "root.hint";
```

```
};
zone "localhost" in {
  type master;
  file "localhost.zone";
};
zone "0.0.127.in-addr.arpa" in {
  type master;
  file "127.0.0.zone";
};
include "/etc/named.conf.include";
```

В SUSE named по умолчанию работает в среде chroot в каталоге /var/lib/named. Если в ваших конфигурационных файлах стоят ссылки на другие файлы, не ставьте слово chroot перед названием этого каталога!

Начало работы и тестирование

Прежде чем в первый раз запустить named, необходимо обеспечить правильную конфигурацию файлов /etc/resolv.conf и /etc/nsswitch.conf. Оба эти файла не относятся к пакету bind, но указывают, как другие программы, расположенные на локальном компьютере, должны использовать сервер имен.

Файл resolv.conf

Файл /etc/resolv.conf работает со всеми сетевыми службами компьютера и указывает, какими серверами имен должны пользоваться эти службы. Как правило, в этом файле содержится IP-адрес сервера имен вашего провайдера или IP-адрес вашего роутера. Чтобы шлюз вместо этого использовал специально настроенный отдельный сервер имен, ключевое слово nameserver в /etc/resolv обязано указывать на адрес 127.0.0.1. Остальные настройки должны остаться без изменений.

```
# /etc/resolv.conf
search sol
nameserver 127.0.0.1
```

Если шлюз выходит в Интернет через роутер с собственным DHCP-сервером или через PPP, то файл /etc/resolv.conf обновляется при каждом соединении. В качестве сервера имен при этом используется сервер имен роутера или интернет-провайдера. Таким образом, любые изменения, вносимые в /etc/resolv.conf вручную, при следующем соединении стираются и специально созданный сервер имен никак не влияет на работу шлюза.

Если доступ к Интернету осуществляется через роутер (DHCP), вам больше подойдет статическая конфигурация сети. В некоторых дистрибутивах в программе конфигурации сети предусматривается возможность использования DHCP, но конфигурацию сервера имен все равно придется выполнить вручную. При работе с PPP необходимо удалить параметр usepeerdns из /etc/ppp/options или /etc/ppp/peers/connection_name. В SUSE есть и другая возможность — можно задать в качестве значения переменной NETCONFIG_DNS_POLICY в /etc/sysconfig/network/config

пустую строку, чтобы все сценарии SUSE «самовольно» не вносили в этот файл изменений.

Запуск/остановка

Команды для автоматического запуска демона сервера имен различаются в разных дистрибутивах. Все они перечислены в разделе 4.5. Сценарий Init-V в Debian и Ubuntu называется `bind9`, в Fedora, Red Hat и SUSE — `named`. Изменения, внесенные в конфигурацию, вступят в силу только тогда, когда вы прикажете серверу имен заново считать конфигурационные файлы:

```
root# /etc/init.d/bind9 reload    (Ubuntu, Debian)
root# /etc/init.d/named reload    (Fedora, Red Hat, SUSE)
```

Журналирование

Для проведения первых тестов нужно активизировать функцию журналирования, если это не предусмотрено в вашем дистрибутиве по умолчанию. Для этого необходимо дополнить один из конфигурационных файлов, например `named.conf.local` в Ubuntu, следующими строками:

```
logging {
  channel "named_log" {
    file "/var/log/named/named.log" versions 10 size 500k;
    severity dynamic;
    print-category yes;
    print-severity yes;
    print-time yes;
  };
  channel "query_log" {
    file "/var/log/named/named-query.log" versions 10 size 500k;
    severity debug;
    print-severity yes;
    print-time yes;
  };
  category default { named_log; };
  category queries { query_log; };
};
```

С помощью следующих команд сначала создаются пустые файлы регистрации и гарантируется, что `named` может изменять файлы:

```
root# mkdir /var/log/named
root# touch /var/log/named/named.log
root# touch /var/log/named/named-query.log
root# chown -R bind:bind /var/log/named       (Debian, Ubuntu)
root# chown -R named:named /var/log/named     (Fedora, Red Hat)
```

В Fedora и Red Hat `named` использует имена учетных записей и групп `named` (а не `bind`). Кроме того, в этих дистрибутивах `named` обычно может вносить изменения только в каталог `/var/named/`, поэтому в него нужно помещать и сами файлы регистрации. По умолчанию для этого применяется `/var/named/data/named.run`.

Тестирование сервера имен

С помощью команды ping можно проверить, работает ли разрешение имен интернет-адресов. Еще лучше проверить, исправно ли функционирует сервер имен, с помощью команды host. В следующих строках показано, как система узнает IP-адреса Yahoo! и как узнать IP-адреса компьютеров:

```
root# host yahoo.com
host yahoo.com
yahoo.com has address 69.147.114.224
yahoo.com has address 209.131.36.159
yahoo.com has address 209.191.93.53
yahoo.com mail is handled by 1 d.mx.mail.yahoo.com.
yahoo.com mail is handled by 1 e.mx.mail.yahoo.com.
...
root# host 69.147.114.224
224.114.147.69.in-addr.arpa domain name pointer b1.www.vip.re3.yahoo.com.
```

Конфигурация DHCP

Если вы используете DHCP-сервер и передаете адреса DNS компьютерам в сети с его помощью, нужно изменить соответствующую строку в dhcpd.conf. В дальнейшем клиенты будут связываться уже не с DNS вашего интернет-провайдера, а с локальным сервером имен (который, в свою очередь, устанавливает соединения со внешними DNS, если адреса еще нет в кэше). Укажите в dhcpd.conf IP-адрес того компьютера, на котором вы только что установили DNS (в приведенном примере это адрес 192.168.0.1).

Кроме того, вам потребуется указать в dhcpd.conf доменное имя локальной сети (в примере это имя sol). После проведения этой операции в /etc/resolv.conf DHCP-клиентов появится строка search sol.

```
# Дополнение в начале /etc/dhcpd.conf
option domain-name-servers 192.168.0.1;
option domain-name "sol";
```

С точки зрения клиента, при создании в локальной сети собственного DNS меняется немногое. Просто клиент теперь связывается с внутрисетевым, а не с внешним DNS.

Разрешение имен локальных компьютеров (динамическая конфигурация)

Теперь конфигурацию DNS необходимо дополнить таким образом, чтобы и сервер имен мог распознавать клиентские компьютеры локальной сети. Для этого DHCP-сервер конфигурируется так, чтобы он мог передавать информацию о полученных из Интернета именах и адресах серверу имен. Информация об адресах важна только в пределах локальной сети и не должна попадать «наружу».

Часто DNS выполняет прямо противоположную задачу: если компьютеры, работающие в вашей сети, имеют адреса, действительные во всем Интернете,

то эти адреса будут известны во всем мире. Если ваш провайдер не занимается устранением подобной проблемы, вы сами должны сконфигурировать DNS так, чтобы адреса ваших компьютеров «не разглашались». В большинстве книг и документов, посвященных DNS, такая операция считается обычной практикой. Однако, поскольку в этой книге я заостряю внимание именно на управлении небольшими локальными сетями, более подробно описывать такие случаи я не буду.

Что касается примера, приведенного в этом разделе, локальная сеть sol находится в адресном пространстве 192.168.0.*. DNS работает на компьютере mars.sol с адресом 192.168.0.1. Для расширения требуются два новых файла зон: `db.sol` для разрешения локальных адресов *.sol в IP-адресах (например, merkur.sol→192.168.0.15), а также `db.192.168.0` для обратного разрешения локальных IP-адресов в именах компьютеров (например, 192.168.0.15→merkur.sol).

Все имена файлов, каталогов, учетных записей и групп далее описаны для Ubuntu. В конце раздела приводятся советы по конфигурации Fedora и SUSE.

Создание ключа

Обмен данными между DHCP-сервером и сервером имен защищается общим ключом. Далее поэтапно показано, как подготовить к работе файл ключей. Сначала с помощью `dnssec-keygen` создается файл ключей:

```
root# dnssec-keygen  -a HMAC-MD5 -b 128 -r /dev/urandom \
                     -n USER DHCP_UPDATER
```

Команда `dnssec-keygen` создает в актуальном каталоге два файла, названия которых начинаются с `Kdhcp`. Определяющее значение имеет файл `*.private`, который может выглядеть следующим образом:

```
root# cat Kdhcp*.private
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: 8D9AcWw2G+sIAV42cgMPwg==
Bits: AAA=
```

Теперь последовательность символов `key` переносится в новый файл `ddns.key`. Пример такого файла можно скопировать из `man dhcpd.conf`.

```
# Файл /root/ddns.key
key DHCP_UPDATER {
  algorithm HMAC-MD5.SIG-ALG.REG.INT;
  secret "8D9AcWw2G+sIAV42cgMPwg==";
};
```

Этот файл копируется в каталоги `/etc/bind` и `/etc/dhcp3`, после чего для него настраиваются следующие права доступа:

```
root# cp ddns.key /etc/bind/
root# cp ddns.key /etc/dhcp3/
root# chown root:bind /etc/bind/ddns.key
```

```
root# chown root:dhcpd /etc/dhcp3/ddns.key
root# chmod 640 /etc/bind/ddns.key
root# chmod 640 /etc/dhcp3/ddns.key
root# ls -l /etc/*/ddns.key
-rw-r----- 1 root bind ... /etc/bind/ddns.key
-rw-r----- 1 root dhcpd ... /etc/dhcp3/ddns.key
```

Конфигурация dhcpd

В следующем листинге в обобщенном виде показано, как нужно дополнить /etc/dhcp3/dhcpd.conf для обеспечения динамической конфигурации. Если в dhcpd.conf содержится несколько групп, то дополнения вносятся вне этих групп.

```
# Дополнения в /etc/dhcpd3/dhcpd.conf
...
include                "/etc/dhcp3/ddns.key";
ddns-updates           on;
ddns-update-style      interim;
ddns-domainname        "sol.";
update-static-leases   on;
zone sol. {
  primary 127.0.0.1;
  key "DHCP_UPDATER"; }
zone 0.168.192.in-addr.arpa. {
  primary 127.0.0.1;
  key "DHCP_UPDATER"; }
```

Поясню этот пример. Параметр include считывает файл ключей, создание которого было описано выше; ddns-updates активизирует динамическое обновление сервера имен; ddns-update-style описывает, каким методом должны производиться обновления. Этот метод пока не стандартизирован, поэтому называется *временным* (метод подробно описан в man dhcpd.conf).

Параметр ddns-domainname указывает доменное имя локальной сети — во всех примерах в книге используется доменное имя sol; update-static-leases заставляет DHCP-сервер переадресовывать на сервер имен и ту информацию, которая поступает с хостов, имеющих статическую конфигурацию. (Как правило, DHCP-сервер переадресовывает только те IP-адреса и имена, которые были присвоены динамически. Устройства, имеющие статическую конфигурацию, в таком случае будут «не замечены» системой.)

Оба определения zone в итоге указывают, какие зоны сервера имен следует обновить. При этом primary указывает IP-адрес сервера имен, а key — используемый ключ (определение ключа находится в ddns.key).

Конфигурация bind

Со стороны DHCP конфигурация уже завершена, а с named еще надо поработать. В первую очередь укажем в конфигурационном файле named.conf.local две новые зоны, которые называются «основными» (master zone). Иначе говоря, теперь named не может обращаться ко внешним источникам для разрешения имен, а сама

отвечает за такое разрешение; `notify no` исключает возможность передачи `named` информации из локальной сети на внешнюю DNS.

Обратите внимание, что в названиях зон адреса указываются в обратном порядке. Вместо 192.168.0.* записывается 0.168.192. Синтаксис `named.conf` требует обязательно указывать `.in-addr.arpa`.

Имена файлов зон (в данном случае `db.sol` и `db.192.168.0`) можно выбирать произвольно. Однако по этим именам, разумеется, должно быть понятно, конфигурация каких областей сети задается.

```
# Изменения в /etc/bind/named.conf.local
include "/etc/bind/ddns.key";
# Эти зоны обновляют DHCP-сервер
zone "sol" {
  type master;
  notify no;
  file "/var/cache/bind/db.sol";
  allow-update { key "DHCP_UPDATER"; };
};
zone "0.168.192.in-addr.arpa" {
  type master;
  notify no;
  file "/var/cache/bind/db.192.168.0";
  allow-update { key "DHCP_UPDATER"; };
};
```

Файл db.sol. Строение файлов зон соответствует указанному образцу, причем, как правило, требуется откорректировать лишь немногие строки. Во второй строке файла `/etc/bind/db.sol` указывается `sol.` и `root.sol.`, где `sol` заменяется доменным именем вашей сети. В строке `NS` указывается имя локального сервера имен (оканчивается точкой!), а в следующей строке — его IP-адрес. При использовании статической конфигурации вы можете задать и другие имена компьютеров, а также их IP-адреса. Однако в данном примере этого не требуется. В следующем листинге все изменения, которые следует внести в образец, выделены полужирным:

```
; новая зона файла /etc/bind/db.sol
$TTL 86400
@         IN      SOA     sol.  root.sol.     (
                                1              ; Serial
                           604800              ; Refresh (7 days)
                            86400              ; Retry (1 day)
                          2419200              ; Expire (28 days)
                            86400 )            ; Negative Cache TTL (1 day)
          IN      NS      mars.sol.
mars      IN      A       192.168.0.1
```

db.192.168.0. Первые строки файла `/etc/bind/db.192.168.0` выглядят так же, как и в `db.sol`. Но следующий список построен прямо противоположным образом, то есть строятся взаимосвязи между окончаниями IP-адресов и именами. Поскольку файл ссылается на зону 192.168.0.*, в каждом IP-адресе нужно указать только последнее число.

```
; новая зона файла /etc/bind/db.sol
$TTL 86400
@         IN      SOA     sol.  root.sol.     (
                                1              ; Serial
                           604800              ; Refresh
                            86400              ; Retry
                          2419200              ; Expire
                            86400 )            ; Negative Cache TTL
@         IN      NS      mars.sol.
1         IN      PTR     mars.sol
```

Вы, вероятно, заметили, что файл `named.conf.local` ссылается на файлы зон из каталога `/var/cache/bind`, а не на сам каталог `/etc/bind`, как обычно. Причина заключается в том, что `named` записывает изменения, указываемые DHCP-сервером, в файлы регистрации `*.jnl`. При завершении работы сервера имен изменения сохраняются в файлах `*.db`, так что при новом запуске в вашем распоряжении оказываются самые актуальные версии файлов. Чтобы все работало, сервер имен должен иметь право вносить изменения в файлы `*.jnl`, а они есть лишь в каталоге `/var/cache/bind`. По этой причине теперь нам понадобятся еще две символьные ссылки, которые будут указывать на файлы зон в `/etc/bind`:

```
root# cd /var/cache/bind/
root# ln -s /etc/bind/db.sol .
root# ln -s /etc/bind/db.192.168.0 .
```

Fedora, Red Hat

В Fedora и Red Hat скопируйте `ddns.key` в следующие каталоги:

```
root# cp ddns.key /var/named/
root# cp ddns.key /etc/dhcp/
```

В конфигурационном файле DHCP файл ключей указывается таким образом:

```
# в /etc/dhcp/dhcpd.conf
include  "/etc/dhcp/ddns.key";
```

В Fedora и Red Hat для файлов зон `db.sol` и `db.192.168.0` символьные ссылки не требуются, но эти файлы нужно сохранить в специальном каталоге `/var/named/dynamic`. Тогда изменения, которые потребуется внести в `named.conf`, будут выглядеть так:

```
# в /etc/named.conf
include "/var/named/ddns.key";
zone "sol" {
  type master;
  notify no;
  file "/var/named/dynamic/db.sol";
  allow-update { key "DHCP_UPDATER"; }; };
zone "0.168.192.in-addr.arpa" {
  type master;
  notify no;
  file "/var/named/dynamic/db.192.168.0";
  allow-update { key "DHCP_UPDATER"; }; };
```

SUSE

В SUSE и DHCP-сервер, и сервер имен работают в среде chroot. Поэтому файл ddns.key нужно скопировать в следующие каталоги:

```
root# cp ddns.key /var/lib/dhcp
root# cp ddns.key /var/lib/named
```

Файл конфигурации DHCP не содержит данных о каталогах, а ссылается прямо на ddns.key:

```
# in /etc/dhcpd.conf
include    "ddns.key";
```

Файлы зон db.sol и db.192.168.0 должны находиться в каталоге /var/lib/named/dyn. Ссылки не нужны. Файл named.conf дополняется следующими строками:

```
# в конце /etc/named.conf
include "ddns.key";
zone "sol" {
  type master;
  notify no;
  file "dyn/db.sol";
  allow-update { key "DHCP_UPDATER"; }; };
zone "0.168.192.in-addr.arpa" {
  type master;
  notify no;
  file "dyn/db.192.168.0";
  allow-update { key "DHCP_UPDATER"; }; };
```

Техническая поддержка

Практически с ходу named начинает работать как кэширующий сервер имен, не требуя никаких дополнительных изменений конфигурации. Это происходит благодаря наличию файла корневых серверов, который, в зависимости от дистрибутива, может называться db.root, named.ca или root.hint. В этом файле содержится список центральных DNS (так называемых *корневых серверов*), которые разбросаны по всему миру. Не исключено, что после установки отдельные записи из этого файла будут уже не актуальны, но как минимум часть адресов должна работать.

В любом случае вам потребуется регулярно обновлять этот файл. Для этого необходимо выполнить следующую команду. Она запрашивает корневой сервер A из актуального списка корневых серверов (если этот сервер сейчас недоступен, измените A на B, C и т. д.).

```
root# dig @A.ROOT-SERVERS.NET
; <<>> DiG 9.4.2-P2 <<>> @A.ROOT-SERVERS.NET
...
;; QUESTION SECTION:
;.                          IN      NS
;; ANSWER SECTION:
.                   518400  IN      NS    A.ROOT-SERVERS.NET.
```

```
.                       518400  IN      NS      K.ROOT-SERVERS.NET.
.                       518400  IN      NS      H.ROOT-SERVERS.NET.
...
;; ADDITIONAL SECTION:
A.ROOT-SERVERS.NET.     3600000 IN      A       198.41.0.4
A.ROOT-SERVERS.NET.     3600000 IN      AAAA    2001:503:ba3e::2:30
B.ROOT-SERVERS.NET.     3600000 IN      A       192.228.79.201
...
```

Если все сработает, просто переадресуйте вывод `dig` в файл `/etc/bind/db.root` (сначала создайте резервную копию!).

```
root# cd /etc/bind
root# cp db.root db.root.bak
root# dig @A.ROOT-SERVERS.NET > db.root
```

17.8. Интеграция WLAN в сеть

В принципе в том, чтобы дополнить обычную сеть функциями WLAN, нет ничего сложного. Простейшее решение — применить ADSL-WLAN-роутер. Если вы настроили собственный интернет-шлюз так, как это было описано выше, то можете просто подключить к сетевому концентратору точку доступа к WLAN (рис. 17.2) или использовать вместо обычного концентратора локальной сети WLAN-роутер. В конфигурации шлюза ничего менять не надо. Нужно лишь настроить шифрование WLAN. В настоящее время лучше всего использовать метод шифрования WPA2 и пароль подлиннее.

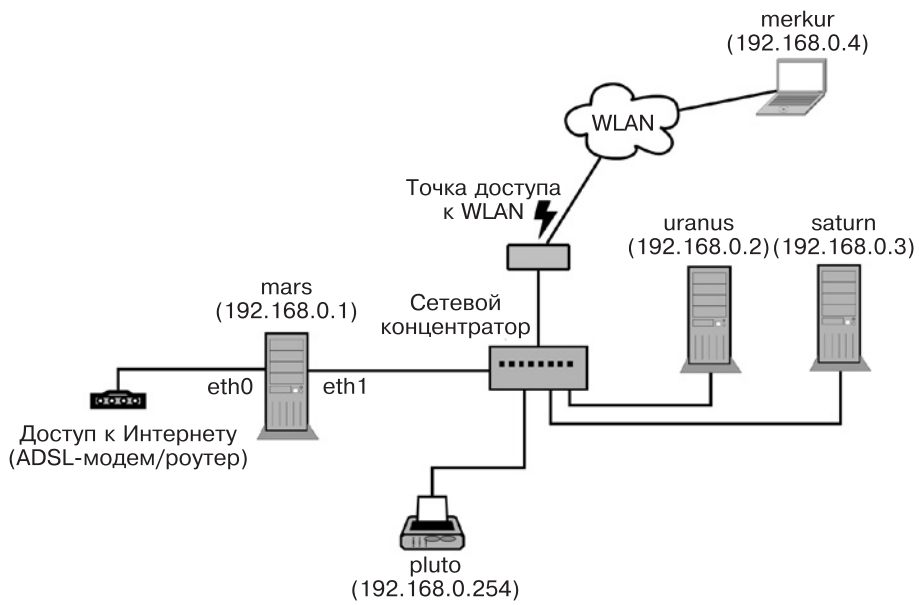

Рис. 17.2. Создание WLAN-соединения через сетевой концентратор

К сожалению, против такого решения есть два аргумента, связанных с безопасностью.

- Все пользователи сети (независимо от того, подключаются они через LAN или WLAN) используют одну и ту же подсеть 192.168.0.*. А использовать некоторые сетевые службы (например, службы для работы с каталогами — NFS или Samba) только в безопасной сети LAN, но не использовать их в незащищенной сети WLAN невозможно.
- Степень безопасности всей сети приравнивается к степени безопасности применяемой WLAN. При шифровании WLAN действует принцип наименьшего общего знаменателя: вы можете использовать метод шифрования WPA2, считающийся в наше время наиболее надежным, лишь в том случае, если все клиенты совместимы с WPA2.

Собственная подсеть WLAN с применением VPN. Можно подключить точку доступа WLAN не только к концентратору локальной сети, но и напрямую к компьютеру-шлюзу (рис. 17.3). Для этого на компьютере-шлюзе нужно установить третью сетевую карту.

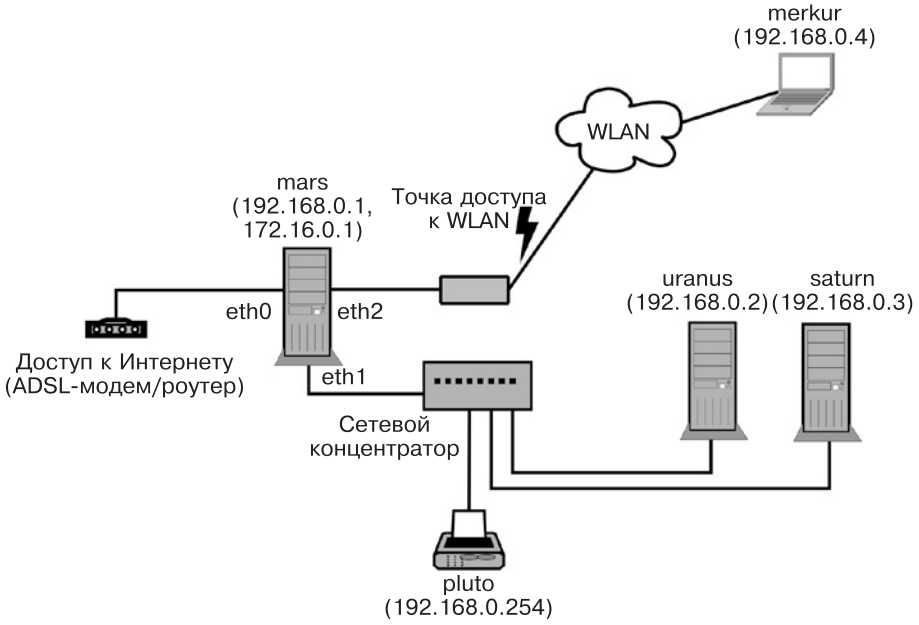

Рис. 17.3. Прямое подключение WLAN к интернет-шлюзу

Существенное преимущество второго варианта перед первым заключается в том, что во втором случае WLAN и LAN используют два совершенно независимых сетевых адресных пространства, например, 192.168.0.* для защищенной сети LAN и 172.16.0.* для незащищенной WLAN. При использовании второго метода непосредственное соединение клиентов LAN и WLAN становится невозможным, а это уже некоторое достижение в области безопасности. Обмен любыми

данными между LAN и WLAN осуществляется через сервер, а на сервере установлен брандмауэр, который с точностью определяет, что разрешено, а что — нет. Кроме того, мы получаем возможность использовать в двух сетях разные сетевые службы, например доступ к Интернету и SSH осуществляется и через LAN, и через WLAN, а доступ к Samba или MySQL возможен лишь через LAN.

Этот вариант становится особенно изящным, когда для защиты соединения WLAN между сетевым сервером и клиентом WLAN создается виртуальная частная сеть (VPN): так обеспечивается «бесшовная» интеграция клиента WLAN и обычной сети без каких-либо компромиссов в области безопасности. О том, как создается VPN, рассказано в разделах 16.10 (клиентская сторона) и 18.5 (сервер).

18 Безопасность

В этой главе рассказано, какие меры необходимо принимать для обеспечения безопасности вашего компьютера или локальной сети. Итак, мы рассмотрим следующие вопросы:

- базовая безопасность сетевых служб (TCP-Wrapper, выполнение в учетных записях с ограниченными правами, средах `chroot`);
- защита трафика TCP/IP путем внедрения фильтрации пакетов на базе `iptables` (брандмауэр);
- безопасный обмен информацией между двумя компьютерами с помощью виртуальных частных сетей (VPN);
- безопасный доступ к Интернету («защита от детей») с помощью фильтра (Squid и DansGuardian);
- безопасность важных сетевых служб с помощью SELinux или AppArmor.

Предлагаемые ниже решения ни совершенны, ни универсальны. Как и во многих других ситуациях, с которыми мы сталкивались в книге, на тему безопасности можно было бы сказать гораздо больше, но размеры этой главы ограничены. Основная моя цель — пробудить ваш интерес к этой теме. Для начала рекомендую посетить следующие сайты:

- http://linuxsecurity.com/;
- http://www.cert.org/;
- http://lwn.net/Security;
- http://www.redhat.com/docs/manuals/enterprise/.

18.1. Основы построения сетей и их анализ

Приступая к обеспечению безопасности компьютера, нужно иметь представление о том, как работают сетевые службы, какие службы действуют в настоящее время, какие порты открыты и т. д. В этом разделе мы рассмотрим основы работы с TCP/IP и некоторые программы, позволяющие анализировать текущее состояние сети, например, составлять список всех действующих сетевых соединений. Прежде всего рассмотрим табл. 18.1, в которой в обобщенном виде представлены важнейшие сокращения.

Интернет-протокол

Практически все распространенные сетевые службы базируются на IP-пакетах. Если, например, интернет-пользователь хочет обратиться к вашему компьютеру через FTP, то компьютер запускает FTP-клиент. Этот клиент посылает на ваш компьютер специальные пакеты. Если на вашем компьютере установлен FTP-сервер, он принимает эти IP-пакеты и реагирует на запрос, пересылая свои IP-пакеты клиенту.

Таблица 18.1. Важнейшие сокращения сетевых терминов

Сокращение	Значение
DNS	Служба доменных имен
HTTP	Протокол передачи гипертекста
ICMP	Протокол управления сообщениями в Интернете
IP	Интернет-протокол
NFS	Сетевая файловая система
TCP	Протокол управления передачей
UDP	Протокол пользовательских датаграмм

Кроме самих данных в IP-пакетах содержится (в том числе) еще четыре важных фрагмента информации: IP-адрес отправителя, порт отправителя, адрес назначения и порт получателя. Благодаря этим данным становится известно, откуда приходит пакет и куда он должен быть направлен.

IP-адреса и порты

Мы уже понимаем, зачем нужен IP-адрес (см. главу 16). IP-порты применяются для идентификации различных служб. Например, для запроса веб-документа обычно используется порт 80. Номера портов — это 16-битные числа. Порты вплоть до 1024 считаются привилегированными и зарезервированы для серверных служб (например, для HTTP-сервера). Остальные порты могут использоваться и клиентами, но и среди них есть несколько номеров, которые не должны применяться клиентом, так как, в свою очередь, зарезервированы для выполнения определенных целей.

Для многих IP-номеров портов в `/etc/services` заданы псевдонимы. В табл. 18.2 перечислены важнейшие номера портов, а также имена, под которыми они обычно используются (если такие имена есть), и краткое объяснение.

IP-протоколы

Существуют различные протоколы для работы с IP-пакетами: большинство интернет-служб используют TCP. Этот протокол требует подтверждения о получении пакета. Но бывают и протоколы, которым такое подтверждение не нужно. К их числу относится, например, ICMP (используется программой `ping`) и UDP (используется DNS и NFS).

Фильтр IP-пакетов

IP-пакеты могут создаваться локальными программами или приходить на компьютер извне — через сетевой или PPP-интерфейс. Ядро решает, как поступить с пакетами. Упрощенно говоря, оно может либо отбросить эти пакеты, либо переадресовать работающим программам или другим интерфейсам. При этом описанные выше характеристики пакетов могут использоваться в качестве критериев для принятия решений. Чтобы применить такой фильтр пакетов на практике, необходимо сообщить ядру, как оно должно поступать с различными IP-пакетами. Для этого в ядре, начиная с версии 2.4, используется команда `iptables`, о применении которой в этой книге есть отдельный раздел — 18.4.

Таблица 18.2. Важнейшие IP-порты

Название	Порт	Функция
ftp	20,21	FTP
ssh	22	SSH
smtp	25	Электронная почта
domain	53	DNS
bootps	67	DHCP
bootpc	68	DHCP
http	80	Сеть
kerberos	88	Kerberos
pop3	110	Электронная почта
portmap	111	Portmap (для NFS)
ntp	123	Время (сетевой протокол синхронизации времени)
netbios-ns	137	Служба имен Microsoft/NetBIOS
netbios-dgm	138	Служба датаграмм Microsoft/NetBIOS
netbios-ssn	139	Служба доступа к файлам Microsoft (SMB, Samba)
imap	143	Электронная почта
https	443	Сеть
microsoft-ds	445	Файловая система CIFS (SMB, Samba)
printer	515	Печать с использованием LPD/LPR
ipp	631	Печать с использованием IPP/CUPS
swat	901	Администрирование Samba
rmi	1099	Удаленный вызов методов (Java)
pptp	1723	PPTP/VPN
nfs	2049	NFS
	3128	Squid (сетевые прокси)
mysql	3306	Сервер базы данных MySQL
	5353	Конфигурация сети с помощью Zeroconf/Bonjour
	5999-6003	X-дисплей
	9100	Сетевой принтер HP-JetDirect

Определение активных сетевых портов

Принцип работы большинства сетевых служб заключается в том, что эти службы «наблюдают» за определенным портом. Если на этот порт приходят IP-пакеты, то конкретная служба занимается обработкой пришедшей информации и отвечает на нее. Пакеты, которые были присланы на «ненаблюдаемые» порты, просто игнорируются и поэтому не представляют опасности. Для того чтобы оценить степень опасности, которой подвергается компьютер, нужно получить список всех наблюдаемых портов (справедливо и обратное — злоумышленник, атакующий компьютер, в первую очередь попытается узнать номера активных портов).

Netstat. При определении сетевой активности локального компьютера очень помогает команда `netstat`. В зависимости от того, с какими параметрами команда вызывается, она выдает массу различной информации.

В первом примере (сервер mars) показаны активные соединения (`established`) или наблюдаемые порты (`LISTEN`). Кратко опишу параметры: `a` отображает неактивные порты, `tu` выводит только ту информацию, которая касается портов TCP и UDP, `pe` дополнительно отображает номер процесса и учетную запись, под которой он выполняется. Ради экономии места вывод сокращен.

```
root# netstat -atupe
Active Internet connections (servers and established)
Proto Local Address          Foreign Addr  State   User    PID/Prog name
tcp   *:nfs                  *:*           LISTEN  root    -
tcp   *:54980                *:*           LISTEN  root    -
tcp   *:ldap                 *:*           LISTEN  root    5842/slapd
tcp   *:3142                 *:*           LISTEN  root    5904/perl
tcp   localhost:mysql        *:*           LISTEN  mysql   5785/mysqld
...
tcp6  [::]:ssh               [::]:*        LISTEN  root    5559/sshd
tcp6  mars.sol:ssh           merkur.so...  ESTAB   root    7729/0
udp   *:nfs                  *:*                   root    -
udp   mars.local:netbios-ns  *:*                   root    6231/nmbd
udp   mars.sol:netbios-ns    *:*                   root    6231/nmbd
udp   *:netbios-ns           *:*                   root    6231/nmbd
udp   mars.local:netbios-dgm *:*                   root    6231/nmbd
udp   mars.sol:netbios-dgm   *:*                   root    6231/nmbd
udp   *:netbios-dgm          *:*                   root    6231/nmbd
udp   *:domain               *:*                   root    5537/dnsmasq
udp   *:55350                *:*                   avahi   5604/avahi-...
...
```

Кратко опишу этот вывод: на тестовом компьютере работают серверы Samba, NFS, Kerberos, LDAP, Dnsmasq, CUPS, MySQL и SSH. Итак, если компьютер напрямую подключен к Интернету (без брандмауэра), то любой потенциальный агрессор злорадно потирает руки. Существует множество программ, безопасность которых несовершенна.

Следующая команда возвращает список активных TCP и UDP-соединений вместе с именами пользователей и названиями процессов:

```
root# netstat -tuep
Active Internet connections (w/o servers)
Proto Local Address   Foreign Address  State       User     PID/Program name
tcp   localhost:57450 localhost:ldap   ESTABLISHED root     6233/smbd
tcp   localhost:ldap  localhost:57450  ESTABLISHED openldap 5842/slapd
tcp6  mars.sol:ssh    merkur.sol:45368 ESTABLISHED root     7729/0
```

lsof. Если требуется узнать, какие программы используют порты TCP и UDP, вам также пригодится команда `lsof`. В форме `derFormlsof -i [протокол]@[хост-имя][:порт]` команда возвращает список процессов, использующих указанные сетевые ресурсы. Две следующие команды отображают все процессы, применяющие протокол UDP или порт 22:

```
root# lsof -i udp
ntpd       3696  ntp    16u   IPv4    9026  UDP *:ntp
ntpd       3696  ntp    17u   IPv6    9028  UDP *:ntp
ntpd       3696  ntp    18u   IPv6    9031  UDP ip6-localhost:ntp
portmap    4745  daemon  3u   IPv4   12931  UDP *:sunrpc
rpc.statd  4764  statd   5u   IPv4   12962  UDP *:700
rpc.statd  4764  statd   7u   IPv4   12970  UDP *:39146
...
root# lsof -i :22
COMMAND  PID USER FD TYPE DEVICE  SIZE  NODE  NAME
sshd    5559 root 3u IPv6        14097   TCP  *:ssh (LISTEN)
sshd    7729 root 3r IPv6        33146   TCP  mars.sol:ssh->merkur.sol:45368
(ESTABLISHED)
```

nmap. Команды `netstat` и `lsof` могут выполняться только на локальном компьютере, то есть злоумышленники не смогут ими воспользоваться. Но враг может применить так называемый сканер портов. Такие программы рассылают пакеты на важнейшие порты компьютера и на основании ответа определяют, какие службы (и какие именно версии этих служб) работают на компьютере. Представленная здесь команда `nmap` — известнейший, но далеко не единственный такой сканер. В большинстве дистрибутивов она устанавливается при первом запуске.

В следующих строках показано, какие результаты `nmap` возвращает для сервера mars. Команда `nmap` выполнялась на другом компьютере в той же локальной сети. Вывод сокращен ради экономии места.

```
root# nmap -v -A mars
Starting Nmap 4.62 ( http://nmap.org ) at 2009-03-20 09:43 CET
Initiating ARP Ping Scan at 09:43
Scanning 192.168.0.1 [1 port]
...
Discovered open port 53/tcp on 192.168.0.1
Discovered open port 21/tcp on 192.168.0.1
...
Completed SYN Stealth Scan at 09:43, 0.29s elapsed (1715 total ports)
Initiating Service scan at 09:43
```

```
Scanning 9 services on mars.sol (192.168.0.1)
...
Host mars.sol (192.168.0.1) appears to be up ... good.
Interesting ports on mars.sol (192.168.0.1):
Not shown: 1706 closed ports
PORT    STATE   SERVICE         VERSION
21/tcp  open    ftp vsftpd      2.0.6
22/tcp  open    ssh OpenSSH     4.7p1 Debian 8ubuntu1.2 (protocol 2.0)
53/tcp  open    domain dnsmasq  2.41
111/tcp open    rpcbind
139/tcp open    netbios-ssn     Samba smbd 3.X (workgroup: SOL)
389/tcp open    ldap OpenLDAP   2.2.X
445/tcp open    netbios-ssn     Samba smbd 3.X (workgroup: SOL)
749/tcp open    rpcbind
2049/tcp open   rpcbind
MAC Address: 00:14:6C:8E:D9:71 (Netgear)
Device type: general purpose
Running: Linux 2.6.X
...
```

Для `nmap` также существует графический пользовательский интерфейс, который в зависимости от дистрибутива может находиться в пакете `zenmap` или `nmap-frontend`.

ВНИМАНИЕ

Сканирование портов воспринимается многими администраторами как покушение на взлом. Никогда не посылайте запросов с таких программ, как nmap, на незнакомые компьютеры! Однако nmap — это удобный и практичный инструмент, позволяющий обнаружить бреши в защите собственной сети.

18.2. Основы защиты сетевых служб

В предыдущем разделе было показано, как можно быстро получить информацию о выполняемых в данный момент сетевых службах. Следующий шаг заключается в том, чтобы как можно надежнее защитить эти службы.

- Деинсталлируйте все сетевые службы, которые вам не нужны. Службы, которые не установлены, не работают и поэтому совершенно безопасны.
- Часто при работе с самыми нужными сетевыми службами бывает достаточно лишь предоставить доступ к службе всего для нескольких определенных клиентов (которые, в частности, находятся в локальной сети). Например, практически исключен случай, в котором вам пришлось бы предоставлять доступ к службам сервера печати в Интернете.

  Что касается Apache, Samba, MySQL и многих других «крупных» служб, то меры защиты нужно предпринимать в соответствующем конфигурационном файле. К счастью, есть несколько сетевых служб, которые для управления доступом обращаются к библиотеке TCP-Wrapper. Эта библиотека позволяет осуществлять централизованную конфигурацию.

- Необходимые сетевые службы должны выполняться с минимальным набором прав. Их выполнением занимаются сценарии Init-V, поставляемые вместе с дистрибутивом. Если это возможно и целесообразно, запустите службы без прав администратора с учетной записи, созданной специально для этих целей, или в среде `chroot`, которая не позволит обращаться к файлам из-за ограничений каталогов.
- В качестве дополнительного уровня защиты рекомендуется использовать специальный брандмауэр, предназначенный для фильтрации пакетов, который, в соответствии с определенными правилами, блокирует пакеты, приходящие из Интернета на адреса различных служб (см. раздел 18.3).
- Ни одна программа не защищена от ошибок. Программные ошибки позволяют потенциальным агрессорам завершать ваши программы с помощью отправки на компьютер специальных пакетов, а в особо тяжелых случаях даже выполнять на вашем компьютере свои команды. Чтобы свести к минимуму связанный с этим риск, ядро может наблюдать за выполнением программ на основании заранее заданных правил. Такой метод называется *мандатным управлением доступом* — Mandatory Access Control, кратко — MAC. В Linux для реализации этой функции используется два метода: SELinux и AppArmor (см. разделы 18.8 или 18.9 соответственно).

Обновления, журналирование. Невозможно обеспечить достаточную безопасность действующей конфигурации компьютера за один раз — только с помощью регулярных обновлений вы сможете поддерживать ваше ПО в актуальном состоянии. Рекомендуется регулярно просматривать файлы регистрации вашего компьютера.

Библиотека TCP-Wrapper

На сервере локальной сети обычно не следует делать все сетевые службы глобально доступными. Вполне достаточно обеспечить доступ к службам в рамках локальной сети. Некоторые сетевые службы обращаются для обеспечения такой базовой защиты к библиотеке TCP-Wrapper. К подобным службам относятся, в частности, SSH-сервер и NFS-сервер, а в SUSE — и служба CUPS. Службы, которые запускаются *демоном интернет-сервисов*, также с пользой применяют библиотеку TCP-Wrapper (см. раздел 14.16).

Файлы /etc/hosts.allow и hosts.deny

Файлы `/etc/hosts.allow` и `/etc/hosts.deny` определяют, с каких компьютеров какими службами можно управлять. Настройки действуют только для тех сетевых служб, которые применяют для контроля доступа библиотеку TCP-Wrapper или команду `tcpd`. По умолчанию оба файла пусты, то есть никакие ограничения не действуют.

Сначала библиотека TCP-Wrapper интерпретирует `hosts.allow`: если в этом файле прямо указано, что доступ разрешен, то осуществляется контроль. В противном случае библиотека интерпретирует `hosts.deny`: если в этом файле доступ запрещен, то клиент получает сообщение об ошибке. Внимание: во всех случаях, для которых отсутствуют правила и в `/etc/hosts.allow`, и в `/etc/hosts.deny`, доступ разрешается!

Чтобы обеспечить максимально безопасную конфигурацию, нужно запретить запуск любых сетевых служб — это делается с помощью команды `all:all` в `/etc/hosts.deny`. Благодаря команде `spawn` любая попытка запустить сетевую службу будет протоколироваться в файле `/var/log/deny.log`.

Теперь из файла `/var/log/deny.log` вы сможете узнать, кто и когда пытается использовать сетевые службы вашего компьютера (запись в этом файле — необязательно сигнал об атаке; вполне возможно, что кто-то просто ошибся при написании IP-адреса).

```
# /etc/hosts.deny
# по умолчанию запретить все службы, любые попытки соединения
# протоколировать
ALL : ALL : spawn (echo Attempt from %h %a to %d at $(date) \
                   >> /var/log/deny.log)
```

Следующий шаг — разрешить в файле `/etc/hosts.allow` использование определенных служб. Приведенная далее в качестве примера конфигурация позволяет:

- получать с локального компьютера (localhost) доступ к любым службам;
- обмениваться информацией по `ssh` с любого компьютера, находящегося в сети (это касается любых компьютеров, подключенных к Интернету);
- использовать NFS и SWAT в локальной сети.

В примере предполагается, что сервер доступен под именами mars и mars.sol, что локальная сеть работает в адресном пространстве 192.168.0.* и что все компьютеры используют домен *.sol. Я просто даю шаблон, по которому вы можете активизировать для работы в локальной сети или в глобальном масштабе и другие сетевые службы, которые ранее отключили:

```
# /etc/hosts.allow
# разрешить выполнение отдельных служб
ALL     : localhost mars mars.sol          : ALLOW
Sshd    : ALL : ALLOW
portmap : 192.168.0.0/255.255.255.0 *.sol  : ALLOW
mountd  : 192.168.0.0/255.255.255.0 *.sol  : ALLOW
swat    : 192.168.0.0/255.255.255.0 *.sol  : ALLOW
# только в SUSE
cupsd   : 192.168.0.0/255.255.255.0 *.sol  : ALLOW
```

Синтаксис, применяемый в `hosts.allow` или `hosts.deny`, на основании данных примеров уже должен быть понятен. Каждая запись состоит из двух частей, разделяемых двоеточием. В первой части указывается служба, во второй — IP-адрес или сетевое имя, в третьей части — итоговое действие. В `man 5 hosts_access` содержится более подробное описание синтаксиса.

Обеспечение поддержки библиотеки TCP-Wrapper

Строка `cupsd` в SUSE требуется потому, что сервер печати CUPS в этом дистрибутиве компилируется с поддержкой библиотеки TCP-Wrapper (чего не скажешь о большинстве других дистрибутивов). Но при этом необходимо учитывать, что доступ к сетевому принтеру дополнительно управляется CUPS-специфичным файлом `/etc/cups/cups.conf` (см. раздел 20.9).

С помощью команды ldd вы легко можете сами определить, использует ли та или иная программа библиотеку TCP-Wrapper (libwrap). Результаты для cupsd и sshd в SUSE 11.2 и Ubuntu 9.10 выглядят так:

```
user$ ldd /usr/sbin/cupsd | grep wrap          (SUSE)
     libwrap.so.0 => /lib64/libwrap.so.0 (0x00007f1f3fece000)
user$ ldd /usr/sbin/sshd | grep wrap
     libwrap.so.0 => /lib64/libwrap.so.0 (0x00007fbf12455000)
user$ ldd /usr/sbin/cupsd | grep wrap          (Ubuntu)
user$ ldd /usr/sbin/sshd | grep wrap
     libwrap.so.0 => /lib/libwrap.so.0 (0xb7f85000)
```

Запуск сетевых служб без прав администратора

Для того чтобы такие программы, как Apache и MySQL, без проблем выполняли за вас вашу работу, нет необходимости запускать их с правами администратора. В большинстве дистрибутивов для таких служб предусмотрены специальные учетные записи, названия которых от дистрибутива к дистрибутиву различаются. Например, в Ubuntu Apache выполняется под учетной записью www-data и поэтому может обращаться лишь к тем файлам, которые доступны для чтения с этой учетной записи. Вы можете в этом убедиться с помощью команды ps axu (но один экземпляр Apache все же должен работать с правами администратора; он отвечает лишь за запуск других экземпляров Apache и не выполняет никаких других задач).

```
root# ps axu | grep apache2
root ... /usr/sbin/apache2 -k start
www-data ... /usr/sbin/apache2 -k start
www-data ... /usr/sbin/apache2 -k start
...
```

Поскольку сценарии Init-V, как правило, выполняются с правами администратора, для запуска сетевого демона с другой учетной записи требуется специальный механизм. В простейшем случае процесс необходимо запустить в форме su *имя учетной записи* -c *демон*. Но для большинства сетевых процессов все же предусмотрены более тонкие механизмы, в ходе которых программа инициализируется с правами администратора и только потом переходит к работе с учетной записью, для которой определено меньшее количество прав. В некоторых программах, например в syslog, предусмотрен специальный параметр, позволяющий указать нужную учетную запись. У Apache, MySQL и некоторых других серверных служб, которые запускаются в нескольких экземплярах, управляющий процесс сохраняет за собой права администратора. Но обычно такой процесс выполняет нетипичные задачи (как правило, запуск и остановку других экземпляров).

Запуск сетевых служб в среде chroot

Команда chroot *каталог команда* запускает указанную команду, причем *каталог* используется в качестве корневого каталога. Команда может обращаться только к тем

файлам, которые находятся в пределах этого каталога. Чтобы гарантировать, что программа не сможет «выбраться» из своей «клетки», ее нужно выполнять с учетной записи, имеющей ограниченные права (то есть учетная запись root не подходит).

Правда, на практике сетевые службы запускаются не с помощью `chroot`, а посредством специального параметра, предназначенного для указания `chroot`-каталога. В этом каталоге должны находиться все необходимые библиотеки, конфигурационные файлы и т. д. Все эти файлы в нужный каталог перед запуском скопирует сценарий Init-V.

В SUSE по умолчанию DHCP-сервер и сервер имен запускаются в средах `chroot`. В качестве DHCP-сервера в данном случае используется специальная обновленная версия `dhcpd`, в которой `chroot`-каталог можно задавать с помощью дополнительного параметра `-chroot`. Что касается сервера имен, то в данном случае для указания `chroot`-каталога применяется параметр `-t`.

Если за сетевой службой наблюдает SELinux или AppArmor и правила обеспечения безопасности сформулированы правильно, то можно обойтись и без применения среды `chroot`, а если она и используется, то никак не способствует повышению безопасности. В Fedora и Red Hat `chroot`-каталоги по умолчанию не применяются, а система полностью полагается на соблюдение правил SELinux.

18.3. Брандмауэры: введение в проблему

Термин «брандмауэр» у всех на устах, но общепринятого определения этого феномена не существует. Функции брандмауэра могут выполняться оборудованием: в таком случае под брандмауэром обычно понимается компьютер, стоящий на стыке локальной сети и Интернета. Многие ADSL-роутеры могут выполнять простейшие функции брандмауэров.

Нередко брандмауэром может быть и программный пакет, установленный на компьютере и при условии правильной конфигурации повышающий безопасность компьютера. Во многих дистрибутивах содержатся многофункциональные инструменты, предназначенные для настройки конфигурации брандмауэра.

В этой книге под брандмауэром понимается совокупность методов, повышающих надежность обмена информацией, проходящей по TCP/IP через фильтр пакетов. Такой фильтр анализирует все сетевые пакеты, приходящие на компьютер, а также пакеты, уходящие с компьютера в сеть. В зависимости от того, все ли правила соблюдаются, пакеты могут быть пропущены или блокируются. Конфигурация подобного фильтра пакетов подробно рассмотрена в следующем разделе. Но сначала разберемся с терминологией.

Брандмауэры для частных ПК

Еще несколько лет назад стремление защитить домашний ПК брандмауэром могло показаться абсурдным. Но со временем ситуация изменилась: сегодня большинство частных ПК постоянно подключены к Интернету, доступны по фиксированному IP-адресу, назначаемому провайдером, а значит, подвергаются опасности.

Если, к примеру, на компьютере действует SSH-сервер, то злоумышленник может попытаться войти в сеть через этот сервер. Для этого агрессоры используют сценарии, автоматически подбирающие логины, просто подставляя слова из словаря. Таким образом, хороший пароль дорогого стоит!

Другая опасность таится во WLAN: на настоящий момент хорошо защищенными можно считать только те сети WLAN, в которых применяется механизм WPA2, и то лишь при условии, что используемый пароль является достаточно длинным и сложным (см. раздел 16.3).

Вы можете возразить, что атака на ваш компьютер не имеет смысла, так как находящиеся на нем данные вряд ли кого-то заинтересуют. Может быть, и так. Но не каждая атака предпринимается с целью выведать данные, а потом манипулировать ими. Часто злоумышленник хочет установить у вас на компьютере маленькую программку, которой позже сможет воспользоваться. Жертвами подобных атак становятся миллионы компьютеров с Windows, которые могут как бы «удаленно управляться» злоумышленниками.

Брандмауэры для локальных сетей

Обычно корпоративные локальные сети больше нуждаются в обеспечении безопасности, чем домашние ПК. Одновременно создаются лучшие условия для построения нужной инфраструктуры. На практике в фирменной локальной сети за выход в Интернет и обеспечение безопасности часто отвечает отдельный компьютер. Все остальные сетевые службы работают на других компьютерах. Эта концепция изображена на рис. 18.1.

В очень небольших сетях функции брандмауэра и сетевого сервера может выполнять один компьютер (см. рис. 17.1). Но такой подход не является оптимальным, так как на этом компьютере придется запустить и эксплуатировать множество сетевых служб, которые могут быть использованы для нанесения вреда сети. Правда, такое обеспечение безопасности все равно лучше, чем полное отсутствие брандмауэра.

В очень больших сетях часто бывает не один, а даже два брандмауэра. Первый брандмауэр служит лишь для обеспечения базовой безопасности, но является проницаемым для таких интернет-протоколов, как HTTP или FTP. Сетевое пространство в таком случае называется *демилитаризованной зоной* (DMZ). Этот термин означает, что внутри сети предпринимаются лишь ограниченные меры обеспечения безопасности. Как правило, в этой зоне располагается веб-сервер, а также другие сетевые серверы, которые должны быть общедоступны (то есть могут быть найдены через Интернет).

Демилитаризованная зона отделяется от оставшейся части локальной сети вторым брандмауэром. Уже за ним располагаются все другие службы, отвечающие за работу локальной сети и абсолютно недоступные извне. Однако конфигурация многоступенчатого брандмауэра — очень обширная тема, выходящая за рамки этой книги. Руководства по конфигурации таких барьеров даются в специальной литературе.

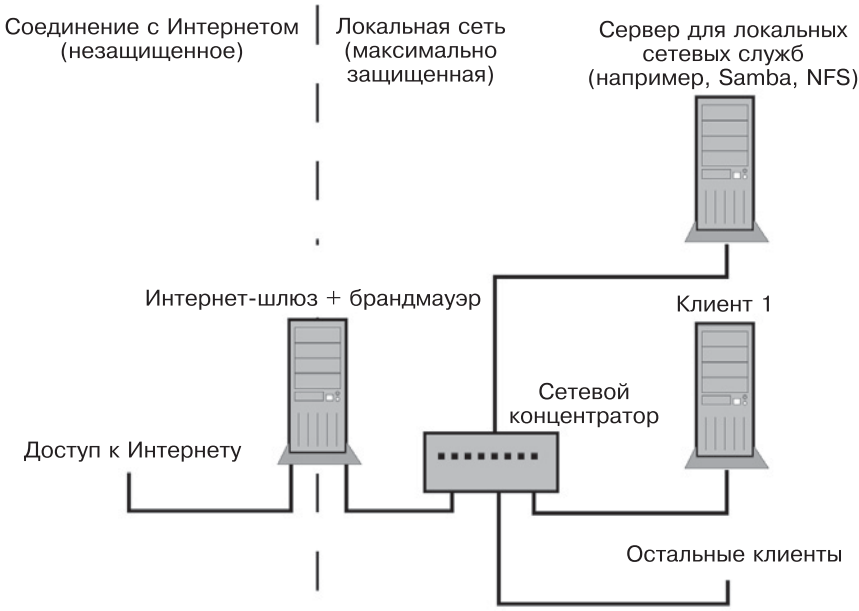

Рис. 18.1. Брандмауэр для локальной сети

Помощь в конфигурации: фильтрация пакетов щелчком кнопкой мыши

В следующем разделе мы рассмотрим, как из бесчисленных команд `iptables` составить фильтр пакетов, имеющий форму сценария. Однако для создания такого сценария вам понадобится внимательно изучить соответствующий материал. Обычному пользователю Linux такие знания определенно не потребуются — во многих дистрибутивах система выполняет эту часть работы за пользователя: удобные конфигурационные инструменты позволяют создавать несложные брандмауэры, просто щелкая кнопками мыши — часть этой работы выполняется уже в ходе установки системы.

Конфигурационные инструменты — это хорошая идея. Однако за совсем простой настройкой фильтра пакетов скрывается больше ноу-хау в фильтрации пакетов, чем вы можете себе представить (если не займетесь вопросом фильтрации пакетов совсем плотно). Часто результат бывает более эффективным, чем собственное решение.

Проблема заключается в самой реализации процесса: созданный таким образом фильтр пакетов, по существу, является *«черным ящиком»*. Если и существует документация о том, как именно функционирует такой фильтр, то весьма скудная. Вы не знаете, от чего вас защищает такой фильтр, и не знаете, какие побочные эффекты он дает при работе. Может случиться так, что вы будете несколько дней ломать голову, почему же в сети не работает вывод на печать, и обнаружите, что

этого просто не допускает фильтр пакетов. В таком случае попытка соответствующим образом откорректировать фильтр пакетов, скорее всего, потерпит фиаско, так как фильтр недостаточно документирован, а конфигурация недостаточно гибка.

Debian. В Debian по умолчанию не предусмотрены брандмауэры, соответственно, отсутствуют и конфигурационные инструменты для их создания.

Fedora, Red Hat. При установке Fedora и Red Hat по умолчанию создается брандмауэр, блокирующий все приходящие извне попытки соединения. Для создания конфигурации запустите через меню Система ▸ Администрирование ▸ Брандмауэр программу `system-config-firewall`. Теперь можно определить конкретные службы (например, SSH) и сетевые интерфейсы (например, интерфейс LAN) как безопасные и освободить их от действия брандмауэра. Для того компьютера, который выполняет функции шлюза, можно дополнительно задать интерфейс маскарадинга. Брандмауэр запускается в рамках процесса Init-V:

- файл правил — `/etc/sysconfig/iptables`;
- сценарий Init-V — `/etc/init.d/iptables`.

В дистрибутивах SUSE брандмауэр также создается по умолчанию, причем интерфейс для связи с Интернетом автоматически присваивается внешней зоне. Конфигурация осуществляется в модуле YaST Безопасность ▸ Брандмауэр. При этом отдельные вкладки открываются щелчком на записи в левой части окна YaST, где обычно расположены файлы справки, а не на отдельных диалоговых окнах, как это бывает в других случаях. Если компьютер служит шлюзом LAN, присвойте интерфейс LAN внешней зоне (это делается на вкладке Интерфейсы) и установите на вкладке Маскарадинг одноименный флажок. Как и в Fedora, брандмауэр запускается в рамках процесса Init-V:

- файл правил — `/etc/sysconfig/SuSEfirewall`;
- сценарий Init-V — `/etc/init.d/SuSEfirewall2*`.

Ubuntu. В Ubuntu по умолчанию брандмауэр не создается, отсутствуют и соответствующие конфигурационные инструменты, но есть команда `ufw` (*«несложный брандмауэр»*). Она позволяет задавать правила для брандмауэра с применением гораздо более простого синтаксиса, чем в `iptables`. Кроме того, предполагается, что в будущих версиях Ubuntu одновременно с установкой сетевых служб будут инсталлироваться соответствующие `ufw`-правила, обеспечивающие безопасность. Разумеется, до этого еще надо дожить: пока `ufw` применяется не так широко.

Если вам уже приходилось работать с фильтрами пакетов, то с `ufw` вы быстро добьетесь того, чего хотите. Кратко опишу синтаксис этой команды (более подробно этот вопрос рассмотрен в `man ufw`): `ufw enable` активизирует брандмауэр. В дальнейшем он будет автоматически активизироваться при каждом запуске компьютера. Команда `ufw disable` снова деактивизирует брандмауэр. Команда `ufw default allow|deny` указывает, как в принципе следует поступать со входящими пакетами: принимать или отклонять (как правило, действует параметр `deny`). Дополнительно, с помощью `ufw allow|deny n` или `ufw allow|deny service` вы определяете правила, которые будут действовать для отдельных портов или протоколов.

Все правила сохраняются в конфигурационных файлах в /etc/ufw. Команда ufw status выдает информацию о текущем состоянии брандмауэра.

```
user$ sudo -s
root# ufw enable
root# ufw allow ssh
root# ufw status
Firewall loaded
To           Action    From
-- ------ ----
22:tcp       ALLOW     Anywhere
22:udp       ALLOW     Anywhere
```

Более подробно эта тема раскрыта на следующих сайтах:

- http://wiki.ubuntuusers.de/ufw;
- http://doc.ubuntu.com/ubuntu/serverguide/C/firewall.html;
- https://wiki.ubuntu.com/UbuntuFirewall.

Firestarter. Это популярная, не зависящая от конкретного дистрибутива программа для конфигурации брандмауэров. При первом запуске открывается ассистент для создания базовой конфигурации. Сначала выбирается интерфейс, через который все компьютеры соединяются с Интернетом. Благодаря флажку Запуск с выбором брандмауэра брандмауэр автоматически активизируется, как только устанавливается соединение с Интернетом. Если ваш компьютер обращается за сетевыми параметрами на DHCP-сервер, вам также потребуется установить флажок Получать IP-адрес от DHCP-сервера.

На втором этапе вы можете конфигурировать ваш компьютер как шлюз. При необходимости Firestarter одновременно может сконфигурировать DHCP-сервер dhcpd, но для этого предварительно требуется установить специальный пакет. Завершите работу ассистента, нажав кнопку Сохранить, но не Закончить, иначе отконфигурированные настройки будут сброшены.

Брандмауэр сразу же становится активен и в дальнейшем автоматически запускается при включении компьютера или при установлении соединения с Интернетом. Теперь ни один компьютер, находящийся в локальной сети или в Интернете, не может начать обмен информацией с вашим компьютером. Часто эта настройка оказывается излишне жесткой. Потребуются дополнительные правила, которые позволят другим компьютерам обмениваться информацией с вашим.

Чтобы задать новые правила, выполните Система ▸ Управление системой ▸ Быстрый запуск, откройте вкладку Безопасность, выберите Правила для поступающего трафика и щелкните кнопкой мыши на области списка Разрешать подключения или Разрешать службу. Только теперь станет активной кнопка Добавить указание, которая открывает конфигурационное диалоговое окно. На рис. 18.2 показано, как определяется правило, позволяющее всем компьютерам с адресами 192.168.0.* обмениваться информацией с SSH-сервером вашего компьютера. Новое правило активизируется нажатием кнопки Применить указание.

Более подробная информация о конфигурации и применении Firestarter дается на сайте http://www.fs-security.com/.

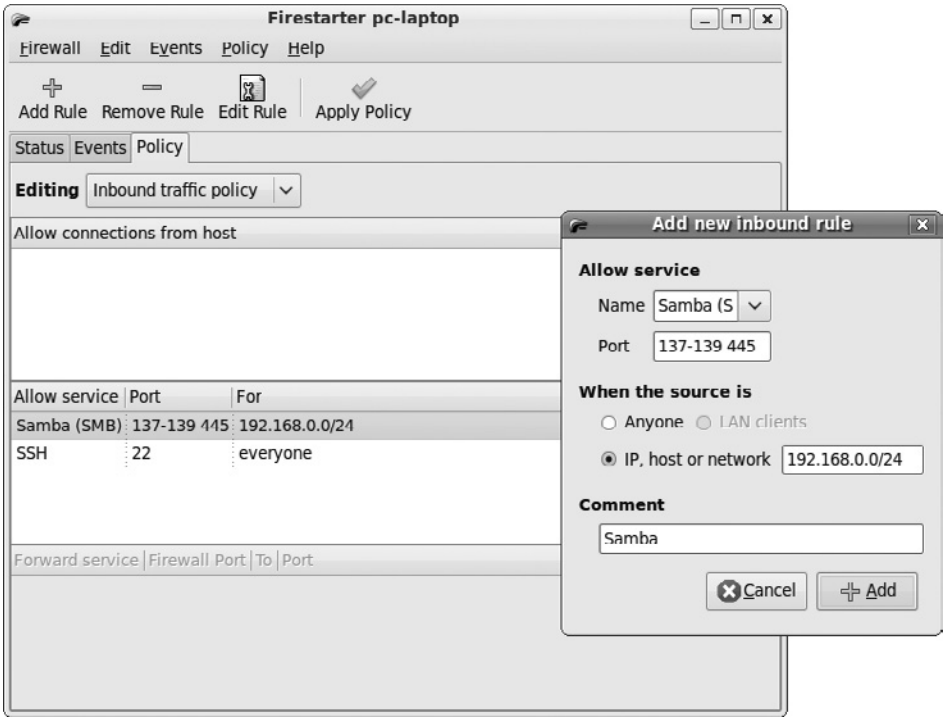

Рис. 18.2. Разрешаем доступ к сетевым каталогам Windows для компьютеров из адресного пространства 192.168.0.*

Ссылки. Кроме упомянутых выше программ есть еще множество инструментов, которые помогут вам настроить конфигурацию брандмауэра. Существуют инструменты с графическим пользовательским интерфейсом и без него. Посмотрите следующие сайты:

- http://www.fwbuilder.org/;
- http://www.simonzone.com/software/guarddog/;
- http://firehol.sourceforge.net/;
- http://www.shorewall.net/.

Количество правил фильтра. Команда `iptables -L | wc -l` позволяет оценить, какое количество правил используется при работе активного в данный момент брандмауэра. Итоговое число позволяет судить о том, насколько сложен брандмауэр, но ничего не говорит о степени его надежности! Лучше всего полностью подавить весь сетевой трафик, а для этого обычно достаточно одного-двух правил.

Сетевой фильтр

Внутри ядра обработкой правил брандмауэра занимается система, называемая *сетевым фильтром*. На рис. 18.3 в очень упрощенном виде показано, какими путями IP-пакеты могут передвигаться в системе фильтрации пакетов. Более под-

робная схема дается по следующему адресу: http://open-source.arkoon.net/kernel/kernel_net.png.

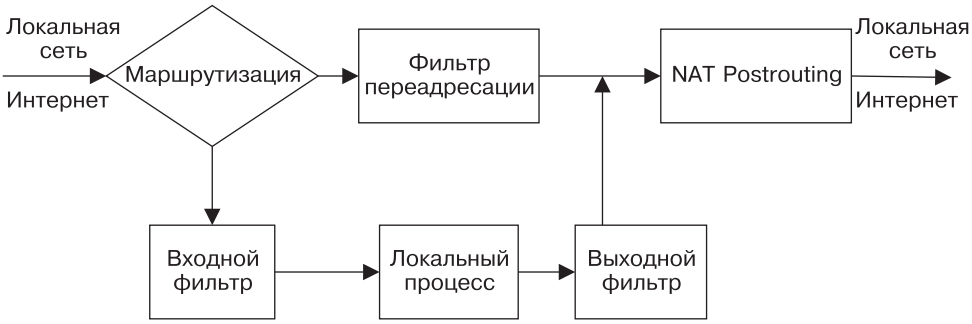

Рис. 18.3. Упрощенное представление системы iptables/netfilter

В следующем списке очень кратко описаны состояния IP-пакета в ядре.

- **Маршрутизация.** Основываясь на информации об IP и адресе порта, ядро решает, должен пакет обрабатываться на локальном компьютере или его следует передать через сетевой интерфейс на другой компьютер (который может находиться как в локальной сети, так и в Интернете).
- **Входной фильтр.** На основании определенных правил проверяется, следует ли обработать пришедший пакет с помощью локальных программ (например, сетевых демонов).
- **Локальный процесс.** В этом окне символически изображены все программы, обрабатывающие IP-пакеты на локальном компьютере либо создающие IP-пакеты (то есть это все сетевые службы, например, ftpd, httpd и т. д.).
- **Выходной фильтр.** На основании отдельных правил определяется, может ли IP-пакет снова покинуть ядро.
- **Фильтр переадресации.** Этот фильтр определяет, какие из пакетов, которые следует только переадресовать (но не обрабатывать), могут пройти через ядро.
- **NAT Postrouting.** Если компьютер должен предоставлять другим компьютерам доступ в Интернет путем маскарадинга, этот механизм выполняет нужные операции с IP-пакетами других компьютеров.

Фильтр пакетов на рис. 18.3 соединяется только с окнами Входной фильтр, Выходной фильтр, Фильтр переадресации и, возможно, NAT Postrouting. В остальных частях изображения описаны сетевые функции ядра или обычные сетевые функции, работающие в локальной системе и никак не связанные с работой фильтра пакетов.

ВНИМАНИЕ

На многих схематических изображениях брандмауэра слева рисуется полный опасностей Интернет, в центре — брандмауэр, а справа — надежная и безопасная локальная сеть. Так вот, на рис. 18.3 все построено иначе! Пакеты, которые слева приходят в компьютер, происходят как из локальной сети, так и из Интернета. То же касается и пакетов, которые справа проходят через брандмауэр.

Действия

За переадресацию пакетов отвечает ядро — независимо от того, приходят они с сетевого интерфейса или создаются на компьютере одной из локальных программ. Ядро может совершать на различных уровнях фильтрующей системы три разных действия.

- `Deny` — переадресация пакетов отклоняется без запроса о подтверждении (можно сказать, что при этом пакет удаляется и больше не существует). Отправитель никогда не узнает, что произошло с его пакетом.
- `Reject` — переадресация пакетов отклоняется с запросом о подтверждении. С пакетом происходит то же, что и в первом случае, но отправитель с другим ICMP-пакетом получает уведомление о том, что его пакет не был принят.
- `Accept` — пакет переадресовывается.

Таблицы

В принципе сетевой фильтр построен так: каждый IP-пакет проходит через различные части ядра, где в определенных пунктах проверяется на основании установленных правил. При соответствии правилам пакет переадресовывается, в противном случае — удаляется или отправляется назад. Сетевой фильтр управляется тремя таблицами.

- Таблица фильтра — обычно в ней содержится вся система правил для отдельных пакетов (брандмауэр).
- Таблица трансляции сетевых адресов — действует лишь в том случае, если в ядре активизирована функция маскарадинга. Она обеспечивает возможность различных изменений адресов (трансляции сетевых адресов) для пакетов, приходящих в ядро извне либо покидающих ядро.
- Таблица mangle — также позволяет выполнять различные операции с IP-пакетами. Таблица служит для выполнения специальных задач и далее в книге рассматриваться не будет.

Цепочки правил (chains)

В каждой из упомянутых таблиц предусмотрено несколько цепочек правил:

- таблица фильтра — Input, Forward, Output;
- таблица трансляции сетевых адресов — Prerouting, Output и Postrouting;
- таблица mangle — Prerouting и Output.

Из восьми этих цепочек правил на рис. 18.3 отображены только четыре самые важные.

ПРИМЕЧАНИЕ

Цепочки правил не зависят друг от друга, то есть существует две разные цепочки Prerouting и три разные цепочки Output. Однако в документации часто пишут, что речь идет просто о цепочке Output, не указывая, к какой именно таблице она относится. В таких случаях всегда имеются в виду цепочки таблицы-фильтра — далее мы увидим, что эта таблица является наиболее важной.

Такая же трактовка касается и команды iptables: там с помощью параметра -t можно указать желаемую таблицу. Если этого параметра нет, то команда будет автоматически применена к таблице-фильтру.

Когда идущий через ядро IP-пакет сталкивается с цепочкой правил, ядро по порядку проверяет его на соответствие всем правилам. Как только одно из определенных правил совпадает с характеристикой пакета, над пакетом выполняется предусмотренное на этот случай действие (например, пакет переадресовывается, удаляется, отправляется обратно и т. д.). Только если с характеристиками пакета не совпадает ни одно из указанных правил, выполняется действие, заданное в системе по умолчанию. В зависимости от конфигурации по умолчанию также может быть задано одно из трех действий: переадресовать, удалить, отправить обратно.

Исходное состояние

В исходном состоянии в ядре активна только таблица-фильтр с тремя цепочками Input, Forward, Output. Ни одна из этих цепочек не содержит дополнительных правил и для всех трех по умолчанию задано действие по переадресации.

Команда iptables

Итак, искусство создания фильтра пакетов заключается в том, чтобы определить для каждой релевантной цепочки фильтра действие, выполняемое по умолчанию, а также несколько правил. Для этого применяется команда `iptables`. Конкретный пример ее использования приведен в следующем разделе. Более подробно эта проблема, как обычно, освещена в Интернете:

- http://www.netfilter.org/;
- http://www.linuxguruz.org/iptables/;
- http://people.netfilter.org/rusty/unreliable-guides/.

Система nftables

Система сетевого фильтра входит в состав ядра с 2001 года. Со временем обнаружились определенные недостатки этой системы, и уже ведется работа над системой следующего поколения nftables. Однако пока еще рано судить, когда и в какой форме она заменит сетевой фильтр. Информацию вы можете найти на сайте http://lwn.net/Articles/324989/.

18.4. Как самостоятельно построить брандмауэр с помощью iptables

В данном разделе будет рассмотрен процесс программирования собственного брандмауэра для фильтрации пакетов для шлюза, то есть для компьютера, предоставляющего всем остальным компьютерам локальной сети доступ в Интернет (см. главу 17). На рис. 18.1 показано исходное положение. Задача брандмауэра заключается в том, чтобы полностью заблокировать порты, представляющие опасность, и разрешать обмен информацией через другие порты только при условии, что такой обмен информацией инициирован изнутри системы (то есть из локальной сети). Наконец, сценарий брандмауэра также занимается активизацией функций маскарадинга.

Пример был выполнен и протестирован в Ubuntu. Если вы работаете с другим дистрибутивом, то вам обязательно нужно сначала деактивизировать брандмауэр, работающий в вашем дистрибутиве! Кроме того, нужно откорректировать сценарий Init-V в соответствии с правилами, действующими в вашем дистрибутиве (см. главу 14).

Сценарий Init-V

При запуске либо остановке работы брандмауэра и, разумеется, на этапе проведения испытаний, вам пригодится сценарий Init-V `myfirewall`. Он сохраняет путь, идущий из `iptables` и `sysctl`, в переменных `IPT` и `SYS`, а затем проверяет, существуют ли три необходимых конфигурационных файла.

Затем сценарий считывает файл `/etc/default/myfirewall`. Только если брандмауэр активизирован, сценарий продолжает работу. Команды для запуска и остановки работы брандмауэра находятся в файле `/etc/myfirewall/*.conf`. Обратите внимание, что правила остановки выполняются и при запуске! Они как бы «сбрасывают показатели» системы сетевого фильтра, благодаря чему гарантируется, что при выполнении правил запуска не возникнет никаких конфликтов с правилами, которые могли быть определены ранее.

```
#!/bin/sh -e
# собственный сценарий Init-V /etc/init.d/myfirewall
### BEGIN INIT INFO
# Provides:        firewall
# Required-Start:  networking
# Required-Stop:
# Default-Start:   S
# Краткое описание: запуск брандмауэра и маскарадинга
### END INIT INFO
# Основные функции
. /lib/lsb/init-functions
IPT=$(which iptables)
SYS=$(which sysctl)
# Конфигурационные файлы
# (Выполняется проверка для /etc/myfirewall/myfirewall-start.conf
# и /etc/myfirewall/myfirewall-stop.conf ...)
if [ ! -e /etc/default/myfirewall ]; then
  echo "/etc/default/myfirewall is missing"
  exit 0
fi
. /etc/default/myfirewall
if [ $MFW_ACTIVE != "yes" ]; then
  echo "Firewall disabled in /etc/default/myfirewall"
  exit 0
fi
# Функции для start, stop и restart
case "$1" in
  start|restart)
    log_begin_msg "Starting firewall and masquerading ..."
    . /etc/myfirewall/myfirewall-stop.conf
    . /etc/myfirewall/myfirewall-start.conf
```

```
    log_end_msg 0
    ;;
  stop)
    log_begin_msg "Stopping firewall and masquerading ..."
    . /etc/myfirewall/myfirewall-stop.conf
    log_end_msg 0
    ;;
  *)
    log_success_msg "Usage: xxx {start|stop|restart}"
    exit 1
    ;;
esac
exit 0
```

Для того чтобы этот сценарий автоматически выполнялся при запуске системы, создайте следующую ссылку:

```
root# cd /etc/rcS.d
root# ln -s ../init.d/myfirewall S41myfirewall
```

Конфигурация. При работе сценарий обращается к трем конфигурационным файлам:

- `/etc/myfirewall/myfirewall-start.conf` — запуск брандмауэра;
- `/etc/myfirewall/myfirewall-stop.conf` — остановка брандмауэра;
- `/etc/default/myfirewall` — базовые настройки.

В `/etc/default/myfirewall` содержится множество базовых настроек, регулирующих работу брандмауэра. Параметр `MFW_ACTIVE` определяет, должен ли брандмауэр активизироваться при запуске системы; `MFW_MASQ` указывает, следует ли активизировать функцию маскарадинга (см. раздел 17.3):

```
# Файл /etc/default/myfirewall
# Запуск брандмауэра yes/no
MFW_ACTIVE=yes
# Запуск маскарадинга: yes/no
MFW_MASQ=yes
# Локальная сеть
MFW_LAN=eth1
MFW_LAN_IP=192.168.0.0/24
# Интернет
MFW_INET=eth0
```

Остановка работы брандмауэра

После завершения работы по администрированию подробнее рассмотрим те правила, по которым работает брандмауэр. Сценарий, останавливающий работу брандмауэра, выглядит так:

```
# Файл /etc/myfirewall/myfirewall-stop.conf
# Сбросить iptables
$IPT -P INPUT ACCEPT
$IPT -P OUTPUT ACCEPT
$IPT -P FORWARD ACCEPT
```

```
$IPT -P POSTROUTING ACCEPT -t nat
$IPT -P PREROUTING ACCEPT -t nat
$IPT -P OUTPUT ACCEPT -t nat
$IPT -F
$IPT -F -t nat
$IPT -X
# Остановить маскарадинг
$SYS -q -w net.ipv4.ip_forward=0
```

Здесь мы видим исходное состояние `iptables`: команда `iptables -P` устанавливает по умолчанию для всех фильтров действие Accept; `iptables -F` удаляет все имеющиеся правила, причем для таблицы трансляции сетевых адресов требуется отдельная команда; `iptables -X` удаляет все заданные пользователем цепочки правил. Теперь в сетевом фильтре разрешен любой сетевой трафик.

Запуск брандмауэра

Разумеется, сценарий `myfirewall-start.conf` гораздо интереснее. Он начинается с правила, разрешающего доступ к SSH-серверу и к Интернету. Это правило имеет смысл лишь в том случае, когда компьютер напрямую соединен с Интернетом (а не через ADSL-роутер или другой компьютер) и администрирование должно осуществляться извне. Если на компьютере предлагается не только SSH, но и другие общедоступные службы, то вы должны специально разрешить и использование этих служб.

Порядок, в котором следуют команды `iptables`, имеет определяющее значение, так как интерпретация правил прекращается сразу же после того, как то или иное правило будет выполнено! Существует две возможные стратегии формулирования правил для брандмауэра: сначала вы можете выборочно разрешить использование отдельных пакетов, а потом заблокировать все остальные пакеты с помощью заключительного правила. Можно также сразу запретить некоторые наиболее опасные пакеты, а в последнем правиле указать, что использование всех остальных пакетов допускается. Именно по второму варианту строится заданный по умолчанию алгоритм работы команды `iptables`, принимающей пакеты при условии, что отсутствуют правила, которые бы прямо это запрещали.

Закрытие портов

Цикл `for` полностью блокирует некоторые порты от всей информации, приходящей из Интернета. При необходимости вы можете сами дополнить этот список портов.

```
# Файл /etc/myfirewall/myfirewall-start.conf
# Разрешить доступ из Интернета к SSH-серверу (порт 22)
$IPT -A INPUT -i $MFW_INET -p tcp --dport 22 -j ACCEPT
# полностью закрыть определенные порты
#   23 (telnet)
#   69 (tftp)
#  135 (Microsoft DCOM RPC)
#  139 (NetBIOS/Samba/т.д.)
#  445 (Файловая система CIFS для Samba/SMB)
#  631 (ipp/CUPS)
```

```
# 1433 (Microsoft SQL Server)
# 2049 (NFS)
# 3306 (MySQL)
# 5999-6003 (X-Displays)
for PORT in 23 69 135 139 445 631 1433 2049 3306 \
            5999 6000 6001 6002 6003; do
  $IPT -A INPUT  -i $MFW_INET -p tcp --dport $PORT -j DROP
  $IPT -A OUTPUT -o $MFW_INET -p tcp --dport $PORT -j DROP
  $IPT -A INPUT  -i $MFW_INET -p udp --dport $PORT -j DROP
  $IPT -A OUTPUT -o $MFW_INET -p udp --dport $PORT -j DROP
done
```

Цепочка правил wall

В следующем примере задается новая цепочка правил под названием `wall`. Эта цепочка — одновременно очень изящная и эффективная защита от новых соединений, устанавливаемых извне. Первое правило `wall` гласит, что должны приниматься все пакеты, которые относятся к уже существующему соединению.

Второе правило позволяет принимать пакеты, инициирующие новое соединение, если это соединение устанавливается не через интернет-интерфейс. Благодаря этому правилу мы, например, можем установить из локальной сети SSH-соединение с компьютером.

Третье правило указывает, что все пакеты, не соответствующие предыдущим правилам, должны отклоняться. Иначе говоря, это правило можно сформулировать как «Все, что не разрешено, — запрещено!». Тогда потенциальному агрессору, которых хочет прорваться на ваш компьютер из Интернета, не удастся даже запустить SSH-соединение (разумеется, то же касается и любых других сетевых служб — HTTP, FTP, Telnet и т. д.).

Две заключительные команды сценария указывают, что все пакеты, проходящие через фильтры входа и переадресации, проверяются на основании правил `wall`:

```
# Файл /etc/myfirewall/myfirewall-start.conf (продолжение)
# Эта цепочка правил блокирует все попытки соединений,
# инициируемые извне (из Интернета)
$IPT -N wall
$IPT -A wall -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -A wall -m state --state NEW -i ! $MFW_INET -j ACCEPT
$IPT -A wall -j DROP
# Эта цепочка правил применяется для INPUT и FORWARD
$IPT -A INPUT -j wall
$IPT -A FORWARD -j wall
```

Маскарадинг

Наконец, активизируем маскарадинг:

```
# Файл /etc/myfirewall/myfirewall-start.conf (продолжение)
# Активизация маскарадинга
if [ $MFW_MASQ = 'yes' ]; then
  $IPT -A POSTROUTING -t nat -o $MFW_INET -s $MFW_LAN_IP -j MASQUERADE
  $SYS -q -w net.ipv4.ip_forward=1
fi
```

18.5. VPN: введение

Виртуальной частной сетью называется сетевое соединение двух или нескольких компьютеров поверх имеющейся сети. Этот метод наиболее замечателен тем, что сеть VPN является защищенной, тогда как лежащая в ее основе передающая среда, например Интернет или WLAN, остается незащищенной (в данном случае под словом «незащищенная» подразумевается, что злоумышленнику сравнительно просто считать передаваемую информацию и даже манипулировать ею).

«Виртуальный» в данном контексте означает, что частная сеть строится на базе уже имеющейся сети. Этот процесс часто называют термином «туннель»: в имеющейся сети создается защищенный снаружи туннель, через который происходит обмен информацией между компьютерами, образующими частную сеть. Трафик в туннеле отличается от остального сетевого трафика тем, что он использует особый протокол, особое шифрование и т. д. Далее перечислены различные методы создания виртуальных частных сетей.

«Частный» здесь означает, что новая сеть лучше защищена от окружающей среды, а значит, и от возможных вторжений. Иначе говоря, сеть является «частной», в отличие от условно-общедоступного Интернета и недостаточно защищенных сетей WLAN.

Пользователь VPN может видеть обе сети (после того, как выполнит `ifconfig`). В примерах, которые приводятся в этой книге, используется базовая (общедоступная) сеть с адресным пространством 192.168.0.*. При конфигурации клиентов очень важно то, что незащищенная базовая сеть применяется только как среда для передачи информации защищенной виртуальной частной сети.

Разумеется, шифрование данных, передаваемых по VPN, требует определенной вычислительной работы и соответствующего администрирования. Один VPN-клиент оказывает на сеть незначительную нагрузку, но общая нагрузка на процессор VPN-сервера может быть очень серьезной, если в виртуальной частной сети одновременно устанавливается много соединений.

Технологии VPN

Существует несколько возможностей реализации VPN. В следующем списке перечислены важнейшие из них, а также упомянуты некоторые тематические сайты.

- **IPsec, Openswan.** IPsec — это протокол для безопасной передачи данных между двумя компьютерами, который может использоваться для создания виртуальной частной сети. IPsec — составная часть нового интернет-протокола IPv6, но в основном пригоден и для работы с действующим в настоящее время IPv4. Недостатком IPsec является его сложность: для решения даже сравнительно простых задач требуется долгая и утомительная конфигурация, поэтому мы не будем рассматривать этот протокол.

  IPsec в версии 2.6 и выше интегрирован прямо в ядро. Для управления функциями IPsec требуется, в числе прочих, команда `ipsec`, входящая в состав пакета `openswan`. IPsec также поддерживается многими другими операционными

системами (в том числе Windows 2000/XP/Vista и выше). Информацию о IPsec и Openswan вы можете найти на следующих сайтах:

- http://www.openswan.org/;
- http://www.ipsec-howto.org/;
- http://lartc.org/howto/lartc.ipsec.html.

- **L2TP.** Протокол туннелирования PPP-соединения уровня 2 — плод совместной работы Microsoft и Cisco. Он объединяет принципы работы PPTP и протокола Cisco L2F (протокол эстафетной передачи на втором уровне). В самом протоколе L2TP не предусмотрено никаких надежных методов аутентификации, поэтому он применяется в комбинации с IPsec (L2TP/IPsec). При этом IPsec отвечает за аутентификацию и шифрование, а L2TP — за управление туннелем. Более подробные сведения о L2TP вы найдете на следующих сайтах:
  - http://www.jacco2.dds.nl/networking/freeswan-l2tp.html;
  - http://sourceforge.net/projects/openl2tp/.
- **PPTP.** Туннельный протокол типа точка-точка совмещает черты протокола точка-точка (PPP), который первоначально разрабатывался для интернет-соединений через модем и ISDN, и зашифрованного туннеля (http://www.poptop.org/). PPTP не только подходит для передачи IP-пакетов, но и совместим с другими протоколами (например, Novell IPX, AppleTalk и т. д.).

  Протокол PPTP был разработан компанией Microsoft, документация по нему находится в открытом доступе (RFC 2637), поэтому интегрировать компьютеры с Windows в PPTP-VPN можно без проблем. Обычно не требуется устанавливать и дополнительные драйверы. Драйвер нужно обновить только в Windows 9x/ME. PPTP также поддерживается различными внешними устройствами (например, некоторыми КПК, совместимыми с WLAN).

  Первые версии PPTP были весьма несовершенны с точки зрения безопасности. Но сегодня некоторые такие недостатки уже исправлены. Если вы используете достаточно длинный пароль (минимум 12 символов!) и применяете систему аутентификации MS-CHAP версии 2, то в большинстве случаев PPTP оказывается достаточно надежным.
- **OpenVPN.** Это достаточно простой VPN-демон, не требующий специальных модулей ядра, в отличие от IPsec и CIPE (http://openvpn.sourceforge.net/). Обмен данными происходит с помощью зашифрованных UDP-пакетов. Для доступа к трафику VPN в Linux используются специальные устройства tun и tap. Значительным достоинством OpenVPN является его сравнительно несложная конфигурация. OpenVPN работает в Linux, Windows (2000 и выше) и в различных UNIX-производных системах (Solaris, *BSD, Mac OS X).
- **PPPD и SSH.** Туннели можно создавать и с помощью SSH. Такой туннель применяется для зашифрованного обмена информацией между двумя компьютерами. Обычно SSH-туннель имеет «ширину» всего один IP-порт. Только при комбинации SSH с PPP-демоном можно направить через SSH-туннель весь IP-трафик. Создаваемая таким образом сеть VPN концептуально похожа на

решение с PPTP, но, в отличие от последнего, более надежна. Комбинация SSH и PPPD подробно рассмотрена на следующих сайтах:

- http://www.oreillynet.com/pub/a/wireless/2001/02/23/wep.html;
- http://www.tldp.org/HOWTO/VPN-HOWTO/.

Перечисленные выше VPN-технологии отличаются друг от друга в первую очередь тем, на каком уровне происходит шифрование сетевого трафика. При использовании IPsec шифрование и построение туннеля осуществляется уже на уровне IP, то есть является *низкоуровневым*. В остальных вариантах для передачи зашифрованных данных, напротив, используются стандартные TCP- и UDP-пакеты. Такой подход к решению проблемы не требует вмешательства в ядро, но может вызывать проблемы, связанные с маскарадингом и некоторыми IP-протоколами (например, FTP).

VPN-технологий много, и выбрать из них не так-то просто. В этой книге мы работаем в основном с конфигурацией PPTP, так как она сравнительно проста.

Топологии сетей VPN

Очевидно, что если существуют различные варианты реализации VPN-соединений, то есть и много возможностей соединения компьютеров по VPN. В следующем списке перечислены некоторые варианты.

- **Соединение точка-точка между двумя компьютерами.** В этом простейшем случае мы просто создаем безопасное соединение между двумя компьютерами поверх незащищенной сети (WLAN, Интернет). Часто можно обойтись и без конфигурации VPN, применив обычный SSH.
- **Сценарий клиент/сервер.** В данном случае многочисленные клиенты должны обращаться за информацией к центральному серверу. Такой метод часто именуется «странствующий воин» (road warrior scenario) и предназначен для сотрудников, которые работают удаленно и часто бывают в пути, а при этом им требуется безопасное соединение с сервером фирмы через ноутбук.
- **Сценарий сервер/сервер.** Для обмена конфиденциальными данными VPN можно устанавливать между двумя серверами, которые географически находятся друг от друга достаточно далеко. Клиенты обеих сетей обмениваются информацией только с теми серверами, с которыми они соотнесены. Такой вариант топологии обычно применяется в тех случаях, когда следует связать два удаленных друг от друга офиса фирмы (например, один находится в Европе, а другой — в США) через обычное интернет-соединение с помощью VPN.

В этой книге будет описан только сценарий клиент/сервер, причем мы предполагаем, что соединение между клиентами и сервером производится через WLAN.

Независимо от применяемого варианта может отличаться еще кое-что, а именно то, какая часть IP-трафика будет направлена через VPN: при работе с WLAN обычно бывает целесообразно пустить весь IP-трафик через VPN. При применении сценария сервер/сервер, напротив, лучше передавать через VPN только те данные, которые важны для обеспечения безопасности (например, для синхронизации баз

данных или файловых систем). Все остальные интернет-службы сервер предоставляет своим клиентам напрямую. Поскольку от данных факторов также зависят функции брандмауэра и роутера, мы имеем практически бесконечное количество конфигурационных возможностей, которые не являются предметом этой книги. По многим VPN-решениям уже есть отдельные книги, в которых те или иные методы рассмотрены детально.

18.6. Реализация VPN с помощью PPTP

В следующих абзацах, а также на рис. 18.4, обобщены условия, необходимые для построения VPN.

- VPN-сервер устанавливается на том же компьютере, на котором находятся интернет-шлюз и брандмауэр. Базовая конфигурация описана в главе 17, посвященной работе шлюза.
- VPN-сервер служит для защиты информации, передаваемой по локальной сети WLAN (сценарий клиент/сервер). VPN-соединение извне (через Интернет) не допускается.
- Точка доступа WLAN подключается к серверу через специальный сетевой интерфейс. Конфигурация DHCP-сервера такова, что на этом интерфейсе компьютерам присваиваются адреса из адресного пространства 172.168.0.* (см. раздел 17.6).
- Трафик, идущий через сетевой интерфейс WLAN, насколько это возможно, блокируется брандмауэром. Принимаются только те пакеты, которые необходимы для DHCP-конфигурации клиентов WLAN и для VPN. WLAN-специфичная часть брандмауэра описана в подразделе «Конфигурация брандмауэра для PPTP-сервера» этого раздела.

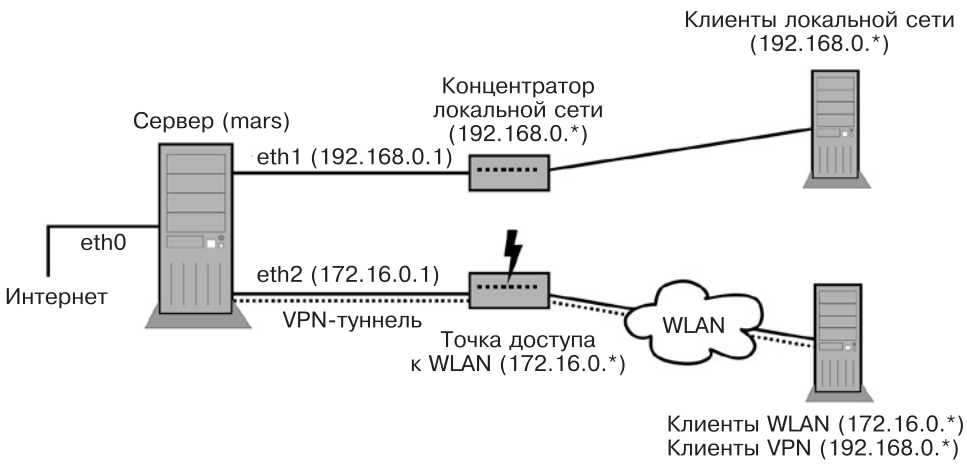

Рис. 18.4. Топология сети VPN

Локальная сеть состоит из двух частей: защищенной (LAN) и незащищенной (WLAN):

- 192.168.0.* — обычная локальная сеть (частная);
- 172.16.0.* — беспроводная сеть (общедоступная, WLAN).

На сервере нужно оборудовать *три* сетевых интерфейса, которые будут работать при подобной конфигурации следующим образом:

- `eth0` — 10.0.0.1 (доступ в Интернет через ADSL-роутер);
- `eth1` — 192.168.0.1 (LAN);
- `eth2` — 172.16.0.1 (WLAN).

Сервер использует сеть WLAN только для работы с DHCP и VPN. Все остальные сетевые функции предоставляются клиентам WLAN через VPN-туннель. Клиенты WLAN через VPN подключаются к сети 192.168.0.*, таким образом, с помощью VPN они органично вплетаются в локальную сеть.

Проблемы, связанные с сервером имен. Принципиальное ограничение PPTP-VPN заключается в том, что хотя VPN-клиенты и могут использовать локальную сеть, их имена при этом остаются неизвестными. Причина этого проста: IP-адреса клиентам раздает не DHCP-сервер, а PPTP-сервер. А сервер имен (то есть `bind`) ничего об этом не узнает. В отличие от `dhcpd`, `pptpd` не поддерживает подобного обмена информацией с сервером имен. В результате сервер имен не может соотнести хост-имена VPN-клиентов и их IP-адреса.

Конфигурация сети на сервере

Теперь мы должны добавить к конфигурации сети, представленной в главе 17, третий сетевой интерфейс — `eth2`, который должен быть сконфигурирован статически и иметь IP-адрес 172.16.0.1. Если вы работаете с Debian или Ubuntu, то строки в `/etc/network/interfaces` должны выглядеть так:

```
# Файл /etc/network/interfaces
...
# Интерфейс для подключения к точке доступа WLAN
auto eth2
iface eth2 inet static
  address 172.16.0.1
  netmask 255.255.255.0
```

Конфигурация DHCP. Теперь DHCP-сервер должен обслуживать интерфейс WLAN eth2 и присваивать клиентам WLAN адреса из диапазона 172.16.0.*. Для этого необходимо отдельно сгруппировать настройки LAN в файле `dhcpd.conf`, а потом добавить в него вторую группу настроек, описывающих работу WLAN. Обратите внимание, что во второй сети не нужно настраивать параметры `routers` и `domain-name-servers`. WLAN в данном случае служит просто носителем для VPN, сам сетевой трафик и трафик из Интернета по ней не проходят. Информация о сервере имен и маршрутизации в данном случае лишь добавит вам проблем.

```
# /etc/dhcp3/dhcpd.conf
# Конфигурация для двух раздельных сетей
```

```
# Глобальные параметры
...
# Сеть 1 (192.168.0.*)
group { # LAN
  option broadcast-address 192.168.0.255;
  option subnet-mask 255.255.255.0;
  option routers 192.168.0.1;
  option domain-name-servers 192.168.0.1;
  # Динамическое адресное пространство
  subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.2 192.168.0.239; }
}
# Сеть 2 (172.16.0.*)
group { # WLAN
  option broadcast-address 172.16.0.255;
  option subnet-mask 255.255.255.0;
  # Динамическое адресное пространство
  subnet 172.16.0.0 netmask 255.255.255.0 {
    range 172.16.0.2 172.16.0.254; }
}
```

Внесенные изменения активизируются следующей командой:

```
root# /etc/init.d/dhcp3-server restart
```

ВНИМАНИЕ

Вместо точки доступа к WLAN вы можете использовать WLAN-роутер, но для этого нужно деактивизировать DHCP-сервер!

Несмотря на долгие эксперименты, мне не удалось заставить PPTP-сервер работать вместе с dnsmasq. Причина в том, что конфигурация недостаточно приспособлена к решению таких задач, как объединение двух сетей (LAN и WLAN) с разными конфигурационными параметрами.

Настройка PPTPD-сервера

Теперь на серверном компьютере нужно установить программу PPTP-сервера, которая обычно находится в пакете `pptpd`. Конфигурационные настройки этой программы выполняются сразу в нескольких файлах.

Файл pptpd.conf

В `/etc/pptpd.conf` содержатся важнейшие настройки для `pptpd`. Обязательно нужно указать локальный IP-адрес сервера и адресное пространство VPN-клиента. В данном случае речь идет об адресах, которые нужно соединить друг с другом посредством VPN. Если вы используете в локальной сети и DHCP-сервер, убедитесь, что динамические адресные пространства серверов DHCP и PPTP не пересекались. При необходимости откорректируйте `dnsmasq.conf`.

```
# /etc/pptpd.conf
option /etc/ppp/pptpd-options    # Параметры pptptd
localip 192.168.0.1
remoteip 192.168.0.240-253
listen 172.16.0.1
```

Параметры pptpd

В /etc/ppp/pptpd-options содержатся все PPP-специфичные параметры. Название этого файла, в принципе, может быть любым, но обязано совпадать с командой option в /etc/pptpd.conf.

```
# /etc/ppp/pptpd-options
name pptpd          # Такое же название должно быть указано
                    # во втором столбце файла chap-secrets
lock
require-mschap-v2
require-mppe-128
refuse-pap
refuse-chap
refuse-mschap
proxyarp
ms-dns 192.168.0.1  # PPTP-клиенты должны использовать этот DNS
```

Необходимо более подробно объяснить значение параметров pptpd: настройка name отвечает за верную интерпретацию логина и пароля PPP. Указанное здесь имя должно совпадать с именем, указанным во втором столбце файла chap-secrets.

Благодаря различным командам require и refuse для идентификации может применяться только протокол **mschap версии 2, разработанный Microsoft. Применяя** его вместе с настройкой require-mppe-128 (128-битное шифрование), мы гарантируем максимально надежное исполнение PPTP.

Благодаря параметру proxyarp IP-адрес клиента VPN вносится в таблицу ARP (протокол разрешения адресов) сервера, поэтому всем остальным компьютерам, находящимся в сети, открывается доступ к VPN-клиенту.

С помощью ms-dns указывается адрес того сервера имен, которым VPN-клиенты должны пользоваться для разрешения сетевых имен. Хотя название этого параметра и начинается с ms (Microsoft), он действует и для клиентов, использующих Linux.

СОВЕТ

Не забывайте, что PPP-демон pppd также учитывает все параметры, содержащиеся в /etc/ppp/options (но параметры из /etc/ppp/pptpd-options имеют приоритет). Если возникнут проблемы с конфигурацией PPTP-сервера, в первую очередь посмотрите файл /etc/ppp/options. В нашем примере предполагается, что файл /etc/ppp/options пуст.

Если появятся проблемы с VPN-соединением, добавьте в pptpd-options еще одну строку с ключевым словом debug. Тогда pptpd и pppd будут записывать в /var/log/messages подробные статусные сообщения и сообщения об ошибках.

Файл chap-secrets

Последний конфигурационный файл называется /etc/ppp/chap-secrets. В нем содержатся имена и пароли, по которым PPTP-сервер опознает клиентов VPN. В этом файле четыре столбца. В первом находится логин, в третьем — пароль. Чтобы пароль был достаточно надежным, его длина не должна быть менее 12 символов.

Во втором столбце располагается опознавательный код для сервера (см. настройку name в pptpd-options). В четвертом столбце можно указать либо звездочку,

либо IP-адрес. В первом случае PPTP-сервер присвоит клиенту IP-адрес из диапазона `remoteip`, во втором — применит указанный IP-адрес, который должен находиться вне диапазона `remoteip` во избежание конфликтов.

```
# /etc/ppp/chap-secrets
# Логин      Имя pptpd    Пароль            IP-адрес клиента
"vpnclient"  "pptpd"      "vpntestpassw"    *
```

Измененная конфигурация вступает в силу после перезапуска PPTP-сервера:

```
root# /etc/init.d/pptpd restart
```

Чтобы демон `pptpd` автоматически запускался и завершал работу при начале и окончании работы компьютера, нужно создать соответствующие ссылки для системы Init-V, которые будут отличаться в зависимости от дистрибутива (см. раздел 4.5).

Статус

Теперь клиентский компьютер должен установить соединение с PPTP-сервером (в разделе 16.10 описано, как это делается). При каждом удачном соединении на сервере создается новый `ppp`-интерфейс, то есть `ppp0`, `ppp1` и т. д.

Состояние этих интерфейсов можно узнать с помощью команды `ifconfig`. В следующих строках кратко обобщены интерфейсы, работающие на компьютере mars.sol. В данном примере `lo` — это петлевой интерфейс, `eth0` — интерфейс для соединения с Интернетом, `eth1` — интерфейс для LAN, `eth2` — интерфейс для WLAN, а `ppp0` — VPN-интерфейс для первого клиента PPTP.

```
root# ifconfig
lo   Protocol:Local loop
     inet Addr:127.0.0.1 Mask:255.0.0.0
     ...
eth0 Link encap:Ethernet-Hardware-Address 00:16:17:cd:c3:81
     inet Adr:10.0.0.1 Bcast:10.0.0.255 Mask:255.255.255.0
     ...
eth1 Link encap:Ethernet-Hardware-Address 00:14:6c:8e:d9:71
     inet Adr:192.168.0.1 Bcast:192.168.0.255 Mask:255.255.255.0
     ...
eth2 Link encap:Ethernet-Hardware-Address 00:4f:4e:0f:8e:a0
     inet Adr:172.16.0.1 Bcast:172.16.255.255 Mask:255.255.0.0
     ...
ppp0 Link encap:Point-to-Point-Connection
     inet Adr:192.168.0.1 P-z-P:192.168.0.240 Mask:255.255.255.255
     ...
```

Конфигурация брандмауэра для PPTP-сервера

Из соображений безопасности рекомендуется закрыть интерфейс WLAN для всего трафика, кроме абсолютно необходимого. Следующие команды пропускают через интерфейс WLAN только пакеты DHCP и PPTP. DHCP использует протокол UDP на портах 67 и 68. PPTP использует протокол TCP на порте 1723, а также собственный протокол GRE (общая инкапсуляция маршрутов).

Требуемые правила брандмауэра легко встраиваются в сценарий, показанный в начале раздела 18.4: для этого сначала нужно дополнить /etc/default/myfirewall и указать в его переменной MFW_WLAN название интерфейса WLAN:

```
# Файл etc/default/myfirewall
...
MFW_WLAN=eth2
```

Кроме того, добавьте в файл /etc/myfirewall/myfirewall-start.conf следующие команды:

```
# Файл etc/myfirewall/myfirewall-start.conf
...
# Принять порты 67 и 68 для UDP (DHCP)
$IPT   -A INPUT   -i   $MFW_WLAN   -p   udp   --dport   67 -j ACCEPT
$IPT   -A OUTPUT  -o   $MFW_WLAN   -p   udp   --dport   67 -j ACCEPT
$IPT   -A INPUT   -i   $MFW_WLAN   -p   udp   --dport   68 -j ACCEPT
$IPT   -A OUTPUT  -o   $MFW_WLAN   -p   udp   --dport   68 -j ACCEPT
# Принять порт 1723 для TCP (PPTP control)
# Принять протокол GRE (идентификатор 47, PPTP data)
$IPT   -A INPUT   -i   $MFW_WLAN   -p   tcp   --dport   1723 -j ACCEPT
$IPT   -A OUTPUT  -o   $MFW_WLAN   -p   tcp   --sport   1723 -j ACCEPT
$IPT   -A INPUT   -i   $MFW_WLAN   -p   gre   -j ACCEPT
$IPT   -A OUTPUT  -o   $MFW_WLAN   -p   gre   -j ACCEPT
# Блокировать все остальные порты WLAN-интерфейса
$IPT   -A INPUT   -i   $MFW_WLAN   -p   tcp   -j DROP
$IPT   -A OUTPUT  -o   $MFW_WLAN   -p   tcp   -j DROP
$IPT   -A INPUT   -i   $MFW_WLAN   -p   udp   -j DROP
$IPT   -A OUTPUT  -o   $MFW_WLAN   -p   udp   -j DROP
```

Конфигурация брандмауэра для PPTP-клиента

Для того чтобы исключить возможность того, что сетевые службы, работающие на клиентских компьютерах (например, Samba или сервер MySQL), передадут через WLAN важные данные, можно защитить интерфейс WLAN брандмауэром. Правила очень похожи на правила для сервера, представленные выше. Отличия есть только в том, как осуществляется обмен данными. Соответствующий сценарий может выглядеть так:

```
#!/bin/sh
IPT=/sbin/iptables
WLAN=eth1
# Принять порты 67 и 68 для UDP (DHCP)
$IPT   -A INPUT    -i   $WLAN   -p   udp   --dport   67   -j ACCEPT
$IPT   -A OUTPUT   -o   $WLAN   -p   udp   --dport   67   -j ACCEPT
$IPT   -A INPUT    -i   $WLAN   -p   udp   --dport   68   -j ACCEPT
$IPT   -A OUTPUT   -o   $WLAN   -p   udp   --dport   68   -j ACCEPT
# Принять порт 1723 для TCP (PPTP control)
# Принять протокол GRE
$IPT   -A INPUT    -i   $WLAN   -p   tcp   --sport   1723   -j ACCEPT
```

```
$IPT   -A OUTPUT   -o   $WLAN   -p   tcp   --dport   1723   -j ACCEPT
$IPT   -A INPUT    -i   $WLAN   -p   gre   -j ACCEPT
$IPT   -A OUTPUT   -o   $WLAN   -p   gre   -j ACCEPT
# Блокировать все остальное
$IPT   -A INPUT    -i   $WLAN   -p   tcp   -j DROP
$IPT   -A OUTPUT   -o   $WLAN   -p   tcp   -j DROP
$IPT   -A INPUT    -i   $WLAN   -p   udp   -j DROP
$IPT   -A OUTPUT   -o   $WLAN   -p   udp   -j DROP
```

18.7. Создание сетевого фильтра с помощью Squid и DansGuardian

Squid — это так называемый кэш-посредник. Это означает, что данная программа сохраняет страницы в особый буфер обмена, расположенный на локальном компьютере, и управляет доступом к ним. Squid может выполнять три функции.

- **Контроль доступа.** Squid может полностью заблокировать определенные страницы для конкретных пользователей, предоставлять доступ к Интернету лишь в определенные часы и т. д. Для распознавания «опасных» сайтов Squid обычно используют вместе с сетевым фильтром DansGuardian. Таким образом, Squid и DansGuardian помогают ограничить доступ в Интернет дома или в общественных местах (например, в школах) и сделать его более безопасным.
- **Кэш.** Если несколько пользователей обращаются к одной и той же веб-странице через кэш-посредник, то такая программа позволяет избежать многократной передачи одного и того же файла. Это ускоряет процесс чтения часто используемых сайтов и уменьшает трафик, идущий к провайдеру. Сегодня, в эпоху Web 2.0, вы все равно не ощутите ни значительного ускорения передачи данных, ни снижения объема закачек.
- **Журналирование/наблюдение.** Squid обеспечивает централизованное наблюдение за локальной сетью, в частности позволяет узнать, кто какие сайты посещает и насколько часто. В некоторых фирмах, где требования к безопасности очень высоки, это может быть оправданно.

В этой главе мы уделим основное внимание первому из трех перечисленных пунктов и лишь бегло коснемся функций кэширования и журналирования. Чтобы избавить вас от возможных разочарований, сразу предупреждаю: Squid и DansGuardian чудес не делают и никак не отменяют воспитательной и просветительской работы с пользователями!

Ссылки. На сайте Squid выложено пособие по конфигурации программы, а также очень подробный документ со списком часто задаваемых вопросов. На немецком языке вышел хороший учебник по Squid, написанный Дирком Дитхардтом (Dirk Dithardt) и доступный как в электронном, так и в печатном варианте:

- http://www.squid-cache.org;
- http://www.squid-handbuch.de/hb/.

Конфигурация и запуск

Работа Squid управляется файлом /etc/squid/squid.conf. Этот конфигурационный файл, поставляемый вместе со Squid, отлично документирован, но ужасающе велик — содержит около 3500 строк! Чтобы вам было легче в нем ориентироваться, сохраните резервную копию оригинального конфигурационного файла, а потом удалите все комментарии (grep с помощью параметра -v может отфильтровать все строки, которые начинаются с символа комментария #, а команда cat удалит все пустые строки). Тогда конфигурационный файл ужмется всего до 50 строк.

```
root# cd /etc/squid
root# mv squid.conf squid.conf.orig
root# grep -v '^#' squid.conf.orig | cat -s > squid.conf
```

По умолчанию squid.conf настроен так, что кэшем может пользоваться только компьютер localhost, причем для буфера обмена резервируется 8 Мбайт оперативной памяти и 100 Мбайт на жестком диске, а каждое обращение к /var/spool/squid заносится в файл регистрации.

Если вы собираетесь использовать Squid в первую очередь как сетевой фильтр, то можно отключить функции кэширования и журналирования. Кроме того, вам следует открыть всем компьютерам локальной сети доступ к Squid. Для этого с помощью acl localnet задайте переменную localnet, а в ней укажите адресное пространство локальной сети. Комбинация с http_access позволяет всем компьютерам локальной сети использовать кэш (вместо localnet можно задавать любое другое имя). В следующем фрагменте изменения, внесенные в конфигурационный файл, поставляемый с Ubuntu, выделены полужирным шрифтом:

```
# /etc/squid/squid.conf
# Пример конфигурации для использования программы в качестве сетевого фильтра
# Хост-имя компьютера, на котором работает squid
visible_hostname mars.sol
# Определения
acl all src 0.0.0.0/0.0.0.0
acl localhost src 127.0.0.1/255.255.255.255
acl to_localhost dst 127.0.0.0/8
acl localnet src 192.168.0.0/255.255.255.0
acl manager proto cache_object
acl SSL_ports port 443          # https
acl SSL_ports port 563          # snews
acl SSL_ports port 873          # rsync
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443         # https
acl Safe_ports port 70          # gopher
acl Safe_ports port 210         # wais
acl Safe_ports port 1025-65535  # Незарегистрированные порты
acl Safe_ports port 280         # http-mgmt
acl Safe_ports port 488         # gss-http
acl Safe_ports port 591         # filemaker
acl Safe_ports port 777         # multiling http
acl Safe_ports port 631         # cups
```

```
acl Safe_ports port 873         # rsync
acl Safe_ports port 901         # SWAT
acl purge method PURGE
acl CONNECT method CONNECT
# Правила доступа
http_access allow manager localhost
http_access deny manager
http_access allow purge localhost
http_access deny purge
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localhost
http_access allow localnet
http_access deny all
icp_access allow all
# Прокси-порт
http_port 3128
# Размер кэша в оперативной памяти 64 Мбайт, кэш на жестком диске не предусмотрен
cache_mem 64 MB
cache_dir
# По умолчанию действует: cache_dir ufs /var/spool/squid 100 16 256
# Деактивизировать функцию журналирования
access_log none
# По умолчанию действует:
# access_log /var/log/squid/access.log squid
# Кэш отсутствует, если в URL есть cgi-bin
hierarchy_stoplist cgi-bin ?
acl QUERY urlpath_regex cgi-bin \?
no_cache deny QUERY
# Срок существования файлов не ограничен
refresh_pattern ^ftp:     1440  20%  10080
refresh_pattern ^gopher:  1440  0%   1440
refresh_pattern .         0     20%  4320
# Прочие настройки
acl apache rep_header Server ^Apache
broken_vary_encoding allow apache
extension_methods REPORT MERGE MKACTIVITY CHECKOUT
hosts_file /etc/hosts
coredump_dir /var/spool/squid
```

Чтобы внесенные изменения вступили в силу, перезапустите Squid:

```
root# /etc/init.d/squid reload
```

Если ваш провайдер предлагает кэш-посредник в качестве отдельной услуги и вы можете либо обязаны его применять, то укажите его IP-адрес и порт с помощью ключевого слова `cache_peer` (как в следующем примере). Если номер ICP-порта неизвестен, попробуйте 0 или 7. Если не получается, дополнительно укажите параметр `default` или `no-query`.

```
# Вышестоящий кэш-посредник (например, интернет-провайдера)
# cache_peer <хост-имя> <тип> <прокси-порт> <icp-порт> <параметры>
cache_peer www-proxy.provider.de parent 8080 7 default
```

Если вы хотите, чтобы сайты локального сервера не сохранялись в буфере обмена, используйте ключевое слово no_cache. В следующем примере предполагается, что на компьютере uranus работает локальный веб-сервер:

```
# Прямой доступ к локальному веб-серверу uranus.sol
acl mars dstdomain .uranus .uranus.sol
no_cache deny uranus
```

ПРИМЕЧАНИЕ

Есть еще не меньше сотни параметров, которые могут влиять на работу Squid или на его функции журналирования. Если вы имеете на этот счет особые пожелания, посмотрите файл squid.conf или документацию по адресу www.squid-cache.org.

Первое испытание. По умолчанию Squid использует порт 3128. Чтобы пользователи вашей локальной сети действительно могли работать с прокси, они должны изменить его конфигурацию в своих браузерах. В Firefox можно открыть Инструменты ▸ Настройки ▸ Дополнительные ▸ Сеть, нажать кнопку Настроить и установить флажок Ручная настройка сервиса прокси. В качестве прокси-сервера требуется указать тот компьютер, на котором работает Squid (например, mars.sol), номер порта — 3128.

Если вы не отключили в Squid функции журналирования, то в файле /var/log/squid/access.log можно проверить, правильно ли работает программа. Если все нормально, то все переданные данные будут зарегистрированы в этом файле.

Конфигурация прозрачного кэш-посредника

Пока использование кэш-посредника происходит на добровольной основе. Клиентам (пользователям Интернета) нужно просто настроить связь по прокси в своем браузере, чтобы Squid начал работать. Если настройки прокси не изменить, то пользователи будут работать как бы «мимо кэша».

Такая ситуация нас не устраивает по двум причинам: многие конечные пользователи просто не сумеют сами изменить настройки прокси. Кроме того, чтобы Squid перестал фильтровать трафик, пользователям понадобится сбросить настройки прокси.

Чтобы избежать таких неприятностей, ядро Linux можно настроить так, чтобы весь трафик HTTP, который обычно идет через IP-порт 80, автоматически перенаправлялся на кэш-посредник. Такой кэш называется прозрачным. Это означает, что кэш без всякой дополнительной конфигурации автоматически применяется пользователем при каждом обращении к HTTP. Для описанной здесь конфигурации необходимо, чтобы Squid был установлен на том же компьютере, на котором работает интернет-шлюз локальной сети.

Файл squid.conf

В squid.conf необходимо изменить всего одну строку. В http_port дополнительно к номеру порта укажите ключевое слово transparent, а потом перезапустите Squid.

```
# Изменение в /etc/squid/squid.conf
...
http_port 3128 transparent
```

Активизация переадресации

Если вы еще этого не сделали, когда настраивали конфигурацию маскарадинга, можете активизировать функцию переадресации в ядре:

```
root# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Переадресация портов

Теперь нужно активизировать переадресацию IP-пакетов: все IP-пакеты, которые должны уйти с компьютера и адресованы на порт 80, следует переадресовать на тот IP-порт, где работает Squid. Команда не помещается на одной строке, поэтому я разделил ее на две части. Интерфейс `eth1` заменяется интерфейсом, через который шлюз соединяется с локальной сетью.

```
root# iptables -A PREROUTING -t nat -i eth1 -p tcp \
               --dport 80 -j REDIRECT --to-port 3128
```

Если все сделано правильно, то теперь каждый клиент, работающий в сети, автоматически использует Squid для передачи веб-страниц. Проверьте, на самом ли деле при доступе клиентского компьютера к Интернету в `/var/log/squid/access.log` появляются новые записи.

В заключение вставьте команду `iptables` в сценарий Init-V, который автоматически выполняется при запуске компьютера. На моем сервере эта команда интегрирована в сценарий брандмауэра (см. начало раздела 18.4):

```
# В /etc/myfirewall/myfirewall-start.conf
...
if [ $MFW_MASQ = 'yes' ]; then
  # Маскарадинг
  $IPT -A POSTROUTING -t nat -o $MFW_INET -s $MFW_LAN_IP \
       -j MASQUERADE
  # Прозрачный кэш-посредник
  $IPT -A PREROUTING -t nat -i eth1 -p tcp --dport 80 \
       -j REDIRECT --to-port 3128
  # Функция переадресации в ядре
  $SYS -q -w net.ipv4.ip_forward=1
fi
```

ВНИМАНИЕ

Принцип работы конфигурации таков, что прокси работает только на клиентских компьютерах локальной сети, но не работает на том компьютере, где установлен прозрачный прокси! Другими словами: автоматическая защита прокси не действует на локальном компьютере, но действует на всех остальных компьютерах в сети. Чтобы использовать прокси и на своем локальном компьютере, вам потребуется изменить на нем в браузере настройки прокси.

Создание сетевого фильтра с помощью DansGuardian

Правила `squid.conf` позволяют блокировать отдельные веб-страницы. Но создать по такому принципу полноценный сетевой фильтр будет очень сложно. Правда, нет

необходимости заново изобретать велосипед. Программа DansGuardian позволяет защитить ваших детей от опасности, которой в наши дни в Интернете предостаточно.

Основная функция этой программы заключается в том, что она анализирует текст сайтов на предмет наличия в них слов, связанных с порнографией, и блокирует переадресацию сайта после превышения определенной границы. Этот механизм достаточно интеллектуален. Поскольку DansGuardian перерабатывает содержимое сайтов, программа работает независимо от списков заблокированных сайтов, которые могли устареть. Обзор других признаков этой программы приводится на следующем сайте: http://dansguardian.org/.

Установка

В большинстве дистрибутивов DansGuardian можно установить в виде отдельного пакета. Поскольку программа также блокирует загрузку большинства известных вирусов, вместе с DansGuardian устанавливается вирусный фильтр ClamAV (эта программа предназначена для защиты клиентов Windows, работающих в локальной сети; для Linux в настоящее время вирусов не существует).

DansGuardian — это демон, запускаемый и останавливаемый обычными командами (см. раздел 4.5). По умолчанию программа ожидает поступления веб-запросов на порт 8080. Если такой запрос поступает, то DansGuardian связывается с программой Squid (порт 3128) и анализирует сайт. Если эта страница окажется безобидной, демон переадресует ее через порт 8080 обратно в браузер. Еще более изящным является представленный ниже вариант конфигурации, при котором DansGuardian обрабатывает все запросы на порте 80 и по этой причине для клиентов не требуется никакой конфигурации прокси.

Файл dansguardian.conf

Основным конфигурационным файлом программы является `/etc/dansguardian/dansguardian.conf`. Как правило, его можно оставить без изменений — в том виде, в котором он был при поставке системы. В зависимости от дистрибутива в файле может содержаться строка `UNCONFIGURED`; если она есть, то перед ней нужно поставить символ комментария. Если вы хотите, чтобы сообщение `'access denied'` выводилось по-русски, используйте настройку `language='russian'`. Если в вашей сети все компьютеры работают с Linux, отключите с помощью `virusscan=off` сканер антивируса, который по умолчанию проверяет все закачки на наличие вирусов. Для обеспечения взаимодействия с прокси Squid нужно указать IP-адрес и номер порта.

```
# /etc/dansguardian/dansguardian.conf (выборочно)
reportinglevel = 3
languagedir = '/etc/dansguardian/languages'
language = 'german'
loglevel = 1
logexceptionhits = on
logfileformat = 1
filterport = 8080
```

```
proxyip = 127.0.0.1
proxyport = 3128
...
```

Завершив конфигурацию, запустите DansGuardian:

```
root# /etc/init.d/dansguardian start
```

Прозрачная защита

По умолчанию DansGuardian работает только с обращениями, поступающими на порт 8080. Чтобы обеспечить прозрачную защиту всех обращений к Интернету, идущих через порт 80, нужно еще раз немного изменить сценарий брандмауэра. Для этого замените `eth1` интерфейсом вашей локальной сети:

```
# Дополнение сценария брандмауэра
#
# Переадресовывать к DansGuardian (Port 8080)
# все пакеты, идущие на порт 80 (прозрачный сетевой фильтр)
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 \
         -j REDIRECT --to-port 8080
```

Как это уже было описано в предыдущем разделе, Squid нужно сконфигурировать как прозрачный кэш-посредник. Кроме того, доступ к Squid нужно ограничить таким образом, чтобы со Squid мог обмениваться информацией только тот компьютер, на котором работает DansGuardian. Если и Squid, и DansGuardian работают на одном и том же компьютере и вы применяете конфигурацию, описанную в этой главе, нужно удалить строку `http_access allow localnet`. В противном случае все пользователи локальной сети смогут так настроить конфигурацию прокси в своем браузере, что браузер будет обмениваться данными напрямую со Squid через порт 3128 и обходить, таким образом, DansGuardian!

Конфигурация сетевого фильтра

Сами фильтрующие функции DansGuardian управляются из файла `/etc/dansguardian/dansguardianf1.conf`. Сначала этот файл указывает на различные заранее сконфигурированные файлы, содержащие ключевые слова, заблокированные сайты и т. д. Параметр `naughtynesslimit` определяет границу, начиная с которой сайт блокируется. Это значение тем выше, чем больше ключевых слов или их комбинаций будет найдено в тексте.

```
-# /etc/dansguardian/dansguardianf1.conf (выборочно)
# Размещение файлов, предназначенных для фильтрации содержимого
bannedphraselist    = '/etc/dansguardian/bannedphraselist'
weightedphraselist  = '/etc/dansguardian/weightedphraselist'
exceptionphraselist = '/etc/dansguardian/exceptionphraselist'
bannedsitelist      = '/etc/dansguardian/bannedsitelist'
greysitelist        = '/etc/dansguardian/greysitelist'
exceptionsitelist   = '/etc/dansguardian/exceptionsitelist'
bannedurllist       = '/etc/dansguardian/bannedurllist'
```

```
greyurllist         = '/etc/dansguardian/greyurllist'
exceptionurllist    = '/etc/dansguardian/exceptionurllist'
bannedregexpurllist = '/etc/dansguardian/bannedregexpurllist'
bannedextensionlist = '/etc/dansguardian/bannedextensionlist'
bannedmimetypelist  = '/etc/dansguardian/bannedmimetypelist'
picsfile            = '/etc/dansguardian/pics'
contentregexplist   = '/etc/dansguardian/contentregexplist'
# Лимит пошлости
# В качестве ориентира:
# 50 для маленьких детей, 100 для школьников,
# 160 для старших тинейджеров.
naughtynesslimit = 50
...
```

Если DansGuardian блокирует сайт, то браузер выводит соответствующее сообщение (рис. 18.5). Внешний вид и содержимое страницы, открывающейся при блокировке контента, можно отрегулировать в файле `/etc/dansguardian/languages/german/template.html`.

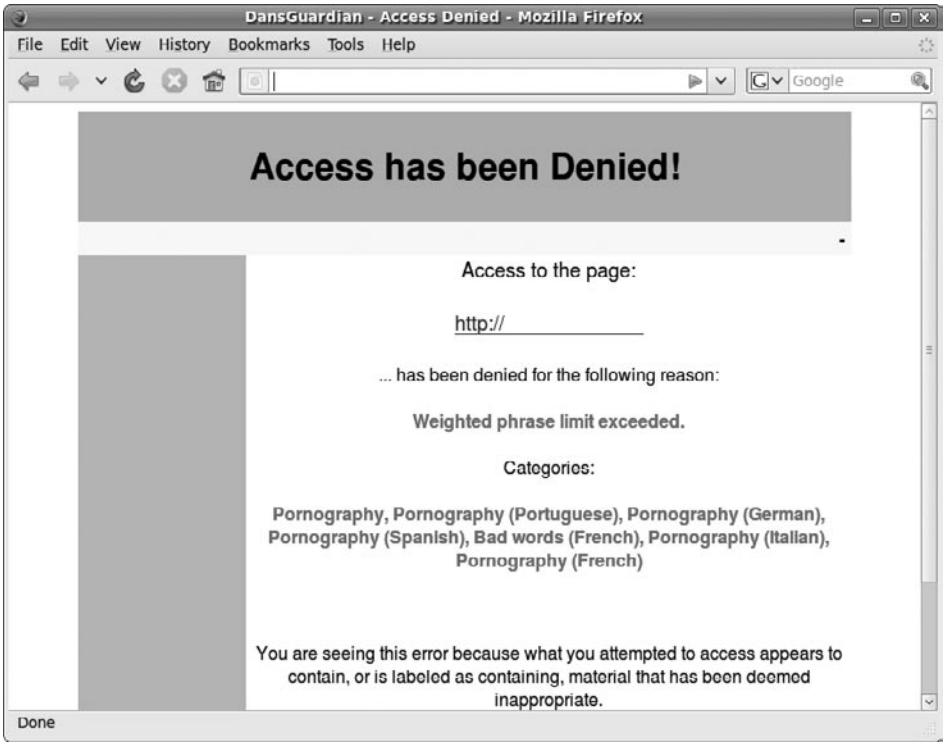

Рис. 18.5. DansGuardian и Squid блокируют доступ к сайту

В табл. 18.3 перечислены важнейшие файлы, находящиеся в каталоге `/etc/dansguardian`. В этих файлах, а также в названных в них включаемых файлах можно перечислить и другие понятия, сайты и типы файлов.

Таблица 18.3. Конфигурационные файлы DansGuardian

Файл	Содержание
bannedextensionlist	Блокировать файлы с указанными расширениями
bannedmimetypelist	Блокировать указанные типы файлов
bannedphraselist	Запрещенные ключевые слова
weightedphraselist	Негативные ключевые слова
bannedsitelist	Полностью блокировать указанные сайты
bannedurllist	Блокировать эти страницы
exceptioniplist	Никогда не блокировать запросы, поступающие с этих компьютеров (административный доступ к сомнительным сайтам)
exceptionsitelist	Принимать такие сайты без дополнительного контроля
exceptionurllist	Принимать такие страницы без дополнительного контроля
exceptionphraselist	Положительные ключевые слова
greysitelist	Принимать такие сайты, но проверять их текстовое содержимое

При анализе текста DansGuardian различает запрещенные ключевые слова (`bannedphraselist`), при обнаружении которых доступ к странице сразу же блокируется, и подозрительные ключевые слова (`weightedphraselist` и `exceptionphraselist`). В базовой конфигурации полностью запрещенных ключевых слов нет, так как самые подозрительные слова иногда могут попадаться на совершенно безобидных сайтах. Вместо этого DansGuardian применяет довольно обширные списки слов (см. файл `/etc/dansguardian/phraselists`). На основании этих списков программа подводит суммарный итог встречаемости всех негативных и положительных ключевых слов. Если итог превышает заданное пороговое значение, то доступ блокируется. Подробности о том, как вычисляются оценочные суммы блокируемых сайтов, описаны в файле регистрации `/var/log/dansguardian/access.log`.

В пакете DansGuardian *нет* списков сетевых адресов, которые должны блокироваться! В файлах `bannedsitelist` и `bannedurllist` документированы только принципы построения синтаксиса. Но есть фирмы, которые позволяют регулярно скачивать обновляемые списки за определенную плату. Самая известная из них — URLblacklist, собирающая в различных Squid-совместимых текстовых файлах ссылки на неблагонадежные сайты (http://urlblacklist.com/). Списки относятся к различным категориям. Фильтрационные списки можно бесплатно скачивать, чтобы научиться с ними работать, но обновление этих списков платное.

При использовании стандартной конфигурации DansGuardian блокирует в числе прочих закачки заархивированных файлов, MP3, образов дисков (ISO) и т. д. Но для повседневной работы с Linux такие ограничения определенно слишком жесткие, а в некоторых дистрибутивах они даже мешают работе системы управления пакетами. Просмотрите файлы `bannedmimetypelist` и `bannedextensionlist` и поставьте символы комментария перед всеми типами файлов, загрузку которых вы хотите разрешить! Вы также можете, настраивая переменные `bannedextensionlist` и `bannedmimetypelist`, просто указать в `dansguardianf1.conf` пустой файл.

Чтобы реагировать на запросы типа DansGuardian блокирует сайт xxx, но он совершенно безобиден, вы как администратор должны уметь обходить DansGuardian. Это проще всего сделать, указав в конфигурационном файле `exceptionip` фиксированный IP-адрес вашего компьютера в локальной сети.

ПРИМЕЧАНИЕ

Не переоценивайте эффективность DansGuardian! Даже настройка naughtynesslimit = 50 (для маленьких детей) не гарантирует полной защиты от порнографических сайтов и совсем не дает защиты от порнографических изображений.

При стандартной конфигурации учитываются в основном английские ключевые слова. Для блокировки большинства порнографических сайтов этого обычно бывает достаточно, но еще нужно заблокировать экстремистские правые сайты и сайты, призывающие к насилию, и для этого придется поработать вручную. Начать нужно с файла /etc/dansguardian/weightedphraselist.

Независимо от этого, фильтр легко обходится динамически создаваемым текстом (JavaScript/Flash), а также не воспринимает содержимого внешних файлов (PDF и т. д.). Доступ к изображениям и видео остается полностью незащищенным. Это же касается всех интернет-служб, работа которых не связана с Глобальной сетью (электронная почта, новостные ленты, чат и т. д.).

18.8. SELinux

Обычно в Linux используется традиционная система управления правами доступа: каждая программа выполняется под учетной записью определенного пользователя. На основании учетной записи система узнает, к каким файлам (и файлам-устройствам) может обращаться программа.

Обычные программы работают с учетной записью пользователя, запустившего ту или иную программу. Сетевые службы, серверы баз данных и т. д. запускаются с учетной записи администратора, но из соображений безопасности практически сразу после запуска переводятся на другую учетную запись с ограниченными правами.

Система прав в UNIX очень проста, но возможности ее конфигурации ограничены. Если злоумышленнику удастся перехватить управление одной из ваших программ, то он сможет получить доступ к огромному количеству данных, которые программе обычно не требуются. Ситуация особенно осложняется, если злоумышленнику удается завладеть правами администратора или он найдет лазейки, которые позволят выполнять на вашем компьютере чужой код с правами администратора. Так хакер получает неограниченную власть над вашим компьютером, может устанавливать на нем свои программы и запускать их.

Возможно, вы спросите, как злоумышленник может заполучить контроль над вашими программами. Для этого практически всегда используются ошибки, содержащиеся в программном коде. Например, при пересылке данных искусственно выполняется так называемое переполнение буфера. Эта ошибка, в свою очередь, применяется для того, чтобы добавить в программу чужой код и выполнить его. Конечно, есть и другие методы, но все они связаны с поиском и использованием ошибок в программе, снижающих ее надежность и позволяющих использовать ее не по назначению либо манипулировать ею.

Меры предосторожности. Идеальных программ, совсем не содержащих ошибок, нет и в ближайшем будущем не будет (здесь я не говорю о простейших и самых миниатюрных программах). По этой причине с течением времени были разработаны всевозможные методы, позволяющие свести к минимуму риски, связанные с программными ошибками. К наиболее проверенным мерам предосторожности относится отказ от использования демонов с правами администратора. Следует

также устанавливать как можно меньше программ или сценариев с битом `setuid` (см. также раздел 4.5), не позволяющих выполнять код в стеке (для этих целей в Red Hat было разработано расширение ядра Exec Shield), и т. д.

Основы SELinux

Программой следующего поколения является расширение ядра SELinux, которое первоначально было разработано Агентством национальной безопасности США как свободная программа. Работа этого расширения заключается в том, что SELinux, основываясь на заданных правилах, наблюдает за выполнением программ. Этот метод называется *обязательным контролем доступа* — Mandatory Access Control (кратко — MAC).

Если правило нарушается, то SELinux запрещает выполнять операцию или регистрирует предупреждение. При этом расширение SELinux не завершает работу программы, нарушающей правила. От программы зависит, как она отреагирует на ситуацию, в которой не сможет обратиться к определенному файлу или воспользоваться сетевым интерфейсом.

Правила MAC обеспечивают значительно более тщательный контроль безопасности, чем система прав доступа UNIX. С помощью MAC программе можно запретить доступ к определенным каталогам и сетевым функциям, независимо от прав доступа или учетных записей, с которыми она работает. Поскольку эти правила отслеживаются на уровне ядра, они действуют и в том случае, когда программа выходит из-под контроля из-за ошибки или недоработки в области безопасности.

Агентство национальной безопасности США — это информационная служба, которая обеспечивает в том числе надзор за обменом электронной информацией и ее шифрование. Итак, хотя SELinux в определенной степени и происходит из сфер «тайной полиции», в дополнении Linux такими функциями нет ничего предосудительного — код SELinux открытый, контролируется многочисленными независимыми экспертами и официально входит в состав ядра версии 2.6 и выше.

Правила SELinux

Без необходимых правил расширение SELinux бесполезно. Таким образом, от качества правил напрямую зависит, насколько надежнее станет система, оборудованная SELinux. Из распространенных дистрибутивов пока только Red Hat вложил достаточно времени и усилий в разработку таких правил. Все испытания проводятся в системе Fedora, которая в данном случае играет роль экспериментального полигона. Те правила, которые будут признаны практичными в Fedora, входят в состав новых версий Red Hat Enterprise (RHEL).

В этом разделе расширение SELinux описано на базе Fedora 11. Насколько широко SELinux закрепится за пределами Red Hat и Fedora, предположить сложно. В Novell и SUSE, а также в Ubuntu используется альтернативное решение AppArmor — о нем речь пойдет в следующем разделе. Правда, не так давно SUSE свернула программу разработки AppArmor и интегрировала SELinux в дистрибутив настолько глубоко, что работать с этим расширением в принципе возможно и без перекомпиляции ядра. Остается лишь гадать, является ли этот шаг началом конца AppArmor.

Критика SELinux

Расширение SELinux небезупречно. Два основных момента, из-за которых оно подвергается критике, таковы.

- Файлы нужно помечать расширенными атрибутами (EA, см. раздел 3.8), чтобы обеспечить их правильное взаимодействие с SELinux. Для этого требуются файловые системы, поддерживающие расширенные атрибуты (NFS к их числу не относится!). Кроме того, возникают проблемы с обновлениями и резервным копированием.
- Основная проблема SELinux — исключительная сложность. Для одного только обеспечения безопасности основных сетевых служб требуются тысячи правил. Лишь немногие эксперты способны судить об эффективности этих правил. По существующим наборам правил не хватает документации. По этим причинам среднестатистический пользователь Linux обычно оказывается не в состоянии приспособить SELinux к своим нуждам.

Расширение SELinux по праву снискало среди большинства своих пользователей славу «черного ящика». Из-за сложности программы ее установка обычно не обходится без ошибок. В результате при возникновении первых же нестыковок возникает соблазн просто выключить компьютер, а потом начать все заново.

Ссылки. SELinux более подробно описывается на следующих сайтах:

- http://www.nsa.gov/research/selinux/;
- http://sourceforge.net/projects/selinux/;
- http://www.crypt.gen.nz/selinux/faq.html;
- http://fedoraproject.org/wiki/SELinux;
- http://www.os-t.de/HTML-SELinux/buch.html.

Альтернативы

Наиболее популярной альтернативой SELinux является система, применяемая в SUSE и Ubuntu — AppArmor (см. раздел 18.9). Кроме того, в версии ядра 2.6.25 появилась еще одна система MAC под названием Smack (http://lwn.net/Articles/252562/). Но пока еще рано судить, сможет ли Smack реально конкурировать с SELinux и AppArmor.

Внутренняя организация и принцип работы SELinux

SELinux базируется на двух основных предпосылках: во-первых, это правильное обозначение всех файлов и процессов с помощью так называемого контекста защиты, во-вторых, это правила, которые должны выполняться наблюдаемыми процессами.

Контекст защиты

Принцип работы SELinux основан на том, что любой объект (например, файл) и любой субъект (например, процесс) связаны с контекстом защиты. С файлами

контекст защиты сохраняется в виде расширенных атрибутов. В таком случае информация о защите непосредственно связана с файлом и не зависит от имени файла. Контекст защиты определенного файла проще всего узнать с помощью команды `ls -Z`. В качестве альтернативы можно попробовать `etfattr -n security.selinux -d` *`имя_файла`*.

```
user$ ls -Z /usr/sbin/httpd
-rwxr-xr-x root root system_u:object_r:httpd_exec_t:s0 /usr/sbin/httpd
user$ ls -Z /etc/httpd/conf/httpd.conf
-rw-r--r-- root root system_u:object_r:httpd_config_t:s0 /etc/httpd/conf/httpd.conf
user$ getfattr -n security.selinux -d /etc/httpd/conf/httpd.conf
getfattr: Removing leading '/' from absolute path names
# file: etc/httpd/conf/httpd.conf
security.selinux="system_u:object_r:httpd_config_t:s0\000"
```

Для работы SELinux совершенно необходимо, чтобы со всеми файлами был сохранен правильный контекст защиты. Чтобы этот контекст работал и тогда, когда вы уже после установки будете создавать новые файлы, во многих каталогах существуют правила SELinux, которые автоматически присваивают нужные контексты файлам, сохраняемым в этих каталогах. Если данный механизм не работает автоматически, то вы можете сами настроить правильный контекст с помощью команды `chcon`:

```
user$ chcon user_u:object_r:user_home_t тестовый_файл
```

При работе с процессами вместо слова «контекст» часто применяется термин «домен». Чтобы определить контекст (домен) того или иного процесса, используйте команду `ps axZ`. Как правило, процесс принимает контекст той учетной записи, с которой он был запущен. Но контекст также можно автоматически изменить сразу после запуска с помощью правила SELinux. Это бывает необходимо в тех случаях, когда определенная программа (например, Firefox) должна получать заданный контекст независимо от того, какая учетная запись ее запустила.

```
user$ ps axZ | grep httpd
user_u:system_r:httpd_t:s0  3758 ?   Ss    0:00 /usr/sbin/httpd
user_u:system_r:httpd_t=s0  3760 ?   S     0:00 /usr/sbin/httpd
...
```

Контекст защиты состоит из трех или четырех частей, разделяемых двоеточиями:

`пользователь:роль:тип:mls-компонент`

Самая важная часть — третья, в которой указывается тип файла или процесса. Большинство правил SELinux интерпретируют эту информацию. Подробное описание всех четырех частей контекста защиты находится здесь: http://fedoraproject.org/wiki/SELinux/SecurityContext.

Правила

В обобщенном виде контекст правил SELinux выглядит так:

```
allow тип1_t тип2_t:класс { операции };
```

Приведу пример: следующее правило разрешает процессам, имеющим тип контекста `httpd_t`, создавать новые файлы в каталогах, имеющих тип контекста `httpd_log_t`.

```
allow httpd_t httpd_log_t:dir create;
```

Как правило, в отдельно взятой конфигурации SELinux таких правил десятки тысяч! Чтобы система работала быстрее, SELinux обычно ожидает правил не в виде текста, а в двоичной форме. Проводя аналогию с программированием, правила можно сравнить с результатом компилирования. Общее описание этапов, которые необходимо выполнить, чтобы добавить к существующей конфигурации SELinux собственный модуль правил, дается в часто задаваемых вопросах по SELinux: http://fedora.redhat.com/docs/selinux-faq-fc5/.

Со временем сформировались некоторые устойчивые наборы правил, предназначенные для достижения определенных целей.

- `Strict` — в Fedora 2 набор правил `strict` был активен и содержал правила для выполнения всех процессов, но создавал, таким образом, больше проблем, чем решал.
- `Targeted` — в Fedora 3 и выше по умолчанию стал использоваться набор правил `targeted`. Его правила защищают только специально выбранные сетевые службы.

  К сожалению, очень плохо документирован механизм, в соответствии с которым осуществляется защита, а также не описано, какую контекстную информацию должны иметь данные и какие управляющие параметры существуют (булевые, см. ниже). Для некоторых важнейших служб в справке `man` есть отдельные страницы: `ftpd_selinux`, `httpd_selinux`, `named_selinux`, `nfs_selinux`, `samba_selinux` и т. д. Полный список выдает следующая команда:

  ```
  user$ rpm -qd selinux-policy | grep man8
  ```

- MLS — в качестве альтернативы можно использовать разработанный для сервера набор правил MLS (многоуровневая безопасность) (пакет `selinux-policy-mls`). Этот набор был разработан для того, чтобы RHEL мог получить сертификат класса EAL 4. Такой сертификат в США требуется при выпуске определенных (часто — военных) приложений, хотя фактическое улучшение безопасности при этом не так и велико. Более подробная информация приводится на следующих сайтах:
  - http://fedoraproject.org/wiki/SELinux/FedoraMLSHowto;
  - http://en.wikipedia.org/wiki/Common_Criteria;
  - http://en.wikipedia.org/wiki/Evaluation_Assurance_Level.

При разработке наборов правил SELinux используется механизм *reference policy*, созданный компанией Tresys (http://oss.tresys.com/projects/refpolicy). Он очень удобно применяется в среде разработки. Все действующие наборы правил SELinux были созданы с помощью этого механизма.

Параметры SELinux (булевые)

Итак, вы уже понимаете, что вносить изменения в наборы правил достаточно сложно. Чтобы у нас было определенное поле для маневра, не требующее изменения

правил, в наборе targeted предусмотрены булевые параметры, которые можно изменять «на ходу». В Fedora/Red Hat для этого обычно используется графический пользовательский интерфейс systemconfig-selinux (рис. 18.6). В качестве альтернативы можно попробовать команду getsebool, узнающую значения булевых параметров; setsebool предназначена для изменения таких параметров.

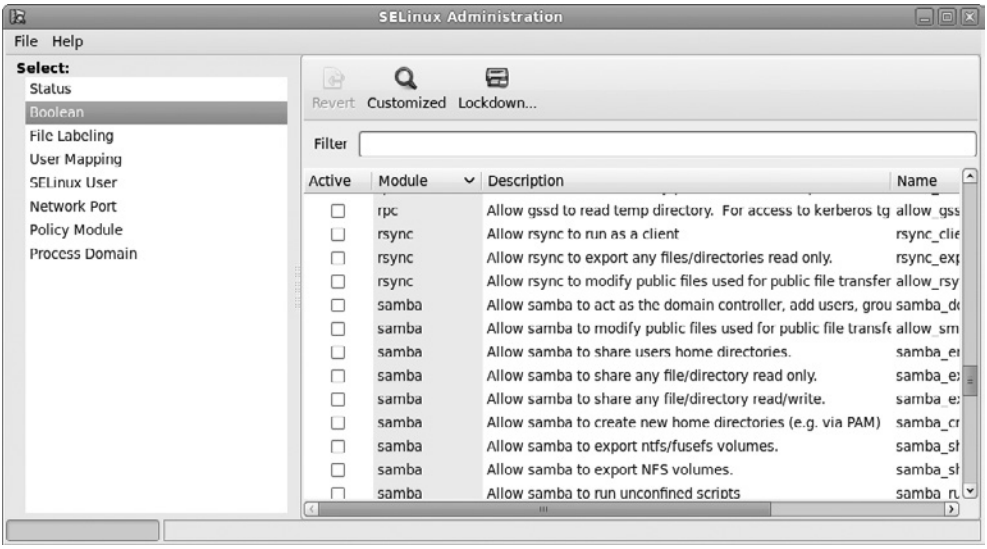

Рис. 18.6. Изменение параметров SELinux

Запуск и конфигурация

SELinux входит в состав ядра, поэтому специально запускать программу (например, через систему Init-V) не требуется. Необходимость в демоне SELinux или других фоновых процессах также отпадает.

Конфигурация производится в файлах, расположенных в каталоге /etc/selinux. Определяющее значение имеет файл /etc/selinux/config. В нем указывается, в каком режиме работает SELinux (enforcing, permissive или disabled) и какой набор правил действует (strict или targeted). Изменения, внесенные в этот файл, вступают в силу только после перезапуска.

```
# /etc/selinux/config
SELINUX=enforcing
SELINUXTYPE=targeted
```

Статус

Команда setstatus определяет текущий статус SELinux. На моем испытательном компьютере, где SELinux работает в режиме targeted, получены следующие результаты:

```
root# sestatus
SELinux status:                 enabled
```

```
SELinuxfs mount:           /selinux
Current mode:              enforcing
Mode from config file:     enforcing
Policy version:            24
Policy from config file:   targeted
```

Устранение проблем, связанных с SELinux

В принципе, существуют следующие возможности реагирования на нарушение правил SELinux:

- в наборе правил находится параметр, который может быть причиной проблемы, а затем его настройки корректируются с помощью `system-config-security`;
- изменяется контекстная информация файла, вызывающего проблемы;
- набор правил изменяется или дополняется; для этого требуются глубокие знания SELinux, и информации, сообщаемой в этом разделе, недостаточно;
- SELinux просто отключается.

Проблемы, вызываемые SELinux, не всегда очевидны. Например, если скопировать дерево каталогов, содержащее HTML-файлы, из каталога NFS в каталог `/var/www/html` командой `cp -a`, то Apache больше не сможет прочитать эти файлы. Причина заключается в том, что при использовании параметра `cp -a` вместе с файлами копируются и расширенные атрибуты, и контекстная информация SELinux. Из-за этого файлы, скопированные в `/var/www/html`, не могут автоматически получить нужную контекстную информацию в соответствии с правилом SELinux. Контекстная информация HTML-файлов в таком случае имеет вид `system_u:object_r:nfs_t`, в то время как правильный вариант — `user_u:object_r:httpd_sys_content_t`. Для того чтобы избежать подобных проблем, измените `cp -a` на `cp -r`.

Сам Apache «ничего не знает» о существовании SELinux. Программа просто обнаруживает, что не может прочитать некоторые файлы, и выдает непонятное сообщение об ошибке You don't have permission to access <filename> (У вас нет прав на доступ к <имя файла>). Только взглянув в `/var/log/messages`, вы поймете, что проблемы с доступом возникли из-за SELinux.

```
root# less /var/log/messages
...
May 19 08:15:12 mars setroubleshoot: SELinux is preventing the
  httpd from using potentially mislabeled files (test.html).
  For complete SELinux messages run sealert -l 2f35f9a0-bd1b-40ab-9779-3a640b99ef10
```

Если в момент нарушения правил вы работаете в KDE или Gnome, то на панели отображается предупреждающий значок программы `seapplet`. Щелчком кнопкой мыши можно запустить программу `setroubleshoot`, которая исключительно точно и понятно описывает проблему и даже предлагает правильное решение (рис. 18.7). Теперь контекстную информацию пострадавших файлов нужно правильно настроить с помощью команды `restorecon`:

```
root# restorecon -R -v /var/www/html/*
```

Отключение SELinux

Если требуется активизировать SELinux лишь ненадолго, то запустите `system-config-security` и включите режим `Permissive`. Расширение SELinux продолжит работу и будет регистрировать нарушения правил в `/var/log/messages`. Но в таком режиме SELinux допускает ошибки и не блокирует программы, которые их совершают. Аналогичное действие оказывает команда `setenforce 0`.

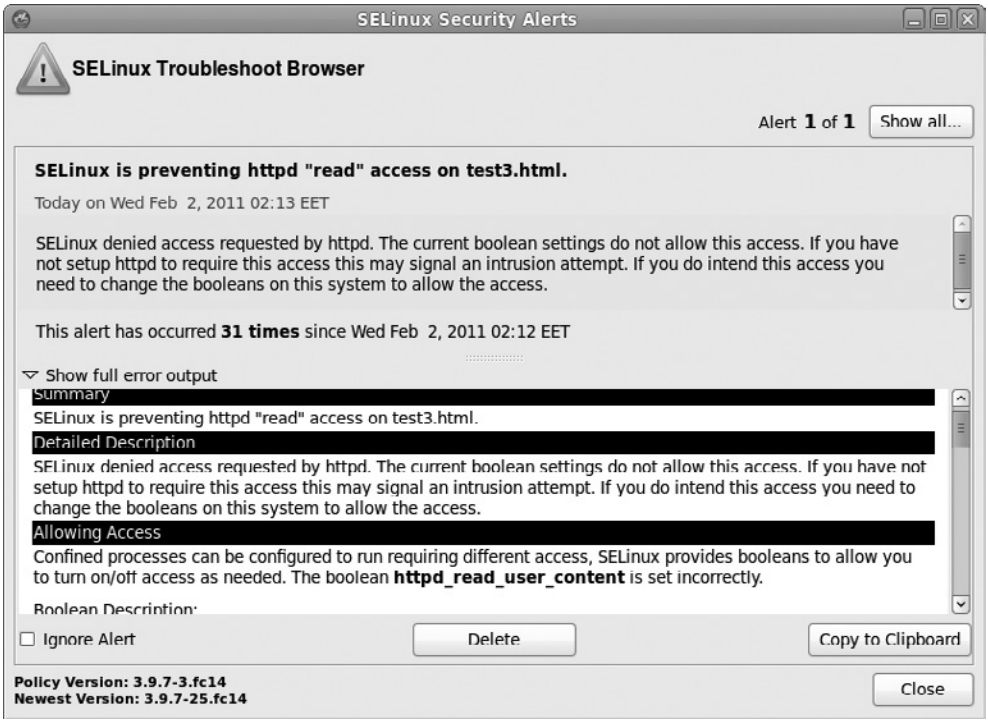

Рис. 18.7. Сообщение об ошибке SELinux

Разумеется, SELinux в `system-config-security` можно полностью отключить (настройка `Disabled`). Но это стоит делать лишь в тех случаях, когда вы больше не планируете работать с этим расширением. Причина заключается в том, что, как только вы отключаете SELinux, все его правила больше не могут присваивать контекстную информацию новым файлам. Если позже снова включить SELinux, то файлы, в которых нет контекстной информации, будут вызывать проблемы. При последующей корректировке контекстных данных вам поможет команда `restorecon`, но подобный процесс утомителен и чреват ошибками.

Если SELinux будет вызывать проблемы уже при старте и мешать запускать компьютер, то в ядре нужно установить параметр `selinux=0`, не позволяющий системе SELinux запускаться. Активизация происходит только после перезапуска системы. Еще можно применить параметр `enforcing=0`, тогда SELinux запускается, а нарушения правил не регистрируются.

18.9. AppArmor

Для того чтобы не адаптировать к своему дистрибутиву сложную систему SELinux, Novell в 2005 году приобрела фирму Immunix и переименовала ее программу Subdomain (предназначенную для обеспечения безопасности) в AppArmor. Компания Novell привела AppArmor в соответствие с лицензией GPL и разработала несколько модулей YaST для администрирования. С тех пор AppArmor играет во всех дистрибутивах Novell и SUSE ту же роль, что и SELinux в Fedora и Red Hat: расширение ядра отслеживает работу всех выполняемых процессов и гарантирует, что эти процессы выполняют установленные правила безопасности. AppArmor также является MAC-системой обеспечения безопасности.

Особенности AppArmor

В отличие от SELinux, правила безопасности AppArmor основаны на абсолютных именах файлов, поэтому отдельного обозначения файлов с помощью расширенных атрибутов не требуется. Кроме того, AppArmor работает и с теми системами, которые не поддерживают расширенных атрибутов. В правилах AppArmor можно ставить джокерные символы, поэтому в типичных практических ситуациях для работы AppArmor требуется значительно меньше правил, чем для SELinux. В этом проявляется главное достоинство программы AppArmor по сравнению с SELinux: AppArmor гораздо проще.

К сожалению, у AppArmor есть и недостатки.

- Эксперты по безопасности из Red Hat придерживаются мнения, что при указании в правилах абсолютных путей вы постоянно подвергаете систему риску (защиту AppArmor можно обойти, переименовав файлы и каталоги; разумеется, для этого агрессор должен иметь достаточно широкие права).
- В отличие от SELinux, программа AppArmor не входит в состав официального ядра. Осенью 2009 года разработчики Canonical активно пытались добиться интеграции AppArmor в ядро.
- Набор правил AppArmor не так масштабен, как в SELinux. По умолчанию защита распространяется на меньшее количество программ. Хотя пользователю и проще самостоятельно создавать и изменять правила, такая «самодельная» безопасность производит впечатление некоторого непрофессионализма.
- Осенью 2007 года Novell свернула разработку AppArmor. В настоящее время этим занимаются только специалисты Canonical.

Ссылки. В этом разделе сделано лишь введение в AppArmor. Более подробная информация дается на следующих сайтах:

- https://help.ubuntu.com/community/AppArmor;
- http://de.opensuse.org/Apparmor_Details;
- http://developer.novell.com/wiki/index.php/Apparmor_FAQ;
- http://forge.novell.com/modules/xfmod/project/?apparmor;
- http://www.novell.com/documentation/apparmor/ (пособия в формате PDF).

Внутренняя организация и практическое применение AppArmor

В следующих моделях рассмотрен вариант AppArmor, поставляемый с openSUSE 11.2. Замечания, касающиеся Ubuntu, даются в конце раздела.

Конфигурация и запуск

AppArmor запускается в начале процесса Init-V сценарием `/etc/init.d/boot.apparmor`, который обращается к функциям, определенным в `/lib/apparmor/rc.apparmor.functions`. Сценарий `boot.apparmor` считывает базовую конфигурацию из каталога `/etc/apparmor` и загружает все данные правил в каталог `/etc/apparmor.d`.

В отличие от SELinux, в AppArmor пока разработаны надежные защитные профили для небольшого количества программ. Отдельные профили, требующие доработки, находятся в каталоге `/etc/apparmor/profiles/extras`. Однако они уже много лет имеют статус экспериментальных и по умолчанию не активизируются.

Изменения, внесенные в конфигурацию, вступят в силу только после того, как вы заново запустите AppArmor или считаете файлы правил:

```
root# /etc/init.d/boot.apparmor restart  (Перезапустить AppArmor)
root# /etc/init.d/boot.apparmor reload   (Заново загрузить правила AppArmor)
```

Правила (профили)

Файлы правил, называемые в AppArmor профилями, создаются в текстовом формате, что упрощает их интерпретацию. Следующие строки описывают правила для демона журналирования событий ядра klogd. Здесь сначала считываются включаемые файлы, а потом — основные признаки (см. `man capabilities`). Правила указывают, какие файлы может использовать программа.

```
# Файл /etc/apparmor.d/sbin.klogd
#включить <tunables/global>
/sbin/klogd {
  # включить <abstractions/base>
  capability sys_admin,
  network inet stream,
  /boot/System.map*         r,
  @{PROC}/kmsg              r,
  @{PROC}/kallsyms          r,
  /dev/tty                  rw,
  /sbin/klogd               rmix,
  /var/log/boot.msg         rwl,
  /var/run/klogd.pid        krwl,
  /var/run/klogd/klogd.pid  krwl,
  /var/run/klogd/kmsg       r,
}
```

В файлах AppArmor может использоваться джокер *, играющий роль подстановочного символа, — им можно заменить любое количество других символов. Сочетание ** имеет подобное значение, но охватывает и символ / и, таким образом, включает все файлы из подкаталогов. Права доступа обозначаются буквами и их

комбинациями, их значение подробно описано в руководстве по администрированию AppArmor. В табл. 18.4 приведены лишь важнейшие из буквенных обозначений. Комбинации с `?x` управляют правами подпроцессов, запускаемых основной программой.

Таблица 18.4. Простейшие права доступа AppArmor

Сокращение	Значение
r	read mode — доступ только для чтения
w	write mode — доступ для чтения и внесения изменений
a	append — позволяет дополнять файл
l	link mode — применяет к жестким ссылкам те же правила, что и к исходному файлу
k	lock — позволяет блокировать файлы
m	allow executable mapping — допускает использование функции mmap
ix	inherent execute mode — программа наследует правила базовой программы
px	discrete profile executemode — для программы в AppArmor предусмотрен собственный профиль безопасности
ux	unconstrainted execute mode — программа выполняется без правил AppArmor

Статус

Команда `aa-status` дает обзор актуального состояния AppArmor:

```
root# aa-status
apparmor module is loaded.
10 profiles are loaded.
10 profiles are in enforce mode.
  /usr/sbin/ntpd
  /usr/sbin/identd
  /sbin/klogd
  /sbin/syslogd
  /sbin/syslog-ng
  /usr/sbin/traceroute
  /usr/sbin/nscd
  /usr/sbin/mdnsd
  /bin/ping
  /usr/sbin/avahi-daemon
0 profiles are in complain mode.
2 processes have profiles defined.
2 processes are in enforce mode :
  /usr/sbin/avahi-daemon (2778)
  /usr/sbin/nscd (2831)
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

При запуске AppArmor файловая система securityfs подключается к каталогу `/sys/kernel/security`. В файлах этой системы содержится информация об активных профилях, количестве возникших нарушений правил и т. д. Подробности о произошедших нарушениях правил приводятся в файлах каталога `/var/log/apparmor`.

Модули YaST

Для управления AppArmor в конфигурационной программе YaST дистрибутива SUSE предусмотрены различные модули (рис. 18.8). В контрольной панели AppArmor программу можно (де)активизировать либо выбрать для отдельных профилей режим `enforce` или `complain`. В режиме `complain` нарушения правил регистрируются, но работа «ошибающейся» программы не прекращается.

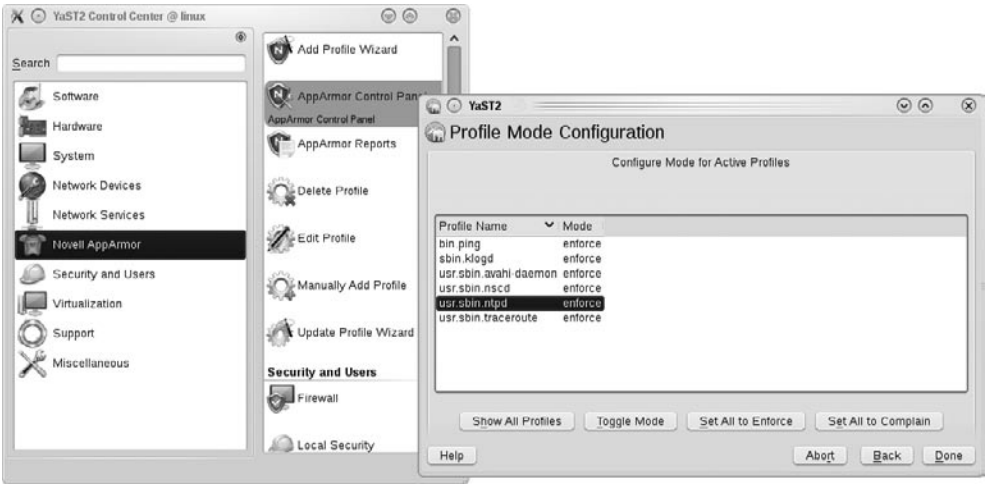

Рис. 18.8. Конфигурация AppArmor с помощью YaST

В модуле Редактировать профиль можно изменять имеющийся профиль. Но, как правило, лучше обрабатывать файл правил в обычном текстовом редакторе. При составлении правил модуль YaST предоставляет только некоторые вспомогательные функции.

Гораздо интереснее модуль Ассистент для добавления профилей: с его помощью вы можете в своеобразном «учебном режиме» сами разработать защитный профиль для программы. Для этого программа запускается, а потом вы пробуете как можно больше ее функций. На этом этапе программа еще не защищена AppArmor, поэтому необходимо полностью исключить возможность атаки во время такого «обучения». Ассистент регистрирует все обращения программы к файлам и предлагает правила, подходящие для защиты программы. Теперь вы должны утвердить или изменить отдельные правила. Полученный в результате профиль будет сохранен под именем `/etc/apparmor.d/programmname`.

С помощью ассистента вам будет достаточно просто создавать собственные защитные профили. Однако полученные таким образом профили, скорее всего, будут неидеальны: с одной стороны, работая с программой, вы вполне можете не проверить некоторые ее функции, а для них будут нужны дополнительные правила. С другой стороны, автоматическое генерирование правил безопасности говорит о невысоком качестве таких правил. Таким образом, после работы ассистента вам предстоит вручную внести еще некоторые доработки профиля безопасности.

Если с имеющимся профилем возникают проблемы, вы можете использовать Ассистент для обновления профилей, чтобы дополнить профиль новыми правилами. Ассистент изучает файлы AppArmor, в которые занесена информация о возникших ошибках, и вырабатывает новые правила на базе этой информации.

Ubuntu

В Ubuntu версии 7.10 и выше также используется AppArmor, более того — в этом дистрибутиве уже полным ходом идет техническая поддержка данного программного пакета. Но по умолчанию в Ubuntu активизируется меньше пакетов, чем в SUSE. В отличие от SUSE, в Ubuntu нет графического пользовательского интерфейса для конфигурации AppArmor.

19 Веб-сервер и корневой сервер

В этой главе описано, как создать собственный веб-сервер, состоящий из Linux, Apache, MySQL и PHP (кратко — LAMP). Как и в других главах, посвященных конфигурации сервера, я ограничусь здесь только вводной информацией. По Apache, MySQL и PHP есть масса специальных пособий, в которых описаны всевозможные варианты их конфигурации и применения.

Корневой сервер. В принципе систему LAMP можно установить на любом компьютере, например в качестве среды разработки веб-приложений. Но для общедоступных сайтов, находящихся в Интернете, обычно арендуется корневой сервер и система устанавливается именно на нем. *Корневым сервером* называется удаленный компьютер, находящийся в вычислительном центре. Таким компьютером вы можете управлять без ограничений (обладая для этого правами администратора). Для опытных администраторов Linux корневой сервер представляет удобную и недорогую возможность создания собственных сайтов. Такой вариант интересен в первую очередь для небольших и средних предприятий, которые подключаются к Интернету через ADSL и, следовательно, не могут позволить себе выделенные серверные службы с собственным доменным именем.

При выборе корневого сервера обращайте внимание на то, насколько респектабелен хостинг-провайдер. Нет ничего хуже, чем плохо налаженная техническая поддержка компьютера, к которому вы сами не имеете доступа. Кроме того, очень важна административная помощь, в частности предоставление веб-интерфейса, позволяющего перезагрузить компьютер после внезапного прекращения работы, или возможность запустить «живую» систему, позволяющую исправить ошибки конфигурации.

Наконец, обратите внимание на то, какие дистрибутивы Linux вы можете установить, не каждый провайдер поддерживает все распространенные дистрибутивы. Для работы с сервером хорошо подходят не только дорогие корпоративные дистрибутивы, но и Debian, Ubuntu LTS и CentOS. Не рекомендую использовать Fedora и openSUSE ввиду слишком непродолжительного времени обслуживания.

SSH. Большинство провайдеров корневых серверов осуществляют минимальную необходимую установку выбранного дистрибутива. Вам сообщаются IP-адрес вашего сервера и пароль для административного доступа, а в дальнейшем вы предоставлены сами себе. Все администрирование производится через SSH и, как правило, без вспомогательных графических конфигурационных инструментов

(некоторые провайдеры предоставляют инструменты администрирования, работающие через Интернет, но здесь мы не будем рассматривать такие программы более подробно).

19.1. SSH

На SSH-сервер можно входить через логин ssh из локальной сети, а также из Интернета. SSH идеально подходит для удаленной технической поддержки компьютеров. Он пришел на смену telnet и rlogin и отличается от двух этих программ значительно большей надежностью. SSH с точки зрения клиента уже был описан в разделе 6.2. Еще более подробно работа с SSH — как со стороны клиента, так и со стороны сервера — описана на сайте openSSH: http://www.openssh.com/.

Установка и запуск

Обычно на корневом сервере SSH-сервер устанавливается по умолчанию. На пользовательских компьютерах, в зависимости от дистрибутива, может понадобиться установить пакет `openssh-server` или `openssh`. SSH-сервер запускается в рамках процесса Init-V. Команды, предназначенные для этого, также варьируются от дистрибутива к дистрибутиву. Они обобщены в разделе 4.5. При первом запуске `sshd` автоматически создает RSA-файлы, предназначенные для шифрования и обмена информацией. Эти файлы сохраняются в каталоге `/etc/ssh/`.

Конфигурация. Конфигурационные файлы `sshd` находятся в каталоге `/etc/ssh`. За серверную конфигурацию отвечает `sshd_config`. Как правило, этот файл можно оставить без изменений, так как SSH-сервер должен заработать с ходу. Обмен информацией по умолчанию осуществляется через IP-порт 22. Если у вас есть особые пожелания, на страницах `man` вы найдете много дополнительной информации.

Безопасный FTP. Составной частью SSH-сервера является SFTP-сервер. Это безопасный аналог обычного FTP-сервера. Обычно функции SFTP предоставляются автоматически, пока работает SSH-сервер. Правда, работать с ними могут только SFTP-совместимые клиенты (например, программа `sftp`). В SFTP предусмотрены лишь пользовательские учетные записи (то есть отсутствует анонимный FTP).

Обеспечение безопасности

В принципе SSH-сервер начинает работать без дополнительной конфигурации. Но в этом кроется значительный риск, который нельзя недооценивать: любой, кто угадает рабочую комбинацию логина и пароля, может войти в систему с вашего компьютера! Взломщики применяют автоматические инструменты, которые ищут серверы в Интернете и пытаются на них зайти. Все подобные действия регистрируются в файле `/var/log/auth.log`. На общедоступных серверах случается по тысяче попыток взлома в день! По этой причине вам не помешает позаботиться о том, чтобы все пользователи применяли как можно более непростые пароли. Например, можно использовать команду `makepasswd` из одноименного пакета, которая создает надежные пароли.

В файле /etc/shadow в зашифрованном виде сохранены все пользовательские пароли, и в нем ни в коем случае не должно быть записей без паролей! Эти записи выделяются тем, что в строке между первым и вторым двоеточием отсутствует текст. Обычно там должен находиться или зашифрованный пароль, или — в системных учетных записях — специальный символ (* или !). Эти символы полностью исключают возможность входа в систему. Если в файле действительно найдется запись без пароля, исправьте эту ошибку, указав *пароль имя*.

Соблюдая перечисленные ниже правила, вы значительно снизите вероятность взлома вашего SSH-сервера. Описанные методы можно применять вместе или по отдельности.

Отсутствие учетной записи администратора

Агрессор пытается заполучить административные права, а проще всего это сделать, войдя в систему с учетной записи администратора. Для этого требуется угадать всего один параметр — пароль администратора. Гораздо безопаснее будет запретить вход в систему от имени администратора, если он осуществляется по SSH. Иначе говоря, по SSH нужно войти в систему под другой учетной записью, а потом перейти на учетную запись администратора с помощью команд su или sudo (проверьте, функционирует ли этот механизм, прежде чем вносить следующее изменение).

```
# Изменение в /etc/ssh/sshd_config
...
PermitRootLogin = no
```

Чтобы данное изменение вступило в силу, нужно заставить sshd заново считать конфигурационные файлы:

```
root# root /etc/init.d/ssh reload
```

В результате такого приема агрессору теперь неизвестны два параметра: *и* логин, *и* пароль!

Изменение SSH-порта

По умолчанию обмен информацией с сервером происходит через порт 22. В строке port вы можете без труда указать другой порт, который в настоящее время не используется. Поскольку многие автоматические инструменты взлома отслеживают только порт 22, вы одновременно избавляетесь от массы проблем, связанных с безопасностью.

Если вы работаете с командой ssh, то при каждом ее применении нужно специально указывать порт для SSH-сервера параметром -p. Обратите внимание, что при работе с scp нужно применять параметр -P, так как -p в этой программе имеет значение preserve и этот параметр сохраняет данные о времени доступа к копируемой информации и другие данные о доступе.

Разумеется, не забывайте и о том, что защита методом изменения порта доступа не является абсолютной. Если ваш сервер серьезно атакован, а не просто кто-то ищет любой плохо защищенный сервер, чтобы установить на нем свой вирус, то атакующие обязательно просканируют все порты. Тогда рано или поздно враг обнаружит ваш SSH-сервер, независимо от того, на каком порту тот работает.

Кроме того, метод изменения порта имеет еще один недостаток, которым нельзя пренебречь. Конфигурация большинства брандмауэров настроена так, что они не перекрывают трафик, идущий через порт 22, а остальные порты вполне могут перекрываться. Если вы работаете на фирме всего пару дней и хотите быстро войти на сервер через SSH, то брандмауэр фирменной сети вас, скорее всего, не пропустит.

Аутентификация с помощью ключей

При работе с SSH-сервером лучше всего использовать для аутентификации не пароль, а специальный ключ. Создание и распределение SSH-ключа подробно описано в разделе 6.2.

Если SSH-соединение, в котором используется ключ (то есть не требуется логин) работает нормально, то можно изменить на сервере конфигурационный файл `sshd_config`. Определяющее значение имеют две строки:

```
# в /etc/ssh/sshd_config
...
PasswordAuthentication no
UsePAM                 no
```

Теперь SSH-аутентификация возможна *только* с помощью ключа. Но следите за тем, чтобы не выгнать себя самого из собственной системы! Если вы потеряете ключ на своем клиентском компьютере, то уже никак не сможете войти на сервер!

Это и есть основной недостаток аутентификации с применением ключей. Если при использовании паролей вы можете войти на сервер с любого компьютера, то при работе с ключами компьютер для входа на сервер должен иметь специальную конфигурацию. Если же у вас нет доступа к этому (этим) компьютерам, например вы в дороге, у вас сломался ноутбук и т. д. вы не сможете войти на сервер. Здесь мы наблюдаем парадокс: чем надежнее защита, тем меньше гибкость системы...

Библиотека TCP-Wrapper

SSH-сервер работает с особой библиотекой, называемой TCP-Wrapper. Эта библиотека с помощью конфигурационных файлов `/etc/hosts.allow` и `/etc/hosts.deny` позволяет указывать, с каких сетевых адресов можно использовать SSH-сервер. Если SSH-сервер установлен на корневом сервере, то такой метод не очень целесообразен, ведь SSH должен быть доступен во всем Интернете. Но если SSH-сервер установлен на сервере локальной сети и администрирование будет выполняться только в ее пределах, то будет очень рационально ограничить таким образом возможности доступа. О библиотеке TCP-Wrapper и ее конфигурационных файлах подробно рассказано в разделе 18.2.

Сценарий DenyHosts

Сценарий `DenyHosts`, написанный на языке Python, отслеживает все попытки входа на SSH. Как только он обнаруживает, что с того или иного IP-адреса поступило много неудачных попыток входа, этот IP-адрес автоматически заносится в файл `/etc/hosts.deny`. В зависимости от конфигурации файл может или остаться там на-

всегда, или через некоторое время (день или неделю) будет оттуда удален. Сценарий DenyHosts эффективно пресекает попытки автоматического входа в систему (но не может противостоять массированной атаке, которая одновременно выполняется с очень большого количества разных компьютеров) и к тому же хорошо зарекомендовал себя на тех серверах, с которыми мне доводилось работать. Для многих дистрибутивов существуют готовые пакеты с этим сценарием. Более подробная информация и самая новая версия сценария содержится на следующем сайте: http://denyhosts.sourceforge.net/.

Сценарий DenyHosts в свою очередь запускается сценарием Init-V и интерпретирует конфигурационный файл /etc/|denyhosts.conf. Очень важно, чтобы DenyHosts наблюдал за правильным файлом регистрации (параметр SECURE_LOG). В следующем листинге приведено несколько строк из конфигурации, действующей на моем сервере kofler.cc:

```
# Файл /etc/denyhosts.conf
SECURE_LOG = /var/log/auth.log
HOSTS_DENY = /etc/hosts.deny
# Снова разблокировать блокированные IP-адреса через 24 часа
PURGE_DENY = 1d
# Блокировать после трех неудачных попыток входа в систему с неправильным логином
DENY_THRESHOLD_INVALID = 3
# Блокировать после пяти неудачных попыток входа в систему с правильным логином
DENY_THRESHOLD_VALID = 5
# Блокировать учетную запись администратора после одной неудачной попытки входа
в систему
DENY_THRESHOLD_ROOT = 1
```

19.2. Apache

Apache — это веб-сервер из мира свободного ПО. Согласно данным netcraft.com, по состоянию на май 2009 года около 47 % всех сайтов работают с Apache. Если же учитывать только миллион наиболее посещаемых сайтов, то доля Apache на рынке оказывается просто баснословной — 68 %. В этой главе мы лишь немного коснемся Apache. Актуальная информация, а также подробная документация по Apache приводится на официальном сайте сервера: http://www.apache.org.

Установка

Обычно установленная копия Apache состоит из нескольких взаимосвязанных пакетов: самого сервера, различных библиотек, плагинов, языков программирования и др. Чтобы облегчить установку, в некоторых дистрибутивах предусмотрен выбор для установки сразу целой группы пакетов.

В Fedora в консоли выполняется команда yum groupinstall 'Web-Server'. В SUSE в разделе YaST, отведенном для управления пакетами, выбирается Сервер ▸ Веб- и LAMP-сервер, а затем устанавливаются все нужные пакеты. В Ubuntu с помощью Synaptic или apt-get устанавливается пакет apache2-mpm-prefork. Вместе с ним устанавливается и несколько других зависящих от него пакетов.

Начиная с версии 2, Apache поддерживает три различных режима многопоточной работы: `perchild`, `prefork` и `worker`. От выбранного метода зависит, насколько эффективно Apache сможет синхронно обрабатывать несколько запросов. При установке Apache вам необходимо выбрать один из трех этих вариантов. Если вы собираетесь использовать вместе с Apache язык программирования PHP, то лучше всего выбрать `prefork`. При работе с другими вариантами возможны ошибки, так как их библиотеки не приспособлены к работе с потоками (http://www.php.net/manual/en/faq.installation.php).

Запуск/остановка. Apache — это демон, который в некоторых дистрибутивах нужно специально запускать. Важнейшие команды для управления им обобщены в разделе 4.5. Названия сценария Init-V в разных дистрибутивах могут быть различными: `apache2` в Debian, SUSE и Ubuntu или `httpd` в Fedora и Red Hat.

Название программы и учетная запись. Названия программы, представляющей собой веб-сервер Apache, в разных дистрибутивах также различаются. По причинам, связанным с безопасностью, веб-сервер, подобно многим другим сетевым демонам, выполняется не от имени администратора, а под другой учетной записью. Чтобы узнать ее название, лучше всего выполнить `ps axu`. В табл. 19.1 показано, что названия программы и учетной записи в различных дистрибутивах неодинаковы.

Таблица 19.1. Название программы, учетная запись и каталог DocumentRoot Apache

Дистрибутив	Название программы	Учетная запись	DocumentRoot
Debian, Ubuntu	apache	www-data	/var/www
Fedora, Red Hat	httpd	apache	/var/www/html
SUSE	httpd2-thread«метод»	wwwrun	/srv/www/htdocs

Тестирование. Чтобы проверить, все ли работает, откройте браузер и введите в адресную строку `http://localhost/` или `http://`*`имя_сервера`*`/`. Теперь вы должны увидеть тестовую страницу веб-сервера (рис. 19.1).

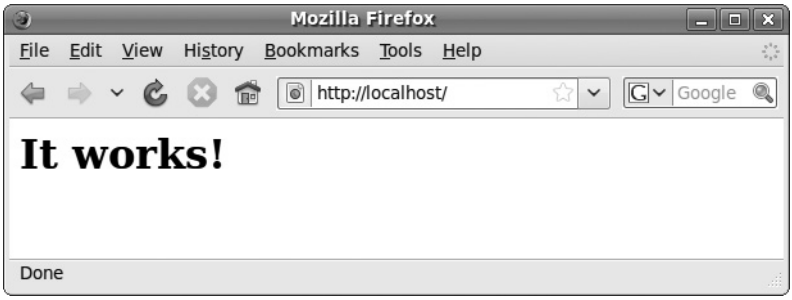

Рис. 19.1. Тестовая страница Apache на компьютере с Ubuntu

Собственные HTML-страницы. Чтобы вместо тестовой страницы на экране появилась главная страница вашего сайта, нужно сохранить HTML-файлы в каталоге документов Apache. Этот каталог также по-разному называется в различных дистрибутивах (ключевое слово `DocumentRoot` в конфигурационных файлах, см. табл. 19.1). Ваши HTML-файлы должны быть доступны для чтения с той учетной записи, под которой выполняется веб-сервер Apache!

Конфигурация

Размеры книги не позволяют подробно описать конфигурацию Apache. Но хотя бы рассмотрим, где располагаются конфигурационные файлы различных дистрибутивов и как производятся простейшие настройки.

Ранее конфигурация Apache выполнялась в файле `httpd.conf`, точное местоположение которого зависело от дистрибутива. Но со временем этот конфигурационный файл становился все более путаным. Вместе с этим возрастала сложность автоматизированной технической поддержки, в частности активизации и деактивизации плагинов.

В результате в большинстве дистрибутивов стал использоваться другой метод — настройки распределялись между несколькими файлами, а затем считывались из различных каталогов с помощью команд `include` (табл. 19.2–19.4). Такой метод позволяет сделать отдельные файлы более понятными, но сама система становится совсем запутанной. Кроме того, становится практически невозможно перенести настройки одного дистрибутива в другой. Если вы ищете в конфигурационных файлах определенное ключевое слово, то лучше всего действовать так:

```
user$ cd /etc/httpd (или) cd /etc/apache2
user$ find -type f -exec grep -i -q ключевое_слово ^; -print
```

Таблица 19.2. Конфигурация Apache в Debian и Ubuntu

Файлы	Содержание
/etc/apache2/apache2.conf	Исходный пункт
/etc/apache2/httpd.conf	Пользовательская конфигурация
/etc/apache2/ports.conf	Отслеживаемые порты, обычно — 80
/etc/apache2/mods-available/*	Доступные дополнительные модули
/etc/apache2/mods-enabled/*	Ссылки на активные дополнительные модули
/etc/apache2/conf.d/*	Прочие конфигурационные файлы
/etc/apache2/sites-available/*	Доступные сайты (виртуальные хосты)
/etc/apache2/sites-enabled/*	Ссылки на активные сайты
/etc/apache2/envvars	Переменные окружения для сценария Init-V

Таблица 19.3. Конфигурация Apache в Fedora и Red Hat

Файлы	Содержание
/etc/httpd/conf/httpd.conf	Исходный пункт
/etc/httpd/conf/magic	Конфигурация типов MIME (для mod_mime)
/etc/httpd/conf.d/*.conf	Файлы для конфигурации модулей

Таблица 19.4. Конфигурация Apache в SUSE

Файлы	Содержание
/etc/apache2/httpd.conf	Исходный пункт
/etc/apache2/*.conf	Глобальные конфигурационные файлы
/etc/apache2/sysconf.d/*.conf	Автоматически генерируемые системные конфигурационные файлы
/etc/apache2/conf.d/*.conf	Прочие конфигурационные файлы
/etc/apache2/vhosts.d/*.conf	Сайты (виртуальные хосты)
/etc/sysconfig/apache2	Основные настройки

В Debian/Ubuntu в каталоге mods-available содержится коллекция файлов *.load и *.conf для различных модулей Apache. Для активизации других модулей создайте в mods-enabled ссылки на эти файлы. При управлении ссылками в Debian вам пригодятся специфичные для этого дистрибутива команды a2enmod и a2dismod. В дальнейшем вы сможете активизировать и деактивизировать виртуальные хосты с помощью команд a2ensite и a2dissite. По умолчанию в sites-available содержится только файл default: он конфигурирует стандартную веб-страницу сервера (каталог /var/www), а кроме того, содержит разнообразные базовые настройки для протоколирования ошибок и обращений к страницам.

Механизм работы такой же, как и с модулями: в каталоге sites-available содержатся все конфигурационные файлы для всех хостов, а в sites-enabled находятся соответствующие ссылки.

В SUSE все CONF-файлы из каталога sysconf.d при каждом запуске Apache создаются сценарием Init-V /etc/init.d/apache2 заново! По этой причине вносить изменения в эти файлы бессмысленно. Напротив, вам следует изменить переменные, находящиеся в /etc/sysconfig/apache2. Кроме того, в данном файле определяется, какие модули должны загружаться при запуске Apache (переменная APACHE_MODULES). Если хотите добавить к конфигурационным файлам SUSE собственный файл, укажите его название в переменной APACHE_CONF_INCLUDE_FILES.

Тестирование конфигурации

Изменив синтаксис, с помощью команд httpd -t, httpd2 -t или apache2 -t вы можете проверить, нет ли в конфигурации синтаксических ошибок. В Debian и Ubuntu сначала нужно считать из файла envvars некоторые переменные окружения:

```
root# . /etc/apache2/envvars
root# apache2 -t
Syntax OK
```

После этого прикажите Apache заново считать конфигурационные файлы:

```
root# /etc/init.d/имя_init-сценария reload
```

Переменная ServerName

Обычно веб-сервер Apache запускается сразу. Но в зависимости от настроек конкретной сети вам потребуется изменить или добавить в конфигурационных файлах как минимум одну строку: переменная ServerName должна содержать имя вашего компьютера. Если эта настройка не подействует, укажите UseCanonicalName Off.

```
# в /etc/apache2/httpd.conf          (Debian/Ubuntu)
# или /etc/httpd/conf/httpd.conf     (Fedora/Red Hat)
ServerName mars.sol     # здесь укажите имя вашего компьютера
```

В SUSE имя компьютера записывается в файле /etc/sysconfig/apache2 в переменной APACHE_SERVERNAME.

Стандартная кодировка

Во всех распространенных дистрибутивах Linux автоматически применяется кодировка Unicode UTF-8. Иначе говоря, если вы создаете в текстовом редакторе

новый текстовый файл, в котором есть специальные символы, например немецкие буквы ä, ö, ü или ß, они сохраняются в кодировке UTF-8.

Для Apache в принципе неважно, в какой кодировке сохранены файлы. Программа просто переносит файлы байт за байтом в браузер, запросивший страницу. Но Apache посылает вместе со страницей так называемый *заголовок*, где в числе прочего указано, в какой кодировке создана страница. Браузер интерпретирует эту информацию и использует указанную кодировку при отображении сайта.

Настройка кодировки. Теперь Apache должен правильно указать кодировку. Если этого не получится сделать, пользователь увидит у себя в браузере не ä или ü, а какие-нибудь причудливые комбинации символов. Во избежание такого в Apache предусмотрена возможность настройки конфигурации кодировки.

- `AddDefaultCharset off` — при такой настройке Apache интерпретирует META-тег передаваемого HTML-файла и сообщает браузеру, какая кодировка указана в этом теге. Если файл HTML начинается так, как это показано ниже, то применяется кодировка Unicode UTF-8:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
...
```

- `AddDefaultCharset charset` — Apache сообщает указанную здесь кодировку всем страницам браузера. Настройка действует для HTML- и PHP-файлов. META-тег в HTML-коде игнорируется.
- `AddDefaultCharset charset extension` — так настраивается кодировка для файлов, имеющих определенное расширение. Если указать `AddCharset utf-8 .utf8`, то все файлы, название которых заканчивается на `.utf8`, будут посланы в браузер в кодировке Unicode UTF-8. Для работы `AddCharset` требуется модуль Apache mod_mime, который по умолчанию активен во всех дистрибутивах, рассмотренных далее.

Debian, Ubuntu. Разумеется, стандартная конфигурация в разных дистрибутивах неодинакова. Для глобальной настройки кодировки в Ubuntu предусмотрен конфигурационный файл `/etc/apache2/conf.d/charset`. По умолчанию он пуст, то есть действует `AddDefaultCharset off`.

Кроме того, `AddDefaultCharset` и `AddCharset` можно использовать в конфигурационных файлах для виртуальных хостов (каталог sites-available), а также в файлах `.htaccess`, если вам нужна особая конфигурация отдельно взятого хоста или каталога. Но не забывайте, что настройки кодировки, указываемые в `.htaccess`, учитываются лишь в тех случаях, когда для веб-каталога задан параметр `AllowOverride All` или `FileInfo`.

Fedora, Red Hat. В Fedora и Red Hat также может применяться `AddDefaultCharset UTF-8`. Эта настройка находится в файле `/etc/httpd/conf/httpd.conf`. В том же файле находится настройка `AllowOverride None` для каталога `/var/www/html`.

SUSE. В SUSE отсутствуют особые функции для задания кодировки в конфигурационных файлах. Таким образом, действует `AddDefaultCharset off`, то есть кодировка определяется только по данным, указанным в META-теге HTML-файла. Для настройки `AddDefaultCharset` хорошо подходит файл `/etc/apache2/mod_mime-defaults.conf`.

Кроме того, в SUSE действует `AllowOverride None` для каталога `/srv/www/htdocs`. Эту настройку можно изменить в файле `/etc/apache2/default-server.conf`.

Обеспечение безопасности при работе дома или внутри организации

Если вы собираетесь использовать Apache только в рамках отдельно взятой фирмы или в локальной сети, то вам потребуется запретить доступ к серверу извне. Это можно сделать, используя брандмауэр (см. главу 18) либо изменив несколько строк в конфигурационных файлах.

<Directory>. В принципе управление доступом к отдельным веб-каталогам происходит в разделах `<Directory>` конфигурационных файлов (как обычно, их расположение различается от дистрибутива к дистрибутиву). При этом, если имеются настройки для корневого каталога, все остальные каталоги соблюдают их по умолчанию. Затем в иных разделах `<Directory>` для конкретных каталогов могут быть указаны и другие настройки, отличающиеся от стандартных. Из следующего примера должно быть понятно, каковы значения ключевых слов `Order`, `Deny` и `Allow`.

Приведенный далее код действует так, что все веб-страницы, находящиеся в стандартном каталоге, могут быть запрошены только страницами, расположенными в локальной сети. Кроме того, перестают работать символьные ссылки, так как они зачастую серьезно угрожают безопасности сети (обратите внимание, что `deny,allow` указывается без пробела!). Действующие строгие правила немного смягчаются для каталога `/var/www/html/public`. Благодаря параметру `Indexes` в браузере отображается список файлов, находящихся в том или ином каталоге, если в нем отсутствует файл `index.html`.

```
# Более надежная стандартная конфигурация, задаваемая для всего дерева каталогов
<Directory />
  Options None              # Без параметров
  AllowOverride None        # .htdocs не оказывает никакого влияния
  Order deny,allow          # Сначала запретить, затем разрешить
  Deny from all             # Запретить все, но...
  Allow from 192.168.0.0/8  # Доступ из локальной сети
  Allow from .sol           # Разрешить доступ из локальной сети
  Allow from localhost      # Доступ для localhost
</Directory>
# Разрешить доступ к Интернету на http://<site>/public (для Fedora/Red Hat)
<Directory "/var/www/html/public">
  Options Indexes FollowSymLinks
  Order allow,deny
  Allow from all            # Свободный доступ для всех!
</Directory>
```

Защита веб-каталогов паролем

В предыдущем разделе было описано, как предоставить доступ к веб-серверу только тем компьютерам, которые находятся в локальной сети. Но иногда бывает необходимо не запрещать доступ к серверу полностью, а разрешать его только после ввода верного пароля.

Такой метод может применяться в тех случаях, когда у вас на сайте есть несколько административных страниц (например, PhpMyAdmin для администрирования MySQL, подробнее описано в разделе 19.4). Неважно, где вы (администратор MySQL) сейчас находитесь, — если хотите, то можете получить доступ к этой странице. Одновременно следует исключить возможность выхода обычных пользователей на административные страницы.

Файл с паролем

Для решения таких проблем каталог необходимо защитить с помощью файла с паролем. По возможности файл с паролем нужно сохранять за пределами веб-каталога, чтобы исключить доступ к этому файлу по адресу http://servername/verzeichnis/password-file. В следующем примере предполагается, что файл с паролем сохранен в каталоге /private.

Для создания нового файла с паролем используйте команду htpasswd (htpasswd2 в SUSE) с параметром -c (create). Пароль, разумеется, зашифровывается:

```
user$ cd /private
user$ htpasswd -c passwords.pwd имя_пользователя
New password: ********
Re-type new password: ********
Adding password for user username
```

Остальные пары логин/пароль добавляются командой htpasswd без параметра -c.

```
user$ cd /private
user$ htpasswd passwords.pwd имя2
New password: ********
Re-type new password: ********
Adding password for user username
```

ПРИМЕЧАНИЕ

Не забудьте дать Apache право чтения файла с паролем и каталога, в котором этот файл находится! Из соображений безопасности Apache работает не под учетной записью администратора, а под другой учетной записью (www-date в Debian и Ubuntu, apache в Fedora и Red Hat, wwwrun в SUSE).

Если ваш компьютер защищен SELinux или AppArmor (см. разделы 18.8–18.9), то правила этих систем безопасности также не должны мешать доступу сервера к файлу с паролем.

Существует два варианта конфигурации Apache, при которых пароли действительно учитываются. При использовании первого варианта требуется изменить центральные конфигурационные файлы (httpd.conf и пр.). Второй вариант предусматривает конфигурацию в файле .htaccess, находящемся внутри веб-каталога.

Файл httpd.conf

Чтобы Apache учитывал файл с паролем, нужно вставить защищаемый паролем каталог в httpd.conf в виде отдельного раздела <Directory>. В следующем примере показано, как это делается:

```
# Каталог, защищенный паролем
<Directory "/var/www/html/admin/">
  AuthType Basic
```

```
  AuthUserFile /private/passwords.pwd
  AuthName "admin"
  Require valid-user
  # Другие параметры, если они требуются
</Directory>
```

Параметр AuthName обозначает область, в которую разрешен доступ. Смысл в том, что вам не требуется каждый раз заново входить в систему, если вы хотите получить доступ к разным каталогам, которые защищены одним и тем же паролем. Если вы вошли в систему с определенным обозначением AuthName, то имеете доступ и ко всем остальным каталогам с таким же AuthName.

Настройка Require valid-user означает, что для входа в систему может применяться любое действующее сочетание имени пользователя и пароля. Вы также можете здесь указать, что вход в систему разрешен только определенному пользователю:

```
Require user имя1 имя2
```

Файлы .htaccess

Описанный выше процесс можно осуществить лишь в том случае, если у вас есть доступ к центральным конфигурационным файлам Apache, то есть если вы сами являетесь администратором сети. Если вы не администратор, то можете обеспечить не менее надежную защиту с помощью файла .htaccess, который находится в каталоге, защищенном паролем. В этом файле должны быть указаны те же команды, что и в группе <Directory>, то есть AuthType, AuthUserFile, AuthName и Require.

Обратите внимание — файл .htaccess учитывается лишь при условии, что в httpd.conf в группе <Directory> данного каталога допускается изменение информации, связанной с аутентификацией (вместо AllowOverride AuthConfig также может быть указано AllowOverride All). На веб-серверах, где отдельные пользователи отвечают за собственные сетевые каталоги, это условие обычно выполняется.

```
# Каталог, который можно защитить с помощью .htaccess
<Directory "/var/www/html/admin/">
  AllowOverride AuthConfig
  # Другие параметры, если они требуются
</Directory>
```

19.3. PHP

Сам Apache может передавать только статические веб-страницы. Но на всех современных сайтах используются и динамические веб-страницы. Всякий раз при запросе такой страницы Apache запускает внешнюю программу, перерабатывает код страницы и отправляет в ответ страницу, которая отдельно настраивается для отображения в браузере (например, на странице может быть указано точное время или результат обработки запроса, посланного в базу данных, либо постоянно переключающиеся рекламные баннеры и т. д.).

Для программирования динамических веб-страниц применяется несколько языков, например Perl или Java. Но в мире Linux/UNIX наиболее популярен язык

PHP — *препроцессор гипертекста*. Подробная информация о PHP приводится на следующем сайте: http://www.php.net/.

Основная идея написания сайтов на PHP заключается в том, что в файле с расширением *.php содержится код, написанный как на HTML, так и на PHP. Код PHP начинается тегом <?php и завершается тегом ?>. Если пользователь запрашивает в Интернете страницу, на которой есть PHP, то Apache передает ее интерпретатору PHP. Там и выполняется PHP-код. Результат выполнения этого кода встраивается непосредственно в HTML-файл. Интерпретатор PHP передает полученную в результате страницу обратно Apache, а сервер, в свою очередь, отсылает ее пользователю. Таким образом, в браузере не видно никакого PHP-кода, а только готовая HTML-страница (та же концепция применяется Microsoft в технологии ASP — *активные серверные страницы*).

Hello World! В этой книге я не смогу дать даже вводной информации о языке программирования PHP. Но следующий мини-пример позволяет понять принцип действия PHP. Следующий файл после обработки PHP-интерпретатора превращается в страницу HTML, на которой указано точное время:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//RU">
<html><head>
  <meta http-equiv="Content-Type"
        content="text/html; charset=iso-8859-1" />
  <title>Пример PHP</title>
</head><body>
<p>Текущее время на данном сервере:
  <?php echo date("G:i:s"); ?>
</p>
</body></html>
```

Установка

Если PHP не был установлен вместе с Apache, то вам потребуется, воспользовавшись программой управления пакетами, установить все нужные пакеты php5. Однако не так просто определить, какие пакеты действительно нужны: подобно Apache, PHP часто разделен на несколько пакетов, в которых содержится сам язык и различные важные настройки. Для того чтобы начать работу с этим языком, обычно будет достаточно php5, php5-common, а также libapache2-mod-php5. Если после окончания установки программа управления пакетами не перезапустит Apache, сделайте это сами, чтобы веб-сервер в дальнейшем учитывал добавленный модуль PHP.

Конфигурация

Бесчисленные параметры интерпретатора PHP регулируются в файле php.ini. Как правило, вы можете оставить настройки, заданные по умолчанию. Место расположения данного файла зависит от дистрибутива:

- Debian, Ubuntu — /etc/php5/apache2/php.ini;
- Fedora, Red Hat — /etc/php.ini;
- SUSE — /etc/php5/apache2/php.ini.

Тестирование

Чтобы проверить, работает ли установленная копия PHP, создайте файл `phptest.php`, в котором должна быть лишь одна строка кода:

```
<?php phpinfo(); ?>
```

Скопируйте этот файл в каталог DocumentRoot (см. раздел 19.2) и убедитесь, что Apache может читать этот файл. Хотя файлы PHP, по существу, являются сценарными, прав на чтение будет достаточно. Права на исполнение файла (биты доступа `x`) не требуются.

Теперь в браузере вы можете просмотреть страницу http://localhost/phptest.php (рис. 19.2). Она будет очень велика, и на ней будут перечислены всевозможные параметры настройки Apache и PHP (из соображений безопасности не рекомендуется помещать такую страницу в свободном доступе в Интернете — здесь содержится масса информации о конфигурации вашего компьютера).

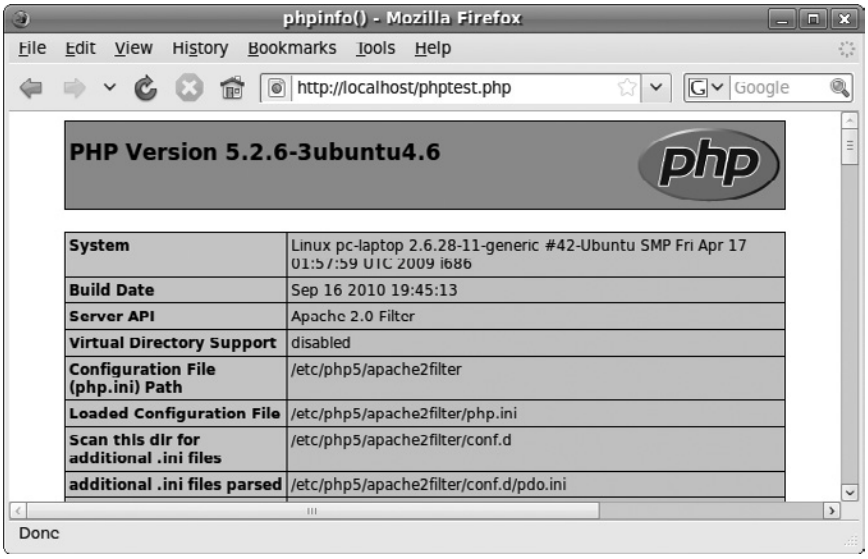

Рис. 19.2. Тестовая страница PHP

Проблемы

Если вместо тестовой страницы с кодом PHP вы получаете предложение загрузить PHP-файл, то причина ошибки, скорее всего, заключается в том, что в качестве веб-адреса было указано имя файла (например, `/srv/www/htdocs/phpinfo.php`). В таком случае файл считывается прямо из локальной системы и не обрабатывается Apache и интерпретатором PHP. Веб-адрес должен начинаться с http://!

Еще одна причина возникновения ошибки может заключаться в неверной конфигурации Apache. Не забыли ли вы перезапустить сервер Apache после того, как установили PHP или изменили конфигурационные файлы?

Если что-то не получается, посмотрите, какая информация содержится в кэше браузера, — это может дать ключ к разгадке. Вместо того чтобы снова запрашивать у Apache страницу (маловероятно, что это сработает), браузер считает ее из собственного внутреннего кэша. Ради обеспечения безопасности перезапустите программу и очистите кэш.

19.4. MySQL

В настоящее время MySQL — наиболее популярная база данных из всех, что используются в Linux. MySQL быстро функционирует, ее техническая поддержка не отличается сложностью, и к тому же эту базу данных легко программировать. MySQL особенно активно применяется в сочетании с PHP для создания динамических веб-страниц. Различные готовые веб-приложения (Joomla!, TYPO3 и т. д.) построены на базе PHP и MySQL.

По сравнению с коммерческими базами данных, например Oracle или IBM DB/2, в MySQL предлагается меньше функций, но для решения большинства практических задач набора ее функций вполне достаточно. Информация о MySQL, а также очень подробный учебник имеется по следующему адресу: http://dev.mysql.com.

Лицензия

MySQL, а также ее драйверы для работы с различными языками программирования соответствуют лицензии GPL. Использование MySQL в рамках отдельно взятой фирмы (без передачи или коммерческой реализации) является бесплатным. Но учитывайте, что для передачи коммерческих проектов *с закрытым кодом*, не соответствующих GPL и построенных на основе MySQL, требуется коммерческая лицензия MySQL-сервера. Вопрос лицензирования MySQL подробно рассмотрен на следующем сайте: http://www.mysql.com/about/legal/.

В начале 2009 года компания Oracle объявила о намерении приобрести Sun — владельца MySQL. Как это отразится на будущем MySQL, пока неясно. Но уже сейчас можно с уверенностью говорить о том, что наряду с официальной MySQL появятся так называемые *ветвления* (forks), то есть новые проекты, полученные на базе открытого кода. Особенно активно действует основатель MySQL Monty Widenius, который создал для этих целей *Open Database Alliance* (http://opendatabasealliance.com/).

Книга о PHP и MySQL. В этом разделе будет рассмотрена только установка MySQL. Разумеется, можно рассказать еще об очень многих вещах, касающихся организации баз данных, управления пользовательскими правами, программирования приложений, работающих с базами данных и администрирования. Если вы жаждете познакомиться с этими темами поближе, то к вашим услугам множество книг и пособий. Хотелось бы указать, что я также не остался в стороне и выпустил в издательстве Addison-Wesley две такие книги: по MySQL и по PHP и MySQL. Содержание и пробные главы из этих книг можно скачать на сайте http://www.kofler.cc/.

Установка и обеспечение безопасности

Во всех распространенных дистрибутивах есть пакеты с MySQL. Обратите внимание, что сам сервер базы данных, его библиотеки, инструменты администрирования и другое устанавливаются в различных пакетах. Далее указано, какие пакеты вам могут понадобиться:

- Debian, Ubuntu — `mysql-client`, `mysql-common`, `mysql-server`;
- Fedora, Red Hat — `mysql`, `mysql-server`;
- SUSE — `mysql`, `mysql-client`, `libmysqlclientn`.

В Fedora целесообразно устанавливать не отдельные пакеты, а заранее составленную группу пакетов MySQL:

```
root# yum groupinstall mysql
```

Запуск/установка

MySQL — это демон, который в отдельных дистрибутивах нужно специально запускать. Команда, используемая для этого, в различных дистрибутивах называется по-разному. В обобщенном виде эта информация дана в разделе 4.5. В зависимости от дистрибутива сценарий Init-V называется `mysql` или `mysqld`.

Файлы базы данных сервера MySQL обычно сохраняются в каталоге `/var/lib/mysql`. Файлы регистрации обычно находятся в каталогах `/var/log/syslog` (Debian, Ubuntu), `/var/lib/mysql/`*`хост-имя`* (SUSE) или `/var/log/mysql*`.

Конфигурация

Конфигурация сервера MySQL выполняется в файле `/etc/my.cnf` или `/etc/mysql/my.cnf` (Debian, Ubuntu). Этот файл уже приготовлен для решения повседневных задач. Ради экономии места я не буду описывать здесь все ключевые слова этого файла, но остановлюсь на некоторых деталях, важных с точки зрения безопасности.

Как правило, конфигурация разбивается на несколько фрагментов указанием вида `[`*`имя`*`]`. Далее мы рассмотрим только раздел `[mysqld]`, касающийся самого сервера. Требования, предъявляемые к этому разделу, вступают в силу только после перезапуска сервера MySQL. Остальные разделы относятся к конфигурации различных клиентских программ.

- `old_passwords=1` — благодаря данной настройке MySQL сохраняет внутренние пароли так, как это делалось в версиях MySQL 4.0 и ниже. В некоторых дистрибутивах (например, в Fedora 11) эта настройка все еще используется по умолчанию, позволяя обойти проблемы совместимости, возникающие при работе со старыми экземплярами MySQL. Однако настройка сильно снижает защищенность системы доступа к MySQL.

  Опытные пользователи MySQL, которые могут работать и без оглядки на старые версии MySQL, по возможности должны использовать вариант `old_passwords=0` или просто удалить эту команду из `my.cnf` *до того*, как настраивать какие-либо пароли! Чтобы измененные настройки вступили в силу, перезапустите сервер MySQL. Подробная базовая информация о старой и новой системах аутентифи-

кации дается в учебнике по MySQL по адресу http://dev.mysql.com/doc/refman/5.1/en/password-hashing.html.

- `bind-address=127.0.0.1` — при такой настройке сетевое соединение с сервером MySQL можно установить лишь с локального компьютера (ни с одного другого компьютера из тех, что находятся в локальной сети или Интернете). Если MySQL должна использоваться только локальными программами (например, PHP), то данная настройка помогает повысить безопасность. В Debian и Ubuntu настройка действует по умолчанию, а в остальных дистрибутивах ее нужно добавить, если это возможно.
- `skip-networking` — закрывает любой доступ из сети к серверу MySQL, даже сетевые соединения, устанавливаемые с локального компьютера. Таким образом, это еще более ограничительная настройка, чем `bind-address=127.0.0.1`. В данном случае соединение с сервером могут устанавливать только локальные программы, обменивающиеся информацией с сервером MySQL через так называемый сокет-файл, например программы, написанные на PHP и на C. Программы, которые обмениваются с MySQL-сервером информацией по протоколу TCP/IP (например, все программы Java), не смогут работать с MySQL-сервером! При возникновении сомнений используйте `bind-address=127.0.0.1`.

Пароль администратора

В Debian и Ubuntu в ходе установки сервера MySQL вам понадобится указать пароль администратора. Как и в Linux, в MySQL такой пароль имеет особое значение и предоставляет неограниченные права администрирования.

Кроме того, в Debian и Ubuntu создается учетная запись debian-sys-main, защищенная паролем, который генерируется случайным образом и сохраняется в файле `/etc/mysql/debian.cnf`, где в виде незашифрованного текста он доступен только администратору Linux. Сценарий `/etc/mysql/debian-start`, необходимый для запуска MySQL, обращается к этой учетной записи, поэтому учетную запись debiansys-main ни в коем случае нельзя отключать! При изменении пароля нужно соответствующим образом обновить и `debian.cnf`.

В Fedora, Red Hat и SUSE такая дополнительная защита MySQL не предусмотрена, поэтому после завершения установки любой пользователь, применив логин root, может без пароля установить соединение с MySQL. Для этого применяются следующие команды:

```
user$  mysql -u root
mysql> UPDATE mysql.user SET password=PASSWORD('xxx') WHERE user='root';
mysql> FLUSH PRIVILEGES;
mysql> exit
```

Если в дальнейшем вы планируете выполнять команды SQL с помощью `mysql`, войти в систему нужно так:

```
user$ mysql -u root -p
Enter password: ******
```

Логины и пароли, используемые в Linux и MySQL, никак не связаны друг с другом. Из соображений безопасности ни в коем случае не применяйте для одного

и того же пользователя Linux и MySQL одинаковый пароль. Часто пароли MySQL сохраняются в программном коде, поэтому защитить их гораздо сложнее, чем обычные пароли Linux.

Анонимные пользователи MySQL

Теперь пользователь MySQL root защищен паролем. Но в некоторых дистрибутивах с базой данных могут работать и анонимные пользователи. Это означает, что на сервер MySQL может войти любой пользователь с любым логином. Хотя права анонимных пользователей сильно ограничены, такой пользователь может быть опасен и лучше не допускать его в систему.

Чтобы узнать, есть ли в системе анонимные пользователи, выполните команду `mysql`. У анонимных пользователей в результате выполнения команды `SELECT` столбец `user` не будет содержать никакой информации. С помощью команды `DELETE` таких пользователей можно удалить. Команда `FLUSH PRIVILEGES` сразу же активизирует подобное изменение базы данных.

```
user$ mysql -u root -p
Enter password: *******
mysql> SELECT user, host, password FROM mysql.user;
+------+----------------+-------------------------------------------+
| user | host           | password                                  |
+------+----------------+-------------------------------------------+
| root | localhost      | *FACF054603403E6836B8DCFCB1EAC269746E8720 |
| root | uranus-suse111 | *FACF054603403E6836B8DCFCB1EAC269746E8720 |
| root | 127.0.0.1      | *FACF054603403E6836B8DCFCB1EAC269746E8720 |
|      | localhost      |                                           |
|      | uranus-suse111 |                                           |
+------+----------------+-------------------------------------------+
mysql> DELETE FROM mysql.user WHERE user='';
mysql> FLUSH PRIVILEGES;
mysql> exit
```

Первые тесты

Создание новой базы данных и новых учетных записей пользователей. Чтобы протестировать MySQL, вы должны знать язык SQL, предназначенный для работы с базами данных. В данном случае я предполагаю, что вы действительно его знаете. Цель команды, показанной ниже, — создать новую базу данных `mydatabase` и новую учетную запись пользователя newuser, который будет иметь право доступа к этой базе. Для решения таких задач лучше всего использовать программу `mysql`. Она нужна примерно для того же, что и Shell в Linux: принимает команды MySQL, переадресовывает их на MySQL-сервер и, наконец, отображает результат. При этом все команды MySQL должны заканчиваться точкой с запятой.

```
user$ mysql -u root -p
Enter password: ******
```

```
...
mysql> CREATE DATABASE mydatabase;
mysql> GRANT ALL ON mydatabase.* TO newuser@localhost
       IDENTIFIED BY 'xxxxxxxxx';
mysql> exit
```

Пользователь newuser может проводить с новоиспеченной базой данных любые другие тесты. Как и в Linux, в MySQL лучше как можно меньше работать под учетной записью root.

Создание таблиц и заполнение их данными. С помощью следующих команд newuser создает новую таблицу (`CREATE TABLE`), вставляет в нее определенные наборы данных (`INSERT`) и, наконец, просматривает все наборы данных (`SELECT`). Особое значение в данной таблице имеет столбец `id`: сервер MySQL самостоятельно вставляет туда каждый новый набор данных, присваивая ему уникальный номер, который потом используется для идентификации этого набора данных.

```
user$ mysql -u newuser -p
Enter password: ******
mysql> USE mydatabase;
mysql> CREATE TABLE mytable (
         id INT NOT NULL AUTO_INCREMENT,
         txt VARCHAR(100),
         n INT,
         PRIMARY KEY(id));
mysql> INSERT INTO mytable (txt, n) VALUES('abc', 123);
mysql> INSERT INTO mytable (txt, n) VALUES('efgsd', -4);
mysql> INSERT INTO mytable (txt, n) VALUES(NULL, 0);
mysql> SELECT * FROM mytable;
+----+-------+------+
| id | txt   | n    |
+----+-------+------+
| 1 | abc    | 123  |
| 2 | efgsd  |  -4  |
| 3 | NULL   |   0  |
+----+-------+------+
mysql> exit
```

Администрирование MySQL

В принципе все административные операции при работе с MySQL можно выполнять с помощью текстовых команд `mysql` и `mysqladmin`, но это не очень удобно. В качестве альтернативы можно воспользоваться одним из многих интерфейсов, которые помогают создавать базы данных, вводить данные, управлять пользователями MySQL и т. д. К официальным инструментам администрирования относятся MySQL Administrator (рис. 19.3), MySQL Query Browser и новая программа MySQL Workbench. В большинстве дистрибутивов эти программы предоставляются в виде отдельных пакетов.

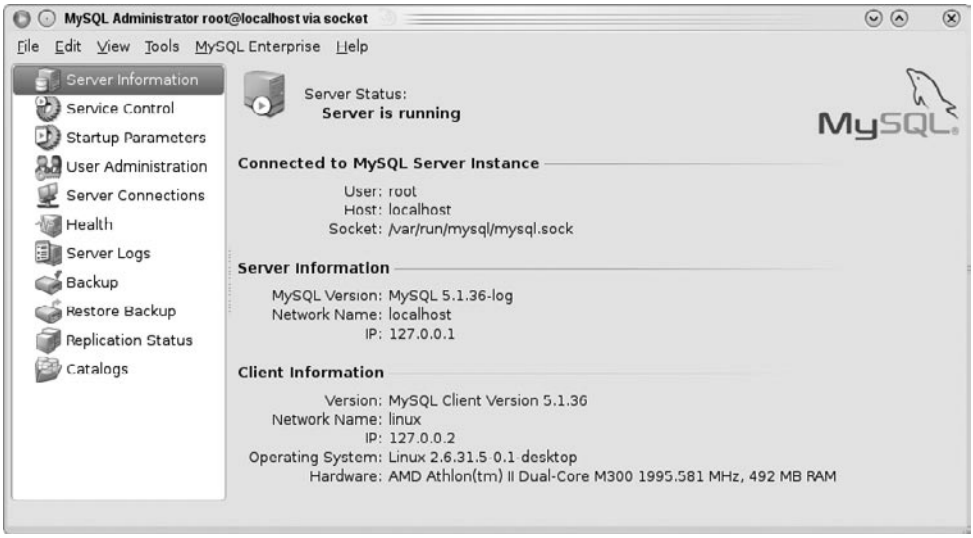

Рис. 19.3. Окно программы MySQL Administrator

PhpMyAdmin

Если PHP и MySQL уже работают, то самое время установить `PhpMyAdmin`. Это пакет файлов, написанных на PHP, которые вместе образуют очень удобный пользовательский интерфейс для администрирования MySQL. Основное преимущество `PhpMyAdmin` по сравнению с другими инструментами администрирования заключается в том, что он обеспечивает удаленную поддержку MySQL. Вам даже не нужен доступ к администрируемому сайту через Интернет.

Установка

В большинстве дистрибутивов, в частности в Debian, Fedora и Ubuntu, можно установить `PhpMyAdmin` как обычный пакет, а затем обращаться к нему по адресу `localhost/phpmyadmin` или в Fedora — `localhost4/phpmyadmin`.

В SUSE, если вы работаете с актуальной версией, вам потребуется самостоятельно установить `PhpMyAdmin`. Для этого скачайте с сайта phpmyadmin.net архив TAR и распакуйте его в каталог, к которому имеет доступ Apache. Команда `tar` создает новый архив `phpMyAdmin-n`, который лучше всего сразу превратить в каталог phpmyadmin — такое название легче запоминается (следующие команды действуют только в SUSE, в других дистрибутивах вам потребуется указывать для `cd` соответствующий стартовый каталог DocumentRoot).

```
root# cd /srv/www/htdocs/
root# tar xzf phpMyAdmin-3.2.0-rc1-all-languages.tar.gz
root# mv phpMyAdmin-3.2.0-rc1-all-languages phpmyadmin
```

Убедитесь, что Apache имеет право считывать файлы! При необходимости дополнительно выполните `chmod -R` или `chown -R`.

Применение

Конфигурация PhpMyAdmin такова, что перед началом работы вы должны представиться под логином для MySQL и соответствующим паролем. В зависимости от того, какие права доступа имеет пользователь MySQL, указанный в config.inc.php, вы, возможно, сможете в полном объеме администрировать MySQL с помощью PhpMyAdmin: программа позволяет создавать новые базы данных и таблицы, а также модифицировать и удалять имеющиеся, вводить наборы данных, модифицировать их, удалять, управлять правами доступа к MySQL, импортировать данные CSV, экспортировать таблицы (например, в целях резервного копирования) и т. д. (рис. 19.4).

Рис. 19.4. PhpMyAdmin

19.5. FTP-сервер

Вместе с большинством веб-серверов используется FTP-сервер, который в зависимости от сайта может выполнять две задачи: во-первых, он обеспечивает скачивание крупных файлов, которые предоставляются на сайте; во-вторых, используется при технической поддержке и обновлении сайта, в частности с его помощью очень удобно закачивать файлы на сайт.

Безопасность

FTP — очень старая программа. Часто ее протокол не может работать с брандмауэрами и маскарадингом. Ситуация осложняется еще и тем, что при установлении

соединения между FTP-клиентом и сервером имя пользователя и пароль передаются в незашифрованном виде. От этого у любого специалиста, неравнодушного к вопросам безопасности, волосы встают дыбом!

Разумеется, для FTP уже давно существуют безопасные аналоги. Одним из наиболее известных является описанный далее SSH-сервер с SFTP (Secure FTP), который в числе прочего обеспечивает и передачу данных. Проблема заключается, в основном, в клиентской стороне: существует сравнительно немного понятных для среднего пользователя программ, которые могут работать со SFTP, поэтому FTP, несмотря на недостаточную защищенность, все еще очень широко применяется.

Еще одна альтернатива — стандарт WebDAV, дополняющий протокол HTTP и упрощающий передачу данных в обоих направлениях. Например, Apache поддерживает WebDAV в комбинации с модулем `mod_dav`. Информацию об этом вы найдете на следующих сайтах:

- http://httpd.apache.org/docs/2.2/mod/mod_dav.html;
- http://wiki.ubuntuusers.de/Apache/webdav.

Если вы не желаете отказываться от традиционного FTP-сервера, можно сконфигурировать его как «только анонимный» FTP-сервер. В таком случае при входе в систему не будет передаваться никаких важных данных. Однако FTP будет использоваться далеко не в полном объеме. Тогда вы уже не сможете применять его для обычной технической поддержки сайта.

vsftpd. Существует множество FTP-серверов. Наиболее популярна в настоящее время программа `vsftpd`. Во всех распространенных дистрибутивах она предоставляется в виде пакета. Сочетание vsftpd означает «very secure FTP daemon»[1]. Однако не воспринимайте слова «very secure» буквально, ведь даже самому безопасному FTP-серверу присущи традиционные недостатки FTP.

Запуск демона. Программу `vsftpd` можно запускать двумя способами: либо как самостоятельный демон, через систему Init-V, либо с помощью демона `xinetd`. В большинстве дистрибутивов по умолчанию задан первый вариант. Для этого в конфигурационном файле `vsftpd.conf` должна содержаться команда `listen=YES`. Чтобы запустить или остановить FTP-сервер, используйте команды, обычные для того или иного дистрибутива (см. раздел 4.5).

Конфигурация

Конфигурация `vsftpd` выполняется в файле `/etc/vsftpd.conf` или `/etc/vsftpd/vsftpd.conf`. Обычно по умолчанию при анонимном FTP-доступе применяется режим «только для чтения». Иначе говоря, клиенты FTP могут скачивать файлы, а закачивать не могут. Если вам нужно, чтобы на FTP можно было входить не только анонимно, но и под определенным логином, то установите для `locale_enable` значение `YES`. Если при использовании такой формы FTP вы также хотите разрешить закачку файлов, установите для `write_enable` значение `YES`. Если в файле `vsftpd.conf` содержится строка `tcp_wrappers=Yes`, то программа `vsftpd` интерпрети-

[1] «Очень безопасный демон FTP» (англ.).

рует файлы /etc/hosts.allow и /etc/hosts.deny (см. раздел 18.2). В следующем примере обобщаются важнейшие настройки vsftpd.conf:

```
# /etc/vsftpd.conf или /etc/vsftpd/vsftpd.conf
...
local_enable=YES / NO        # Допуск FTP-логина
write_enable=YES / NO        # Разрешение на закачку файлов
...
anonymous_enable=YES / NO    # Разрешение анонимного доступа по FTP
anon_upload_enable=YES / NO  # Разрешение закачки файлов,
                             # в том числе при анонимном доступе по FTP
...
listen=YES / NO              # Запуск в качестве демона Init-V-Damon (YES)
или xinetd (NO)
tcp_wrapper=YES / NO         # Интерпретация hosts.allow и hosts.deny
```

Испытание FTP. Теперь FTP должен заработать. Выполните на сервере команду ftp localhost, чтобы проверить, правильно ли был запущен сервер. При этом учитывайте, что root, как правило, не должен указывать FTP-логин.

Анонимный доступ по FTP

Если в vsftpd.conf допускается анонимный доступ по FTP, vsftpd принимает в качестве логина имена anonymous и ftp в комбинации с любым паролем. Обычно в качестве пароля указывается адрес электронной почты, но vsftpd этого не контролирует.

После входа в систему FTP-клиент получает доступ к файлам домашнего каталога пользователя Linux ftp. Местоположение этого каталога указывается в /etc/passwd:

- Debian, Ubuntu — /home/ftp;
- Fedora, Red Hat — /var/ftp/;
- SUSE — /srv/ftp/.

Закачка файлов через анонимный FTP. Если вы разрешаете загружать в систему файлы через анонимный FTP, обратите внимание на то, что в каталоге с данными FTP имеется лишь один каталог, предоставляющий право на внесение изменений, например /var/ftp/upload в Fedora или Red Hat. Этот каталог должен принадлежать пользователю FTP и из соображений безопасности не давать прав на чтение файлов:

```
root# mkdir /var/ftp/upload
root# chown ftp upload
root# chmod 730 upload
```

Таким образом, любому пользователю разрешается закачивать файлы в систему, а также отправлять системному администратору электронное сообщение с разъяснениями, для чего нужен закачанный файл. Остальные пользователи не могут ни просмотреть, ни скачать этот файл из каталога upload. Если не принять таких мер безопасности, то ваш каталог для закачек FTP можно будет использовать для нелегального обмена файлами.

FTP для администратора и других особых категорий пользователей

По соображениям, связанным с безопасностью, администратор (root) и некоторые другие особые пользователи, в частности daemon, lp или nobody, не могут применять FTP. Конфигурация, при которой реализуется такой запрет, отличается от дистрибутива к дистрибутиву.

В Fedora и Red Hat защита логина происходит двумя способами одновременно. С одной стороны, для контроля логина `vsftpd` обращается к PAM (подключаемым модулям аутентификации). PAM интерпретируют файл `/etc/pam.d/vsftpd`, указывающий на файл `etc/vsftpd/ftpusers`. В последнем содержится список всех логинов, которые *не могут* использовать FTP.

С другой стороны, в FTP применяется и внутрисистемный контроль логинов — все пользователи, названные в `/etc/vsftpd.user_list`, блокируются. Такой контроль логинов активизируется в `vsftpd.conf` с помощью `userlist_enable=YES` и `userlist_deny=YES` (действует по умолчанию).

В Debian, SUSE и Ubuntu `vsftpd` обращается к логину также через PAM. Здесь `/etc/pam.d/vsftpd` всегда указывает на `/etc/ftpusers`. В этом файле содержится список всех логинов, с которых нельзя использовать FTP.

20 Сервер локальной сети

В этой главе будет описана конфигурация сетевых служб, которые обычно предоставляются в локальной сети. Вы узнаете, как обмениваться по сети файлами, находясь дома, в офисе или в организации, совместно использовать принтер, синхронизировать время на всех компьютерах и др. Детально будут рассмотрены следующие службы.

- **Файловый сервер (NFS, Samba).** Сетевая файловая система обеспечивает простой обмен файлами между компьютерами с UNIX/Linux. Если вам хотелось бы или требуется обеспечить совместимость сети с Windows, вы можете — в качестве альтернативы или вместе с NFS — использовать Samba. С ее помощью можно работать со всеми каталогами и принтерами, находящимися в сети, по протоколу SMB.
- **Сервер времени (NTP).** При одновременном доступе к файлам в сети важно, чтобы на всех компьютерах были синхронные показатели времени. Для этого следует настроить в сети собственный NTP-сервер, обеспечивающий в локальной сети точное соблюдение и синхронизацию времени.
- **Система печати (CUPS).** Эта служба обеспечивает связь компьютера с принтером вне зависимости от того, как именно должен использоваться принтер — только для распечатки документов с локального компьютера или во всей сети. CUPS применяется во всех распространенных дистрибутивах.
- **Синхронизация времени (NTP).** При одновременном доступе к файлам важно, чтобы время протекало на всех компьютерах сети по возможности синхронно. Для этого применяется сетевой протокол синхронизации времени.

20.1. NFS 3

Сетевая файловая система (NFS) позволяет предоставлять компьютерам, находящимся в сети, доступ к локальным каталогам других компьютеров, также расположенных в сети. В Linux/UNIX NFS обычно применяется для управления файлами и каталогами, к которым открыт общий доступ (в Windows они называются *совместно используемыми* — shared).

Базовые функции NFS входят в состав ядра, таким образом достигается максимальная скорость работы. В качестве альтернативы вы можете воспользоваться NFS-сервером для пользовательского пространства, но он уже практически не применяется, и в этой главе мы не будем его рассматривать.

NFS ядра поддерживает версии NFS 3 и 4. NFS 4 пока еще не очень распространена, а ее конфигурация коренным образом отличается от NFS 3, поэтому в данном разделе мы остановимся только на NFS 3. В следующем разделе будет сделано краткое введение в NFS 4.

NFS реализуется на базе *удаленных вызовов процедур (RPC)*. Функционирование NFS обеспечивается с помощью целого набора программ. Далее перечислены важнейшие компоненты этого набора:

- `nfsd` (для NFS 3) и `nfsd4` (для NFS 4) — это демоны NFS, которые реализуются с помощью потоков ядра;
- `portmap` — отвечает за установление соединения между NFS-клиентом и сервером, а также за динамическое присваивание клиентам номеров портов UDP (только в NFS 3);
- `rpc-mountd` — обрабатывает `mount`-запросы клиентов (только в NFS 3).

Как правило, вам не придется беспокоиться о работе этих программ — когда NFS установлена и активизирована, все необходимые программы автоматически запускаются в ходе выполнения процесса Init-V.

Ссылки. Отличное руководство по NFS с массой примеров, касающихся отладки и обеспечения безопасности, находится по следующему адресу: http://nfs.sourceforge.net/nfs-howto/.

Установка и конфигурация

Прежде чем вы сможете настроить NFS-сервер, убедитесь, что в системе установлены необходимые пакеты. По умолчанию их обычно не хватает. Названия пакетов различаются в зависимости от дистрибутива:

- Debian, SUSE, Ubuntu — `nfs-kernel-server`;
- Fedora, Red Hat — `nfs-utils`, `rpcbind`.

Конфигурация осуществляется в трех файлах — `/etc/exports`, `/etc/hosts.allow` и `/etc/hosts.deny`.

Файл /etc/exports

Файл `/etc/exports` — это основной конфигурационный файл NFS. Он определяет, какой компьютер к каким файлам будет иметь доступ и каким именно будет этот доступ. Компьютеры можно указывать либо по IP-адресам, либо по сетевым именам. IP-адреса можно маскировать (например, с помощью 192.168.0.0/255.255.255.0 или 192.168.0.0/24). В названиях компьютеров могут содержаться джокерные символы * (например, *.sol), а в IP-адресах — нет.

В следующем примере показано, как предоставить доступ к каталогу `/usr/local` всем клиентам, имеющим в сети IP-адреса из пространства 192.168.0.*, но без права изменения файлов этого каталога. Компьютер uranus.sol, кроме того, имеет пра-

во читать каталог /usr/share и вносить в него изменения. Запись /usr/local в exports разделена на две части, так как не помещается на одной строке:

```
# /etc/exports на компьютере mars.sol
# ro = read only, rw = read write
/usr/local 192.168.0.0/24(ro,async,no_subtree_check) \
           *.sol(ro,async,no_subtree_check)
/usr/share uranus.sol(rw,async,no_subtree_check)
```

Синтаксис /etc/exports понятен из предыдущих строк. За названием каталога и хост-именем или IP-адресом в скобках следуют различные параметры NFS, наиболее важные из которых кратко перечислены ниже (несколько других параметров описаны в man exports).

- ro (read-only) или rw (read-write) — указывают, что доступ предоставляется только для чтения или для чтения и внесения изменений.
- sync или async — определяют момент времени, в который NFS-сервер подтверждает изменения, внесенные в файл. По умолчанию действует sync. При такой настройке информация сохраняется лишь тогда, когда файл действительно сохранен. Параметр async гораздо эффективнее, но вместе с тем и ненадежнее. При доступе для внесения изменений скорость работы sync и async отличается радикально (до 10 раз), поэтому на практике async применяется довольно часто.
- Благодаря insecure NFS-сервер реагирует и на те клиентские запросы, которые приходят с IP более 1024. Этот параметр требуется в тех случаях, когда в сети имеются компьютеры, работающие с Apple: по умолчанию Mac OS X использует для обмена информацией с NFS-сервером порты с номером выше 1024.
- no_subtree_check или subtree_check — указывают, должен ли NFS-сервер тестировать поддерево. Коротко расскажу об этом процессе: если по NFS экспортируется каталог файловой системы (но не вся файловая система), то NFS-сервер тестирует поддерево, определяя, таким образом, находится ли в экспортируемом каталоге конкретный файл. Затем NFS-сервер передает клиенту информацию о фактическом местоположении файла. Если позже файл на сервере будет переименован, то на клиентском компьютере возникнут проблемы. По этой причине по умолчанию в современных версиях NFS-сервера тестирование поддерева отключено. И все же параметр no_subtree_check нужно указать, чтобы не получать при запуске системы соответствующих предупреждений от сервера.

  Если хотите, можете активизировать тестирование поддерева. В man exports это рекомендуется делать в первую очередь с каталогами, файлы в которых переименовываются редко и экспортируются в режиме «только для чтения».
- Пользователь root, конечно, может использовать NFS, как и любой другой пользователь, но в экспортированных каталогах из соображений, связанных с безопасностью, он имеет только права пользователя nobody (UID=65534 и GID=65534). Если вы хотите дать root его обычные права, укажите в /etc/exports параметр no_root_squash.

Если NFS-сервер уже работает, то изменения, вносимые в /etc/exports, вступают в силу только после выполнения команды exportfs:

```
root# exportfs -a
```

Файлы /etc/hosts.allow, /etc/hosts.deny

В файлах hosts.allow и hosts.deny указывается, какие компьютеры имеют право обращаться к NFS-серверу. Информация, содержащаяся в /etc/exports, важна только для тех пользователей, которые в принципе могут связаться с NFS-сервером. В этом отношении файлы hosts.allow и hosts.deny занимают первое место в иерархии средств защиты доступа. Синтаксис этих файлов описан в разделе 18.2. Для NFS-сервера релевантны записи portmap и mountd.

Настройки, характерные для отдельных дистрибутивов

Кроме стандартных конфигурационных файлов вы можете указать и такие настройки, которые являются специфичными для отдельных дистрибутивов:

- Debian, Ubuntu — /etc/defaults/nfs-common, /etc/defaults/nfs-kernel-server;
- Fedora, SUSE, Red Hat — /etc/sysconfig/nfs.

Запуск

В Debian и Ubuntu сервер NFS по умолчанию запускается сразу же после установки. В Fedora, Red Hat и SUSE вам придется помочь компьютеру, как и при запуске других служб Init-V, выполнив следующие команды (см. раздел 4.5):

```
root# chkconfig --level 35 nfs on    (Fedora, Red Hat)
root# /etc/init.d/nfs start
root# insserv nfsserver              (SUSE)
root# /etc/init.d/nfsserver start
```

Клиенты NFS

После запуска сервера можете перейти на другой компьютер и с помощью команды mount проверить, все ли работает. В Debian и Ubuntu сначала потребуется установить пакет nfs-common.

```
user@uranus$ mkdir /test
user@uranus$ mount -t nfs mars:/usr/share /test
user@uranus$ ...
user@uranus$ umount /test
```

Если возникнут проблемы, убедитесь, что NFS не блокируется брандмауэром. NFS 3 использует протоколы TCP и UDP на портах 111 (portmap) и 2049 (nsfd), а также на тех портах, которые в данный момент свободны (rpc.*d). Более подробно применение NFS со стороны клиента рассматривается в разделе 13.12.

Определение статуса NFS

Чтобы узнать, работают ли демоны, необходимые для эксплуатации компьютера в качестве NFS-сервера, выполните команду rpcinfo. Результат должен выглядеть примерно так, как в следующем примере:

```
root# rpcinfo -p
Program   Vers   Proto    Port
```

```
 100000      2     tcp     111   portmapper
 100024      1     udp   33781   status
 100003      3     udp    2049   nfs
 100003      4     udp    2049   nfs
 100021      1     udp   58658   nlockmgr
 100005      1     udp   41096   mountd
...
```

Чтобы узнать, какие клиенты в настоящее время обращаются к NFS-серверу, выполните команду `showmounts`. В следующем примере показано, что к серверу mars в данный момент обращается только один клиент с IP-адресом 192.168.0.15:

```
root# showmount -a
All mount points on mars:
192.168.0.0/24:/usr/share
192.168.0.15:192.168.0.0/24
```

Результаты `showmount -a` основаны на файле `/var/lib/nfs/rmtab`, которым занимается `rpc.mountd`. К сожалению, этот файл часто бывает неполон либо содержит старые, уже не актуальные записи (также см. `man rpc.mountd`). По этим причинам результаты выполнения `showmount` не универсальны. Кроме того, учитывайте, что `showmount` в принципе учитывает только каталоги NFS 3!

UID и GID

Для управления правами доступа к файлам и каталогам в NFS 3 применяются UID и GID. Такая система проста, но работает лишь при условии, что на сервере и на всех клиентах соблюдается однозначное соответствие между пользователями, группами и их ID-номерами.

При управлении пользователями вручную достичь однозначного соответствия между UID и GID обычно бывает очень сложно и для этого требуется особая тщательность: создавая на любом компьютере новую пользовательскую учетную запись, нужно вручную задавать UID и GID. Кроме того, вам понадобится центральный справочный список всех уже розданных номеров UID и GID. Если при управлении пользователями вы допустите ошибки или любую небрежность, то сразу же столкнетесь с неприятными последствиями: например, если пользователи peter@merkur и birgit@neptun, работающие на разных компьютерах, имеют одинаковый UID 1234, то оба будут иметь одинаковые права при доступе к каталогу NFS. А такого лучше не допускать.

Раньше подобная проблема решалась путем синхронизации файлов `/etc/passwd`, `/etc/group` и `/etc/shadow` на всех компьютерах локальной сети с помощью NIS (сетевой информационной службы). Настроить NIS достаточно просто, но она считается устаревшей и ненадежной. Гораздо лучше применять централизованное управление пользовательскими данными (логин, пароль, групповая отнесенность, UID и GID и т. д.) с помощью службы LDAP. LDAP означает *«облегченный протокол доступа к каталогу»* — это протокол для управления иерархическими данными. В качестве LDAP-сервера в Linux обычно применяется программа openLDAP. К сожалению, конфигурировать и эксплуатировать сервер LDAP достаточно сложно. Этот процесс подробно описан мной в книге *Ubuntu Server*, вышедшей в издательстве Addison-Wesley.

Проблемы с кодировкой

Как уже было указано в разделе 13.12, протокол NFS 3 не отвечает за соблюдение кодировки в названиях файлов и интерпретирует эти названия просто как последовательности байт. Если на NFS-сервере и на клиентских компьютерах используются различные кодировки, то все символы, не относящиеся к ASCII, могут отображаться неправильно.

20.2. NFS 4

В версии NFS 3 существуют принципиальные проблемы: с точки зрения безопасности протокол оставляет желать лучшего, управление пользователями на базе UID/GID непригодно для работы с большими сетями, защитить NFS брандмауэром сложно, при наличии в названиях файлов символов Unicode клиенты, не поддерживающие Unicode, не могут их правильно отображать и т. д.

Для устранения этих недостатков протокол для NFS 4 был разработан полностью заново: поддерживается управление правами через списки контроля доступа (ACL), а также безопасная аутентификация доступа с применением Kerberos и SPKM-3. В NFS автоматически решаются задачи, связанные с блокировкой и подключением устройств, так что для этого не требуется отдельных RPC-демонов. NFS 4 правильно отображает символы Unicode, содержащиеся в именах файлов. Весь обмен информацией осуществляется через порт TCP 2049, благодаря чему упрощается конфигурация брандмауэра (без UDP и динамических портов).

К сожалению, у NFS 4 также есть свои недостатки: конфигурация протокола сложна, а свободно распространяемого клиента для Windows пока нет. В течение долгого времени NFS также считался недоработанным и нестабильным, но эти начальные сложности уже преодолены.

В этом разделе рассматривается серверная и клиентская конфигурация системы аутентификации, то есть описывается, как быстрее всего начать работу с NFS 4. На практике зачастую требуется применять NFS 4 вместе с LDAP и Kerberos — как правило, это касается сетей с большим количеством пользователей. Такой усложненный вариант конфигурации я не смогу здесь описать, так как этого не позволяют размеры книги.

Конфигурация сервера

В современных версиях Linux по умолчанию поддерживается NFS 4. Но вам обязательно следует убедиться, что на компьютере запущена служба `rpc.idmapd`. Она соотносит имя пользователя NFS и UID/GID. Запускается ли эта служба и как именно запускается, зависит от дистрибутива.

- В Debian и Ubuntu за запуск `rpc.idmapd` отвечает сценарий Init-V `nfs-common`. Он автоматически запускает данную службу при условии, что в системе существует файл `/etc/exports` (на сервере) или в `/etc/fstab` содержится последовательность символов `nfs4` (клиент). При необходимости службу можно запустить и принудительно, указав для этого в `/etc/default/nfs-common` настройку `NEED_IDMAPD=yes`.

- В Red Hat и Fedora `rpc.imapd` запускается по умолчанию с помощью `/etc/init.d/nfs`. Никакой специальной конфигурации не требуется.
- В SUSE необходимо убедиться, что `/etc/sysconfig/nfs` содержит настройку `NFS4_SUPPORT=yes` (обычно по умолчанию это так). Тогда `rpc.idmapd` запускается сценарием Init-V `/etc/init.d/nfskernel`.

Конфигурация `rpc.idmapd` выполняется в файле `/etc/idmap.conf`. Как правило, его можно оставить в том виде, в котором он был при поставке.

Каталоги

Конфигурация каталогов, которые должны экспортироваться по NFS, построена совершенно иначе, нежели в NFS 3: все каталоги должны быть подчинены одному корневому каталогу, который играет роль файловой псевдосистемы. Эту концепцию проще понять на примере. Предположим, нам нужно экспортировать каталоги `/data/audio` и `/data/pictures/photos`. Для этого создаем три новых каталога, а в качестве корневого используем `/nfsexport` (разумеется, название этого каталога может быть произвольным).

```
root# mkdir /nfsexport
root# mkdir /nfsexport/audio
root# mkdir /nfsexport/photos
```

Затем мы подключаем к системе каталоги `/data/audio` и `/data/pictures/photos` как новые подкаталоги `nfsexports`, поэтому теперь содержимое каталога `/data/audio` можно просмотреть в `/nfsexport/audio` и аналогичным образом `/data/pictures/photos` — в `/nfsexport/photos`.

```
root# mount -t none -o bind /nfsexport/audio/  /data/audio/
root# mount -t none -o bind /nfsexport/photos/ /data/pictures/photos/
```

Чтобы в дальнейшем этот процесс проходил автоматически, добавьте в `/etc/fstab` (на сервере) две следующие строки:

```
# /etc/fstab
...
/data/audio         /nfsexport/audio  none bind 0 0
/data/bilder/photos /nfsexport/photos none bind 0 0
```

Файл /etc/exportfs

Закончив подготовительные работы, вы, наконец, можете изменить `/etc/exports`. Этот файл отвечает за работу NFS 4, причем синтаксис практически аналогичен NFS 3 (см. подраздел «Установка и конфигурация» раздела 20.1). Добавляется лишь два параметра.

- Корневой каталог NFS 4 обозначается параметром `fsid=0`. В системе может быть только один корневой каталог (еще раз отмечу: в NFS 4 невозможно экспортировать каталоги, которые находятся за пределами корневого каталога).
- Параметр `crossmnt` также указывается лишь для корневого каталога. Благодаря этому параметру при подключении подкаталогов их содержимое остается видимым для клиентов и тогда, когда корневой каталог на клиентском компьютере остается не подключен. Вместо параматера `crossmnt` корневого каталога

можно указывать для любых каталогов параметр nohide — он помогает достичь такого же результата.

```
# /etc/exports
...
/nfsexport        192.168.0.0/24(rw,async,no_subtree_check,fsid=0,crossmnt)
/nfsexport/audio  192.168.0.0/24(ro,async,no_subtree_check)
/nfsexport/photos 192.168.0.0/24(rw,async,no_subtree_check)
```

Как обычно, exportfs -a гарантирует, что уже действующий NFS-сервер будет учитывать новые записи.

Клиентская конфигурация

При работе с клиентом вы также должны обеспечить запуск и работу rpc.idmapd. Метод при этом применяется тот же, что и в ходе конфигурации сервера. В дальнейшем вы сможете использовать каталоги NFS с помощью команды mount. Следующие команды подключают в локальную файловую систему все дерево каталогов nfsexport в точке /media/nfsdata. Обратите внимание, что в качестве файловой системы необходимо задать NFS 4 (-t nfs4), чтобы можно было адресовать корневой каталог NFS, указав просто /, а не /nfsexport!

```
root# mkdir /media/nfsdata
root# mount -t nfs4 mars:/ /media/nfsdata
root# ls /media/nfsdata
audio photos
```

В качестве альтернативы можно импортировать только подкаталог:

```
root# mkdir /media/photos
root# mount -t nfs4 mars:/photos /media/photos
```

20.3. Основы Samba

Samba — это пакет программ, помогающий интегрировать в одну сеть компьютеры Windows и Linux/UNIX. Название Samba — это переосмысленная аббревиатура SMB, которая, в свою очередь, обозначает протокол *«блок сообщений сервера»*. В сфере Windows этот протокол используется для того, чтобы компьютеры, расположенные в одной сети, могли обращаться к данным друг друга, а также совместно использовать принтер.

С помощью Samba вы также можете предоставить для использования в сети файлы или каталоги с компьютеров с Linux. Пользователи, работающие с компьютерами Windows, Linux или Apple, смогут обращаться к этим файлам. Тонко дифференцированная система аутентификации и распределения прав определяет, кто имеет право на чтение и изменение файлов. Таким образом, Samba — это центральная узловая станция, предназначенная для обмена данными в локальной сети: на фирме, в организации или дома.

Часто Samba воспринимается как соединительное звено между мирами Windows и Linux. Но это слишком узкий подход: SMB часто применяется в средах, состоящих только из компьютеров с Linux или UNIX, так как упрощает работу: программы для просмотра файлов в Gnome и KDE по умолчанию поддерживают SMB, таким образом, с помощью CIFS вы можете подключать каталоги SMB непосредственно в дерево каталогов Linux и т. д. Если вы ищете UNIX-подобную альтернативу, которая работает не на основе Microsoft, то вам лучше всего использовать NFS. Но для применения NFS клиентские компьютеры должны иметь одинаковую пользовательскую конфигурацию, а это означает несовместимость с Windows.

В этом разделе будут рассмотрены основные функции Samba 3.n с точки зрения сервера. Если вы хотите использовать многочисленные «продвинутые» функции, например аутентификацию через LDAP или использование Samba в качестве первичного контроллера домена, обратитесь к специальной литературе. Много полезной информации имеется на сайте Samba: http://www.samba.org/.

Если хотите более подробно изучить технические вопросы (протоколы, механизмы блокировки и т. д.), рекомендую начать с книги *Using Samba*, написанной в соавторстве с Джеральдом Картером (Gerald Carter), Джеем Тс (Jay Ts) и Робертом Экстайном (Robert Eckstein) и вышедшей в издательстве O'Reily (третье издание, 2007 год). Английская версия книги в формате HTML (второе издание) имеется в Интернете по адресу http://www.samba.org/samba/docs/using_samba/toc.html.

Вам также помогут следующие сайты:

- https://help.ubuntu.com/community/SettingUpSamba;
- http://samba.sernet.de/.

Основы

Прежде чем перейти к конфигурации сервера Samba, разберемся в концепциях, лежащих в основе этого сервера, и узнаем, что означают основные связанные с ним термины.

NetBIOS

SMB работает на базе протокола NetBIOS. NetBIOS — это *сетевая базовая система ввода-вывода*. Ее разработку начала компания IBM. Со временем протокол NetBIOS был существенно обновлен. Первоначально NetBIOS состояла из трех служб.

- **Служба имен.** Этот метод обмена данными можно сравнить с DNS, используемым в UNIX/Linux. Управление именами может осуществляться с помощью сервера имен NetBIOS (NBNS) или децентрализованно. При использовании этого метода каждый запускающийся клиент посылает сообщение остальным клиентам сети и сообщает, под каким именем он начал работать.
- **Служба сеансов.** Этот механизм обмена информацией, подобно TCP, обеспечивает аккуратный обмен данными между двумя компьютерами — в форме пакетов. При этом проверяется целостность пакетов, и пакеты, которые содержат ошибки или потерялись, запрашиваются заново.

- **Служба датаграмм.** При использовании этого варианта службы сеансов система не проверяет, в порядке ли приходящие данные. Но у службы датаграмм есть и определенное достоинство — приходящие таким образом данные могут одновременно рассылаться на несколько компьютеров.

WINS

В Windows задачи NBNS решаются с помощью *Службы имен Интернета для Windows (WINS)*. Сервер WINS можно настроить с помощью Samba или актуальной версии Windows (но не с Windows 9x). При этом, если в системе уже имеется сервер доменных имен, Samba может работать с ним.

Разрешение имен в Windows, в отличие от UNIX, может функционировать без применения специального сервера имен. Для этого необходимо рассылать всем компьютерам сети пакеты с датаграммами, поэтому такой метод становится все менее эффективным по мере того, как растут сети с Windows.

Имена компьютеров, используемые в NetBIOS и TCP/IP, не зависят друг от друга. Иными словами, возможен случай, когда компьютер, передавая данные по разным протоколам, будет при этом иметь разные имена. На практике таких случаев, разумеется, следует избегать. Различные имена не только вызывают путаницу, но и не позволяют использовать некоторые функции.

Просмотр сети и ресурсов

Откуда клиенту с Windows известно, какие еще компьютеры есть в сети? Для этого используется функция *просмотра сети и ресурсов*. Под этим понимается управление компьютерами, находящимися в сети. Для того чтобы таким управлением не нужно было заниматься каждому компьютеру, управление передается так называемой *главной системе просмотра ресурсов*. В больших сетях также работает одна или несколько *резервных систем просмотра ресурсов*. Кроме того, при необходимости Samba может выступать в качестве *системы просмотра сети и ресурсов*.

В системе не определяется жестко, какой именно компонент будет заниматься просмотром сети и ресурсов. Один из компьютеров в сети назначается для этой цели динамически — в зависимости от того, какие из компьютеров в настоящее время находятся в сети и какой из них наилучшим образом приспособлен для данных целей (как правило, это компьютер с новейшей версией операционной системы). Это совсем не означает, что система автоматически выберет WINS-сервер, если он вообще есть в сети. Таким образом, функция просмотра сети и ресурсов реализуется очень децентрализованно.

К сожалению, ПК с Windows далеко не всегда удается синхронизировать список компьютеров (browsing list) и реальное состояние сети. Причина этого заключается в том, что клиенты управляют кэшем, в котором в целях снижения нагрузки на сеть сохраняется последнее рабочее состояние локальной системы. Кэш не всегда удается обновить сразу — бывает, что на это требуется определенное время. Иначе говоря, если вы не видите с компьютера с Windows или другого компьютера, на котором работает Windows или Linux, но знаете, что искомый компьютер точно работает, то проблема заключается в несовершенстве функции просмотра сети (скорейшее решение такой проблемы, как это часто бывает при работе с Windows, — перезагрузка компьютера, который не удается найти).

Права доступа и системы обеспечения безопасности

Название *«совместно используемые ресурсы»* может обозначать и каталоги, и принтеры, которые предоставляются в пользование другим компьютерам через NetBIOS. Английский вариант термина — *shares*, кроме того, мы будем говорить о каталогах и объектах. Существуют различные способы управления доступом, определяющие, какими ресурсами может пользоваться тот или иной компьютер.

- **Защита на уровне доступа к совместно используемым ресурсам.** При использовании простейшей формы управления правами доступа каждый каталог и принтер получает собственный пароль (разумеется, объекты могут предоставляться и без пароля). Такой метод все еще иногда применяется в некоторых небольших сетях. Самый очевидный недостаток — требуется очень много паролей: если каждому из 10 компьютеров должно быть предоставлено в пользование по три объекта, мы уже получаем 30 паролей. В больших сетях при применении такого метода достаточно скоро воцаряется хаос.
- **Защита рабочей группы.** Данный метод, являющийся более совершенной формой предыдущего, позволяет взаимный доступ к объектам лишь при условии, что два компьютера относятся к одной и той же рабочей группе. Это повышает безопасность работы, но не кардинально: при отсутствии централизованного администрирования любой компьютер может «представиться» членом любой рабочей группы. Иначе говоря, защита на уровне доступа к совместно используемым ресурсам функционирует по децентрализованному одноранговому принципу (в противоположность методу клиент/сервер, применяемому на уровне пользователей и уровне доменов, который все сильнее тяготеет к централизации).
- **Защита на уровне пользователя.** Для реализации такой защиты пользователь должен применять для входа на клиентский компьютер логин и пароль. В результате, если пользователь захочет взять для работы определенные данные с сервера Samba или другого ПК с Windows, основанием для доступа являются действующий логин и пароль этого пользователя. Кроме того, оба компьютера должны относиться к одной и той же рабочей группе.

  Таким образом, мы уже не присваиваем пароль каждому сетевому объекту. Теперь логин и пароль связаны с пользователем (со списком пользователей, перечисленных по именам, либо со всеми пользователями, входящими в определенную группу). Если в Samba задействуется защита на уровне пользователя, то вам потребуется отдельная база данных, в которой будут храниться имена пользователей, групповая отнесенность и пароли.

  Приведу пример: пользователь *X* работает на компьютере *A*. Чтобы *X* мог запрашивать данные с сервера Samba *S*, то *X*, *A* и *S* должны быть зарегистрированы в системе как пользователи (с одним и тем же именем и паролем). Теперь допустим, что компьютер *A* сломался. Пользователь *X* переходит на компьютер *B*. Чтобы пользователь *X* мог получить доступ к своим данным на *S* с компьютера *B*, на компьютере *B* потребуется создать новую учетную запись — опять же с паролем.

Если пользователь *X* решит изменить свой пароль, такое же изменение потребуется выполнить на сервере *S*, а следовательно, и на всех клиентских компьютерах (*A*, *B*, *C* и т. д.). Другими словами, при таком построении сети децентрализованное управление паролями и децентрализованная аутентификация остаются непреходящими проблемами.

- **Защита на уровне доменов.** В версии Windows NT 4 в мире сетей Microsoft появились так называемые домены. Концепция управления доступом очень напоминает защиту на уровне пользователя. Различия касаются способов управления базой пользовательских данных и метода осуществления аутентификации.

  При входе в систему пользователь обращается к пользовательской базе данных, которая централизованно управляется сервером. Права доступа управляются с помощью так называемого регистрационного маркера. Входя в систему, клиент получает регистрационную информацию, которая остается действительной во всей сети до тех пор, пока пользователь не выйдет из сети. Он не замечает разницы, но внутри системы такое отличие становится фундаментальным и позволяет эксплуатировать сервер со значительно большей эффективностью.

  Для обеспечения защиты на уровне доменов необходимо, чтобы в сети был *первичный контроллер домена (PDC)*. В крупных сетях кроме PDC может использоваться несколько *резервных контроллеров домена (BDC)*, чтобы вся система не останавливалась, если PDC выйдет из строя.

- **Active Directories.** Для того чтобы упростить управление крупными сетями, Microsoft дополнила механизм защиты на уровне домена функцией Active Directory. Для аутентификации (вход в систему, управление паролями и т. д.) теперь применяется облегченный протокол доступа к каталогу (LDAP). С его помощью сеть и ее домены можно построить иерархически. Кроме того, управление именами компьютеров в такой ситуации выполняется с помощью сервера доменных имен, как это обычно делается в Linux, и без применения WINS.

На клиентском уровне Samba поддерживает все пять перечисленных систем защиты. В настоящее время сервер Samba еще не может выполнять роль «Active Directory — Домен — Сервер». Разработчики планируют внедрить необходимые дополнения в версии Samba 4. Пока сложно сказать, когда она выйдет. В этой книге мы рассмотрим только защиту на уровне пользователя. Таким образом, сервер Samba мы понимаем как *изолированный* (standalone) сервер.

ПРИМЕЧАНИЕ

В конфигурационном файле Samba имеется несколько параметров, определяющих, какую систему защиты будет использовать Samba. Важнейший из этих параметров — security. Обращаю ваше внимание на то, что его настройки могут не совпадать с перечисленными выше вариантами один к одному! По умолчанию действует настройка security=user, то есть защита на уровне пользователя. Но эта настройка сохраняется и при конфигурации PDC. Остальные параметры важны для того, чтобы Samba производила аутентификацию и при входе пользователей на компьютеры с Windows.

Централизованная или децентрализованная топология сервера

Если не учитывать систему защиты, существует две принципиально различные стратегии обмена данными между компьютерами в сети с применением Samba.

- **Централизованная топология.** Центральный сервер Samba предоставляет всем пользователям доступ к каталогам. Задача администратора — настроить права доступа к отдельным каталогам так, чтобы в системе имелись и закрытые (частные) каталоги отдельных пользователей, и относительно открытые каталоги для обмена данными в пользовательских группах.

  Преимущества такого варианта конфигурации заключаются в том, что данные централизованно сохраняются на сервере и отдельные пользователи не должны сами заниматься созданием сетевых каталогов. У этой конфигурации есть и недостатки: система получается недостаточно гибкой, любые изменения вносятся администратором. Кроме того, выход сервера из строя чреват катастрофическими последствиями для всей сети.

- **Децентрализованная топология.** При использовании такого варианта каждый компьютер, который должен предоставить данные для совместной работы в сети, делает это сам. И в Windows, и в Linux (точнее говоря, в Gnome и KDE) система помогает в этом пользователю относительно простыми диалоговыми окнами.

  Преимущество такого варианта конфигурации — децентрализованный подход, при котором каждый отвечает сам за себя и не нуждается в помощи администратора. По мере роста сети конфигурация, разумеется, становится все более запутанной, централизованное резервное копирование оказывается практически неосуществимым.

Я придерживаюсь такого мнения, что на предприятии более целесообразно применять централизованную топологию сети. Если же вам нужно просто скопировать пару файлов с одного компьютера на другой в частной сети, то, конечно же, вполне хватит и временной децентрализованной конфигурации, настраиваемой через диалоговое окно общего доступа из Gnome или KDE. Основная проблема заключается в том, что такие диалоговые окна Gnome или KDE недоработаны и плохо функционируют.

20.4. Samba: базовая конфигурация и ввод в эксплуатацию

Установка

Во многих дистрибутивах для сервера и для клиента применяются отдельные пакеты. Обычно клиентские пакеты устанавливаются по умолчанию, поэтому доступ к сетевым каталогам должен функционировать с ходу. Но если вы хотите предоставить в общее пользование сами сетевые каталоги, то вам не обойтись без серверных функций, которые в большинстве дистрибутивов упакованы в пакете `samba`.

В некоторых дистрибутивах также предоставляются пакеты Samba 4. Но эти пакеты ориентированы исключительно на разработчиков, которые сами хотят

испробовать функции Samba. **Samba 4 не предназначена для решения практических рабочих задач!**

Запуск

Службы Samba предоставляются с помощью двух фоновых процессов:

- `nmbd` — используется для внутрисистемного управления и как сервер имен, а также отвечает за просмотр сети и ресурсов; его можно использовать и в качестве главной системы просмотра ресурсов или как WINS-сервер;
- `smbd` — это клиентский интерфейс, предоставляющий клиентам доступ к каталогам, принтерам и текущему списку компьютеров.

Оба процесса запускаются системой Init-V. Названия сценариев Init-V зависят от применяемого дистрибутива. Если в вашем дистрибутиве по умолчанию не предусмотрен запуск Samba сразу же после установки, то обратитесь к разделу 4.5 — там даются советы, как запускать сценарии и конфигурировать автоматический запуск.

- Debian, Ubuntu — `/etc/init.d/samba` запускает оба процесса;
- Fedora, Red Hat, SUSE — `/etc/init.d/smb` запускает `smbd`, `/etc/init.d/nmb` запускает `nmbd`.

smb.conf. Основной конфигурационный файл Samba называется `/etc/samba/smb.conf`. Файл состоит из глобального раздела для основных настроек (`[global]`), а также из любого количества других разделов, предназначенных для предоставления ресурсов (каталогов, принтеров и т. д.) в совместное пользование. Каждый раздел начинается с *`[имя_ресурса]`*. Комментарии могут вводиться символом # или ;. Нельзя ставить комментарий сразу после настройки параметра, то есть каждый комментарий должен начинаться с новой строки.

Далее с небольшими сокращениями показан фрагмент стандартной конфигурации **Samba в Ubuntu. В остальных дистрибутивах этот файл (без учета комментариев)** часто еще короче, поскольку настройки, задаваемые по умолчанию, в нем отдельно не указываются. В следующем примере опущены разделы `[printers]` и `[print$]`, обеспечивающие доступ к принтеру и его драйверам (см. раздел 20.10).

```
# Файл /etc/samba/smb.conf в Ubuntu 9.10
[global]
  workgroup              = WORKGROUP
  server string          = %h server (Samba, Ubuntu)
  dns proxy              = no
  log file               = /var/log/samba/log.%m
  max log size           = 1000
  syslog                 = 0
  panic action           = /usr/share/samba/panic-action %d
  passdb backend         = tdbsam
  unix password sync     = yes
  passwd program         = /usr/bin/passwd %u
  passwd chat            = ...
  pam password change    = yes
  map to guest           = bad user
  usershare allow guests = yes
```

Идентификация сервера. С помощью `workgroup` определяется имя рабочей группы. Вероятно, это первая настройка, которую вам придется изменить, чтобы вписать здесь имя вашей рабочей группы, в которой будет действовать Samba.

Настройка `server string` указывает, под каким именем будет идентифицироваться сервер. Здесь `%h` заменяется хост-именем.

WINS. Благодаря настройке `dns proxy=no` Samba, работающая как WINS-сервер, не будет обращаться к DNS для разрешения хост-имен Windows. Если в вашей сети LAN будет локальный сервер имен, установите для этого параметра значение `yes`. Стандартное значение `no` пригодится вам только в тех случаях, когда в сети нет локального сервера имен, к которому можно быстро обратиться.

Журналирование. Параметры `log file`, `max log size` и `syslog` определяют, какие данные будет регистрировать Samba и где. Журналирование подробно рассматривается в одноименном подразделе далее.

При аварийном завершении работы Samba выполняется сценарий `panic-action`. Он отсылает администратору электронное сообщение, в котором содержится информация о возникших ошибках. Если на сервере не установлена почтовая система, то `panic-action` будет бездействовать.

Пароли. Настройка `passdb backend` указывает, как Samba должна работать с паролями. На выбор предлагаются варианты `smbpasswd` (обычный текстовый файл), `tdbsam` (TDB, относительно простая база данных) или `ldap-sam` (LDAP). Обычно в небольших или средних сетях (примерно до 250 клиентов) лучше всего использовать значение `tdbsam`. Систему `smbpasswd`, которая раньше была достаточно популярна, в настоящее время применять не рекомендуется, так как в ней нельзя сохранять расширенные атрибуты, применяемые в Windows NT 4 и выше (SAM Extended Controls). Управление паролями Samba более подробно рассмотрено в разделе 20.5.

Ключевые слова `unix password sync`, `passwd chat` и `pam password change` указывают, должна ли Samba сравнивать свои пароли с паролями Linux и, если должна, то как именно. Этот метод подробно описан в разделе 20.5.

Гости. С помощью `map to guest` и `usershare guest` определяется, как Samba должна поступать с пользователями, не прошедшими аутентификацию, то есть с пользователями, которые пытаются войти в систему под недействительным именем или паролем. Значение этих и некоторых других `guest`-параметров описывается в разделе 20.6.

Модель защиты. Возможно, вы заметили, что в последнем листинге отсутствует указание на модель защиты. По умолчанию в Samba, а значит, и в базовой конфигурации Ubuntu, действует защита на уровне пользователя (`security=user`). Если вы желаете применить иную модель защиты, ее нужно настроить как значение параметра `security`.

Стандарты Samba

В Debian и Ubuntu в файле `smb.conf` содержатся некоторые команды, отсутствующие в предыдущем листинге, поскольку они избыточны. Например, пароли обязательно шифруются на протяжении уже многих лет, поэтому `encrypt passwords=true` просто документирует стандартную настройку. Вас может запутать настройка `obey pam restrictions=yes`: она влияет на управление паролями лишь в том случае, когда пароли не шифруются. Поскольку в нашем случае пароли шифруются, эта настройка игнорируется.

Далее нам предстоит познакомиться еще со многими параметрами Samba, но, конечно же, не со всеми. Подробная информация обо всех возможностях настройки дается в `man smb.conf`.

Изменения конфигурации, статус

Чтобы изменения, внесенные в `smb.conf`, вступили в силу, Samba должна заново считать конфигурационные файлы:

```
root# /etc/init.d/samba reload
```

testparam. Если вы вносите в `smb.conf` значительные изменения, проверьте с помощью команды `testparam`, нет ли в коде синтаксических ошибок:

```
root# testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[printers]"
Processing section "[print$]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions <Return>
[global]
  server string = %h server (Samba, Ubuntu)
  ...
```

Если выполнить `testparam` с параметром `-v`, то команда выдаст очень длинный список всевозможных способов настройки `smb.conf`. Этот список может вам пригодиться в том случае, когда вы не знаете точно, какие настройки действуют по умолчанию (то есть какие из параметров действуют без вашего вмешательства в систему).

Чтобы узнать актуальное состояние сервера Samba, используйте `smbstatus`. Эта команда также выводит список всех соединений, которые активны в настоящий момент.

Защита Samba

Samba всегда должна иметь такую конфигурацию, чтобы сетевые каталоги действительно были видны и могли использоваться лишь в пределах локальной сети. В идеальном случае вам следует снаружи заблокировать брандмауэром TCP-порты, с которыми работают серверы Samba и Windows: 135, 139 и 145, а также UDP-порты 137 и 138.

Интерфейсы

Занимаясь конфигурацией Samba, не теряйте бдительности. Специально укажите с помощью `interfaces`, через какие сетевые интерфейсы сервер должен обмениваться информацией с остальными компьютерами. Интерфейсы указываются не по именам, а по адресным пространствам, используемым этими интерфейсами. Не забудьте о `localhost`, иначе на сервере не будут работать такие инструменты администрирования, как smbclient или SWAT. Этот параметр имеет значение лишь в том

случае, если ваш компьютер оборудован несколькими сетевыми интерфейсами (на многих компьютерах имеются интерфейсы не только для физических сетевых адаптеров, но и для виртуальных сетевых адаптеров различных программ виртуализации!). По умолчанию Samba обслуживает *все* сетевые интерфейсы.

Чтобы настройки, сделанные в `interfaces`, вступили в силу, вам потребуется специально активизировать настройку `bind interfaces only`.

Хосты

Далее вы можете перечислить в `hosts allow` те компьютеры, которые имеют право обмениваться информацией с Samba. Хост-имена, IP-адреса и IP-адресные пространства в этом списке отделяются друг от друга пробелами. В `hosts allow` можно сделать еще более точную выборку, чем в `interfaces`. Дополнительно в `hosts deny` можно указать отдельные хосты и адреса, которые нельзя использовать Samba.

Гости

Наконец, настройка `map to guest` запрещает всем пользователям, которые не прошли аутентификацию на сервере, любой доступ в сеть. В некоторых случаях может быть очень целесообразно создать для гостей отдельные каталоги, файлы в которых можно читать и даже изменять без аутентификации; однако если этого не требуется, нужно изначально закрыть гостям доступ в сеть.

```
# /etc/samba/smb.cnf
[global]
  ...
  bind interfaces only = yes
  interfaces           = 192.168.0.0/24 localhost
  hosts allow          = clientA clientB clientC
  map to guest         = never
```

Журналирование

Службы Samba `smbd` и `nmbd` регистрируют глобальные события в файлах `/var/log/samba/log.smbd` и `log.nmbd`. Ни название, ни местоположение этих файлов регистрации нельзя изменить с помощью `smb.conf`.

Параметр `log file` в глобальном разделе `smb.conf` указывает, где должны сохраняться клиентские уведомления. Благодаря предварительной настройке `/var/log/samba/log.%m` для каждого клиента, который обращается к Samba, создается свой файл регистрации с именем `log.hostname`. Параметр `max log size=1000` ограничивает максимальный размер 1000 Кбайт. Если файл регистрации превышает этот размер, то Samba переименовывает его — `name.old`. Параметр `syslog=0` не означает, что `syslog` не применяется, а значит, что в `/var/log/syslog` должны регистрироваться только сообщения об ошибках. Могут также применяться значения `1`, `2`, `3` и т. д., если вы хотите регистрировать предупреждения, замечания и сообщения об отладке.

logrotate. Будьте осторожны, применяя программу `logrotate` (см. раздел 9.7): согласно настройке, действующей в Debian, SUSE и Ubuntu по умолчанию, эта программа раз в неделю архивирует файлы `log.smbd` и `log.nmbd` и одновременно

удаляет все архивные версии, которым больше двух месяцев. Однако logrotate игнорирует клиентские файлы регистрации log.hostname, которые прирастают значительно быстрее. Аналогичный конфигурационный файл в Fedora и Red Hat построен более рационально, и в нем учитываются *все* файлы регистрации, находящиеся в /var/log/samba:

```
# /etc/logrotate.d/samba в Fedora и Red Hat
/var/log/samba/* {
  notifempty
  olddir /var/log/samba/old
  missingok
  sharedscripts
  copytruncate
}
```

В качестве альтернативы вы можете использовать в smb.conf настройку log file=/var/log/samba/log.smbd. Таким образом, smbd будет регистрировать в одном и том же файле и глобальные, и клиентские уведомления. Если вы не ставите своей задачей отыскать ошибки в конфигурации Samba, лучше всего остановиться именно на этом решении.

Сетевая конфигурация с помощью SWAT

Для того чтобы не изменять конфигурационный файл Samba smb.conf в текстовом редакторе, можете применить *Samba Web Administration Tool* (кратко — SWAT). Этапы установки SWAT и введения его в эксплуатацию различаются в зависимости от дистрибутива. В любом случае перед установкой сохраните резервную копию smb.conf.

Debian, Ubuntu

SWAT находится в одноименном пакете, который необходимо специально установить. Вместе с ним устанавливается пакет openbsd-inetd. Чтобы он учитывал изменения, автоматически вносимые в конфигурационный файл /etc/inetd.conf, перезапустите openbsd-inetd:

```
root# /etc/init.d/openbsd-inetd restart
```

Для работы SWAT нужно, чтобы у учетной записи root был пароль. В Ubuntu по умолчанию это не так. Выполните sudo passwd root и задайте надежный пароль администратора!

Fedora, Red Hat

SWAT находится в пакете samba-swat, который устанавливается дополнительно. Одновременно с ним устанавливается пакет xinetd. Чтобы активизировать SWAT, замените в /etc/xinet.d/swat строку disable=yes строкой disable=no и запустите xinetd.

```
root# /etc/init.d/xinetd start
root# chkconfig --add xinetd
```

SUSE

В SUSE SWAT содержится в пакете `samba`. SWAT нужно специально активизировать, заменив в `/etc/xinet.d/swat` строку `disable=yes` строкой `disable=no`. Кроме того, не забудьте запустить `xinetd`:

```
root# insserv xinetd
root# /etc/init.d/xinetd start
```

Применение

SWAT использует миниатюрный веб-сервер для обмена информацией через порт 901 (Apache не требуется!). Чтобы начать работу со SWAT, введите в адресной строке браузера адрес `http://localhost:901`. В Fedora 11 указывается `http://localhost4:901`, чтобы обмен данными происходил через IPv4.

После этого появляется окно для входа в систему, в котором вы должны представиться как root. Остальные пользователи не имеют достаточных прав для изменения конфигурационного файла Samba и могут только узнать статус Samba, а также изменить собственный пароль для Samba. В эпоху Web 2.0 пользовательский интерфейс SWAT может показаться несколько старомодным (рис. 20.1).

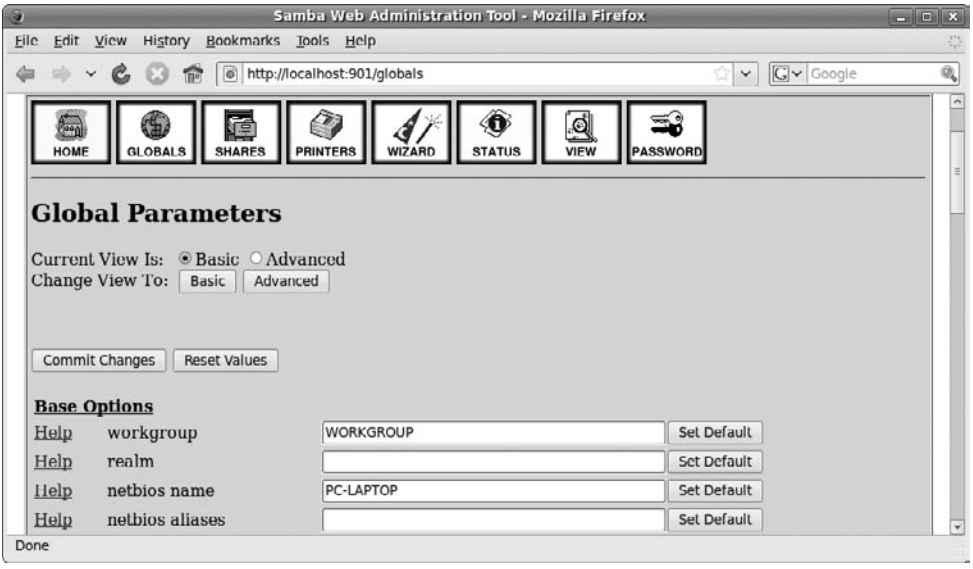

Рис. 20.1. Конфигурация Samba с помощью SWAT

Если вы сохраняете изменения, внесенные в конфигурацию, то SWAT удаляет из `smb.conf` все имеющиеся комментарии, изменяет порядок следования имеющихся записей и самовольно меняет некоторые настройки. По этим причинам я рекомендую создавать резервную копию `smb.conf` до начала установки.

Среди специалистов по Samba SWAT имеет дурную репутацию. Этот инструмент лишь незначительно облегчает конфигурацию. Если вам удалось произвести конфигурацию `smb.conf` с помощью SWAT, то вы с тем же успехом могли бы

изменить эту конфигурацию и вручную, но без ограничений, которые возникают при использовании SWAT.

Защита

При входе в систему имя пользователя и пароль передаются в незашифрованном виде, поэтому SWAT из соображений безопасности следует применять только на локальном компьютере или в локальной сети. В Debian и Ubuntu для этого необходимо изменить файлы `/etc/hosts.deny` и `hosts.allow` так, как это описано в разделе 18.2 (в следующем примере показан случай с локальной сетью, работающей в адресном пространстве 192.168.0.*):

```
# Файл /etc/hosts.allow
swat : 192.168.0.0/24
# Файл /etc/hosts.deny
swat : ALL
```

В Fedora, Red Hat и SUSE такого шага делать не требуется, так как на основании строки `only_from=127.0.0.1` в `/etc/xinet.d/swat` соединение со SWAT может устанавливать только localhost.

20.5. Управление паролями в Samba

По умолчанию в Samba действует настройка `security=user`, то есть защита на уровне пользователя. Например, чтобы пользователь peter мог работать с сетевым каталогом, должны выполняться следующие условия.

- На сервере должна быть учетная запись Linux с именем peter. Она требуется для управления правами доступа. Samba обращается к правам доступа, действующим в Linux, чтобы определить, какие пользователи имеют право на чтение и изменение файлов.

  Учетная запись Linux должна быть неактивна. Часто из соображений безопасности бывает нужно задать для учетной записи недействительный пароль. Таким образом, исключается вероятность, что пользователь Samba войдет на сервер.
- peter и его пароль должны содержаться в базе данных пользователей Samba. По техническим причинам учетные записи Samba (имя пользователя, пароль и другие данные) управляются независимо от учетных записей Linux. Пароль учетной записи Linux не имеет для Samba никакого значения.
- В `smb.conf` должны быть указаны сетевые каталоги, которые может использовать peter (подробнее об этом читайте в разделе 20.6).

Пароли Samba

Прежде чем пользователь локальной сети сможет получить доступ к каталогу, он должен пройти идентификацию на сервере Samba. При установлении соединения с клиентом Windows для этого сообщаются логин Windows и зашифрованная последовательность символов — пароль. На клиентских компьютерах, использующих

Linux, эти данные нельзя получить, зная учетную запись Linux, поэтому обычно на экран выводится окно для ввода имени пользователя и пароля Samba. Затем, как и с клиентами Windows, процесс продолжается: пароль на сервер Samba отправляется в зашифрованном виде, а не открытым текстом.

Из соображений безопасности пароль нельзя восстановить, имея зашифрованную последовательность символов, в виде которой он передается на сервер. Эта последовательность на сервере Samba сравнивается с другой последовательностью, зашифрованной аналогичным образом. Если обе последовательности символов совпадают, значит, совпадают и пароли.

Внутренний интерфейс базы данных smbpasswd. Для работы серверу Samba требуется база данных, в которой хранятся имена пользователей и пароли, с помощью которой осуществляется аутентификация. Раньше для этих целей зачастую применялся обычный текстовый файл (`passdb backend=smbpasswd`). В SUSE этот внутренний интерфейс по-прежнему применяется и пароли сохраняются в файле `/etc/samba/smbpasswd`.

Внутренний интерфейс TDB. В большинстве других дистрибутивов для небольших сетей используется интерфейс TDB (`passdb backend=tdbsam`). В более крупных сетях данными учетных записей лучше управлять через LDAP, но этот вопрос мы отдельно обсуждать не будем.

TDB означает «простая база данных» (trivial database) и представляет собой двоичный формат для сохранения наборов данных. Существенное преимущество этого формата по сравнению с обычными файлами `smbpasswd` заключается в том, что кроме логина и пароля в нем можно сохранять и другие данные, в частности атрибуты. К таким данным относятся, например, SAM Extended Controls, улучшающие совместимость с современными версиями Windows.

От дистрибутива зависит, где именно физически сохраняются пароли:

- Debian, Ubuntu — `/var/lib/samba/passdb.tdb`;
- Fedora, Red Hat —`/var/lib/samba/secret/passdb.tdb`.

При необходимости вы можете указать в `smb.conf` другой файл с помощью настройки `passdb backend=tdbsam:«имя_файла»`. Команда `smbpasswd` создает новую учетную запись Samba или изменяет ее пароль; `pdbedit` обеспечивает доступ ко всей информации, касающейся учетной записи: таким образом, администраторы могут создавать списки всех пользователей Samba (`pdbedit -L -v`), указывать для каждой учетной записи специальные атрибуты и т. д. Обе команды интерпретируют `smb.conf` и работают со всеми системами паролей (то есть с `smbpasswd`, `tdbsam` и `ldapsam`).

Итак, чтобы пользователь peter мог работать с каталогом Samba, вы, как системный администратор Linux, должны создать в Samba учетную запись peter. Для этого используйте команду `smbpasswd`. В качестве пароля укажите ту последовательность символов, которая является паролем пользователя peter в Windows. Чтобы изменить уже имеющийся пароль Samba, используйте `smbpasswd` без параметра `-a`.

```
root# smbpasswd -a peter
New SMB password: ******
Retype new SMB password: ******
Added user peter.
Password changed for user peter.
```

Обратите внимание — чтобы команда sbmpasswd работала, в локальной сети также должен существовать пользователь Linux peter (файл /etc/passwd). При необходимости новые пользователи Linux добавляются с помощью команд useradd или adduser.

Синхронизация паролей Samba и Linux

Обычно при работе в сети бывает необходимо, чтобы пользователи каталогов Samba могли регистрироваться прямо на сервере, например для работы с SSH и обработки своих файлов с применением команд Linux. В таких случаях, разумеется, пароли Samba и Linux обязательно должны совпадать. Наконец, было бы очень неудобно вносить каждое изменение пароля по нескольку раз (в самых сложных случаях трижды: для клиента Windows, сервера Samba и учетной записи Linux).

К сожалению, выполнить синхронизацию паролей не так просто, поскольку для шифрования паролей Samba используется иной алгоритм, нежели для паролей Linux. Поэтому управление паролями Samba и Linux происходит отдельно.

ПРИМЕЧАНИЕ

Хотя алгоритмы и разные, у них есть одна общая черта: сохраняемые последовательности символов позволяют проконтролировать пароль, но не позволяют его реконструировать. По этой причине преобразовать или конвертировать сохраненные пароли из одной системы в другую невозможно.

Наиболее популярное решение этой проблемы заключается в том, чтобы при каждом вызове smbpasswd со стороны клиента (для изменения пароля) параллельно с этим изменялся и пароль для Linux. В Ubuntu smbpasswd уже содержит необходимые для этого настройки:

```
[global]
...
  unix password sync  = yes
  pam password change = yes
  passwd chat         = *Enter\snew\s*\spassword:* %n\n
                        *Retype\snew\s*\spassword:* %n\n
                        *password\supdated\ssuccessfully*
```

Команда unix password sync=yes активизирует синхронизацию. При этом по причине использования pam password change применяется PAM. PAM — это подключаемые модули аутентификации, представляют собой собрание библиотек для администрирования паролей. Параметр passwd program, который требовался ранее и задавал путь к программе passwd, в PAM не имеет значения и игнорируется. Параметр passwd chat описывает обмен данными между Samba и PAM. Эта последовательность символов в предыдущем листинге была разделена на три строки, но в smb.conf она указывается одной строкой.

К сожалению, синхронизация связана со множеством ограничений.

- Команда smbpasswd выполняется конкретным пользователем, а не администратором! Причина: если smbpasswd выполняет root, он переходит к работе непосредственно с базой данных Samba (этот механизм действует и в том случае,

когда сервер Samba не работает). Если же командой управляет обычный пользователь, эта команда связывается с сервером Samba, который и выполняет нужную нам задачу, в том числе синхронизирует пароль.

- Команда `smbpasswd` принимает даже самые плохие пароли (например, состоящие из пробела или только из одной буквы). А система паролей Linux или PAM, напротив, требует хотя бы минимально качественного пароля и не принимает совсем простых вариантов. В результате пароль Samba может измениться, а пароль Linux останется прежним. С этого момента синхронизация паролей закончится и любая новая попытка изменить пароль Linux потерпит неудачу. Чтобы снова синхронизировать пароли, администратору потребуется заново настроить пароли Samba и Linux для конкретного пользователя.
- Синхронизация функционирует лишь в одном направлении: от Samba к Linux. Если же пользователь изменит свой пароль к Linux с помощью `passwd`, пароль Samba останется прежним. В Интернете есть руководства, помогающие направить синхронизацию и от Linux к Samba, например здесь:
  - http://www.debianforum.de/forum/viewtopic.php?f=9&p=651961;
  - http://tinyurl.com/5fax2v.

Из личного опыта я не рекомендую выполнять описанную выше синхронизацию паролей. При этом очень легко допустить ошибку и создать гораздо больше проблем, чем вы можете решить, применив такую синхронизацию. Используйте настройку `unix password sync=no`.

Соотнесение пользователей Linux и Windows

В Windows именем пользователя может служить практически любая последовательность символов (до 128 знаков). В Linux длина имени пользователя, напротив, составляет всего 32 знака — без специальных символов и пробелов. Если пользователь Windows применяет имя, которое не может быть именем пользователя в Linux, две этих последовательности необходимо соотнести через специальный файл. Название этого файла указывается в `smb.conf` с помощью параметра `username map`:

```
# /etc/samba/smb.conf
[global]
  ...
  username map = /etc/samba/smbusers
```

В каждой строке файла `smbusers` сначала указывается имя пользователя в Linux, затем символ = и одно или несколько имен пользователя в Windows. Имена, содержащие пробелы, задаются в кавычках. Файл можно использовать и в том случае, если нескольким пользователям Windows соответствует один пользователь Linux.

```
# /etc/samba/smbusers
peter = "Peter Mayer"
...
```

ВНИМАНИЕ

Файл /etc/samba/smbusers может вывести из строя систему безопасности Samba или Linux, если имя root будет присвоено другому пользователю, кроме администратора! Обратите внимание, что этот файл разрешено изменять только пользователю root:

```
root# chmod 644 /etc/samba/smbusers
```

Все вместе

Предположим, в вашей локальной сети есть компьютер с Windows, на котором работает Петер Майер, и на этом компьютере используется логин Peter Mayer. На сервере Linux, где установлена Samba, есть учетная запись peter и программа `smbusers`, которая соотносит друг с другом Peter Mayer и peter. При таких условиях мы получим следующие сочетания логина и пароля.

- Логин в Windows — Peter Mayer и пароль Windows. Пароль Windows используется при работе в пределах локального компьютера с Windows. Петер может изменить свой пароль в Windows.
- Логин на сервере (Linux) — peter и пароль Linux. Пароль Linux применяется для входа непосредственно на сервер, если учетная запись Linux активна и не заблокирована. Петер может изменить свой пароль, войдя на сервер через SSH и применив команду `passwd`.
- Доступ к сетевым каталогам — Peter Mayer или peter и пароль для Samba. Пароль Samba используется для работы с сетевыми каталогами. Он должен совпадать с паролем Windows. Если это не так, то, когда Петер захочет получить доступ к сетевому каталогу, в Windows откроется окно для входа в систему.

  Петер может изменить пароль с помощью команды `smbpasswd` после входа в SSH. Если на сервере установлен SWAT, то Петер также может изменить свой пароль в браузере. В любом случае в SWAT Петер должен входить с тем паролем, который используется для *Linux*, и только после этого он сможет изменить пароль, применяемый в *Samba*. Следует признать, что схема довольно запутанная. Если учетная запись Linux заблокирована, то Петер никак не может сам изменить свой пароль для Samba!

Таким образом, только те пользователи, которые отлично подкованы технически, могут сами изменять все три своих пароля — и лишь при условии, что соответствующая учетная запись Linux активна. Для всех остальных пользователей действует правило: однажды заданные пароли никогда изменяться не будут. С точки зрения безопасности это, конечно же, совсем нехорошо.

20.6. Samba: сетевые каталоги

В предыдущем разделе мы рассмотрели, какие предпосылки должны выполняться, чтобы пользователь мог войти в Samba. Если сказать кратко, то у пользователя должны быть учетная запись в Linux и пароль к Samba. Однако пока остается открытым вопрос, какие ресурсы увидит и сможет применить пришедший пользователь. За это отвечают разделы `[`*имя_ресурса*`]` в `smb.conf`.

Пользовательские каталоги

Определение каталога, к которому будет иметь доступ конкретный пользователь, выглядит так:

```
# в /etc/samba/smb.conf
...
[directory1]
  user      = peter
  path      = /data/dir1
  writeable = yes
```

Такая настройка позволяет пользователю peter, а также всем другим пользователям, которые в соответствии с smbusers работают с данной учетной записью, читать каталог /data/dir1 и вносить в него изменения. В файловом менеджере этот ресурс называется directory1 (то есть последовательность символов, указанная в квадратных скобках).

Значения ключевых слов понятны: user — это имя пользователя. Вместо user также могут использоваться ключевые слова users или username. Можно указать несколько имен пользователя, разделяя их запятыми.

Слово path указывает, какой каталог сервера предоставляется в совместное использование. Если path специально не указать, то Samba по умолчанию предоставляет в совместное использование домашний каталог указанного пользователя. С помощью writeable=yes мы разрешаем вносить в каталог изменения. Без этого параметра пользователь имеет доступ только для чтения.

В принципе, для всех видов доступа действуют права, задаваемые в Linux. Иными словами, если в /data/dir1 есть файл, принадлежащий пользователю root, то пользователь peter, как правило, может только читать этот файл, но не может изменять. Такое же ограничение действует для всех пользователей сетевых каталогов.

Домашние каталоги

Если файловый сервер настроен для работы с большим количеством пользователей, то проще всего предоставить каждому пользователю Linux возможность видеть и изменять свой домашний каталог. Чтобы вам не пришлось заносить в smb.conf бесчисленное множество записей вида:

```
[username]
  User      = username
  writeable = yes
```

в smb.conf предусмотрена следующая сокращенная форма записи:

```
# в /etc/samba/smb.conf
...
[homes]
  writeable  = yes
  browseable = no
```

Таким образом, домашний каталог пользователя, активного в настоящий момент, виден под именем этого пользователя. Если применить параметр `browseable=no`, то пользователь не перестанет видеть свой домашний каталог, как это может показаться. Просто при этом каталог не будет отображаться в системе дважды: один раз под именем пользователя (например, peter), а один раз — под homes.

Групповые каталоги

Пользовательские и домашние каталоги позволяют пользователю сохранять свои файлы прямо на сервере, но при этом отсутствует возможность обмена данными, ведь эти каталоги невидимы для других пользователей, а значит, и недоступны. Для обмена данными применяются групповые каталоги, которыми могут пользоваться все члены определенной группы. Отнесение пользователей к группам происходит в рамках управления группами в Linux. Группа задается ключевым словом `user` и записью `@имя_группы`.

```
# в /etc/samba/smb.conf
...
[salesdata]
  user           = @sales
  path           = /data/sales
  writeable      = yes
  force group    = +sales
  create mask    = 0660
  directory mask = 0770
```

При настройке доступа к групповым каталогам особенно важно правильно указать, кто имеет право пользоваться какими ресурсами. Это же касается новых файлов и каталогов, которые будут создаваться впоследствии. Благодаря `force group=+sales` новые файлы и каталоги будут попадать в группу `sales` (а не в стандартную группу создающего их пользователя). Если пользователь не является членом группы `sales`, он не получит доступа к этому каталогу.

ВНИМАНИЕ

В вышеуказанном случае никогда не применяйте настройку force group = sales (то есть без плюсика), иначе любой акт доступа к каталогу будет осуществляться в Samba так, как будто активный в данный момент пользователь является членом группы sales — даже тогда, когда с точки зрения Linux пользователь совсем не входит в эту группу. Иными словами, параметр force group = sales означает, что пользователи, совсем не относящиеся к группе sales, получают право чтения и изменения файлов каталога. В групповых каталогах такая ситуация практически всегда неожиданна и может очень негативно повлиять на безопасность системы!

Параметры `create mask` и `directory mask` гарантируют, что члены группы смогут читать и изменять все новые файлы и каталоги, создаваемые другими членами той же группы (восьмеричное число соответствует значению `chmod` — см. `man chmod`). Если новые файлы или каталоги должны предоставляться членам группы только для чтения, но не для изменения, используйте значения `0440` и `0550`.

Каталоги, находящиеся в свободном доступе

Еще более «либеральным» является доступ к открытому для всех каталогу: каждый пользователь, который может пройти аутентификацию в Samba, может читать файлы из такого каталога. Доступ для изменения в данном случае отключается:

```
# в /etc/samba/smb.conf
...
[share]
  path      = /data/share
  read only = yes
```

Разумеется, вы можете настроить для каталогов со свободным доступом и возможность внесения изменений (`writable=yes`). По умолчанию все пользователи имеют право читать файлы, созданные другими пользователями, но не могут их изменять. Для изменения этой ситуации используйте настройку `force group` и два параметра `mask`. Чтобы задействовать неограниченные взаимные права чтения и изменения файлов в группе, укажите `create mask=0666` и `directory mask=0777`.

Доступ для пользователей, не прошедших аутентификацию

Во всех предыдущих примерах предполагается, что пользователь может пройти аутентификацию в Samba. Однако Samba допускает и такую конфигурацию, при которой к каталогам получают доступ пользователи, не прошедшие аутентификацию. Они именуются *гостями* (guest). За работу с гостями отвечают глобальные настройки, обобщенные в следующем листинге:

```
# в /etc/samba/smb.conf
[global]
  ...
  map to guest           = bad user
  usershare allow guests = yes
  guest account          = nobody
```

Благодаря настройке `map to guest=bad user` попытки входа в систему под несуществующим логином автоматически присваиваются виртуальному пользователю Samba guest. По умолчанию в системе нет никаких сетевых каталогов или других ресурсов, которые может использовать guest. Настройка `usershare allow guests=yes` позволяет предоставлять отдельные каталоги для использования посетителями с учетной записью guest (с помощью `guest ok` или `guest only`). Параметр `guest account` указывает, каким пользователям Linux соответствуют гости. В большинстве дистрибутивов, в том числе в Debian и Ubuntu, для этого предусмотрен пользователь nobody.

Каталоги, которые должны быть открыты для использования гостями, обозначаются с помощью параметра `guest ok=ok`. Как правило, такие каталоги следует защищать от изменений — для этого используется настройка `read only=yes`. Можно

сказать, что гости смогут читать или изменять те же каталоги, что и пользователь Linux nobody.

```
[guest]
  path      = /data/guest
  guest ok  = yes
  read only = yes
```

Вариант, аналогичный `guest ok`, это `guest only=yes`: при такой настройке с каталогом могут работать не только гости, но и пользователи, прошедшие аутентификацию. Если вы вообще не хотите допускать гостей к использованию ресурсов Samba, используйте в разделе `[global]` настройку `map to guest=never`.

Предоставление каталогов в общий доступ с помощью Gnome и KDE

Если вы хотите быстро и без лишних сложностей открыть каталог локальному пользователю через Samba, вам, конечно же, не захочется вручную вносить изменения в `smb.conf`. В таком случае каталог можно предоставить в совместное использование через специальное окно Gnome или KDE. К сожалению, в течение уже многих лет эти окна работают довольно плохо.

Nautilus/Gnome

Nautilus поддерживает предоставление каталогов в свободный доступ при условии, что в системе установлено расширение `nautilus-share`. В современных версиях Ubuntu по умолчанию так и есть. Теперь в Nautilus нужно щелкнуть на каталоге правой кнопкой мыши, чтобы вызвать контекстное меню. Команда меню Параметры общего доступа открывает конфигурационное диалоговое окно (рис. 20.2). Этим окном можно пользоваться, не имея прав администратора.

Рис. 20.2. Предоставление каталога с помощью Nautilus/Gnome

Nautilus использует при работе достаточно новый конфигурационный механизм, который Samba поддерживает только в версии 3.0.32 и выше: при работе с этим механизмом параметры каждого отдельного каталога, предоставляемого в общее пользование, сохраняются в отдельном конфигурационном файле в каталоге `/var/lib/samba/usershares`. Права доступа к этому каталогу настроены так, что все пользователи, относящиеся к определенной группе (в Ubuntu — `sambashare`), могут сохранять в нем новые файлы. Готовый конфигурационный файл выглядит так же, как и в следующем примере:

```
#ВЕРСИЯ 2•
# Файл /var/lib/samba/usershares/testdir
path          = /myhome/kofler/testdir
comment       =
usershare_acl = S-1-1-0:R
guest_ok      = n
```

Еще остается открытым вопрос, кто сможет использовать предоставленный каталог. Если установить флажок Разрешить доступ гостю, то любой посетитель получит доступ к вашему сетевому каталогу. Если же снять этот флажок, то пользователь получит доступ к каталогу, только если войдет в систему под своим именем и паролем. Этот механизм работает только с пользователями, для которых на данном компьютере существует учетная запись, и лишь в том случае, если для них с помощью `smbpasswd` был задан пароль Samba. На этом этапе `nautilus-share` вам уже не поможет. Далее `smbpasswd` уже требует для работы права администратора или должна выполняться в консоли с помощью программы `sudo`. Таким образом, здесь интуитивная часть конфигурации свободного доступа заканчивается.

Dolphin/KDE

В файловых менеджерах Dolphin и Konqueror, применяемых в KDE, сетевой каталог можно создать на вкладке Общий доступ окна Свойства (рис. 20.3). Для этого в дистрибутиве должны быть установлены и Samba, и пакет `kdenetwork-filesharing`. Во многих дистрибутивах при стандартной конфигурации эти условия не выполняются, поэтому, если вы нажмете кнопку Разрешить общий доступ к файлам, система просто проигнорирует это действие и даже не выведет сообщения об ошибке.

Когда все условия будут выполнены, потребуется в зависимости от дистрибутива ввести пароль администратора или ваш личный пароль. В открывшемся окне Общий доступ к файлам вы можете выбрать обычное или расширенное предоставление каталога в общее пользование — в ходе испытаний я не нашел разницы между этими вариантами. Нажав кнопку Добавить, вы выбираете каталог, к которому нужно открыть доступ (диалоговое окно, к сожалению, не выбирает автоматически тот каталог, который активен в настоящий момент).

Параметры `PUBLIC` и `WRITEABLE` определяют, все ли пользователи будут иметь доступ к каталогу и смогут его изменять. Как это обычно бывает в KDE, кнопка Другие параметры Samba предлагает на выбор еще около сотни параметров и их вариантов. Маловероятно, что все они понадобятся обычному пользователю. В отличие от Gnome, настройки, касающиеся каталога, предоставляемого в свободный доступ, сохраняются непосредственно в файле `smb.conf` (а не в `/var/lib/samba/usershares`).

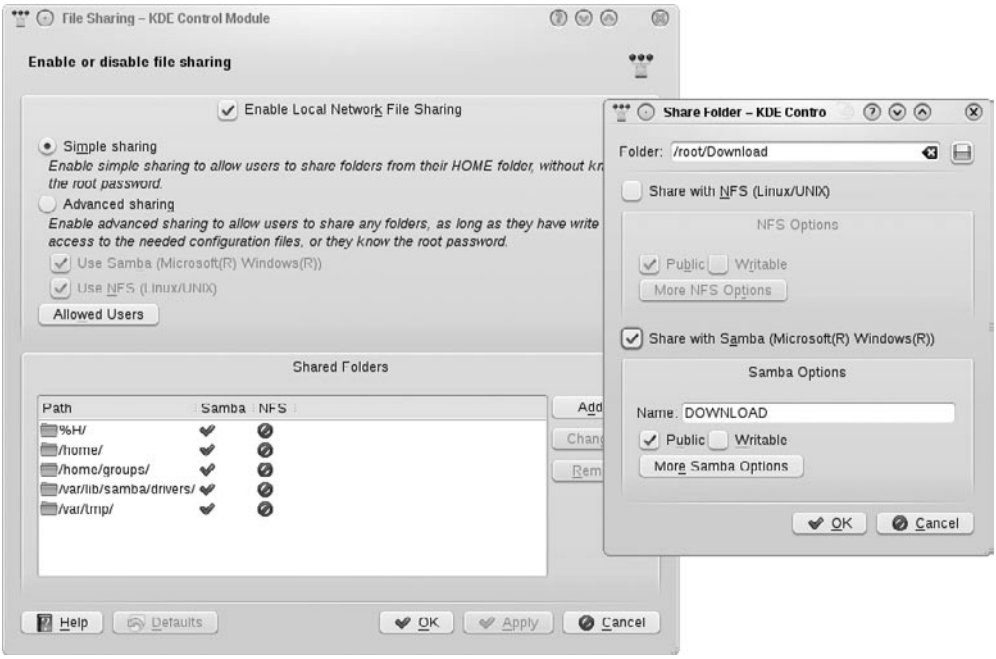

Рис. 20.3. Предоставление каталога в общий доступ с помощью Dolphin/KDE

Кроме того, в KDE не решен вопрос с управлением пользователями Samba: если каталог не должен предоставляться в общее пользование, то вы, как и в Gnome, должны перейти в консоль и создать необходимые пользовательские учетные записи Samba, применив права администратора.

20.7. Samba: домашний сервер/сервер мультимедиа

В этом разделе приведен простой пример конфигурации центрального сервера Samba. Представим себе компьютеризированное домохозяйство, в котором на трех компьютерах, используемых родителями и двумя детьми, накапливается все больше данных: цифровые фото, MP3, школьные работы, бухгалтерия и т. д. При децентрализованном хранении данных возникают некоторые проблемы:

- не выполняется регулярное резервное копирование (что делать, если ноутбук потеряется по дороге в школу или сломается?);
- к общим данным обращаться достаточно сложно. Например, мы запланировали подарить бабушке на следующий день рождения альбом лучших семейных фотографий за последние годы. Но цифровые фото разбросаны по трем компьютерам и лежат в полном беспорядке. Похожие проблемы возникают и на вечеринках, когда вдруг оказывается, что песни, которые все хотят услышать, находятся на другом компьютере.

- обмен данными между компьютерами усложнен и происходит в основном через флешки.

Наконец, сын, увлекающийся Linux, предлагает решить эти проблемы, оборудовав центральный домашний сервер/сервер мультимедиа. Сервер можно интегрировать в домашнюю сеть по беспроводной сети WLAN. При необходимости этот компьютер можно одновременно использовать как интернет-роутер и как брандмауэр.

В данном случае нас заинтересует только конфигурация с применением Samba: каждый член семьи получит собственный сетевой каталог, в котором он может считывать и изменять любые данные. Сохраняемые в этих каталогах файлы являются частными (с той оговоркой, что сын — системный администратор — может прочитать и изменить любой файл). Для общего обмена данными создается еще пять каталогов. Родители могут обращаться к parents, дети — к children, а все члены семьи вместе — к family, audio и photos. Разумеется, каталоги audio и photos можно сделать просто подкаталогами family, но если выделить для них отдельные сетевые каталоги, то работать будет удобнее. При необходимости можно создать дополнительные учетные записи пользователей и каталоги.

Создание пользователей и групп Linux

В качестве имен пользователей мы будем использовать слова mother, father, daughter, son. На практике, конечно же, будут применяться настоящие имена, но здесь мы от этого отказались, чтобы вам не пришлось запоминать имена членов выдуманной семьи. В командах `useradd` благодаря параметру `--create-home` создается каталог `/home/«имя»`. Затем в него копируется содержимое каталога `/etc/skel`. Если вам не нужны эти файлы, то можете сразу их удалить (последняя команда):

```
root# groupadd parents
root# groupadd children
root# groupadd family
root# useradd --create-home --groups parents,family father
root# useradd --create-home --groups parents,family mother
root# useradd --create-home --groups children,family son
root# useradd --create-home --groups children,family daughter
root# rm -rf /home/{father,mother,daughter/son}/*     (необязательно)
```

Поскольку команда `useradd` выполнялась без пароля, новые пользователи были автоматически заблокированы (то есть они не могут войти в систему). Так и планировалось: мы не собирались давать членам семьи возможность входить на сервер под своими именами, так как в этом нет никакой необходимости.

Вместе с каждым пользователем создается и одноименная группа, являющаяся для него стандартной. Кроме того, новые пользователи попадают в группы family и parents или children. Таким образом, отец будет относиться к группам father, parents и family, мать — к группам mother, parents и family и т. д. Если позже вы захотите отнести пользователя еще к одной группе, то лучше всего воспользоваться следующей командой:

```
root# usermod -a -G newgroup user
```

Создание учетной записи пользователя Samba

Теперь создадим учетные записи пользователей Samba, чтобы каждая такая запись имела пароль:

```
root# smbpasswd -a father
New SMB password: *******
Retype new SMB password: *******
root# smbpasswd -a mother
...
root# smbpasswd -a son
root# smbpasswd -a daughter
```

Создание каталогов

При создании каталогов для общих файлов важно правильно настроить владельцев файлов и права доступа к файлам и каталогам, иначе в дальнейшем доступ к файлам может не функционировать. Благодаря первой команде `chmod 770` только члены группы могут читать каталог и вносить в него изменения. Вторая команда закрывает доступ к домашнему каталогу для любых пользователей, кроме его владельца.

```
root# mkdir /shared-data
root# mkdir /shared-data/{parents,children,family,audio,photos}
root# cd /shared-data
root# chown :parents parents/
root# chown :children children/
root# chown :famile family/ audio/ photos/
root# chmod 770 *
root# chmod 770 /home/{father,mother,son,daughter}
```

Важное достоинство общего файлового сервера заключается в том, что он облегчает централизованное резервное копирование. При этом защитить нужно только каталоги /home и shared-data.

Конфигурация Samba

В следующем примере показан конфигурационный файл `smb.conf`. Синхронизация паролей и любой доступ гостей к Samba деактивизируются. Настройки различных каталогов должны быть понятны на основе материала, изложенного в разделе 20.6.

```
# /etc/samba/smb.conf для домашнего сервера
[global]
  workgroup          = home
  server string      = %h server (Samba, Ubuntu)
  security           = user
  passdb backend     = tdbsam
  unix password sync = no
  invalid users      = root
```

```
  map to guest       = never
  log file           = /var/log/samba/log.%m
  max log size       = 1000
  syslog             = 0
  dns proxy          = no
  panic action       = /usr/share/samba/panic-action %d
[homes]
  browseable         = no
  writeable          = yes
[parents]
  user               = @parents
  path               = /shared-data/parents
  writeable          = yes
  force group        = +parents
  create mask        = 0660
  directory mask     = 0770
[children]
  user               = @children
  path               = /shared-data/children
  writeable          = yes
  force group        = +children
  create mask        = 0660
  directory mask     = 0770
[family]
  user               = @family
  path               = /shared-data/family
  writeable          = yes
  force group        = +family
  create mask        = 0660
  directory mask     = 0770
[photos]
  user               = @family
  path               = /shared-data/photos
  ... как в [family]
[audio]
  user               = @family
  path               = /shared-data/audio
  ... как в [family]
```

В этой конфигурации есть один декоративный недостаток: все пользователи видят *все* каталоги, находящиеся в открытом доступе, в том числе те, которые предоставлены не для всех пользователей и которые могут использоваться не всеми (например, родители видят каталог children, а дети — каталог parents). На самом деле пользоваться такими каталогами из-за настроек прав доступа удастся не всем, но красивее было бы, если бы эти каталоги вообще не были видны.

Такая возможность в конфигурации Samba, к сожалению, не предусмотрена. Некоторые каталоги можно скрыть с помощью настройки `browseable=no`, но тогда каталоги не сможет увидеть никто, даже их «законные» пользователи. (каталоги при этом можно использовать как и раньше, но нужно точно указать путь вручную).

Параметры `hide unreadable=yes` и `hide unwriteable=yes` в этом случае не помогут: с их помощью можно скрыть *внутри* сетевого каталога все файлы, которые нельзя читать и изменять определенному пользователю. Но сам сетевой каталог останется видим.

20.8. Samba: клиентский доступ

В этом разделе мы рассмотрим следующую проблему: как клиентский ПК, работающий с Windows или Linux, будет обращаться к тем каталогам, которые Samba предоставила в совместное использование. Необходимо, чтобы выполнялось важное предварительное условие: порты TCP 135, 139 и 445, а также порты UDP 137 и 138 не должны блокироваться брандмауэром.

Клиенты Linux

Прежде чем ваши клиенты, работающие с Linux, получат доступ к каталогам Windows, предоставленным Samba, вам, возможно, потребуется установить клиентские инструменты Samba (по умолчанию они обычно установлены). В Ubuntu необходимые программы упакованы в `samba-common`, `smbclient` и `libsambaclient`, в других дистрибутивах эти пакеты имеют немного другие названия.

KDE, Gnome

Для доступа к сетевому каталогу лучше всего использовать файловый менеджер Gnome или KDE. В обеих программах имеется средство для просмотра сети, которое сначала показывает все доступные каталоги Windows. Чтобы попасть в нужный каталог, вам понадобится пару раз щелкнуть кнопкой мыши и, возможно, ввести свои учетные данные (имя пользователя и пароль).

Пока файловый менеджер еще не в состоянии самостоятельно находить сетевые каталоги. В таком случае вы должны точно указать местоположение каталога в адресной строке файлового менеджера. Действует следующий принцип записи: `smb://имя_сервера/название_каталога`.

Нелюбители KDE и Gnome, которые все же ищут графическую систему для доступа к каталогам Windows, могут попробовать программу LinNeighborhood.

CIFS

Другой метод заключается в том, чтобы подключать сетевые каталоги непосредственно в локальное дерево каталогов, пользуясь при этом CIFS (общей межсетевой файловой системой). Разумеется, это нужно делать лишь в тех случаях, когда каталог доступен в течение достаточно долгого времени, то есть обслуживается стабильным сервером.

Для того чтобы подключить к системе внешний каталог, введите одну из двух следующих команд (в зависимости от того, предоставляются ли каталоги Windows на базе имени пользователя):

```
root# mount -t cifs //venus/myshare /media/
root# mount -t cifs -o username=имя //venus/myshare /media/winshare
```

Таким образом, каталог myshare подключается к файловой системе Linux на компьютере venus. Теперь его данные предоставлены в каталоге /media/winshare. Этот каталог должен существовать уже перед выполнением команды mount. При выполнении команд система запросит у вас пароль. Но пароль можно указать и сразу:

```
root# mount -t cifs -o username=имя,password=xxxxxxxx \
      //venus/myshare /media/winshare
```

Если сетевой каталог всегда должен автоматически подключаться к дереву каталогов, добавьте в /etc/fstab соответствующую запись, например:

```
# в /etc/fstab
//venus/myshare /media/winshare cifs defaults 0 0
```

Команда smbclient

Любители текстовых команд также могут просматривать сетевые каталоги с помощью команды smbclient. Она пусть и не очень удобна, но часто помогает отследить источник проблемы, возникающей в Samba.

Команда smbclient -L localhost отображает все ресурсы локального компьютера, предоставленные в общее пользование, перечисляет все видимые рабочие группы локального компьютера и указывает, какой компьютер в какой группе является основным (ведущим). Если ресурс не защищен паролем, то в ответ на запрос пароля просто нажимайте клавишу Enter. Если на локальном компьютере не работает сервер Samba, то вместо localhost укажите имя этого компьютера.

Если smbclient выдает сообщение о невозможности входа (access denied), это обычно означает, что имя пользователя или рабочей группы на вашем компьютере Linux не совпадает с аналогичным именем на компьютере Windows или на сервере Samba. Простейшее решение — сообщить эту информацию smbclient как дополнительные параметры:

```
user$ smbclient -U имя_пользователя -W рабочая_группа -L venus
```

Вы можете также применять smbclient интерактивно, для передачи файлов. Для этого сначала нужно установить соединение между открываемым каталогом и компьютером Windows или сервером Samba. Каталог указывается в привычной для Windows манере — *имя_сервера**название_каталога*. Чтобы оболочка не пыталась обработать символы \, их нужно удваивать. Затем, как и при использовании FTP, вы можете просматривать каталоги командой ls, командой cd переходить в другой каталог, с помощью get переносить данные на локальный компьютер (скачивать), а с помощью put сохранять данные на внешнем компьютере (закачивать). Список важнейших команд выводится с помощью help. Подробное описание команд дается в man smbclient.

```
user$ smbclient -U name -W wgname \\\\venus\\myshare
Password: xxxxxx
Domain=[wgname] OS=[Unix] Server=[Samba 3.0.28a]
smb: > ls
  .           D        0  Thu  Sep   7  17:38:02  2008
  ..          D        0  Thu  Sep   7  17:38:02  2008
  data        D        0  Wed  Apr   5  18:17:11  2008
  file.xy    AR      226  Sat  Dec  14  00:00:00  2008
```

Клиенты Windows

В старых версиях Windows все серверы Samba отображались непосредственно при просмотре сети и ресурсов в Проводнике или в окне для выбора файлов. На то, чтобы все компьютеры приняли изменения, сделанные в сети, может уйти достаточно много времени. Часто заново установленный и сконфигурированный сервер Samba становится виден в сети только через несколько минут. Лучше и надежнее всего в таком случае просто перезапустить компьютер с Windows.

LLTD. К сожалению, Windows Vista, Windows 7 и другие современные версии этой системы не распознают в локальной сети ни сервер Samba, ни старые серверы Windows. Виноват в этом новый протокол, применяемый для обмена сетевыми данными, а именно Link Layer Topology Discovery[1] (кратко — LLTD). Этот протокол очень неплох, так как работает значительно быстрее своих предшественников. К сожалению, он пока не поддерживается ни Samba, ни более ранними версиями Windows. Более подробная информация о LLTD содержится на следующих сайтах:

- http://support.microsoft.com/kb/922120;
- http://www.microsoft.com/whdc/rally/rallyLltd.mspx.

К счастью, доступ к сетевым устройствам, не совместимым с LLTD, возможен и без их автоматического распознавания. Просто задайте в Проводник Windows и укажите имя компьютера в форме `\\имя` вручную.

Разумеется, более красиво было бы обеспечить совместимость сервера Linux с LLTD. Как ни странно, Microsoft разработала программу специально для этой цели и бесплатно предоставила ее исходный код. К сожалению, этот код не отвечает лицензии свободного ПО, поэтому данная программа по умолчанию не поставляется ни с одним из дистрибутивов Linux.

Пока непонятно, появится ли когда-нибудь версия LLTD, отвечающая лицензии свободного ПО.

20.9. Основы CUPS

Для обработки и буферизации заданий вывода информации на печать и для преобразования данных печати в формат принтера в Linux применяется программный пакет CUPS (Common UNIX Printing System). Он используется почти во всех дистрибутивах, о нем и пойдет речь в этом разделе.

Возможно, вас удивляет то, что о такой элементарной теме, как печать, я решил поговорить только в конце книги. Причина в том, что управление печатью в Linux происходит с помощью сетевой службы. Любой принтер, настроенный для работы в Linux, при соответствующей конфигурации также может использоваться для печати другими компьютерами, расположенными в локальной сети.

Работать с локальным принтером несложно, у вас это получится даже без прочтения данного раздела. Информация, содержащаяся далее, адресована в первую очередь тем пользователям, которые хотят понять базовые концепции, на которых построена система печати в Linux.

---

[1] Обнаружение топологии уровня связи (англ.).

Недоступные принтеры. Если какой-либо принтер временно недоступен (например, он может быть выключен), то CUPS переводит задачу *в режим ожидания*. Иначе говоря, CUPS запоминает, что в настоящее время принтером пользоваться нельзя. Проблема заключается в том, что если позже принтер будет включен (или подключен к сети), то CUPS может «не заметить», что принтер снова доступен. Вам придется специально возобновить взаимодействие системы и принтера. Такая возможность предусмотрена во всех пользовательских интерфейсах CUPS. Кроме того, это можно сделать с помощью команды `cupsenable` *`имя_принтера`* или `lpadmin -E -p` *`имя_принтера`*.

Чтобы исключить возможность возникновения такой проблемы, добавьте в файл `/etc/cups/printers.conf` при описании принтера строку `ErrorPolicy retry-job`. Все больше конфигурационных инструментов CUPS используют эту настройку по умолчанию.

Ход печати

PostScript. Вся философия печатной работы в Linux/UNIX основана на принтерах PostScript. PostScrip — это язык программирования, предназначенный для описания содержимого страниц. Принтеры, приспособленные к работе с PostScript, ожидают данных именно в этом формате. Почти все программы Linux, в которых имеется функция вывода на печать, посылают принтеру данные в формате PostScript.

Преимущество PostScript перед другими форматами заключается в том, что при описании используется векторный формат, то есть качество остается хорошим при любом разрешении. Один и тот же файл PostScript будет отпечатан тем четче, чем большее разрешение обеспечивает принтер, поэтому рассматриваемый формат занимает лидирующее место не только в UNIX, но и в современном печатном деле вообще.

Устройства для печати

Проще всего распечатывать информацию, имея принтер PostScript. Пользователь root посылает файл PostScript командой `cp` на устройство того интерфейса, к которому подключен принтер.

```
root# cp файл.ps /dev/lp0        (Параллельный интерфейс)
root# cp файл.ps /dev/ttyS0      (Последовательный интерфейс)
root# cp файл.ps /dev/usb/lp0    (USB-интерфейс)
```

Спулинг печати

Как правило, информацию должен выводить на печать не только администратор, но и обычные пользователи, причем работающие не только на локальном компьютере, к которому подключен принтер, но и на любом компьютере локальной сети. И никто из этих пользователей не хотел бы возиться с именами устройств, хотя обычным пользователям доступ к файлам устройств обычно закрыт. Решением проблем такого рода занимаются так называемые системы спулинга печати (spooling systems). Они решают несколько задач:

- предоставляют простые в использовании команды, которые дают возможность распечатывать информацию, указывая не название устройства, а просто имя принтера;

- при соответствующей конфигурации такие программы позволяют выводить файлы на печать всем пользователям (в том числе работающим в локальной сети);
- они позволяют подключать к одному компьютеру несколько принтеров и управлять этими устройствами;
- если одновременно поступает несколько заданий печати, эти задания буферизуются, образуя так называемые очереди печати (print queues), и выполняются по мере того, как принтер оказывается свободен;
- кроме того, системы спулинга печати способны выполнять некоторые дополнительные функции, например протоколировать, с какого компьютера сколько информации распечатывается.

Наиболее современной и популярной системой спулинга в Linux является CUPS. Раньше вместо CUPS применялись, например, BSD-LPD или LPRng. Независимо от того, какую систему спулинга вы применяете, команда для распечатки файла всегда выглядит одинаково:

```
user$ lpr -Pимя файл
```

Здесь *имя* — это имя принтера (точнее говоря, имя очереди печати). Если не указать параметр `-P`, то печать происходит на принтере, указанном по умолчанию.

Фильтр печати (Ghostscript)

До сих пор мы предполагали, что пользователь работает с принтером PostScript. Но на практике гораздо чаще встречаются такие принтеры, которые не поддерживают PostScript. Чтобы в Linux работали и они, потребуется преобразовать файл из PostScript в формат конкретного принтера. Внутри системы для этого используется программа Ghostscript, или `gs` (см. раздел 4.5).

За вызов `gs` отвечает так называемый фильтр. Это программа (сценарий), обрабатывает данные ввода и выдает данные вывода. В частности, для обеспечения процесса печати фильтр должен передавать `gs` правильные параметры (имя модели принтера, нужное разрешение, нужный размер страницы и т. д). Программа постранично преобразует данные PostScript в растровый формат и передает их дальше — вместе с командами печати, полученными от принтера.

При работе Ghostscript обращается к внешнему драйверу принтера. Важнейший проект по разработке драйверов для Linux называется Gutenprint (ранее — Gimp-print): http://gutenprint.sourceforge.net/.

Фильтр документов

В настоящее время практически все файлы распечатываются в формате PostScript, но иногда бывает нужно распечатать и обычный текстовый или графический файл. Разумеется, текстовый файл можно загрузить в редактор, из которого файл будет отправлен на печать в формате PostScript. Графические файлы также можно преобразовать в PostScript с помощью специального конвертера.

Еще удобнее было бы просто выполнять для таких файлов команду `lpr «файл»`. Чтобы этот механизм работал, система спулинга пытается распознать тип файла, выводимого на печать.

Если это удается сделать и формат файла не PostScript, то файл преобразуется в PostScript с помощью специальных программ. Вызов таких команд преобразования производится через сценарий-фильтр.

Все вместе

Итак, вы подключили к компьютеру струйный принтер (не PostScript) и правильно сконфигурировали его. Имя принтера — pluto. Теперь вы хотите распечатать графический файл `mypicture.png` и выполняете следующую команду:

```
user$ lpr -Ppluto mypicture.png
```

После этого происходят следующие операции:

- `lpr` передает файл в спулинг-систему CUPS;
- CUPS передает файл на фильтр;
- фильтр распознает тип файла (PNG) и преобразует растровый рисунок в формат PostScript;
- данные PostScript передаются Ghostscript, который преобразует их в тот формат, с которым работает принтер pluto;
- после того как принтер pluto напечатает все находящиеся в очереди файлы, он распечатает и изображение `mypicture.png`.

Внутренняя организация CUPS

Как и большинство других сетевых функций, CUPS является фоновым процессом (демоном). Демон печати `cupsd` запускается системой Init-V (см. раздел 4.5).

Конфигурационные файлы

В старых системах печати почти вся конфигурация систем печати осуществлялась в файле `/etc/printcap`. В CUPS этот файл не играет практически никакой роли. Ради обеспечения совместимости он все еще присутствует в системе, но в нем содержится только актуальный список очередей печати (и никаких дополнительных параметров). Вся конфигурация CUPS происходит в файлах каталога `/etc/cups`. В табл. 20.1 перечислены важнейшие файлы.

Таблица 20.1. Конфигурационные файлы в /etc/cups

Файл	Содержание
classes.conf	Определение всех классов
cupsd.conf	Центральный файл конфигурации CUPS
lpoptions	Изменения по сравнению с базовой конфигурацией
mime.convs	Фильтр, применяемый для обработки файлов различных типов
mime.types	Типы файлов для преобразования в PostScript
printers.conf	Определение всех принтеров
ppd/*имя*.ppd	Конфигурация очереди печати *имя*
~/.cups/lpoptions	Персональные настройки (KDE)

В cupsd.conf настраиваются различные установочные каталоги, определяются порт демона CUPS для протокола печати через Интернет (IPP), параметры просмотра принтеров, параметры безопасности, права доступа для клиентов в сети (allow/deny) и т. д.

В каталоге /etc/cups/ppd для каждого имени принтера, приведенного в printers.conf, содержится соответствующий файл PPD. В нем сохраняются все параметры печати (модель принтера и драйвера, такие настройки, как размер бумаги и разрешение и т. д.).

Когда администратор изменяет параметры печати или настройки (размер бумаги, разрешение печати, вертикальный или горизонтальный формат и т. д.), эти изменения сохраняются в файле lpoptions. Изменения действуют для всех пользователей, которые сами не внесли на своем компьютере других изменений. Пользовательские изменения сохраняются в ~/.cups/lpoptions.

В mime.types хранится список всех типов документов, автоматически распознаваемых CUPS и конвертируемых в формат PostScript. В mime.convs указывается, какой фильтр должен применяться (фильтры должны находиться в /usr/lib/cups/filter и представлять собой исполняемые файлы).

ВНИМАНИЕ

Система CUPS очень сложна, поэтому используйте при конфигурации специально предназначенные для этого инструменты. Вручную изменять конфигурацию CUPS могут только профессионалы высшего класса. Информация, которую я сообщил в этом разделе, совсем не является исчерпывающей! Более подробно конфигурация CUPS рассмотрена на следующих сайтах: http://www.cups.org/ и http://localhost:631/help/.

Файлы PPD (описание принтеров PostScript)

Для CUPS каждый принтер является принтером PostScript. Детали, характерные для данного языка — размер полей, разрешение принтера, команды для вызова различных дополнительных функций (например, подача бумаги), особенности (дуплексная печать) и т. д., сохраняются в файлах PPD (PostScript Printer Definition). Формат PPD был разработан компанией Adobe и также применяется на компьютерах Apple и Windows.

Поскольку на самом деле далеко не каждый принтер является устройством PostScript, в файлах CUPS-PPD в виде комментария также содержатся нужная команда GhostScript и все параметры, необходимые для того, чтобы gs могла преобразовать файл PostScript в формат принтера. Далее приведен фрагмент файла PPD для струйного принтера HP DeskJet 6980:

```
*PPD-Adobe: "4.3"
...
*Manufacturer: "HP"
*ModelName: "HP Deskjet 6980 Series hpijs"
*FoomaticIDs: "HP-DeskJet_5650 hpijs"
*FoomaticRIPCommandLine: "gs -q -dBATCH -dPARANOIDSAFER -dQUIET -dNOPAUSE
  -sDEVICE=ijs -sIjsServer=hpijs%A%B%C -dIjsUseOutputFD%Z -sOutputFile=- -"
...
```

Эта информация получена из базы данных ppds.dat, в которой содержатся все известные CUPS записи PPD. Двоичный файл ppds.dat может, в зависимости от

дистрибутива, находиться, например, в каталоге /var/cache/cups. Если вашего принтера в этой базе данных нет и вам не удастся найти совместимую модель, попробуйте найти подходящий PPD-файл в Интернете.

При распечатке файла CUPS сначала преобразует его в формат PostScript. Затем CUPS извлекает из PPD-файла параметр GhostScript для нужного принтера, вызывает gs, которая, в свою очередь, преобразует PostScript в формат нужного принтера. Полученные в итоге файлы отправляются на устройство печати.

HPLIP

Компания Hewlett-Packard в рамках проекта *HP Linux Imaging and Printing* (кратко — HPLIP) сама разрабатывает свободные драйверы для многих принтеров, сканеров и многофункциональных устройств. В качестве лицензии обычно применяется GPL, менее часто — MIT или BSD. Такая активная поддержка свободного ПО — достойный пример для всей компьютерной индустрии. Поскольку многие принтеры HP совместимы с CUPS и без HPLIP, использовать функции HPLIP обычно необязательно. Более подробно о HPLIP рассказано на следующем сайте: http://hplipopensource.com/hplip-web/.

Для HPLIP существует графический пользовательский интерфейс hplip-toolbox, который во многих дистрибутивах находится в отдельном пакете (например, hplip-gui в Ubuntu) и может быть установлен дополнительно. Программа самостоятельно распознает подключенные устройства HP, помогает при их конфигурации и применении. Интерфейс hp-toolbox может в числе прочего отображать количество чернил в картридже для многих струйных принтеров HP — такая функция не предусмотрена в самой системе CUPS.

IPP

CUPS поддерживает IPP (протокол печати через Интернет). Этот протокол намного упрощает работу с принтерами в сети из компьютеров Linux (см. также раздел 20.10). IPP поддерживается во всех распространенных операционных системах. Подробная информация об IPP дается на следующем сайте: http://www.pwg.org/ipp/.

Для совместимости со старыми системами печати в CUPS также предоставляется традиционный демон lpd системы BSD-LPD. Для этого при необходимости запускается программа cups-lpd с помощью xinetd или другого демона службы Интернета. Конфигурация xinetd описана в man cups-lpd.

Спулинг

Все файлы, отосланные на печать, буферизуются в каталоге /var/spool/cups/*, пока печать не завершится. Следите за тем, чтобы данные спула не потерялись при перезапуске Linux. После перезапуска cupsd проверяет, не осталось ли файлов, ожидающих вывода на печать, и снова пытается передать их на принтер.

Библиотека TCP-Wrapper

Как правило, доступ к CUPS регулируется только в /etc/cups/cupsd.conf. Но CUPS может быть скомпилирован и так, что дополнительно применяется библиотека

TCP-Wrapper, например в SUSE. Чтобы проверить, так ли это, выполните команду `ldd`:

```
user$ ldd /usr/sbin/cupsd | grep wrap
...
libwrap.so.0 => /lib64/libwrap.so.0 (0x00007fa6e5c6a000)
```

В этом случае CUPS будет работать лишь при условии, что это не запрещено в `/etc/hosts.deny` и прямо разрешено в `/etc/hosts.allow` (по умолчанию). Но если вы сами изменяете оба файла, не забывайте о CUPS (запись `cupsd`). Конфигурация `/etc/hosts.allow` и `hosts.deny` подробно описана в разделе 18.2.

Веб-интерфейс CUPS

В принципе конфигурационные файлы CUPS можно изменять в текстовом редакторе. Что касается некоторых базовых настроек, они, может быть, и практичны, но я не рекомендую этим заниматься, так как процесс очень сложен. Лучше использовать конфигурационные инструменты для локального компьютера (то есть графический пользовательский интерфейс) или веб-интерфейс CUPS (рис. 20.4).

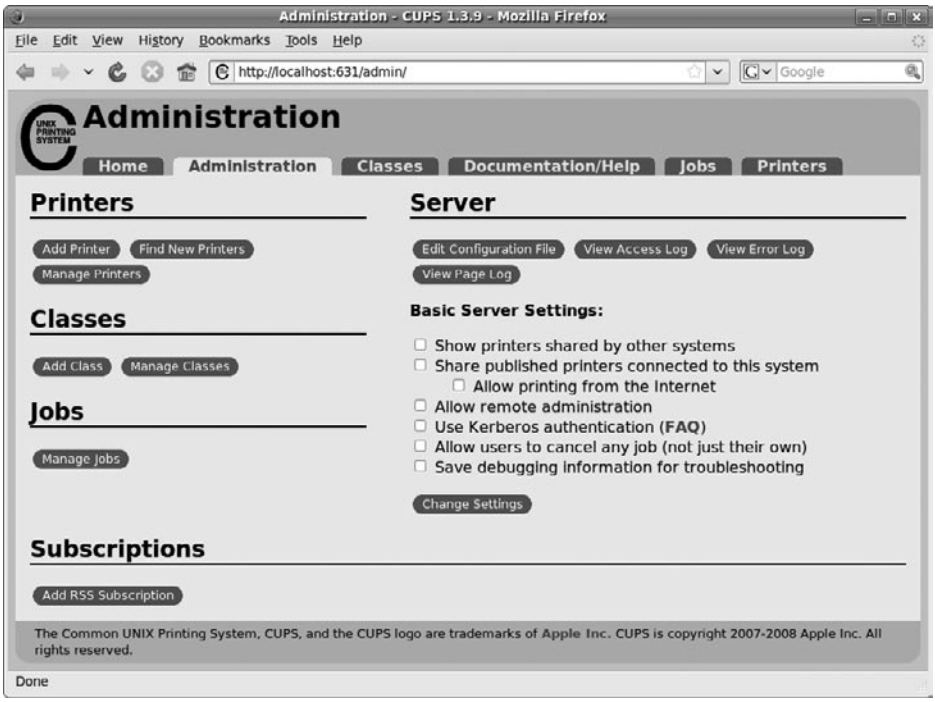

Рис. 20.4. Конфигурация CUPS в браузере

По причинам, связанным с безопасностью, такой интерфейс используется только на локальном компьютере. Стартовая страница открывается по адресу http://localhost:631.

Если у вас на сервере не установлен графический пользовательский интерфейс, можете воспользоваться текстовым браузером (например, lynx). Но более удобно было бы изменить cups.conf так, чтобы доступ к CUPS также был возможен с другого компьютера, находящегося в локальной сети. Для обеспечения большей безопасности заблокируйте выход из локальной сети через порт 631 брандмауэром:

```
# Изменения в /etc/cups/cups.conf
# Порт 631 заменяет настройку 'Listen localhost:631'
Port 631
<Location />
  ...
  Allow @LOCAL
</Location>
<Location /admin>
  ...
  Allow @LOCAL
</Location>
<Location /admin/conf>
  ...
  Allow @LOCAL
</Location>
```

Чтобы измененные настройки вступили в силу, CUPS должен заново считать конфигурацию:

```
root# /etc/init.d/cupsys reload
```

Для работы с административной частью веб-интерфейса нужно войти в систему (с логином для Linux и соответствующим паролем). Теперь вы можете настраивать новые принтеры, управлять задачами печати и т. д.

Администрирование CUPS с помощью команд

Как правило, при работе с принтером используются соответствующие окна тех или иных программ (OpenOffice, Firefox и т. д.), а для управления заданиями печати — соответствующие инструменты Gnome или KDE. Если вы любите работать с командной строкой, можете воспользоваться различными командами, позволяющими печатать файлы и управлять задачами принтера. Эти команды полезны в первую очередь в тех случаях, когда вы хотите автоматизировать выполнение задач принтера с помощью сценариев.

Команды lpr, lpq, lprm и lpc предоставляются не только в CUPS, но и в BSD-LPD и LPRng. Это своего рода общий знаменатель для всех систем спулинга. Но обратите внимание и на то, что в разных системах различные параметры различаются.

Команда lpr

С помощью команды lpr файл распечатывается. Если у вас настроено несколько принтеров, укажите параметром -P без пробелов название очереди печати. На принтере, заданном по умолчанию, параметр -P можно не использовать.

```
user$ lpr -Pимя файл
```

Если файл, предназначенный для печати, уже преобразован в формат конкретного принтера, сообщите `lpr` дополнительный параметр `-l`. Тогда команда будет действовать в обход обычной системы фильтрации и пошлет файлы на принтер в неизменном виде. Если на принтерах, совместимых с PostScript, нужно распечатать файлы PostScript, то подобная операция может сэкономить вам массу времени.

С помощью канала команду `lpr` можно использовать и для того, чтобы распечатать вывод другой команды. Следующая команда распечатывает список файлов, выводимый `ls`, на принтере, заданном по умолчанию:

```
user$ ls -l *.tex | lpr
```

Вместо `lpr` также можно использовать команду `lp` (ее синтаксис описан в `man 1 lp`). Эта команда должна облегчить жизнь тем, кто переходит к работе с CUPS после использования System-V (это другая система спулинга, но в Linux она не играет никакой роли).

Команды lpq, lprm

Все задания печати, которые нельзя выполнить сразу, буферизуются и помещаются в очередь (по одной очереди на каждый из настроенных принтеров). Чтобы просмотреть содержимое очереди, выполните команду `lpq -P`*Римя*.

Задачи принтера, которые вы поставили сами, можно снова удалить с помощью команды `lprm -P`*Римя* `id`, после чего нужно будет задать название очереди и ID-номер задачи. Чтобы узнать верный номер, выполните `lpq`.

```
user$ lpq
FS-1800+ is not ready
Priority   Owner      Job   File(s)        Total volume
1st        kofler     20    evince-print   17408 Bytes
2nd        kofler     21    evince-print   16384 Bytes
user$ lprm 20
user$ lprm 21
```

Команда lpc

Команда `lpc` позволяет более детально контролировать процесс печати. После ее выполнения вы попадаете в интерактивную рабочую среду, в которой можете выполнять команды `status`, `help` и т. д. Команда `topq` изменяет положение задачи печати в листе ожидания. В качестве параметров указывается имя принтера и номер задачи. Часть команд `lpc` (в том числе `topq`) может выполняться лишь администратором. Команды `exit`, `bye` или `quit` завершают работу `lpc`.

Команды lpstat, lpinfo, lpadmin, lpoptions

Команда `lpstat` выводит информацию обо всех принтерах, доступных для CUPS; `lpinfo` дает список всех доступных устройств печати и драйверов к ним. С помощью `lpadmin` производится настройка нового принтера или удаляется имеющаяся конфигурация принтера; `lpoptions` отображает параметры принтеров CUPS или изменяет эти параметры.

```
user$ lpoptions -o PageSize=A4
```

CUPS отключает принтеры, которые недоступны. Чтобы снова активизировать принтер, выполните одну из двух следующих команд:

```
user$ lpadmin -E имя_принтера
user$ accept имя_принтера
```

Если хотите специально отменить принтер, используйте команду `reject`:

```
user$ reject имя_принтера
```

20.10. CUPS: конфигурация принтера

При конфигурации принтера вам могут пригодиться инструменты, интегрированные в Gnome или KDE, средства конфигурации CUPS или специальные конфигурационные программы вашего дистрибутива. Программа `system-config-printer`, обычно поставляемая вместе с Gnome, первоначально разрабатывалась Red Hat. Пользователи SUSE могут применить модуль YaST Аппаратное обеспечение ▸ Принтер.

Драйверы принтера. При конфигурации принтера важнее всего то, совместим ли конкретный принтер с Linux, то есть с CUPS, GhostScript и соответствующими драйверами для принтера. Далее рассказано, насколько хорошо поддерживаются принтеры различных категорий.

- **Лазерные принтеры.** Многие лазерные принтеры совместимы с PostScript и HP (язык принтера PCL). Они оптимально подходят для работы с Linux. То же касается большинства сетевых моделей лазерных принтеров.
- **Принтеры GDI/Windows.** Эти лазерные принтеры (обычно недорогие) были разработаны специально для применения с Windows. Основная идея заключается в том, что программа Windows сначала подготавливает всю страницу, предназначенную для печати, на компьютере, и только потом передает ее на принтер. При компоновке страниц используется графический интерфейс GDI, разработанный Microsoft, поэтому GDI и присутствует в названии принтера. Проблема в том, что по формату, в котором данные передаются с компьютера на принтер, в свободном доступе обычно нет документации. По этой причине такие принтеры в Linux не поддерживаются.
- **Струйные принтеры и фотопринтеры.** Что касается струйных принтеров, то степень поддержки принтера в Linux зависит от конкретной модели. Принтеры HP поддерживаются достаточно хорошо в Linux. HP очень тесно сотрудничает с сообществом Linux, и для большинства моделей существуют свободно распространяемые драйверы. Но самые новые модели могут еще отсутствовать в базе данных принтеров CUPS. Со струйными принтерами других производителей проблем, как правило, больше.

Если вы собираетесь купить новый принтер, то посмотрите сначала следующий сайт, где очень подробно рассмотрена тема использования принтеров в Linux. Там же имеется серьезная база данных по моделям принтеров, поддерживаемым в этой операционной системе: http://www.linuxfoundation.org/en/OpenPrinting.

TurboPrint. Некоторые принтеры, для которых нет свободно распространяемых драйверов, поддерживаются платными драйверами, разработанными компанией

TurboPrint. Такие драйверы помогают повысить качество работы прежде всего с некоторыми фотопринтерами — по сравнению с драйверами, предлагаемыми в CUPS. Относительно недорогие драйверы, а также их бесплатные версии с ограниченной функциональностью предлагаются на следующем сайте: http://www.turboprint.de/.

Конфигурация локального принтера

Независимо от того, какой конфигурационной программой вы пользуетесь, старайтесь выполнять следующие рекомендации.

- Автоматическое распознавание принтера функционирует (если функционирует вообще) только тогда, когда принтер включен. Если вы работаете с несколькими принтерами, то на время конфигурации одного из принтеров выключайте все остальные.
- Чтобы выполнить конфигурацию вручную, нужно указать как минимум интерфейс (параллельный, последовательный, USB, сетевой и т. д.) и модель принтера. Модель выбирается из огромной базы данных.

  Если вашего принтера в этой базе данных нет, попробуйте найти совместимую модель. При работе с лазерным принтером, совместимым с PostScript и HP, можете выбрать в качестве производителя Generic, а затем указать стандарт (например, PostScript или PCL).

  Для большинства новых принтеров, которые еще не внесены в базу данных CUPS, в Интернете можно найти подходящие PPD-файлы. Такой файл можно загрузить в ходе конфигурации. Но учитывайте, что не всякий файл PPD совместим с CUPS либо для работы с принтером может потребоваться новейшая версия CUPS.
- Для работы со многими моделями принтеров на выбор предлагается несколько драйверов. На это может быть две причины. Во-первых, многие принтеры поддерживают по несколько стандартов. Во-вторых, в базе данных CUPS учтены драйверы из многих проектов (GhostScript, Gutenprint и т. д.). Как результат, для некоторых принтеров существует по несколько драйверов, созданных в рамках разных проектов.

  Если вы не уверены в том, какой драйвер лучше всего подойдет вам для решения поставленной задачи, создайте для принтера в системе несколько записей с разными именами. После этого вы сможете с удобством протестировать несколько драйверов. Качество печати зависит от того, что именно вы хотите печатать — текст, чертежи, фотографии и т. д. Кроме того, на качество печати влияют настройки параметров драйвера (например, разрешение в DPI).
- Почти все лазерные принтеры PostScript можно эксплуатировать в режиме совместимости так, чтобы они работали как лазерные принтеры HP-Laserjet. Таким образом, принтер PostScript в большинстве случаев можно сконфигурировать как принтер, совместимый с HP-Laserjet. При подобной конфигурации данные печати, поступающие с вашего компьютера, преобразуются в формат HP-Laserjet, а затем отправляются на принтер. Это кажется неудобным, но как раз при работе со старыми моделями принтеров получится значительно увеличить скорость работы.

Конфигурация сетевого принтера (клиентские настройки)

В этом подразделе даются некоторые советы по конфигурации принтера, который подключается к компьютеру через сеть. Для этого существует достаточно много вариантов, в зависимости от того, к работе с какими протоколами приспособлен сетевой принтер, и от того, как компьютер, не оборудованный сетевыми функциями, соединен с другим компьютером в локальной сети:

- IPP-принтер (управление с помощью Linux/UNIX/Mac OS X с CUPS, протокол печати через Интернет);
- принтер UNIX (управление с помощью Linux/UNIX, протокол LPD);
- принтер Windows (управление через сервер Windows или Samba);
- принтер Novell (управление через компьютер Novell Networks);
- Socket API (например, JetDirect от HP) на IP-порте 9100;
- AppSocket (например, Tektronix);
- протоколы конкретных производителей.

Использование IPP-принтера

Детали конфигурации зависят от того, по какому протоколу осуществляется обмен информацией. Процесс печати в сети построен проще всего, если с обеих сторон применяется CUPS, то есть протокол IPP. Такие принтеры видны со стороны клиента сразу после подключения, без дополнительной настройки, и с ними можно тут же начинать работу.

Команда `lpstat -v` возвращает список всех доступных принтеров. Следующая команда была выполнена на компьютере merkur. На нем установлен и настроен принтер DeskJet-5940. Кроме того, к компьютерам mars и saturn подключены еще два принтера с именами pluto и kyocera:

```
user@uranus$ lpstat -v
Device for DeskJet-5940: parallel:/dev/lp0
Device for pluto:        ipp://mars.sol:631/printers/pluto
Device for kyocera:      ipp://saturn.sol:631/printers/kyocera
```

Вы можете использовать все три принтера сразу с помощью команды `lpr -P`*имя*. Если в сети есть несколько принтеров с одинаковыми именами, это имя нужно задать в форме *имя_принтера*`@`*хост-имя*, то есть, например, `lpr -Plp@jupiter`.

ПРИМЕЧАНИЕ

Чтобы принтеры для работы с CUPS, подключенные к другим компьютерам, были видны на локальном компьютере, CUPS на внешних компьютерах должен иметь ту конфигурацию, которая описана в следующем разделе. Порт 631 не должен блокироваться брандмауэром!

В принципе IPP-принтер может иметь и такую конфигурацию, что использовать его для печати будет можно, но автоматически отображаться в сети он не будет. В таком случае сначала потребуется сконфигурировать принтер локального компьютера. Для этого выбираем тип принтера IPP Network Printer и в качестве адреса указываем ipp://хост-имя/printers/имя_принтера. Если внешние принтеры управляются через Linux/CUPS, укажите для настроек «производитель» и «модель» соответственно RAW и QUEUE. Это означает, что данные PostScript будут без изменений передаваться на внешний компьютер, который, в свою очередь, будет заниматься подготовкой данных для принтера.

Настройка конфигурации других сетевых принтеров

Если внешний сетевой принтер несовместим с IPP, то перед использованием для него нужно задать требуемую конфигурацию со стороны клиента. Для этого применяются те же программы, что и при конфигурации локального принтера, но в качестве типа принтера выбирается Network Printer. Дальнейшая конфигурация зависит от применяемого протокола.

- LPD (UNIX-LPD) — указывается хост-имя компьютера/принтера, а также номер очереди печати (в случае возникновения сомнений просто `lp` или `lp0`).
- SMB (Windows/Samba) — нужно указать хост-имя компьютера, название принтера, а также, возможно, имя пользователя и пароль. Прежде чем начать работу с принтером Windows, установите клиентский инструментарий Samba (если в дистрибутиве для работы и с серверами, и с клиентами, применяются одни и те же инструменты, то установите весь пакет Samba).
- Сокет-протокол или JetDirect — определяется хост-имя или IP-адрес принтера, а также номер порта (как правило, 9100).

В некоторых программах вышеуказанные данные должны вводиться в форме URI-адреса (табл. 20.2). Подробная информация о сетевом принтере, то есть протокол, логин и т. д., сохраняется в файле `/etc/cups/printers.conf`. В следующих строках показана конфигурация сетевого принтера, совместимого с JetDirect:

```
# в /etc/cups/printers.conf
<DefaultPrinter FS-1800+>
  Info Kyocera FS-1800+
  Location pluto
  DeviceURI socket://pluto:9100
  State Idle
  StateTime 1243572198
  Accepting Yes
  Shared Yes
  JobSheets none none
  QuotaPeriod 0
  PageLimit 0
  KLimit 0
  OpPolicy default
  ErrorPolicy retry-job
</Printer>
```

Важнейшее ключевое слово в `printers.conf` — это `DeviceURI`. За ним идет URI-адрес, из которого можно узнать протокол и сетевой адрес. В табл. 20.2 приведено несколько примеров таких адресов и показано, из чего они могут состоять.

Таблица 20.2. Адреса CUPS-URI

Адрес	Значение
usb:/dev/usb/lp0	Локальный USB-принтер
parallel:/dev/lp0	Локальный принтер, подключенный к параллельному интерфейсу

Адрес	Значение
serial:/dev/ttyS0?baud=115200	Локальный принтер, подключенный к последовательному интерфейсу
lpd://хост-имя/имя_принтера	Сетевой принтер LPD
socket://хост-имя:9100	Принтер с сокет-протоколом, например HP JetDirect
smb://хост-имя/имя_принтера	Принтер Windows
smb://рабочая_группа/хост-имя/принтер	Принтер Windows
smb://пользователь:xxx@wg/host/принтер	Принтер Windows
ipp://хост-имя/printers/имя_принтера	IPP-принтер (обычно дополнительная конфигурация не требуется)

Конфигурация сетевого принтера (серверные настройки)

Все больше принтеров (преимущественно лазерных) оснащаются сетевыми функциями. Такие принтеры просто подключаются к локальной сети и сразу после этого любой из компьютеров локальной сети может работать с таким принтером. Но в этом разделе мы рассмотрим иной вопрос: как можно использовать в сети локальный принтер *без* сетевого интерфейса? Вы, конечно, уже догадались, что с помощью CUPS. На компьютере, к которому подключен принтер, CUPS конфигурируется так, что сервер печати предоставляет всем остальным компьютерам сети доступ к локальному принтеру. После этого вы сможете работать с принтером по IPP, используя любую из распространенных операционных систем.

Конфигурация сервера

Как правило, принтер, работающий с CUPS, может использоваться только с локального компьютера. Чтобы и другие компьютеры сети могли работать с этим принтером, выберите в веб-интерфейсе вкладку Контроль, установите флажок Открыть доступ к принтерам, подключенным к этой системе, а затем нажмите кнопку Изменить настройки. Если вам больше нравится самостоятельно изменять конфигурационные файлы, внесите в `cupsd.conf` следующие настройки и перезапустите `cupsys`:

```
# Изменения в /cups/cupsd.conf
Listen 631
Browsing On
BrowseOrder allow,deny
BrowseAddress @LOCAL
<Location />
  ...
  Allow @LOCAL
</Location>
```

Настройка `Listen 631` означает, что CUPS обменивается информацией через сетевой порт 631 (`Listen localhost:631` в данном случае не подходит, так как открывает доступ к принтеру только с локального компьютера).

Благодаря BrowseAddress @LOCAL информация CUPS отправляется на все локальные сетевые интерфейсы (broadcast), но не на интерфейсы, обеспечивающие связь с Интернетом (PPP и т. д.). В качестве альтернативы вы можете с помощью BrowseAddress @IF(eth0) указать и строго определенный сетевой интерфейс. В файле cupsd.conf также предусмотрено несколько других ключевых слов Browse. Например, BrowseAllow и BrowseDeny определяют, какие компьютеры будут *принимать* информацию CUPS. По умолчанию никакие ограничения приема не устанавливаются, и случаи, в которых было бы необходимо или целесообразно менять эти или другие настройки Browse*Xxx*, встречаются редко.

Благодаря Allow @LOCAL другие компьютеры локальной сети действительно смогут работать с принтерами, предоставляемыми CUPS.

Клиентская конфигурация

Чтобы остальные компьютеры автоматически распознавали в сети внешний принтер CUPS, нужно установить флажок Показать доступные принтеры других систем. Он (возможно, сформулирован немного по-другому) может находиться в разных местах — в зависимости от конкретной конфигурационной программы.

В качестве альтернативы можно настроить принтер вручную. Для этого откройте окно конфигурации нового принтера, выберите тип устройства Протокол межсетевой печати и укажите хост-имя сервера. После этого в конфигурационном окне отобразится список всех принтеров, доступных на сервере.

Принтеры CUPS можно использовать и в Windows. Для этого установите в окне конфигурации принтера флажок Создать подключение с помощью принтера к Интернету или сети и укажите следующий адрес: http://mars.sol:631/printers/pluto.

В данном случае, конечно же, нужно заменить mars.sol хост-именем сервера CUPS, а pluto — названием принтера. В качестве драйвера по возможности укажите тот драйвер, с которым действительно работает принтер; если в Windows для вашего принтера нет специального драйвера (что маловероятно), можете использовать любой драйвер PostScript для принтера. В таком случае Windows посылает данные PostScript на сервер CUPS, который преобразует данные в формат для конкретного принтера. Хороший бесплатный драйвер — Adobe Universal PostScript Windows Driver, который не так давно можно было скачать здесь: http://www.adobe.com/support/downloads/detail.jsp?ftpID=1500.

Проблемы

Если на печать выводится беспорядочный текст либо в текст вкраплены нечитаемые графические символы, то причина этого обычно заключается в двойной переработке данных: сначала драйвер Windows преобразует распечатку в формат принтера. Затем данные попадают в CUPS и там форматируются повторно (в CUPS это называется «фильтруются»). Разумеется, что-то может пойти не так.

Чтобы избежать этого, используйте именно тот драйвер, который нужен для работы конкретного принтера в Windows! На сервере CUPS конфигурируйте принтер в виде так называемой *очереди без обработки*, которая просто переадресует полученные данные на принтер без дополнительной обработки. Для этого при конфигурации принтера выберите тип устройства RAW и модель RAW QUEUE.

Кроме того, в конце файла /etc/cups/mime.convs найдите приведенную ниже строку и удалите стоящий перед ней знак комментария. Таким образом, вы разрешите передачу двоичных данных печати на принтер без дополнительной обработки («фильтрации») в CUPS:

```
# В конце /etc/cups/mime.convs
application/octet-stream application/vnd.cups-raw 0 -
```

Конфигурация сетевого принтера Samba (серверные настройки)

Чтобы не предоставлять принтер в локальной сети непосредственно через CUPS, эту задачу можно решать с применением Samba (причем она, в свою очередь, будет обращаться к CUPS). В большинстве дистрибутивов Samba по умолчанию уже имеет соответствующую конфигурацию. В следующем листинге обобщены важнейшие строки из /etc/samba/smb.conf:

```
# Файл /etc/samba/smb.conf
...
# Использовать через Samba все принтеры CUPS
[printers]
  comment      = All Printers
  browseable   = no
  path         = /var/spool/samba
  printable    = yes
  guest ok     = no
  read only    = yes
  create mask  = 0700
```

За сам доступ к принтеру отвечает раздел [printers]. Благодаря browseable=no видны только принтеры, а каталог printers остается невидимым. Путь указывает, где находятся временные файлы печати.

Если вы хотите предоставить в общее пользование не все принтеры, а только один из них, используйте вместо раздела [printers] следующий код. В примере предполагается, что очередь печати данного принтера в /etc/printcap называется lp1, но на клиентах Samba должна быть видна под именем hp3:

```
# Файл etc/samba/smb.conf
...
# Доступ только к принтеру CUPS pluto под именем HP_pluto
[HP_pluto]
  printer      = pluto
  browseable   = no
  path         = /var/spool/samba/
  printable    = yes
  guest ok     = no
  read only    = yes
  create mask  = 0700
```

Samba позволяет предоставлять клиентам Windows драйверы для принтеров. Следующие строки входят в стандартную конфигурацию Samba и задают для этой цели каталог /var/lib/samba/printers:

```
# Файл /etc/samba/smb.conf
...
[print$]
  comment     = Printer Drivers
  path        = /var/lib/samba/printers
  browseable  = yes
  read only   = yes
  guest ok    = no
```

Возникает следующая проблема: каталог с драйверами принтера пуст. Приобретать и настраивать драйверы принтера в формате, который поддерживается во всех распространенных версия Windows, стоит лишь в том случае, когда принтер должен использоваться очень большим количеством клиентов с Windows. В других случаях проще вручную установить драйвер в Windows, воспользовавшись имеющейся в системе базой драйверов. Более подробно эта тема рассмотрена в man cupsaddsmb и на следующем сайте: http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/CUPS-printing.html.

20.11. NTP

Если несколько пользователей обращаются с различных компьютеров к одним и тем же файлам, важно, чтобы все эти пользователи работали по одинаковому времени. Но часы, встроенные в каждый компьютер, обычно не очень точны. *Сетевой протокол времени (NTP)* помогает синхронизировать работу компьютеров в локальной сети. Он обеспечивает выравнивание времени между несколькими компьютерами. В Интернете есть множество общедоступных серверов времени, на которых можно узнать точное время. Существуют две возможности использования сетевого протокола времени.

- Команда ntpdate один раз узнает точное время и настраивает часы компьютера. Если компьютер часто включается и выключается, этого вполне достаточно.
- На сервере, который может непрерывно работать в течение многих недель и месяцев, однократной настройки точного времени обычно недостаточно. Время компьютера постепенно будет все сильнее расходиться с точным временем. В таком случае используйте демон ntpd, который будет регулярно выходить на связь с другими серверами времени и постепенно исправлять локальное время. Одновременно ntpd может служить сервером времени для других компьютеров (например, для всех клиентов, находящихся в локальной сети).

  Даже если вы используете на компьютере ntpd, команда ntpdate очень удобна для первичного установления точного времени. Демон ntpd функционирует только тогда, когда начальное отклонение локального времени от точного не превышает одной минуты.

Более подробно вопрос управления датой и временем рассмотрен на следующих сайтах:

- http://www.ntp.org/;
- http://tldp.org/HOWTO/TimePrecision-HOWTO/;
- http://wwp.greenwichmeantime.com/.

Файл ntpd.conf

В любом дистрибутиве управление сервером синхронизации времени `ntpd` осуществляется в файле `/etc/ntp.conf`. Минимальная конфигурация выглядит так:

```
# Файл /etc/ntp.conf
driftfile /var/lib/ntp/ntp.drift
statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable
# узнавать точное время отсюда
server de.pool.ntp.org
server ch.pool.ntp.org
# По умолчанию запретить доступ к серверу
restrict -4 default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery
# Неограниченный доступ для localhost (включая конфигурацию)
restrict 127.0.0.1
restrict ::1
# Разрешить запросы времени в локальной сети 192.168.0.*
restrict 192.168.0.0 mask 255.255.255.0 nomodify notrap
```

Debian, Ubuntu

От дистрибутива зависит, будут ли запускаться программы `ntpdate` и `ntpd` и как именно. В Debian и Ubuntu `ntpdate` установлена по умолчанию и выполняется при каждом установлении соединения с сетью (сценарий `/etc/network/if-up.d/ntpdate`).

Если вы также хотите выполнять на компьютере `ntpd`, установите пакет `ntp`. Кроме того, добавьте в файл `/etc/ntp.conf` адрес близлежащего и легкодоступного сервера синхронизации времени. По умолчанию в этом файле внесен только один сервер, и обычно этого мало.

С помощью `ntpq -p` вы можете убедиться, что `ntpd` функционирует. В выводе команды самое важное значение имеет столбец `offset`: в нем указывается разница между локальным временем и временем различных эталонных часовых серверов в миллисекундах. Разница должна быть как можно меньше. Чтобы `ntpq -p` возвратила пригодные для работы результаты, `ntpd` должен поработать некоторое время (как минимум, несколько минут). Обратите внимание, что `ntpd` должен не только исправлять небольшие отклонения времени, но и время от времени ускорять или

замедлять ход часов, пока не будет достигнуто точное время. Таким образом, можно избежать резких изменений времени.

```
root# ntpq -p
remote           refid  st  t  when  poll  reach   delay   offset  jitter
==============================================================================
europium.canoni  ...     2  u     2    64      1  21.565  -117.64  0.002
www.alter-provi  ...     2  u     1    64      1  20.436  -118.56  0.002
```

Если отклонение времени превысит 1 секунду (в столбце `offset` будет значение более `1000`), настройте время вручную с помощью `ntpdate`. Для этого ненадолго остановите `ntpd` и сообщите `ntpdate` адрес общедоступного сервера времени:

```
root# /etc/init.d/ntpd stop
root# ntpdate de.pool.ntp.org
28 Jan 10:43:06 ntpdate[16752]: adjust time server 131.234.137.24 offset 3.010946 sec
root# /etc/init.d/ntpd start
```

Fedora, Red Hat

В Fedora и Red Hat при конфигурации NTP используется `system-config-date`. Если активизировать в этой программе NTP, то при старте компьютера будет выполняться сценарий Init-V, запускающий одноименный демон. С помощью `ntpq -p` вы можете убедиться, что все работает. Если начальное отклонение времени окажется слишком большим, ненадолго остановите `ntpd` и синхронизируйте локальное время с помощью `ntpdate`:

```
root# /etc/init.d/ntpd stop
root# /etc/init.d/ntpdate start
root# /etc/init.d/ntpd start
```

SUSE

В SUSE конфигурация NTP состоит из двух частей: в модуле YaST Система ▸ Дата и время можно настроить точное время через NTP (кнопка Изменить) — при этом один раз выполняется `ntpdate`. Чтобы настроить сервер синхронизации времени, запустите модуль YasT Сетевые службы ▸ Настройки NTP и установите флажок Запустить службу Ntp и выполнить начальную загрузку. Важно знать, сколько серверов синхронизации времени предусматривает та или иная конфигурация — один или несколько (по умолчанию используется только локальное время, а этого обычно недостаточно). Теперь `ntpd` будет запускаться сценарием Init-V `ntp`.

Алфавитный указатель

H

I

J

K

L

M

N

P

Q

R

S

T

U

V

W

Y

Z

А

Б

В

Г

Д

З

И

К

Л

М

Н

О

П

P

C

Т

У

Ф

Х

Ц

Ч

Ш

Э

Я

Кофлер М.

Linux. Полное руководство

Перевод с немецкого О. Сивченко

Заведующая редакцией	*К. Галицкая*
Ведущий редактор	*Е. Каляева*
Литературный редактор	*Н. Гринчик*
Художник	*К. Радзевич*
Корректор	*Т. Радецкая*
Верстка	*Г. Блинов*

ООО «Мир книг», 198206, Санкт-Петербург, Петергофское шоссе, 73, лит. А29.

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

Подписано в печать 31.03.11. Формат 70×100/16. Усл. п. л. 64,500. Тираж 2000. Заказ 63.36.

Отпечатано по технологии CtP в ООО «СЗПД».
188306, Гатчина, Железнодорожная ул., 45Б.

ИЗДАТЕЛЬСКИЙ ДОМ
ПИТЕР®
WWW.PITER.COM

ПРЕДСТАВИТЕЛЬСТВА ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР»
предлагают эксклюзивный ассортимент компьютерной, медицинской, психологической, экономической и популярной литературы

РОССИЯ

Санкт-Петербург м. «Выборгская», Б. Сампсониевский пр., д. 29а
тел./факс: (812) 703-73-73, 703-73-72; e-mail: sales@piter.com

Москва м. «Электрозаводская», Семеновская наб., д. 2/1, корп. 1, 6-й этаж
тел./факс: (495) 234-38-15, 974-34-50; e-mail: sales@msk.piter.com

Воронеж Ленинский пр., д. 169; тел./факс: (4732) 39-61-70
e-mail: piterctr@comch.ru

Екатеринбург ул. Бебеля, д. 11а; тел./факс: (343) 378-98-41, 378-98-42
e-mail: office@ekat.piter.com

Нижний Новгород ул. Совхозная, д. 13; тел.: (8312) 41-27-31
e-mail: office@nnov.piter.com

Новосибирск ул. Станционная, д. 36; тел.: (383) 363-01-14
факс: (383) 350-19-79; e-mail: sib@nsk.piter.com

Ростов-на-Дону ул. Ульяновская, д. 26; тел.: (863) 269-91-22, 269-91-30
e-mail: piter-ug@rostov.piter.com

Самара ул. Молодогвардейская, д. 33а; офис 223; тел.: (846) 277-89-79
e-mail: pitvolga@samtel.ru

УКРАИНА

Харьков ул. Суздальские ряды, д. 12, офис 10; тел.: (1038057) 751-10-02
758-41-45; факс: (1038057) 712-27-05; e-mail: piter@kharkov.piter.com

Киев Московский пр., д. 6, корп. 1, офис 33; тел.: (1038044) 490-35-69
факс: (1038044) 490-35-68; e-mail: office@kiev.piter.com

БЕЛАРУСЬ

Минск ул. Притыцкого, д. 34, офис 2; тел./факс: (1037517) 201-48-79, 201-48-81
e-mail: gv@minsk.piter.com

Ищем зарубежных партнеров или посредников, имеющих выход на зарубежный рынок.
Телефон для связи: **(812) 703-73-73**. **E-mail:** fuganov@piter.com

---

Издательский дом «Питер» приглашает к сотрудничеству авторов. Обращайтесь по телефонам: **Санкт-Петербург** – **(812) 703-73-72**, **Москва** – **(495) 974-34-50**

Заказ книг для вузов и библиотек по тел.: (812) 703-73-73.
Специальное предложение – e-mail: kozin@piter.com

Заказ книг по почте: на сайте **www.piter.com**; по тел.: (812) 703-73-74
по ICQ 413763617

ИЗДАТЕЛЬСКИЙ ДОМ ПИТЕР®
WWW.PITER.COM

УВАЖАЕМЫЕ ГОСПОДА!
КНИГИ ИЗДАТЕЛЬСКОГО ДОМА «ПИТЕР»
ВЫ МОЖЕТЕ ПРИОБРЕСТИ
ОПТОМ И В РОЗНИЦУ У НАШИХ
РЕГИОНАЛЬНЫХ ПАРТНЕРОВ.

ДАЛЬНИЙ ВОСТОК

Владивосток
«Приморский торговый дом книги»
тел./факс: (4232) 23-82-12
e-mail: bookbase@mail.primorye.ru

Хабаровск, «Деловая книга», ул. Путевая, д. 1а
тел.: (4212) 36-06-65, 33-95-31
e-mail: dkniga@mail.kht.ru

Хабаровск, «Книжный мир»
тел.: (4212) 32-85-51, факс: (4212) 32-82-50
e-mail: postmaster@worldbooks.kht.ru

Хабаровск, «Мирс»
тел.: (4212) 39-49-60
e-mail: zakaz@booksmirs.ru

ЕВРОПЕЙСКИЕ РЕГИОНЫ РОССИИ

Архангельск, «Дом книги», пл. Ленина, д. 3
тел.: (8182) 65-41-34, 65-38-79
e-mail: marketing@avfkniga.ru

Воронеж, «Амиталь», пл. Ленина, д. 4
тел.: (4732) 26-77-77
http://www.amital.ru

Калининград, «Вестер»,
сеть магазинов «Книги и книжечки»
тел./факс: (4012) 21-56-28, 6 5-65-68
e-mail: nshibkova@vester.ru
http://www.vester.ru

Самара, «Чакона», ТЦ «Фрегат»
Московское шоссе, д.15
тел.: (846) 331-22-33
e-mail: chaconne@chaccone.ru

Саратов, «Читающий Саратов»
пр. Революции, д. 58
тел.: (4732) 51-28-93, 47-00-81
e-mail: manager@kmsvrn.ru

СЕВЕРНЫЙ КАВКАЗ

Ессентуки, «Россы», ул. Октябрьская, 424
тел./факс: (87934) 6-93-09
e-mail: rossy@kmw.ru

СИБИРЬ

Иркутск, «ПродаЛитЪ»
тел.: (3952) 20-09-17, 24-17-77
e-mail: prodalit@irk.ru
http://www.prodalit.irk.ru

Иркутск, «Светлана»
тел./факс: (3952) 25-25-90
e-mail: kkcbooks@bk.ru
http://www.kkcbooks.ru

Красноярск, «Книжный мир»
пр. Мира, д. 86
тел./факс: (3912) 27-39-71
e-mail: book-world@public.krasnet.ru

Новосибирск, «Топ-книга»
тел.: (383) 336-10-26
факс: (383) 336-10-27
e-mail: office@top-kniga.ru
http://www.top-kniga.ru

ТАТАРСТАН

Казань, «Таис»,
сеть магазинов «Дом книги»
тел.: (843) 272-34-55
e-mail: tais@bancorp.ru

УРАЛ

Екатеринбург, ООО «Дом книги»
ул. Антона Валека, д. 12
тел./факс: (343) 358-18-98, 358-14-84
e-mail: domknigi@k66.ru

Екатеринбург, ТЦ «Люмна»
ул. Студенческая, д. 1в
тел./факс: (343) 228-10-70
e-mail: igm@lumna.ru
http://www.lumna.ru

Челябинск, ООО «ИнтерСервис ЛТД»
ул. Артиллерийская, д. 124
тел.: (351) 247-74-03, 247-74-09,
247-74-16
e-mail: zakup@intser.ru
http://www.fkniga.ru, www.intser.ru