Introduction to Linux. A Hands on Guide — Введение в Linux. Руководство по работе

younglinux.info/book/export/html/173

Machtelt Garrels
Garrels.be

Адрес оригинала: http://tille.garrels.be/training/tldp/index.html

<e-mail автора на сайте автора>

ISBN 90-808529-1-0

20100512

Примечание переводчика:

В некоторых местах перевод может хромать, т. к. не являюсь профессиональным переводчиком. Некоторые места, которые мне корректно перевести не удастся, будут помечены специально.

Автор данного руководства (Machtelt Garrels) на данный момент в известность о переводе не поставлена, т. к. нет уверенности, что перевод будет закончен.

Исходя из информации об авторских правах данного руководства осуществление перевода правомерно.

Предисловие

Оглавление

Зачем это руководство?

Многие люди продолжают верить в трудность изучения Linux или в то, что только специалисты могут понять, как работает система Linux. Несмотря на то, что доступно много свободной документации, и она широко распространена в Web, но часто путает, поскольку обычно ориентирована на опытных UNIX или Linux пользователей. В настоящее время, благодаря успехам в развитии, Linux набрала популярность как дома так и на работе. Целью этого руководства является показать людям всех возрастов, что Linux может быть простым и забавным, использоваться для всех видов задач.

Кому следует читать эту книгу?

Это руководство было создано как беглый обзор операционной системы Linux, оно ориентировано на новых пользователей как разведывательный тур и руководство по началу работы с упражнениями в конце каждой главы. Для более продвинутых слушателей оно может быть использовано в качестве настольных рекомендаций и коллекции базовых знаний, необходимых, чтобы приступить к администрированию системы и сети. Эта книга содержит множество примеров из реальной жизни на основе опыта автора как системного и сетевого администратора Linux, инструктора и консультанта. Надеемся, что эти уроки помогут вам получить более глубокое понимание системы Linux и вдохновят вас попробовать что-то по своему усмотрению.

Каждый кто хочет освоить "CLUE" (Command Line User Experience — навыки пользователя командной строки) с Linux (и UNIX вообще) найдет эту книгу полезной.

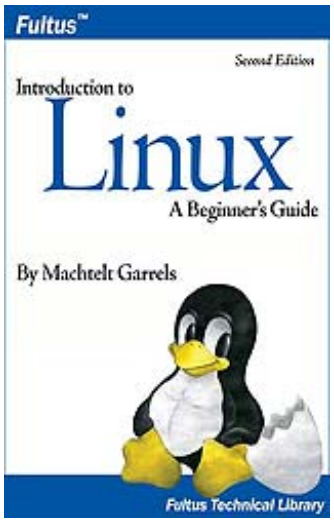
Новые версии и переводы

Этот документ опубликован в разделе руководств Linux Documentation Project на http://www.tldp.org/guides.html ; здесь вы также можете скачать версии в форматах PDF и PostScript.

Последнюю версию можно найти на http://tille.garrels.be/training/tldp/.

Третье издание этого руководства доступно в печатном виде от Fultus.com Books как как книга в мягкой обложке Print On Demand (POD). Fultus распространяет этот документ через Ingram и Baker & Taylor во многих книжных магазинах, в том числе Amazon.com, Amazon.co.uk, BarnesAndNoble.com и Google's Froogle всемирный торговый портал и Google Book Search.

Рисунок 1. Передняя обложка Введения в Linux



Руководство было переведено на хинди:

- Alok Kumar
- Dhananjay Sharma
- Kapil
- Puneet Goel
- Ravikant Yuyutsu

Andrea Montagner перевел руководство на итальянский язык.

Marian Vasile перевел руководство на румынский.

Andr? Nunes перевел руководство на португальский.

Перевод на испанский, Sinhala и Farsi продолжается.

История исправлений

Revision 1.30 20100512 MG
Новое развитие изображения, незначительные обновления, это версия для третьего издания в печатном виде от Fultus.

Revision 1.29 20100412 MG
Исправлены опечатки.

Revision 1.28 20090710 MG
Исправлены опечатки, обновления, заметка о ctime в inodes.

Revision 1.27 20080606 MG
Обновления.

Revision 1.26 20070919 MG
Комментарии читателей, лицензия.

Revision 1.25 20070511 MG

Комментарии читателей, незначительные обновления, E-mail этикет, обновление информации о наличии (спасибо Олегу).

Revision 1.24 2006-11-01 MG
Добавлен индекс терминов, подготовлено для второго печатного издания, добавлена информация по GPG и proxy.

Revision 1.23 2006-07-25 MG and FK
Обновления и исправления, снова удален app5, адаптирована лицензия, чтобы включить в Debian документы.

Revision 1.22 2006-04-06 MG
Глава 8 полностью пересмотрена, глава 10: уточнены примеры, добавлена информация по ifconfig и cygwin, исправлены сетевые приложения.

Revision 1.21 2006-03-14 MG
Добавлены упражнения в главу 11, исправить ошибки в newline, команда завершила обзор по главе 9, незначительные корректировки в главе 10.

Revision 1.20 2006-01-06 MG
Разделена глава 7: материал по аудио теперь в отдельной главе, chap11.xml. Малые изменения, обновления для команд как aptitude, больше на USB памяти, интернет-телефонии, корректировки читателей.

Revision 1.13 2004-04-27 MG
Последнее прочтение перед отправкой всего Fultus для распечатки. Добавлена ссылка на Fultus в разделе Новые Версии, обновлены разделы Условные обозначения и Структура. Незначительные изменения в главах 4, 5, 6 и 8, добавлена информация по rdesktop в главе 10, обновлен глоссарий, заменить ссылки на fileutils с coreutils, спасибо переводчикам с хинди.

Благодарности

Огромное спасибо всем людям, которые поделились своим опытом. И особенно бельгийским пользователям Linux за то, что выслушали меня каждый день и всегда быть щедры на комментарии.

Также особое внимание Tabatha Marshall за сделанную очень тщательную ревизию, проверку орфографии и стиля, а Eugene Crosser за обнаружение ошибок, которые мы двое просмотрели.

И спасибо всем читателям, которые сообщили мне о недостающих темах и которые помогли выделить последние ошибки, нечеткость определений и опечатки, потрудившись прислать мне свои замечания. Благодарность также людям, которые помогают мне сохранять это руководство современным, как Filipus Klutiero который сделал полный обзор в 2005 и 2006 и помогает мне внедрить руководство в коллекцию документов Debian, и Алексей Еременко, который отправил мне основу для главы 11.

Наконец, большое спасибо для добровольцев, которые в настоящее время переводят этот документ на французский, шведский, немецкий, персидский, хинди и многие другие. Это большая работа, которую нельзя недооценивать; я восхищаюсь вашим мужеством.

Обратная связь

Отсутствующая информации, недостающие ссылки, пропущенные буквы? Сообщите об этом по почте, указав этот документ:

<e-mail ищите на сайте автора>

Не забудьте сверится с последней версией сначала!

Информация об авторских правах

Что вам нужно?

Вам потребуется компьютер и носитель с дистрибутивом Linux. Большая часть этого руководства применима для всех дистрибутивов Linux - и UNIX в целом. Помимо времени, нет никаких дополнительных специфических требований.

Краткое HOWTO по установке содержит полезную информацию о том, как получить ПО Linux и установить его на компьютере. Требования к оборудованию и сосуществование с другими операционными системами, также обсуждались.

CD-образы могут быть загружены с различных ресурсов, см. *Приложение A, Куда идти дальше?*.

Интересная альтернатива для тех, кто не осмеливается сделать шаг инсталляции Linux на свои машины - это дистрибутивы Linux, которые можно запустить с CD, такие как дистрибутив Knoppix. Многие другие дистрибутивы, такие как Ubuntu, имеют пробную версия, которую вы можете записать на CD, или вы можете получить CD или USB-накопитель с образом на конференции, выставке или других профессиональных, полупрофессиональных и неофициальных встречах. Если вы пожелаете установить дистрибутив после его испытания, часто присутствует функция установки, которая позволяет легко скопировать данные с CD или USB-накопителя на ваш жесткий диск.

Условные обозначения, используемые в этом документе

В этом документе встречаются следующие типографские и общеупотребимые соглашения:

Таблица 1. Типографские и обиходные соглашения

Text type	Meaning
"Quoted text"	Цитаты людей, цитируемый компьютерный вывод.
`terminal view`	Буквальный компьютерный ввод и вывод захваченый из терминала, обычно предоставляются на светлом фоне серого цвета.
command	Имя команды, которая может быть введена в командной строке.
VARIABLE	Имя переменной или указатель на содержимое переменной, как в $VARNAME.
option	Опция команды как в «-a опция команды **ls**».
argument	Аргумент команды как в «читать **man** *ls*».
prompt	Приглашение пользователю, за которым, как правило, следует команда, которую вы вводите в окне терминала как в hilda@home> **ls** -l
command options *arguments*	Командный синопсис или обычный способ использования, на отдельной строке.
filename	Имя файла или каталога, например, "Измененить /usr/bin директорию".
Key	Клавиши для нажатия на клавиатуре, такие как "набрать **Q** для выхода".
|Button|	Графическая кнопка для щелчка, как кнопка |OK|.
Menu ? Choice	Выбор пункта в графическом меню, например: "Выберите **Help ? About Mozilla** в вашем браузере."
Terminology	Важный термин или концепция: "*Ядро* Linux является сердцем системы".
\	Обратная косая черта в картине терминала или командного синопсиса указывает на незавершенную строку. Иными словами, если вы видите длинную команду, которая разрезана на несколько строк, \ означает "Не нажимайте пока **Enter**!"

See Chapter 1, What is Linux?	Ссылка на связанные темы в этом руководстве.
See The author	Ссылка на внешний web-ресурс.

Используются следующие изображения:

 **Это примечание.** Оно содержит дополнительную информацию или ремарки.

 **Это предостережение.** Оно означает будьте осторожны.

 **Это предупреждение.** Будьте очень осторожны.

 **Это совет.** Советы и хитрости.

Структура этого документа

Это руководство является частью Linux Documentation Project и стремится быть основой для всех других материалов, которые вы можете взять из Project. Так оно предоставляет вам основные знания, потребность в которых возникает у тех, кто хочет начать работу с системой Linux, и в то же время пытается сознательно избежать повторного изобретения горячей воды. Поэтому, вы можете ожидать, что это книга будет снабжена неполными и полными ссылками на источники дополнительной информации о вашей системе в Интернете и в документации к ней.

Первая глава представляет собой введение в тему Linux, а в следующих двух обсуждаются безусловно основные команды. В главах 4 и 5 обсуждаются некоторые более продвинутые, но все еще базовые темы. Глава 6 необходима, чтобы продолжать имея основы, поскольку в ней рассматривается редактирование файлов; эта способность, которая необходима для перехода от новичка Linux к пользователю Linux. В следующих главах обсуждаются несколько более сложных тем, с которыми вам придется иметь дело в повседневном использовании Linux.

Все главы приводятся с упражнениями, которые будут проверкой вашей готовности к следующей главе.

Глава 1, Что такое Linux?: Что такое Linux, как он появился, преимущества и недостатки, что сулит будущее для Linux, кому следует его использовать, установка на компьютер.

Глава 2, Быстрый старт: Начало работы, подключение к системе, основные команды, где найти помощь.

Глава 3, О файлах и файловой системе: Файловая система, важные файлы и каталоги, управления файлами и каталогами, защищая ваших данных.

Глава 4, Процессы: Понимание и управления процессами, процедуры загрузки и завершения работы, отложенные задачи, повторяющиеся задачи.

Глава 5, I/O перенаправление: Что такое стандартный ввод, вывод и ошибка и как эти возможности

использовать из командной строки.

Глава 6, Текстовые редакторы: Почему Вы должны научиться работать с редактором, обсуждение наиболее распространенных редакторов.

Глава 7, Дом милый /home: Настройка графического, текстового и аудио окружения, настройки для пользователей Linux, для которых английский язык является не родным, советы добавления дополнительного программного обеспечения.

Глава 8, Принтеры и печать: Преобразование файлов в формат для печати, передача их на принтер, советы для решения проблем печати.

Глава 9, Основные методы резервного копирования: Подготовка данных для резервного копирования, обсуждение различных инструментов, удаленное резервное копирование.

Глава 10, Сеть: Обзор сетевых инструментов Linux и пользовательских приложений, с коротким обсуждением лежащих в основе служб программ-демонов и обеспечение безопасности сети.

Глава 11, Звук и Видео: В данной главе обсуждаются звук и видео, включая передачу голоса по IP и звукозапись.

Приложение A, Куда идти дальше?: Какие книги читать и сайты посещать, когда вы закончите чтение этой.

Приложение B, DOS в сравнении командами с Linux: Сравнение.

Приложение C, Особенности Shell: Если вы когда-нибудь застряните, эти таблицы могут быть выходом. Кроме того, хороший аргумент, когда ваш босс настаивает на том, что ВЫ должны использовать ЕГО любимую оболочку.

Глава 1. Что такое Linux?

Аннотация

Мы начнем с обзора того, каким образом Linux стала операционной системой, какой она является на сегодняшний день. Мы будем обсуждать ее развитие в прошлом и будущем, внимательнее посмотрим на преимущества и недостатки этой системы. Мы будем говорить о дистрибутивах, об Open Source (открытом исходном коде) в общих чертах и попытаемся объяснить кое-что о GNU.

Эта глава отвечает на такие вопросы как:

- Что такое Linux?
- Где и как Linux начался?
- Linux не та система, в которой все делается в текстовом режиме?
- У Linux есть будущее или это просто надувательство?
- Каковы преимущества использования Linux?
- Каковы недостатки?
- Какие виды Linux существуют и как выбрать тот, который мне подходит?
- Что такое движения Open Source и GNU?

История

UNIX

Чтобы понять популярность Linux, мы должны совершить путешествие во времени, на 30 лет назад ...

Вообразите компьютер размером с дом, даже стадионы. Хотя размеры этих компьютеров создавали серьезные проблемы, была одна вещь, из-за которой было еще хуже: все компьютеры имели разные операционные системы. Программное обеспечение всегда было подогнано для достижения конкретной цели, и ПО для одной системы не работало на другой. Возможность работать на одной системе автоматически не означало, что вы могли бы работать на другой. Это создавало трудности, как для пользователей, так и системных администраторов.

Кроме того, компьютеры были очень дорогими, также затраты совершались сразу после первоначального приобретения просто из-за того, что пользователям нужно было понять как работают компьютеры. Общая стоимость единицы вычислительной мощности была огромна.

Технологически мир был не достаточно продвинутым, поэтому пришлось жить с такими размерами и в течение следующего десятилетия. В 1969 году команда разработчиков в лабораториях Bell Labs начала работу над решением проблемы программного обеспечения, направленной на решение проблем с совместимостью. Они разработали новую операционную систему, которая была

1. простой и элегантной,
2. написанной на языке программирования Си, а не на ассемблере,
3. допускала возможность пересборки кода.

Разработчики Bell Labs назвали их проект "UNIX".

Возможность повторного использования кода было очень важным. До сих пор программный код специально разрабатывался для каждой конкретной компьютерной системы, которая имелась в продаже. Но UNIX была необходима только маленькая часть такого специального кода, который теперь принято называть ядром. Ядро – это единственная часть кода, которую необходимо адаптировать для каждой конкретной системы, и оно составляет основу системы UNIX. Операционная система и все другие функции были созданы вокруг этого ядра и написаны на языке программирования высокого уровня - C. Этот язык был специально разработан для создания системы UNIX. С помощью этого нового "метода" было гораздо легче разработать операционную систему, которая может работать на многих различных типах аппаратного обеспечения.

Поставщики программного обеспечения быстро приспособились, теперь они могли почти без усилий продать в 10 раз больше софта. Сложилось новое невероятное положение вещей: представьте, например, компьютеры разных производителей, соединенные в одной сети, или пользователей, работающих на разных системах и не нуждающихся в дополнительном обучении по использованию другого компьютера. UNIX сделал многое, чтобы помочь пользователям различными систем стать совместимыми.

В течение следующих двух десятилетий развитие UNIX продолжалось. Стало возможным делать больше различных вещей, и больше производителей оборудования и ПО добавляли в свои товары поддержку UNIX.

Первоначально UNIX использовался только в очень больших средах, таких как мэйнфреймы и мини-ЭВМ (заметим, что ПК - это "микро" компьютер). Вы следовало работать в университете, на правительство или в крупных финансовых корпорациях, чтобы получить доступ к системе UNIX.

Но были разработаны меньшие компьютеры, и в конце 80-х у многих людей появились домашние компьютеры. К этому времени было несколько версий UNIX, доступных для архитектуры PC, но ни одна из них не была действительно свободной и что более важно: они были ужасно медленными, поэтому большинство людей выбирали для своих домашних компьютеров MS DOS или Windows 3.1.

Linus и Linux

К началу 90-х домашние ПК наконец стали достаточно мощными, чтобы на них можно было запустить полномасштабный UNIX. Линус Торвальдс, молодой человек, изучающий компьютерную науку в университете Хельсинки, подумал, что было бы неплохо иметь какую-нибудь свободно доступную академическую версию UNIX, и сразу начал писать программный код.

Он начал задавать вопросы, искать ответы и решения, которые помогли бы ему получить UNIX на его компьютере. Ниже приводится одно из его первых сообщений в comp.os.minix, датированное 1991 годом:

```
От: torvalds@klaava.Helsinki.FI (Линус Бенедикт Торвальдс)
Новостная группа: comp.os.minix
Тема: Gcc-1.40 и вопросы POSIX
ID сообщения: <@ 1991Jul3.100050.9886 klaava.Helsinki.FI>
Дата: 3 июля 91 10:00:50 GMT
Привет нетлэндеры,
В связи с проектом, над которым я работаю (в Minix), я заинтересован в ясных
стандартах POSIX. Может ли кто-нибудь указать мне (предпочтительно)
машиночитаемый формат новейших требований POSIX? Не плохо бы
Ftp-сайты.
```

С самого начала целью Линуса было создание свободной системы, которая полностью совместима с оригинальным UNIX. Именно поэтому он просил стандарты POSIX, POSIX по-прежнему является стандартом для UNIX.

В те дни технология plug-and-play еще не была изобретена, но слишком много людей были заинтересованы в собственной системе UNIX, что это было лишь маленьким препятствием. С постоянно растущей скоростью новые драйверы становились доступными для всех видов нового оборудования. Почти сразу после того, как новое оборудование становилось доступным, кто-нибудь покупал его и испытывал на Linux (так постепенно была названа система), выпуская тем самым больше свободного кода для все более широкого спектра аппаратного обеспечения. Эти кодеры не останавливались на своих собственных компьютерах; каждый образец аппаратного обеспечения, который они могли найти, был пригоден для Linux.

Тогда тех людей называли "ботаниками" или "фриками", но это не имело значения для них, поскольку список поддерживаемого оборудования рос все больше и больше. Благодаря тем людям, Linux теперь идеальна не только для работы на новых ПК, но ее также выбирают для старого и экзотического оборудования, которое было бы бесполезно без существования Linux.

Через два года после сообщения Линуса было уже 12000 пользователей Linux. Проект, популярный среди любителей, постоянно рос все это время, не выходя за рамки стандарта POSIX. Все функции UNIX были добавлены в течение следующих нескольких лет, в результате чего на сегодняшний день Linux стала зрелой операционной системой. Linux является полным клоном UNIX, пригодным для использования на рабочих станциях, также как и на средних и высококлассных серверах. Сегодня много важных игроков на рынке аппаратного и программного обеспечения имеют собственную команду разработчиков Linux; у ваших местных продавцов, вы даже можете купить системы с предварительно установленным Linux с официальной поддержкой – даже тогда, когда все еще есть много аппаратного и программного обеспечения, которое не поддерживается.

Современное применение систем Linux

Сегодня Linux присоединился к рынку десктопов. Разработчики Linux первоначально были сосредоточены на сетях и сервисных программах, и офисные приложения стали последним барьером, который был снят. Мы не

хотим принимать то, что Microsoft управляет рынком десктопов, поэтому в течение последних нескольких лет были начаты многие альтернативные проекты с целью сделать Linux приемлемым выбором в качестве рабочей станции, обеспечения удобным интерфейсом пользователя и MS совместимыми офисными приложениями, такими как текстовые процессоры, электронные таблицы, презентации и тому подобное.

С точки зрения сервера, Linux хорошо известен как стабильная и надежная платформа, обеспечивающая базу данных и торговые услуги для таких компаний, как Amazon - известный книжный онлайн магазин, почтовая служба США, немецкая армия и многие другие. Особенно полюбили Linux в качестве межсетевого экрана, прокси-сервера и веб-сервера Интернет-провайдеры и провайдеры интернет-услуг, и вы найдете окно с Linux в пределах досягаемости каждого администратора системы UNIX, который ценит удобное управление станцией. Кластеры Linux-машин использовались в создании таких фильмов, как "Титаник", "Шрек" и другие. В почтовых отделениях они нервные центры, которые распределяют почту, в больших поисковых системах кластеры используются для выполнения поиска в Интернете. Это лишь немного из тысячи трудных работ, которые Linux выполняет изо дня в день по всему миру.

Стоит также отметить, что современный Linux работает ни только на рабочих станциях, средних и мощных серверах, но также на "гаджетах", таких как КПК, мобильные телефоны, a shipload of embedded applications и даже в экспериментальных наручных часах. Это делает Linux единственной операционной системой в мире, которая охватывает такой широкий спектр аппаратного обеспечения.

Интерфейс пользователя

Linux трудный?

Ответ на вопрос, является ли Linux трудным для изучения, зависит от человека, которого вы об этом спрашиваете. Опытные пользователи UNIX ничего не скажу, потому что Linux является идеальной операционной системой для опытных пользователей и программистов, ведь эта система была разработана и разрабатывается такими людьми.

Имеется все, что хороший программист может пожелать: компиляторы, библиотеки, инструменты для разработки и отладки. Эти пакеты поставляются с каждым стандартным дистрибутивом Linux. Компилятор с языка C входит бесплатно - в отличие от многих дистрибутивов UNIX, требующих лицензионные сборы на этот инструмент. Имеются полная документация и руководства, часто с включенными примерами; все это для того, чтобы помочь вам начать работу в кратчайшие сроки. Linux ощущается как UNIX, и переход между UNIX и Linux является обыденной вещью.

В первые дни существования Linux, для начала работы с системой было необходимо быть экспертом. Те, кто освоил Linux, чувствовали себя лучше, чем остальные "lusers", которые еще не увидели свет. Обычной практикой было сказать начинающему пользователю "RTFM" (читай руководства). Хотя руководства были в каждой системе, в них было трудно ориентироваться, и даже если кому-нибудь это удавалось, объяснения были в таких технических терминах, что новый пользователь становился слегка озадаченным от изучения системы.

Использующее Linux сообщество начинало понимать, что Linux никогда не будет важным игроком на рынке операционных систем, пока не наступят серьезные изменения в доступности системы.

Linux для неопытных пользователей

Такие компании, как RedHat, SuSE и Mandriva возникли, обеспечивая дистрибутивы Linux «упаковкой», подходящей для массового потребления. Они объединили множество графических пользовательских интерфейсов (GUI), разработанных сообществом, в целях облегчения управления программами и

сервисами. Сегодня, как у пользователя Linux, у Вас есть все средства знакомства с вашей системой изнутри, однако уже не обязательно обзаводиться этими знаниями, чтобы система соответствует вашим потребностям.

На сегодняшний день вы можете войти через графический интерфейс и запустить все необходимые приложения, не введя ни одного символа, однако у вас по-прежнему есть возможность доступа к ядру системы в случае необходимости. Благодаря своему строению, Linux позволяет пользователю «расти» в системе: она одинаково подходит начинающим и опытным пользователям. Новые пользователи не обязаны решать сложные задачи, в то время как опытные пользователи не обязаны работать так же, как они это делали, когда впервые начали изучение Linux.

Хотя развитие в сфере услуг продолжается, серьезные вещи делаются для пользователей ПК, которые обычно рассматриваются как группа, которая вероятно меньше всего знает, как работает система. Разработчики десктопных приложений затрачивают неимоверные усилия, чтобы создать самые красивые рабочие столы, которые вы когда-либо видели, или сделать вашу Linux-машину похожей на ваш бывший MS Windows или рабочую станцию Apple. Последние разработки также включают поддержку 3D ускорения и USB устройств, обновление систем и пакетов одним щелчком мыши и т. д. Все имеющиеся сервисы Linux старается представить в логической форме, понятной для обычных людей. Ниже приведен краткий список, содержащий некоторые замечательные примеры; на этих сайтах есть много скриншотов, которые дадут вам представление о том, как может выглядеть Linux на десктопе:

http://www.gnome.org
http://kde.org/screenshots/
http://www.openoffice.org
http://www.mozilla.org

У Linux есть будущее?

Open Source (открытое программное обеспечение)

Идея программного обеспечения с открытым исходным кодом достаточно проста: когда программисты могут читать, распространять и изменять код, то он будет доведен до совершенства. Люди могут изменить его, исправлять, отлаживать, они могут делать это на скорости, that dwarfs the performance of software developers at conventional companies. Это ПО будет более гибким и лучше по качеству, чем программное обеспечение, которое был разработано с использованием обычных способов, потому что больше людей испытали его в разнообразных условиях, по сравнению с тем, что может когда-либо себе позволить разработчик закрытого ПО.

Инициатива Open Source начала ясно прорисовываться в коммерческом мире, и очень медленно коммерческие производители начинают осознавать этот момент. Хотя многие ученые и технические специалисты убеждены уже 20 лет, что это путь, только теперь поставщики важных коммерческих приложений, таких как для Интернета, начинают понимать, что они могут получать выгоду от использования открытого исходного кода. Сейчас Linux вырос их прошлой стадии, когда он был почти исключительно академической системой, полезной только для горстки людей с техническим образованием. Теперь Linux предоставляет больше, чем операционная система: существует целая инфраструктура, поддерживающая ряд усилий для создания операционной системы, включая создание и тестирование программ для нее, внедрения всего самого важного для пользователей, предоставление технического обслуживания, обновления, поддержки, настройки, и т. д. Сегодня Linux готова принять вызов быстро меняющегося мира.

Пятнадцать лет опыта к Вашим услугам

Хотя Linux, вероятно, самая известная Open Source инициатива, есть еще один проект, который внес огромный вклад в популярность операционной системы Linux. Этот проект под названием SAMBA, его достижением является реверсный инжиниринг Server Message Block (SMB)/Common Internet File System (CIFS), протокола, используемого для передачи файлов и печати на работающих связанных ПК, изначально поддерживается MS Windows NT, OS/2 и Linux. Теперь пакеты доступны практически для любой системы, обеспечивается взаимосвязь решений в смешанных средах, использующих протоколы MS Windows: Windows-совместимые (вплоть до WinXP) файловые серверы и серверы печати.

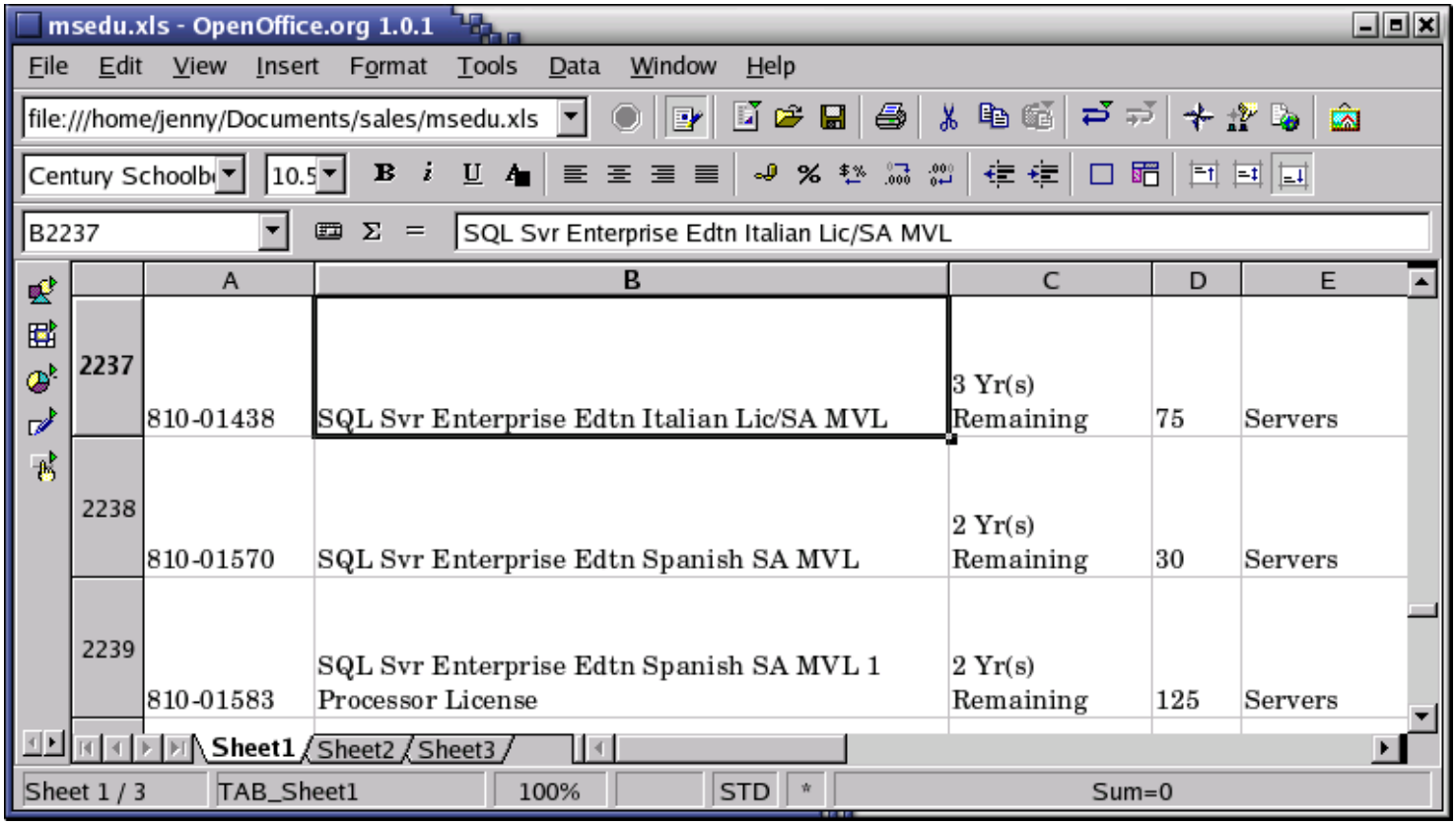
Может быть, даже более успешным, чем SAMBA является проект сервера Apache HTTP. Сервер работает на UNIX, Windows NT и многих других операционных системах. Изначально известный как “A PAtCHy server” («пятнистый сервер»), получившийся на основе существующего кода и серии “заплаток”, готовый код заслужил имя родного американского племени Apache, известного своими превосходными навыками в стратегии военного дела и неисчерпаемой выносливостью. Apache показал себя более быстрым, стабильным и функционально полным, чем многие другие веб-серверы. Apache работает на сайтах, которые посещают миллионы посетителей в день, и хотя разработчики не предоставляют никакой официальной поддержки, сообщество пользователей Apache предоставляет ответы на все ваши вопросы. Коммерческая поддержка в настоящее время оказывается рядом третьих сторон.

В категории офисных приложений доступен выбор комплектов клонов MS Office, начиная от частичной до полной реализации приложений, имеющихся на рабочих станциях MS Windows. Эти инициативы сделали многое, чтобы Linux стал приемлемым для рынка настольных систем, поскольку пользователям не нужно дополнительное обучение, чтобы научиться работать на новых системах. С десктопов приходит похвала обычных пользователей, но не только она, также все их специфические требования, которые все более сложные и взыскательные с каждым днем.

Сообщество Open Source, в основном состоящее из людей, которые вносят свой вклад более половины десятилетия, обеспечивает позицию Linux как важного игрока на рынке десктопов, так и вообще в информационных технологиях. Нанятые сотрудники и добровольцы так усердно работают, что Linux удается сохранять позиции на рынке. Чем больше пользователей, тем больше вопросов. Сообщество Open Source уверенно продолжает выдавать ответы, и следит за их качеством подозрительными глазами, в результате чего рождаются все большие стабильность и доступность.

Составление списка всего доступного программного обеспечения для Linux выходит за рамки данного руководства, поскольку имеются десятки тысяч пакетов. На протяжении этого курса мы представим вам наиболее распространенные пакеты, которые почти все есть в свободном доступе. Для того, чтобы убрать немного страха начинающего пользователя, вот вам скриншот одной из самых разыскиваемых программ. Вы сами можете видеть, что никаких усилий не надо, чтобы пользователи, которые переходят с Windows, чувствовать себя как дома:

Рисунок 1.1. OpenOffice MS-совместимая таблица



Свойства Linux

Плюсы Linux

Многие из преимуществ Linux - это следствие его происхождения, они глубоко укоренились еще в UNIX, за исключением, конечно, первого преимущества:

- Linux свободен:

Говорят, как и бесплатное пиво. Если вы не хотите абсолютно ничего тратить, то вам даже не обязательно платить за CD. Linux может совершенно бесплатно быть полностью загружен из Интернета. Нет регистрационных сборов, никаких расходов за каждого пользователя, зато есть бесплатные обновления и свободный доступ к исходному коду для случая, если вам захочется изменить поведение системы.

Прежде всего, Linux является свободным аналогично понятию свободы слова:
Обычно используемая лицензия - GNU Public License (GPL). В этой лицензии говорится, что каждый кто захочет что-то сделать, имеет право вносить изменения в Linux и в конечном итоге распространять измененную версию с одним условием: после изменения код должен остаться по-прежнему доступным. На практике, вы можете использовать образ ядра, например, чтобы добавить поддержку для машины телепортации или путешествия во времени и продать ваш новый код, при условии, что ваши клиенты смогут также иметь его копию.

- Linux переносим на любую аппаратную платформу:

Продавец, который хочет продать новый тип компьютера и не знает, какая ОС на его новой машине будет работать (скажем, процессор в вашем автомобиле или стиральной машине), может взять ядро Linux и сделать его работающим на данной аппаратуре; это все из-за того, что документация, связанная с деятельностью такого рода находится в свободном доступе.

- Linux был сделан так, чтобы постоянно работать:

Как и UNIX, система Linux предполагает работу без перезагрузки все время. Вот почему большинство задач выполняются в ночное время или запланированы на автоматическое выполнение в относительно спокойные минуты, это приводит к увеличению производительности в загруженные периоды и более сбалансированному использованию оборудования. Это свойство позволяет использовать Linux также в средах, где у людей нет времени или возможности контролировать системы круглосуточно.

- Linux является надежной и универсальной:

Модель безопасности, используемая в Linux, основана на идее безопасности UNIX, которая, как известно, надежна и проверенного качества. Но Linux подходит не только для использования в качестве форта от ударов противника через Интернет: его можно в равной степени приспособить к другим ситуациям, используя те же высокие стандарты безопасности. Ваша продвинутая машина или станция управления будут так же надежно защищены, как и ваш брандмауэр.

- Linux масштабируемый:

От карманного компьютера с 2 МБ памяти до кластеров с петабайтом памяти и сотнями узлов: добавляйте и удаляйте подходящие пакеты - Linux универсален. Вам больше не нужен суперкомпьютер, т.к. вы можете использовать Linux, чтобы сделать больше, используя «строительные» блоки, предоставленные системой. Если вым захочется сделать что-то не такое значительное, например, создать операционную систему для встроенного процессора или просто сделать возможным повторное использование старого 486, Linux также пригодится.

- У ОС Linux и у большинства его приложений очень короткий период отладки:

Поскольку Linux был разработан и протестирован тысячами людей, как ошибки, так и их решения обычно находились довольно быстро. Иногда получается, что проходит всего несколько часов между ее обнаружением и исправлением.

Минусы Linux

- Слишком много различных дистрибутивов:

"Quot capita, tot rationes", как римляне уже сказали: больше людей, больше мнений. На первый взгляд, количество дистрибутивов Linux может показаться страшным или смешным, в зависимости от вашей точки зрения. Но это также означает, что каждый найдет то, в чем он или она нуждаются. И вам не нужно быть экспертом, чтобы найти подходящий релиз.

Обычно каждый пользователь Linux (когда его спрашивают) скажет, что лучший дистрибутив - та версия, которую использует он. Так какую же следует выбрать? Не мучьтесь слишком долго над этим: все релизы содержат более или менее схожий набор базовых пакетов. Над базовыми пакетами добавляются специальные программные решения третьих сторон, например, TurboLinux больше подходит для малых и средних предприятий, RedHat для серверов, а SuSE для рабочих станций. Тем не менее, различия, скорее всего, будут очень поверхностным. Лучшая стратегия заключается в тестировании нескольких дистрибутивов; к сожалению, не у всех есть время для этого. К счастью, есть много советов по вопросу выбора Linux. Быстрый поиск в Google, используя ключевые слова "выбор дистрибутива" выводит десятки ссылок на хорошие советы. Краткая инструкция по установке (Installation HOWTO) также рассматривает вопрос выбора дистрибутива.

- Linux не очень дружелюбен к пользователю и запутан для начинающих:

Надо сказать, что Linux (по крайней мере, ядро системы) менее удобный для эксплуатации пользователем, чем MS Windows и, конечно, труднее, чем MacOS, но ... В свете его популярности, значительные усилия были предприняты, чтобы сделать Linux еще проще в использовании, особенно для новых пользователей. Ежедневно публикуется все больше информации, такой как данное руководство, чтобы помочь заполнить пробел в документации, доступной для пользователей всех уровней.

- Продукт Open Source может быть надежным?

Как может то, что бесплатно быть надежным? Есть ли у пользователей Linux что-то лучшее при использовании Linux или нет, что дает им огромное преимущество по сравнению с пользователями проприетарного программного обеспечения, которые не имеют такой свободы. После долгих периодов тестирования, большинство пользователей Linux приходят к выводу, что Linux не только хорош как свободная ОС, но во многих случаях лучше и быстрее традиционных решений. Если бы Linux не заслуживал доверия, то давно бы исчез, никогда не узнав популярности как сейчас, имея миллионы пользователей. В настоящее время пользователи могут влиять на их системы и делиться своими замечаниями с сообществом, в результате система становится лучше изо дня в день. Это проект, который никогда не заканчивается, и это правда, но в постоянно меняющемся окружении, Linux — это кроме прочего проект, продолжающий стремиться к совершенству.

Особенности Linux

Linux и GNU

Не смотря на большое количество реализаций Linux, в разных дистрибутивах вы найдете много общего; по сути, любую версию Linux можно представить как комплект строительных блоков, из которых вы можете собирать что-то, отвечающее вашим нуждам и взглядам. Установка системы – это только начало длительных отношений. Когда вы начнете думать, что у вас хорошо работающая система, Linux все-равно будет стимулировать ваше воображение и творчество, и чем больше вы будете осознавать мощь системы, тем больше будете пытаться пересмотреть свои ограничения.

Linux'ы могут отличаться в зависимости от дистрибутива, вашей аппаратуры и персональных предпочтений, но база, на которой построены все графические и другие интерфейсы, останется одной и той же. Система Linux основана на инструментах GNU (GNU's Not UNIX), которые предоставляют набор стандартных способов управления и использования системы. Все инструменты GNU имеют открытый исходный код, поэтому могут быть установлены на любой системе. В большинстве дистрибутивов содержатся предварительно скомпилированные пакеты наиболее распространенных инструментов; такими пакетами являются RPM RedHat и Debian-пакеты (также называемые deb или dpkg) на Debian. Вам не обязательно быть программистом, чтобы установить их. Однако, если вы относитесь к тем, кто любит делать что-то самостоятельно, то получите большее удовольствие от Linux, поскольку большинство дистрибутивов поставляются с полным набором инструментов для разработки. Это позволяет инсталлировать новое ПО исключительно из исходного кода. Такой способ дает вам также возможность устанавливать программное обеспечение, даже если оно не существует в виде пакетов, подходящих для вашей системы.

A list of common GNU software: (Список общего программного обеспечения GNU):

- Bash: GNU интерпретатор команд ОС
- GCC: GNU компилятор с языка C
- GDB: GNU отладчик
- Coreutils: набор базовых утилит UNIX-типа, таких как ls, cat и chmod

- Findutils: для поиска файлов
- Fontutils: конвертация шрифтов из одного формата в другой или создание новых шрифтов
- The Gimp: GNU Image Manipulation Program (создание и редактирование изображений)
- Gnome: GNU среда рабочего стола
- Emacs: очень мощный редактор
- Ghostscript and Ghostview: интерпретатор и графический интерфейс для файлов PostScript
- GNU Photo: софт для взаимодействия с цифровыми камерами
- Octave:язык программирования, в первую очередь предназначенный для выполнения численных вычислений и обработки изображений.
- GNU SQL: система управления реляционными базами данных
- Radius: удаленная аутентификация и отчетность сервера
- ...

Также для Linux существует много коммерческих приложений, для получения дополнительной информации об этих пакетах мы отправляем вас к специальной документации. В данном руководстве мы будем обсуждать только свободно доступное ПО, который поставляется (в большинстве случаев) под лицензией GNU.

Для установки отсутствующих или новых пакетов, вам потребуется какой-то способ управления программным обеспечением. Чаще всего реализации включают RPM и dpkg. RPM (RedHat Package Manager) используется на разных системах Linux, даже когда их имя не намекает на это. В dpkg (системе управления пакетами Debian) используется интерфейс под названием apt-get, который также может управлять RPM пакетами. Novell Ximian Red Carpet — третья вариант реализации RPM с графическим интерфейсом. Производители программного обеспечения третьих сторон могут разрабатывать свои собственные процедуры установки, иногда напоминающие InstallShield, и такие, как для MS Windows и для других платформ. При изучении Linux, вы, вероятно, столкнетесь с одной или несколькими из этих программ.

GNU/Linux

Ядро Linux (костяк вашей системы) не является частью проекта GNU, но использует ту же лицензию, что и программное обеспечение GNU. Основная масса утилит и средств разработки (основа вашей системы), которые не являются Linux-специфичными, взяты из проекта GNU. Для того, чтобы любая ОС была полезна, она должна содержать как ядро, так и, по крайней мере, минимальный набор утилит, поэтому часть людей утверждают, что такая система должна называться GNU/Linux.

Чтобы иметь максимально возможную степень независимости от дистрибутивов, на протяжении всего этого курса мы будем обсуждать именно этот вариант Linux. Если мы не говорим о системе GNU/Linux, то названия определенного дистрибутива, версии или программы будет упомянуто отдельно.

Какой дистрибутив мне следует установить?

До инсталляции очень важно определить, какое у вас оборудование. Поскольку каждый дистрибутив Linux содержит основные пакеты и может быть сконструирован для удовлетворения почти любых требований (т.к. везде используется ядро Linux), вам необходимо только учесть, будет ли дистрибутив работать на вашем оборудовании. LinuxPPC, например, был сделан для работы на Apple и других PowerPC и не работает на обычном ПК на базе x86. LinuxPPC работает на новых компьютерах Mac, но вы не можете использовать его на некоторых старых шинах с древней технологией. Еще одна сложная случай — это оборудование Sun, это

могут быть старые SPARC процессоры или более новые UltraSparc, обоим требуются разные версии Linux.

Некоторые дистрибутивы Linux оптимизированы для определенных процессоров, таких как процессоры Athlon, но они в то же время будут прилично работать на стандартных 486, 586 и 686 процессорах Intel. Иногда дистрибутивы для специальных процессоров не так надежны, т. к. меньшее количество людей участвует в их проверке.

Большинство дистрибутивов Linux предлагают ряд программ для обычных PC с специальными пакетами, содержащими оптимизированные ядра для процессоров, основаных на Intel x86. Эти дистрибутивы являются проверенными и поддерживаются на регулярной основе, упор делается на качество серверных реализаций, а также легкую установку и обновление. Примеры - Debian, Ubuntu, Fedora, SuSE и Mandriva, которые на сегодняшний день являются самыми популярными системами Linux и, в общем, считаются простыми в обращении для начинающего пользователя, хотя не лишают специалистов возможности получать максимальную отдачу от своих Linux-машин. Linux также прилично работает на ноутбуках и серверах среднего ряда. Драйверы для нового оборудования добавляются только после всесторонних испытаний, что делает систему более стабильной.

Хотя в одной системе стандартной средой рабочего стола может быть Gnome, другая по умолчанию может предложить KDE. Вообще, как Gnome так и KDE доступны для всех основных дистрибутивов Linux. Для более опытных пользователей есть также другие оконные и десктопные менеджеры.

Стандартный процесс установки позволяет пользователям выбирать из различных базовых установок, таких как "рабочая станция", где устанавливаются все пакеты, необходимые для повседневного использования и разработки, или установка "сервера", где могут быть выбраны различные сетевые сервисы. Опытные пользователи в процессе начальной установки могут устанавливать любые комбинации пакетов, которые они захотят.

Цель этого руководства заключается в описании того, что применимо для всех дистрибутивов Linux. Для вашего же удобства, однако, настоятельно рекомендуется, поначалу придерживаться основных дистрибутивов, поддерживающих все рядовое аппаратное обеспечение и широко используемые приложения по умолчанию. Ниже следует хороший список для начинающих:

- Fedora Core
- Debian
- SuSE Linux
- Mandriva (former MandrakeSoft)
- Knoppix: операционная система, которая работает с вашего CD-ROM'a, и не требуется ничего инсталлировать.

Доступные для загрузки ISO-образы могут быть взяты на LinuxISO.org. Основные дистрибутивы могут быть приобретены в любой порядочном компьютерном магазине.

Резюме

В этой главе мы узнали, что:

- Linux – это одна из реализаций UNIX.
- Linux – операционная система, написанная на языке программирования Си.
- “De gustibus et coloribus non disputandum est” («О вкусах и цветах не спорят»): для каждого найдется Linux.

- Linux использует инструменты GNU, ряд свободно доступных стандартных инструментов для обслуживания операционной системы.

Упражнения

Практическое задание для начинающих: установите Linux на ваш ПК. Перед этим прочитайте руководство по установке для вашего дистрибутива и/или HOWTO по установке.

 **Читайте сообщения!**
Большинство ошибок происходят из-за пренебрежения к чтению информации, выводимой во время установки. Тщательное чтение сообщений во время установки является первым шагом на пути к успеху.

То, что вы должны знать перед началом установки Linux:

- Будет ли этот дистрибутив работать на моем оборудовании?
  В случае возникновения сомнений о совместимости вашего оборудования проверьте информацию на http://www.tldp.org/HOWTO/Hardware-HOWTO/index.html.
- Какая у меня клавиатура (количество клавиш, расположение)? Какая мышь (последовательный/параллельный порт, количество кнопок)? Сколько Мб RAM?
- Устанавливать мне базовую рабочую станцию, сервер, или мне потребуется выбрать специфичные пакеты самому?
- Мне устанавливать с жесткого диска, CD-ROM'а или использовать сеть? Должен ли я для этого внести изменения в BIOS? Требует ли выбранный способ установки загрузочный диск?
- Linux будет единственной системой на данном компьютере, или можно будет загружать пару инсталлированных систем? Стоит ли выделять большой раздел в целях установки виртуальных систем в дальнейшем, or is this a virtual installation itself?
- Принадлежит ли данных компьютер какой-нибудь сети? Какое его имя, IP адрес? Есть ли какие шлюзы серверов или других важных сетевых машин, с которыми должен быть связан мой компьютер?

 **Linux ожидает быть в сети.**
Если сеть не используется или была неверно настроена, то это может привести к медленной загрузке.

- Является ли данный компьютер шлюзом/маршрутизатором/брандмауэром? (Если вы задумались над этим вопросом, то, скорее всего, не является.)
- Разметка диска: пусть программа установки на этот раз сделает это за вас, мы будем обсуждать разделы подробно в Главе 3, «О файлах и файловой системе». *(Примечание переводчика: аккуратнее с этим советом, а то удалите все свои данные.)* Если вам захочется знать все по этому вопросу, то обратитесь к системно-зависимой документации. Если ваш дистрибутив не предлагает по умолчанию разметить диск, то, возможно это означает, что он не подходит для начинающих.
- Будет эта машина запускаться в текстовом или графическом режиме?
- Придумайте хороший пароль администратора (root). Создайте учетную запись обычного пользователя (без привилегированного доступа к системе).
- Нужен ли мне диск для восстановления системы? (рекомендуется)
- Какие я хочу иметь языки программирования?

Полный перечень можно найти на http://www.tldp.org/HOWTO/Installation-HOWTO/index.html.

Если установка была успешной, то в следующих главах мы может продолжить наше обучение.

Глава 2. Быстрый старт

Аннотация

Чтобы получить максимальную отдачу от этого руководства, мы сразу же начнем с практической главы о входе в систему Linux и выполнении некоторых базовых действий.

Мы обсудим:

- Вход в систему
- Выход из системы
- Текстовый и графический режимы
- Изменение вашего пароля
- Навигацию по файловой системе
- Определение типа файла
- Просмотр текстовых файлов
- Поиск помощи

Вход в систему, активация пользовательского интерфейса и выход из системы

Введение

Для работы в системе Linux вам нужно ввести имя пользователя и пароль. Вам всегда придется проходить аутентификацию для доступа к системе. Как было уже упомянуто в упражнениях к Главе 1, "Что такое Linux?", большинство систем Linux для ПК имеют два основных режима работы в ней: либо быстрый и рассудительный текстовый режим консоли (которая выглядит как DOS с мышью, многозадачностью и многопользовательскими возможностями), либо графический режим, который выглядит лучше, но зато съедает больше системных ресурсов.

Графический режим

В настоящее время он присутствует на большинстве настольных компьютеров по умолчанию. Когда появляется запрос на ввод вашего имени пользователя, а затем, в появившемся новом окне, запрос на ввод пароля, вы должны понимать, что будете входить в систему с помощью графического режима.

Перед входом в систему, убедитесь, что курсор находится в окне ввода логина, укажите имя пользователя и пароль, затем нажмите кнопку |OK| или клавишу **Enter**.

 **Осторожнее с аккаунтом root!**
Вообще считается плохой идеей "графический" вход в систему под именем root, т.е. учетной записью администратора системы, т. к. использование графики приводит к запуску множества дополнительных программ, и в случае root'а с большим количеством дополнительных разрешений. Чтобы свести риски к минимуму, для работы в графическом режиме используйте учетную запись обычного пользователя.

Достаточное сведение рисков к минимуму — это общая рекомендация; вход под учетной записью root уместен, когда действительно требуются дополнительные привилегии.

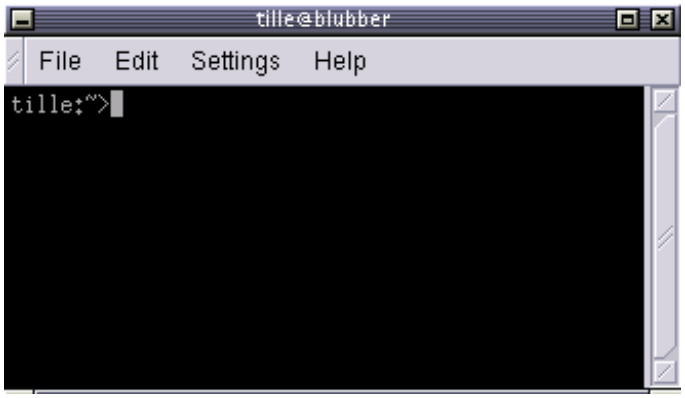
После ввода комбинации вашего имени пользователя и пароля, может потребоваться некоторое время для запуска графической среды; время зависит от скорости процессора компьютера, используемого вами программного обеспечения и ваших личных настроек.

Позже, вам будет необходимо открывать *окно терминала* или, для краткости, *Xterm* (X является названием для лежащей в основе ПО поддержки графической среды). Эту программу можно найти в меню **Приложения ? Утилиты, Системные инструменты** или **Интернет** (все зависит от того, какой оконный менеджер вы используете). Могут быть значки, которые можно использовать для быстрого запуска окна Xterm, кроме того, как правило, в контекстном меню рабочего стола (щелчок правой кнопкой мыши на фоне) вам будет предложено приложение окна терминала.

Во время работы с меню, вы заметите, что многие вещи могут быть сделаны без ввода команд с клавиатуры. Для большинства пользователей старым добрым методом обращения с компьютером будет метод "указал-и-щелкнул". Но это руководство для будущих сетевых и системных администраторов, которым необходимо будет вмешиваться в сердце системы. Такие нуждаются в более мощном инструменте, чем мышь, для решения всех задач, с которыми им придется столкнуться. Этот инструмент есть shell (командная оболочка), а в графическом режиме, открыв окно терминала, мы и запускаем нашу оболочку.

Окно терминал — это ваша панель управления системой. Почти все необходимое делается с помощью этого простого, но мощного текстового инструмента. Окно терминала, когда вы открываете его, всегда должно показывать приглашение. Этот терминал показывает стандартное приглашение, состоящее из логина пользователя и текущего рабочего каталога, обозначенного тильдой (~):

Рисунок 2.1. Окно терминала



Другая распространенная форма приглашения имеет такой вид:

```
[user@host dir]
```

Применительно к данному образцу, *user* - будет заменен на ваш логин, *host* - имя компьютера, на котором вы работаете, и *dir* – указание вашего текущего местонахождения в файловой системе.

Позже мы обсудим приглашения и их поведение в деталях. Сейчас достаточно знать, что приглашения могут отображать любую информацию, но они не являются частью команд, которые вы передаете вашей системе.

Для выхода из системы в графическом режиме, вам следует закрыть все окна терминала и других приложений. После этого, нажмите на значок выхода или найдите аналогичную функцию в меню. Закрывать все совсем не обязательно, система может сделать это за вас, но управление сеансами может "выложить" на экране все открытые приложения снова, когда вы войдете в систему, это занимает больше времени и не всегда является желаемым эффектом. Однако, это поведение можно изменить.

Когда вы видите экран для ввода логина, запрашивающий имя пользователя и пароль, значит, вы успешно вышли из системы.

Gnome или KDE?

Мы уже упоминали несколько раз как рабочий стол Gnome так и KDE. Это два самых популярных способа управления вашим десктопом, хотя есть много, много других. Какой бы вы рабочий стол не выбрали для работы — замечательно, но до тех пор, пока вы не узнаете, как открыть окно терминала. Тем не менее, мы и впредь будем продолжать ссылаться на Gnome и KDE как наиболее популярным способам решения определенных задач.

Текстовый режим

Вы знаете, что в текстовом режиме, когда весь экран черный, на нем присутствуют (в большинстве случаев белые) только символы. При входе текстовый режим обычно показывает некоторую информацию о компьютере, на котором вы работаете, название машины и приглашение, ожидающее вашего входа:

```
Debian GNU/Linux 5.0 fenix tty1
fenix login: _
```

Вход в систему отличается от входа через графический режим в том, что вы должны нажимать клавишу **Enter** после ввода имени пользователя, здесь на экране нет кнопок, чтобы щелкать мышью. После этого вы должны ввести пароль, после чего снова нажать **Enter**. Вы не увидите никаких признаков того, что вы вводите что-то, даже звездочек, и вы не уведите движение курсора. Но это нормально для Linux и делается по соображениям безопасности.

Когда система опознает вас как законного пользователя, вы можете получить еще немного информации, называемой *сообщением (message)* дня, которое может быть чем угодно. Кроме того, в системах UNIX модно отображать "фразы фортуны", которые представляют собой некоторые общие мудрым или не очень (это зависит от вас) мысли. После этого вам будет предоставлена оболочка shell, с таким же приглашением, которое вы наблюдали бы в графическом режиме.

Не входите в систему как root

Также в текстовом режиме: входите как root только для установок и настроек, которые требуют безусловных прав администратора, например, добавление пользователей, установка программного обеспечения, а также выполнения сетевых и других настроек системы. После того как вы закончите, сразу выйдите их этого специального аккаунта и возобновите работу под непривилегированным пользователем. Кроме того, некоторые системы, как Ubuntu, принуждают вас использовать **sudo**, так что вам не нужен прямой доступ к учетной записи администратора.

Выход осуществляется путем ввода команды **logout** с последующим нажатием **Enter**. Вы успешно вышли из системы, когда снова видите экран с предложением войти.

Кнопка питания

Хотя Linux не предназначена для отключения без применения соответствующих процедур по остановки системы, нажатие на кнопку питания равнозначно запуску тех процедур на более новых системах. Тем не менее, выключение старой системы, минуя остановочный процесс, может вызвать серьезное повреждение! Если вы хотите быть уверены, всегда используйте иконку выключения, если вы выходите из графического интерфейса; или, когда видите экран для входа в систему (где вы должны указать свое имя пользователя и пароль), осмотритесь для поиска значка выключения.

Теперь, когда мы знаем, как войти и выйти из системы, мы готовы для наших первых команд.

Абсолютные основы

Команды

Это бегло то, что нам потребуется для начала; позже мы обсудим эти команды более детально.

Таблица 2.1. Команды быстрого старта

Команда	Значение
ls	Отображает список файлов в текущей рабочей директории подобно команде **dir** в DOS
cd directory	Смена директории
passwd	Изменение пароля текущего пользователя
file filename	Отображает тип файла с именем filename
cat textfile	Отображает содержимое textfile на экране
pwd	Отображает текущую рабочую директорию
exit or **logout**	Завершение сеанса
man command	Чтение страниц руководства о **command**
info command	Чтение info-страниц о **command**
apropos command	Поиск в базе данных *whatis* для строк

Общие замечания

Вы набираете эти командыв графическом или текстовом режиме в окне терминала после приглашения, а затем нажимаете **Enter**.

Команды могут выдавать результат сами по себе, например, **ls**. Команда ведет себя иначе, если вы указываете *опцию*, которой, как правило, предшествует тире (-), как в **ls -a**. Такая же опция для другой команды может иметь совершенно иное значение. У программ GNU могут быть длинные опции, которым предшествуют два тире (--), как **ls --all**. У некоторых команд опций нет.

Аргумент(ы) для команды – это характеристики объекта(ов), для которых вы применяете команду. Например, в **ls** /etc каталог /etc является аргументом команды **ls**. Это означает, что вы хотите увидеть содержимое этой директории, а не той, что по умолчанию (содержимое текущего каталога) выводится, если просто набрать **ls** и затем нажать **Enter**. Некоторые команды требуют обязательного наличия аргументов, для других они необязательны.

Путем проверки справочной информации по той или иной команде, вы можете выяснить, принимает ли команда параметры (опции) и аргументы, и какие из них действительны (см. Раздел "Получение помощи").

В Linux, как и в UNIX, каталоги разделяются с использованием вперед наклоненной косой черты (слэша), то же самое используется для веб-адресов (URL). Тщательно структуру каталогов мы обсудим позже.

Символы **.** и **..** имеют специальное значение, когда это касается каталогов. Мы попытаемся выяснить об этом значении, в ходе выполнения упражнений, и еще больше — в следующей главе.

Старайтесь избегать входа в систему и использования аккаунта системного администратора, *root*. Кроме того, выполнение обычной работы, решение большинства задач, включая проверку системы, сбор информации и т.д., могут быть выполнены с использованием обычной учетной записи пользователя без каких-либо специальных привилегий на все. В случае необходимости, например, при создании нового пользователя или установки нового программного обеспечения, предпочтительный способ получения прав администратора — путем переключения ID пользователей, для примера см. Раздел "Путь".

Почти все приведенные в этой книге команды могут быть выполнены без привилегий администратора системы. В большинстве случаев, когда для доступа требуются права администратора, при выполнении команды или запуске программы под непривилегированным пользователем, система предупредит вас или предложит ввести пароль root'a. После того как вы закончите, сразу завершите приложение или сессию, которые были запущены с привилегиями root.

Чтение документации должно стать вашей второй натурой. Особенно в начале важно читать системную документацию, руководства к основным командам, HOWTO и т.д. Поскольку объем документации очень велик, то невозможно включить все связанные документы. Эта книга, в целях стимулирования привычки к чтению справочных страниц, попытается направить вас к наиболее подходящей документации по каждому предмету обсуждения.

Использование возможностей Bash

Некоторые специальные комбинации клавиш позволяют делать некоторые вещи легче и быстрее в оболочке GNU, Bash, которая присутствует по умолчанию почти на любой системе Linux, см. Раздел "Shell". Ниже приведен список наиболее часто используемых возможностей; вам настоятельно советуется возыметь привычку их использования, с тем, чтобы с самого начала получить максимум от вашего познания Linux.

Таблица 2.2. Комбинации клавиш в Bash

Клавиша или комбинация клавиш	Функция
Ctrl+A	Перемещение курсора в начало командной строки.
Ctrl+C	Завершение запущенной программы и возврат к приглашению оболочки, см. Глава 4. Процессы.
Ctrl+D	Выход из текущей сессии оболочки, равносильно набору **exit** или **logout**.
Ctrl+E	Перемещает курсор к концу командной строки.
Ctrl+H	Генерирует символ возврата (**backspace**).
Ctrl+L	Очищает данный терминал.
Ctrl+R	Поиск в истории команд, см. Раздел "Команда grep".

Ctrl+Z	Приостановка программы, см. Глава 4. Процессы.
Стрелка влево и **стрелка вправо**	Перемещает курсор в командной строке на одну позицию влево или вправо, так что вы можете вставлять символы не только в начало и конец.
Стрелка вверх и **стрелка вниз**	Обзор истории. Перейдите на строку, которую вы хотите повторить, если необходимо отредактируйте детали, и нажмите Enter; это сэкономит время.
Стрелка вверх и **стрелка вниз**	Обзор истории. Перейдите на строку, которую вы хотите повторить, если необходимо отредактируйте детали, и нажмите Enter; это сэкономит время.
Shift+PageUp и **Shift+PageDown**	Обзор буфера терминала (просмотр текста, который был «прокручен» на экране).
Tab	Завершение команды или имени файла; когда возможны несколько вариантов, система издаст сигнал из колонок, или, если слишком много возможных вариантов, спросит вас, хотите ли вы увидеть их все.
Tab Tab	Показывает возможные варианты завершения файла или команды.

Последние два пункта в таблице выше, могут потребовать дополнительного пояснения. Например, если вы хотите перейти в каталог directory_with_a_very_long_name, вы не собираетесь набирать такое очень длинное имя, нет. Просто наберите в командной строке **cd dir**, а затем нажмите **Tab** и оболочки завершит имя за вас, если нет других файлов, начинающихся с тех же трех символов. Конечно, если нет других элементов, начинающихся с "d", то вы, также можете просто ввести **cd d** и нажать **Tab**. Если более чем один файл начинается с тех же символов, оболочка просигнализирует вам об этом, после чего вы можете нажать клавишу **Tab** дважды через короткий промежуток времени, и shell представит варианты, которые у вас есть:

```
your_prompt> cd st
starthere        stuff        stuffit
```

В приведенном выше примере, если вы введете "a" после первых двух символов и нажмете **Tab** еще раз, то поскольку нет других возможных вариантов, оболочка завершает имя каталога без необходимости ввода строки "rthere":

```
your_prompt> cd starthere
```

Конечно, вам все равно придется нажимать клавишу **Enter**, чтобы согласиться с выбором.

В том примере если вы введете "u", а затем нажмете **Tab**, оболочка добавит за вас "ff", но затем она запротестует снова, потому что возможны несколько вариантов. Если вы нажмете **Tab Tab** еще раз, то увидите варианты; если вы введете один или более символов, которые делают выбор однозначным для системы, и нажмете **Tab** опять, или **Enter**, когда вы дойдете до конца имени выбираемого вами файла, shell завершит имя файла и переместит вас в тот каталог – если это действительно имя директории.

Это работает для всех имен файлов, которые являются аргументами команд.

То же самое касается завершения имени команды. При вводе **ls** и последующем двойном нажатии клавиши **Tab**, выведется список всех команд из вашего PATH (см. Раздел "Путь"), которые начинаются с этих двух символов:

```
your_prompt> ls
ls            lsdev      lspci       lsraid        lsw
lsattr        lsmod      lspgpot     lss16toppm
lsb_release   lsof       lspnp       lsusb
```

Получение помощи

Будьте осторожны

GNU/Linux приветствует самостоятельность. И как обычно в этой системе есть несколько способов достижения цели. Привычный способ получения помощи — это найти того, кто знает; однако несмотря на терпеливость и миролюбивость сообщества пользователей Linux, почти все будут предполагать, что вы пытались использовать один или несколько описанных в данном разделе методов, прежде чем спрашивать их; способы выражения этой точки зрения могут быть достаточно грубыми, если окажется, что вы не следуете этому основному правилу.

Страницы man

Многие начинающие пользователи бояться справочных (man) страниц, потому что они ошеломляют количеством информации. Однако они хорошо структурированы, вы можете убедиться в этом на примере ниже: **man** man.

Чтение man-страниц обычно происходит в окне терминала в графическом режиме, или, если вы предпочитаете, прямо в текстовом режиме. Введите эту команду после приглашения и нажмите **Enter**:

```
yourname@yourcomp ~> man man
```

Документация для man отобразится на экране после нажатия **Enter**:

```
man(1)                                                          man(1)


NAME
 man - format and display the on-line manual pages
 manpath - determine user's search path for man pages

SYNOPSIS
 man [-acdfFhkKtwW] [--path] [-m system] [-p string] [-C config_file]
 [-M pathlist] [-P pager] [-S section_list] [section] name ...


DESCRIPTION
 man formats and displays the on-line manual pages.  If you specify
 section, man only looks in that section of the manual.
 name is normally the name of the manual page, which is typically the
 name of a  command, function, or file.  However, if name contains a
 slash (/) then man interprets it as a file specification, so that you
 can do man ./foo.5 or even man /cd/foo/bar.1.gz.

 See  below  for  a  description  of where man looks for the manual
```

```
 page files.

OPTIONS
 -C  config_file
lines 1-27
```

Для просмотра следующей страницы нужно нажать пробел. Вы можете вернуться на предыдущую страницу, используя клавишу **b**. Когда вы дойдете до конца, **man** обычно закрывается, и вы возвращаетесь назад к приглашению. Нажмите **q**, если вы хотите покинуть man-страницу, не дойдя до конца, или если просмотр автоматически не завершается в конце страницы.

 **Пейджеры**

Имеющиеся комбинации клавиш для работы с man-страницами зависят от *пейджера*, используемого в вашем дистрибутиве. Большинство дистрибутивов используют less для просмотра man-страниц и их прокрутки. См. Раздел "Меньше (less) значит больше" для более подробной информации о пейджерах.

Каждая man-страница обычно содержит несколько стандартных разделов (что можно наблюдать на примере **man** man):

- Первая строка содержит имя команды, о которой вы читаете, и id раздела, в котором эта man-страница находится. Man-страницы упорядочены по разделам. Команды могут иметь несколько man-страниц, например, man-страницу из пользовательского раздела, man-страницу из раздела администратора системы, и man-страницу из раздела для программиста.
- Имя (**name**) команды и краткое описание приводятся в том виде, который используется при индексировании man-страниц. Вы можете просматривать эту базу при поиске любой строки с помощью команды **apropos**.
- Обзор (**synopsis**) команды представляет специальные обозначения всех опций и/или аргументы, которые эта команда может принимать. Вы можете считать опции способом выполнения команды. Аргументы это то, по отношению к чему вы ее выполняете. У некоторых команд нет опций и аргументов. Необязательные опции и аргументы располагаются между "[" и "]", что указывает на то, что они могут быть опущены.
- Дается более длинное описание (**description**) команды.
- Список опций (**options**) с их описанием. Обычно опции могут быть скомбинированы. Если это не так, раздел уведомит вас об этом.
- Окружение (**environment**) описывает переменные оболочки, которые влияют на поведение этой команды (это есть не у всех команд).
- Иногда предусмотрены разделы специфичные для данной команды.
- Ссылки на другие man-страницы приводится в разделе “SEE ALSO” ("смотри также"). Между круглыми скобками есть номер раздела, в котором находится справочная страница для данной команды. Опытные пользователи часто переходят к разделу "SEE ALSO" используя команду /, за которой следует строка поиска see и нажимают **Enter**.
- Также обычно имеется информация об известных ошибках (аномалиях), а также куда можно сообщить о новых ошибках, если вы их найдете.
- Также может содержаться информация об авторе и авторских правах.

У некоторых команд есть несколько man-страниц. Например, у команды **passwd** есть man-страница в разделе 1, а другая — в разделе 5. По умолчанию отображается man-страница с наименьшим номером.

Если вы хотите увидеть другой раздел, а не тот, что по умолчанию, укажите его после команды **man**:

man 5 *passwd*

Если вы хотите увидеть все справочные страницы о команде, одну за другой, используйте с **man** опцию **-a**:

man -a *passwd*

Таким образом, когда вы дойдете до конца первой man-страницы и нажмите пробел, будет отображаться man-страница из следующего раздела.

Больше информации

Страницы Info

В дополнение к man-страницам, вы можете почитать о команде info-страницы, используя команду **info**. Они обычно содержат более свежую информацию и несколько проще в использовании. Man-страницы некоторых команд ссылаются на info-страницы.

Начните с ввода **info** info в окне терминала:

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

Info: An Introduction
**********************

   Info is a program, which you are using now, for reading
documentation of computer programs.  The GNU Project distributes most
of its on-line manuals in the Info format, so you need a program called
"Info reader" to read the manuals.  One of such programs you are using
now.

   If you are new to Info and want to learn how to use it, type the
command `h' now.  It brings you to a programmed instruction sequence.

   To learn advanced Info commands, type `n' twice.  This brings you to
`Info for Experts', skipping over the `Getting Started' chapter.

* Menu:

* Getting Started::             Getting started using an Info reader.
* Advanced Info::               Advanced commands within Info.
* Creating an Info File::       How to make your own Info file.
--zz-Info: (info.info.gz)Top, 24 lines --Top-----------------------------
Welcome to Info version 4.2. Type C-h for help, m for menu item.
```

Используйте клавиши со стрелками для просмотра текста и перемещения курсора на строку, начинающуюся со звездочки и содержащую ключевое слово, о котором вы хотите получить информацию, и затем нажмите **Enter**. Используйте клавиши **P** и **N** для перехода к предыдущей или следующей теме. Пробел переместит вас на одну страницу дальше, независимо от того, начинается ли новая тема или страница info для другой команды. Используйте **Q** для выхода. Программа **info** содержит больше информации.

Команды whatis и apropos

Краткие пояснения к командам доступны путем использования команды **whatis**, как в примере ниже:

```
[your_prompt] whatis ls
ls                   (1)  - list directory contents
```

Она отображает краткую информацию о команде, и первый раздел в структуре man-страниц, который содержит соответствующую страницу.

Если вы не знаете, где начать и какую man-страницу читать, команда **apropos** даст дополнительную информацию. Скажем, вы не знаете, как запустить браузер, тогда вы можете ввести следующую команду:

```
another prompt> apropos browser
Galeon [galeon](1)  - gecko-based GNOME web browser
lynx           (1)  - a general purpose distributed information browser
             for the World Wide Web
ncftp          (1)  - Browser program for the File Transfer Protocol
opera          (1)  - a graphical web browser
pilot          (1)  - simple file system browser in the style of the
      Pine Composer
pinfo          (1)  - curses based lynx-style info browser
pinfo [pman]   (1)  - curses based lynx-style info browser
viewres        (1x)  - graphical class browser for Xt
```

После нажатия **Enter** вы увидите, что большая часть браузеров входят в состав вашей ОС: и не только веб-браузеры, но также файловые и FTP браузеры, браузеры для документации. Если у вас установлены пакеты разработчика, то могут быть также сопровождающие man-страницы, касающиеся написания программ, связанных с браузерами. Как правило, команды с man-страницы находится в первом разделе, поэтому помечены "(1)", что соответствует их использованию под пользователем. Пользователь, который написал выше команду **apropos**, может в результате попытаться выполнить команды **galeon**, **lynx** или **opera**, т.к. они очевидно связаны с просмотром World Wide Web.

Опция --help

Большинство команд GNU поддерживают --help, которая выдает краткое пояснение того, как использовать команду, и список доступных опций. Ниже результат выдачи с этой опцией для команды **cat**:

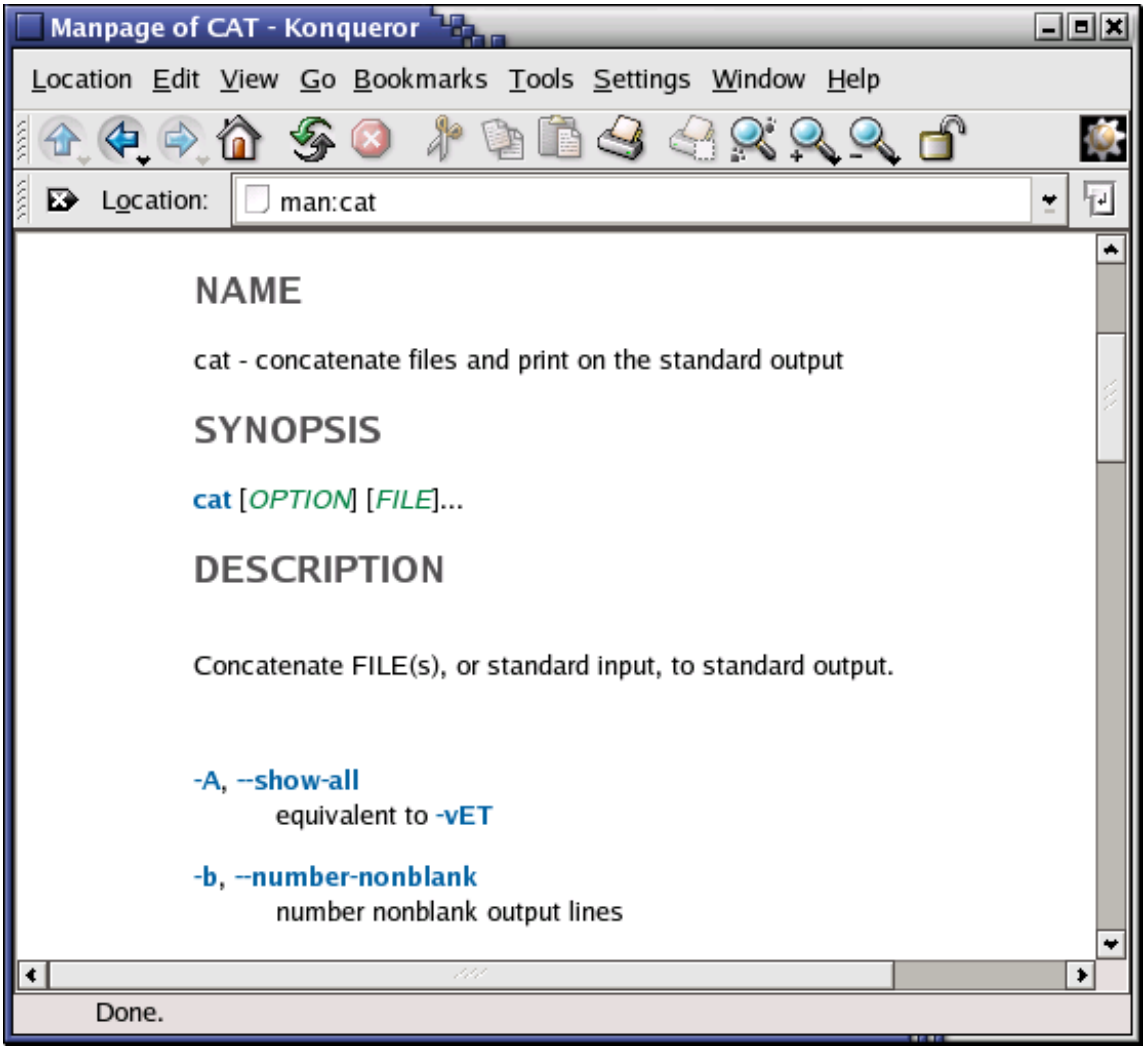
```
userprompt@host: cat --help
Usage: cat [OPTION] [FILE]...
Concatenate FILE(s), or standard input, to standard output.

  -A, --show-all           equivalent to -vET
  -b, --number-nonblank    number nonblank output lines
  -e                       equivalent to -vE
  -E, --show-ends          display $ at end of each line
  -n, --number             number all output lines
  -s, --squeeze-blank      never more than one single blank line
  -t                       equivalent to -vT
  -T, --show-tabs          display TAB characters as ^I
  -u                       (ignored)
  -v, --show-nonprinting   use ^ and M- notation,
```

```
      except for LFD and TAB
      --help     display this help and exit
      --version  output version information and exit

With no FILE, or when FILE is -, read standard input.

Report bugs to .
```

Помощь в графическом режиме

Если вы предпочитаете графический интерфейс пользователя, то не отчаивайтесь. Konqueror, файловый менеджер KDE по умолчанию, обеспечивает безболезненный и красочный доступ к man и info-страницам. Вы можете попробовать "info:info" в адресной строке, и сможете посмотреть info-страницу о команде **info**. Также "man:ls" представит вам man-страницу команды **ls**. У вас даже есть завершение имени команды: вы увидите man-страницы для всех команд, начинающихся с "ls" прокрутив меню. Ввод "info:/dir" в адресной строке отображает все info-страницы, расположенные во вспомогательных категориях. Превосходен включенный в справочник Konqueror Help-контент. Запускается из меню или набрав команду **konqueror** в окне терминала с последующим нажатием **Enter**; см. скриншот ниже.

Рисунок 2.2. Konqueror как help-браузер



Help-браузер в Gnome также очень удобный. Вы можете запустить его выбрав **Приложения ? Справка** в меню Gnome, нажав спасателей значок на вашем рабочем столе, либо введя команду **gnome-help** в окне

терминала. Системную документацию и man-страницы легче просматривать в простом интерфейсе.

Файловый менеджер **nautilus** обеспечивает поиск по индексу man- и info-страниц, они легко просматриваются и взаимосвязаны. Nautilus запускается из командной строки, или нажатием на значке вашего домашнего каталога, или из меню Gnome.

Большим преимуществом GUI для системной документации является то, что вся информация полностью взаимосвязана, так что вы можете переходить по ссылкам в разделе "SEE ALSO" ("См. также") и там, где появляются ссылки на другие man-страницы, и таким образом просматривать и приобретать знания без перерыва в течение нескольких часов.

Исключения

Некоторые команды не имеют отдельной документации, т.к. являются частью другой команды. **cd, exit, logout** и **pwd** как раз такие исключения. Они часть вашей программы shell и называются встроенными командами shell. Для получения информации о них обратитесь к man- или info-страницам вашей оболочки. Большинство начинающих пользователей Linux имеют Bash оболочку. См. Раздел "Shell" для дополнительной информации об оболочках.

Если вы изменяете первоначальную системную конфигурацию, то все равно еще может быть возможным, что man-страницы существуют, но не видны, т.к. окружение вашей оболочки изменилось. В этом случае вам нужно проверить переменную MANPATH. Как это сделать описано см. Раздел "Экспорт переменных".

Некоторые программы или пакеты содержат только набор инструкций или ссылки в каталог /usr/share/doc. Для их отображения см. Раздел "Другие способы просмотра содержимого".

В худшем случае, вы можете случайно удалить документы из системы (надеемся случайно, потому что это очень плохая идея делать это целенаправленно). В этом случае, попробуйте сначала убедиться, что там действительно нет ничего подходящего, используя инструмент поиска (прочитайте Раздел "Поиск файлов"). Если окажется именно так, то возможно, вам придется переустановить пакет, который содержит команду, к которой относится документация, см. Раздел "Установка нового ПО".

Резюме

Linux по традиции работает в текстовом и графическом режимах. Поскольку мощный процессор и оперативная память не так дорого стоят в наши дни, каждый пользователь Linux может позволить себе работать в графическом режиме и обычно так и делает. Это не означает, что вам не надо знать о текстовом режиме: мы будем работать в текстовом окружении на протяжении этого курса, используя окно терминала.

Linux подталкивает своих пользователей к приобретению знаний и независимости. Конечно, для достижения этой цели вам придется читать много документации; поэтому вы заметите, что мы ссылаемся на дополнительную документацию для почти каждой команды, инструмента и проблемы, встречающейся в этой книге. Чем больше документов вы читаете, тем легче будет становиться в дальнейшем, и тем быстрее вы будете перелистывать руководства. Как можно скорее сделайте чтение документации привычкой. Когда вы не знаете ответа на вопрос, обращение к документации должно стать второй натурой.

Мы уже узнали некоторые команды:

Таблица 2.3. Новые команды из главе 2: Основы

Команда	Значение

apropos	Поиск информации о команде или по теме.
cat	Показывает содержимое одного или более файлов.
cd	Переход в другую директорию.
exit	Выход из сессии оболочки shell.
file	Получение информации о содержании файла.
info	Чтение info-страниц о команде.
logout	Выход из сессии shell.
ls	Просмотр содержимого каталога.
man	Чтение страниц руководств о команде.
passwd	Изменение пароля.
pwd	Отображение текущей рабочей директории.

Упражнения

Многое мы узнаем, совершая ошибки и наблюдая, как что-то может пойти не так. Эти упражнения придуманы таким образом, чтобы заставить вас читать некоторые сообщения об ошибках. Порядок, в котором следует выполнять эти упражнения, важен.

Не забывайте использовать возможности Bash в командной строке: выполняя упражнения, старайтесь вводить как можно меньше символов насколько это возможно!

Подключение и отключение

- Определите, работаете ли вы в текстовом или графическом режиме.
  Я работаю в текстовом/графическом режиме. (ненужное зачеркнуть)
- Войдите под своим именем пользователя и паролем, которые вы создали для себя во время установки.
- Выйдите.
- Войти снова, используя несуществующее имя пользователя
  -> Что случилось?

Пароли

Войти снова под вашим именем пользователя и паролем.

- Измените пароль на *P6p3.aa!* и нажмите клавишу **Enter**.
  -> Что произошло?
- Попробуйте еще раз, но на этот раз введите пароль, который до смешного легок, как *123* или *aaa*.
  -> Что случилось?

- Попробуйте еще раз, на этот раз без ввода пароля, а просто нажав клавишу **Enter**.
  -> Что случилось?
- Попробуйте команду **psswd** вместо **passwd**
  -> Что случилось?

Новый пароль.
Пока вы снова не измените ваш пароль на прежний, т.е. тот, который был до этого упражнения, он будет "P6p3.aa!". Измените ваш пароль после этого упражнения!
Обратите внимание, некоторые системы могут не позволить снова изменить пароль на прежний в течение определенного времени или определенного количества изменений пароля, или в обоих случаях.

Каталоги

Вот несколько упражнений, которые помогут вам освоиться.

- Введите команду **cd** blah
  -> Что случилось?
- Введите команду **cd ..**
  Помните о пробеле между "cd" и ".."! Используйте команду **pwd**.
  -> Что произошло?
- Получите список содержимого каталога с помощью команды **ls**.
  -> Что вы видите?
  -> Как вы думаете что это?
  -> Проверьте, используя команду **pwd**.
- Введите команду **cd**.
  -> Что случилось?
- Повторите шаг 2 два раза.
  -> Что случилось?
- Отобразите содержимое этой директории.
- Попробуйте выполнить команду **cd** root
  -> Что случилось?
  -> В какие каталоги у вас есть доступ?
- Повторите шаг 4.
  Знаете ли вы еще один путь узнать, где вы сейчас?

Файлы

- Перейдите в каталог /, а затем в etc . Наберите **ls**; если выдача больше, чем ваш экран, сделайте окно больше, или попробуйте **Shift + PageUp** и **Shift + PageDown**.
- Файл inittab содержит ответ на первый вопрос в этом списке. Попробуйте на нем команду **file**.
  -> Тип файла inittab — это .....
- Используйте команду **cat** inittab и прочитайте этот файл.
  -> Какой режим у вашего компьютера по умолчанию?
- Вернитесь в вашу домашнюю директорию, используя команду **cd**.

- Введите команду **file .**
  -> Вам это помогло определить смысл "."?
- Можете ли вы посмотреть "." с помощью команды **cat**?
- Отобразите помощь для программы **cat**, используя опцию --help. С помощью опции для нумерации выходных линий посчитайте, сколько пользователей перечислено в файле /etc/passwd.

Получение помощи

- Прочитайте **man** ls
- Прочитайте **info** passwd
- Введите команду **apropos** pwd
- Попробуйте **man** или **info** для **cd**
  -> Как бы вы найдете более подробную информацию о **cd**?
- Прочитайте **ls** --help и попробуйте выйти.

Глава 3. О файлах и файловых системах

Аннотация

После начальной разведки в Главе 2, Быстрый старт, мы готовы более подробно обсудить файлы и каталоги системы Linux. Многие пользователи испытывают трудности с Linux, т.к. они недостаточно изучили, где и какие данные хранятся в системе. Мы постараемся пролить свет на организацию файлов в файловой системе.

Мы также составим список наиболее важных файлов и каталогов, будем использовать разные способы просмотра содержимого тех файлов, и узнаем, как могут быть созданы, перемещены и удалены файлы и каталоги.

После выполнения упражнений этой главы, вы будете уметь:

- Описывать схему файловой системы Linux
- Отображать и вводить адреса
- Описывать более важные файлы, в том числе ядро и shell
- Искать потерянные и скрытые файлы
- Создавать, перемещать и удалять файлы и каталоги
- Показать содержимое файлов
- Понимать и использовать различные типы ссылок
- Узнавать свойства файла и изменять права доступа к файлу

Общий обзор файловой системы Linux

Файлы

Общее

Простое описание системы UNIX, также применимое к Linux, заключается в следующем:

"В системе UNIX, все есть файл; а если что-то не файл, то это процесс".

Это утверждение правда, т.к. существуют специальные файлы, которые больше, чем просто файлы (именованные каналы и сокеты, например), но для простоты допускают обобщение и говорят, что все есть файл. Система Linux, как и UNIX, не делает никакой разницы между файлом и каталогом, так как каталог - это просто файл, содержащий имена других файлов. Программы, службы, тексты, изображения и т.д. — все это файлы. В системе Linux устройства ввода и вывода и вообще все устройства считаются файлами.

Для того, чтобы организованно управлять всеми этими файлами, человеку удобно представлять их в виде упорядоченной древовидной структуры на жестком диске, что нам известна из MS-DOS (Дисковая Операционная Система), например. Большие ветви содержат больше ветвей, а в конце ветви содержат листья деревьев, т.е. обычные файлы. В настоящее время мы будем использовать это представление о дереве, но потом мы узнаем, почему это не совсем точный образ.

Виды файлов

Большинство файлов просто файлы, называемые обычными файлами; они содержат обычные данные, например, текстовые файлы, исполняемые файлы (или программы), файлы ввода или вывода программ и т.д.

Хотя предположение, что все что вы обнаружите в системе Linux представляет собой файл достаточно верное, есть некоторые особенности.

- *Каталоги*: файлы, которые представляют собой списки других файлов.
- *Специальные файлы*: механизм использования ввода-вывода. Большинство специальных файлов находятся в /dev, мы их обсудим позже.
- *Ссылки*: механизм обеспечения видимости файла или каталога во множестве частей файлового дерева системы. Мы в деталях поговорим о ссылках.
- *(Домены) сокеты*: особый тип файла, подобный сокетам TCP/IP, обеспечивающий взаимодействие в сети процессов, защищенных контролем файловой системы на доступ.
- *Именованные каналы*: действуют более или менее похоже на сокеты и обеспечивают способ коммуникации между процессами без использования правил поведения сетевых сокетов.

Опция -l команды **ls** отображает тип файла, на что указывает первый символ в каждой выводимой строке:

```
jaime:~/Documents> ls -l
total 80
-rw-rw-r--   1 jaime   jaime   31744 Feb 21 17:56 intro Linux.doc
-rw-rw-r--   1 jaime   jaime   41472 Feb 21 17:56 Linux.doc
drwxrwxr-x   2 jaime   jaime    4096 Feb 25 11:50 course
```

Эта таблица дает обзор символов, характеризующих тип файла:

Таблица 3.1. Типы файлов

Символ	Значение
-	Обычный файл

d	Директория
l	Ссылка
c	Специальный файл
s	Сокет
p	Именованный канал
b	Блочное устройство

Чтобы не нужно было всегда выводить длинный список для просмотра файлового типа, многие системы по умолчанию выдают не просто **ls**, а **ls** -F, который добавляет суффиксы к именам файлов с виде одного из символов “/=*|@”, которые указывают на тип файла. Чтобы было еще легче начинающим пользователям, обе опции и -F и --color обычно комбинируют, см. Раздел, "Дополнительная информация о ls". Для лучшей удобочитаемости в этом документе мы будем использовать **ls** -F.

Как пользователю вам приходится иметь дело только непосредственно с текстовыми файлами, исполняемыми файлами, каталогами и ссылками. Специальные типы файлов существуют для того, чтобы ваша система делала то, что вы требуете от нее и рассматриваются системными администраторами и программистами.

Теперь, прежде чем мы рассмотрим важные файлы и каталоги, нам нужно узнать побольше о разделах.

О разметке

Зачем разделы?

Большинство людей имеют смутное представление о том, что такое разделы, так как каждая операционная система самостоятельно способна их создавать или удалять. Может показаться странным, что Linux использует более чем один раздел на одном диске, даже когда выполняется стандартная процедура установки; для этого должно быть какое-то объяснение.

Одной из целей разделения на разделы является повышение сохранности данных на случай непредвиденных происшествий. Путем разделения жесткого диска на разделы, данные могут быть сгруппированы и разобщены. Когда происходит авария, повреждаются данные только одного раздела, а данные других разделов скорее всего уцелеют.

Этот принцип датируется теми днями, когда у Linux не было журналируемой файловой системы и сбои питания могли привести к катастрофе. По причинам обеспечения надежности и безопасности использование разделов осталось, так нарушение одной части системы, автоматически не означает, что весь компьютер в опасности. В настоящее время это самая важная причина для разметки. Простой пример: пользователь создает скритп, программу или веб-приложение, которые начинают заполнять диск. Если диск содержит только один большой раздел, вся система перестает работать, если диск заполнен. Если пользователь хранит данные на отдельном разделе, то только тот (с данными) раздел будет затронут, в то время как системные разделы и, возможно, другие разделы с данными сохранят функциональность.

Помните, что имеющаяся журналируемая файловая система обеспечивает только защиту данных в случае сбоя питания и неожиданного отключения устройств хранения. Она не защищает ваши данные от испорченных блоков и логических ошибок в файловой системе. В этих случаях, вам следует использовать

RAID (избыточный массив недорогих жёстких дисков) решение.

Схемы и типы разделения

Есть два вида основных разделов в системе Linux:

- *раздел с данными*: обычные данные системы Linux, включая *корневой раздел*, содержащий все данные для старта и запуска системы; и
- *раздел подкачки*: расширение физической памяти компьютера, представляет собой дополнительную память на жестком диске.

Большинство систем содержат корневой раздел, один или несколько разделов с данными, и один или несколько разделов подкачки. Системы в смешанных средах, могут содержать разделы данных других систем, такие как разделы файловых системам FAT или VFAT с данными MS Windows.

Большинство систем Linux во время установки используют **fdisk**, чтобы задать тип раздела. Как вы могли заметить в ходе упражнений к Главе 1, это обычно происходит автоматически. Но однажды вам может так не повести. В таких случаях вам придется выбрать тип раздела вручную и даже вручную сделать настоящее разбиение на разделы. Стандартные разделы Linux имеют номера 82 для раздела подкачки и 83 для данных, который может быть журналируемым (ext3) или обычным (ext2, на старых системах). Утилита **fdisk** имеет встроенную помощь, на случай если вы забудете эти значения.

Наряду с этими двумя, Linux поддерживает множество других типов файловых систем, такие как относительно новая файловая система Reiser, JFS, NFS, FATxx и многие другие файловые системы, изначально доступные на других (проприетарных) операционных системах.

Стандартный корневого раздела (обозначается одиночной косой чертой, /), составляет около 100-500 МБ и содержит системные конфигурационные файлы, большинство основных команд и серверные программы, системные библиотеки, некоторое временное пространство и домашний каталог пользователя с правами администратора. Стандартная установка для корневого раздела требуется около 250 МБ.

Пространство для подкачки (обозначается как *swap*) доступно только для самой системы, и скрыто из виду при обычной работе. Раздел подкачки - это механизм, который обеспечивает, как и на обычных системах UNIX, продолжение вашей работы, что бы ни случилось. Из-за этой дополнительной памяти в Linux вы практически никогда не увидите раздражающих сообщений типа *"Недостаточно памяти, пожалуйста закройте сначала некоторые приложения и попробуйте еще раз"*. Процедура подкачки или виртуальной памяти давно принята операционными системами уже вне мира UNIX.

Используемая память на жестком диске естественно медленнее, чем при использовании реальных микросхем памяти компьютера, но обеспечивает дополнительный комфорт. Мы узнаем больше о разделе подкачки, когда будем обсуждать процессы в Главе 4, Процессы.

Linux обычно рассчитывает на использование удвоенного количества физической памяти в пространстве подкачки на жестком диске. При установке системы, вы должны решить, как собираетесь это сделать. Например на системе с 512 Мб RAM:

- 1-й вариант: один раздел подкачки в 1 Гб
- 2-й вариант: два раздела подкачки по 512 МБ
- 3-й вариант: с двумя жесткими дисками: по одному разделу в 512 МБ на каждом диске.

Последний вариант даст лучшие результаты, when a lot of I/O is to be expected .

Читайте документации к программному обеспечению для специфичных установок. Некоторые приложения, такие как базы данных, могут требовать большего пространства подкачки. Другие, такие как портативные системы, могут вообще не иметь никакой подкачки из-за нехватки жесткого диска. Пространство для подкачки может также зависеть от версии вашего ядра.

Кроме того во многих дистрибутивах ядро находится на отдельном разделе, поскольку это самый важный файл вашей системы. Если это так, вы увидите, что у вас также есть раздел */boot*, содержащий ваше ядро (ядра) и сопутствующие файлы данных.

Остаток жесткого диска(ов) обычно делится на разделы данных, хотя может быть, что все не критичные для системы данные будут находиться на одном разделе, например, при выполнении стандартной установки рабочей станции. Когда некритичные данных разделяется на несколько разделов, то обычно это происходит по следующему принципу:

- раздел для пользовательских программ (*/usr*)
- раздел, содержащий персональные данные пользователей (*/home*)
- раздел для хранения временных данных, таких как очереди печати и почты (*/var*)
- раздел для дополнительного программного обеспечения (*/opt*)

После того как разделы сделаны, можно только добавить дополнительные. Изменение размеров или свойств существующих разделов возможно, но не рекомендуется.

Разделение жесткого диска на разделы определяется системным администратором. На больших системах он или она может даже развернуть один раздел на несколько жестких дисков, используя соответствующее ПО. Большинство дистрибутивов при стандартных установках имеют возможность оптимизироваться под рабочие станции (для обычных пользователей), под обычные цели сервера, но также допускают настройку разделов. Во время процесса установки вы можете определить вашу собственную схему разделов, используя либо специфический инструмент вашего дистрибутива, который обычно запускается в начале в графическом интерфейсе, или **fdisk**, инструмент текстового режима для создания разделов и установки их свойств.

Установка рабочей станции или клиентская установка осуществляется для использования главным образом одним и тем же человеком. Выбранное ПО для установки отражает это и акцент делается на общие пользовательские пакеты, такие как красивые темы рабочего стола, инструменты разработки, клиентские программы для работы с электронной почтой, мультимедийный софт, web и другие службы. Все это объединяется на одном большом разделе, добавляется пространство подкачки в два раза превышает объем оперативной памяти и ваша стандартная рабочая станция готова и обеспечивает наибольший объем дискового пространства для возможности личного пользования, но с недостатком возможной потери целостности данных во время проблемных ситуаций.

На сервере системные данные стремятся отделить от пользовательских данных. Программы различных служб хранятся отдельно от данных, которые они обрабатывают. На таких системах создаются различные разделы:

- раздел со всеми данными, необходимыми для загрузки машины
- раздел с конфигурационными данными и серверными программами
- один или несколько разделов, содержащих серверные данные, такие как таблицы базы данных, почта пользователей, FTP-архив и т.д.
- раздел с пользовательскими программами и приложениями

- один или несколько разделов для конкретных пользовательских файлов (домашние каталоги)
- один или несколько разделов подкачки (виртуальная память)

Службы обычно используют больше памяти и, следовательно, им нужно больше пространства подкачки. Некоторые процессы сервера, такие как связанные с базами данных, могут потребовать больше пространства подкачки, чем обычно; см. специальную документацию для подробной информации. Для повышения производительности swap часто разделяют на несколько разделов.

Точки монтирования

Все разделы подключаются к системе через точки монтирования. Точка монтирования определяет место расположения конкретных данных в файловой системе. Как правило все разделы связаны через раздел *root*. В этом разделе, который обозначается косой чертой (/), создаются каталоги. Эти пустые каталоги будут начальной точкой разделов, которые подключаются к нему. Например: дан раздел, содержащий следующие каталоги:

```
videos/  cd-images/ pictures/
```

Мы хотим подключить этот раздел к файловой системе в каталог /opt/media. Для того, чтобы сделать это, системный администратор должен убедиться, что каталог /opt/media существует в системе. Желательно, это должен быть пустой каталог. Как это делается объясняется далее в этой главе. Затем, используя команду **mount**, администратор может подключить раздел к системе. Если вы посмотрите на содержимое ранее пустой директории /opt/media, оно будет содержать файлы и каталоги, которые имеются на смонтированном носителе (жестком диске или разделе жесткого диска, CD, DVD, флэш-карте, USB или других устройствах хранения).

Во время запуска системы, именно так монтируются все разделы, которые описаны в файле /etc/fstab. Некоторые разделы не монтируются по умолчанию, если они постоянно не подключены к системе, например устройство хранения, используемое в вашей цифровой камере. Если все правильно настроено, то устройство будет смонтировано, сразу же как система замечает, что оно подсоединено, или оно может быть смонтировано пользователем, тогда вам не нужно быть системным администратором, чтобы подключать и отключать устройства к системе и от нее. Пример есть в Разделе "Использование rsync".

На работающей системе информацию о разделах и их точках монтирования можно получить с помощью команды **df** (которая показывает используемое пространство диска). В Linux команда **df** GNU варианта, и поддерживает опцию -h, которая значительно улучшает читаемость. Отметим, что на коммерческих системах UNIX есть свои собственные версии **df** и многих других команд. Обычно у них такое же поведение, хотя версии обычных инструментов от GNU, часто имеют больше функций и они лучше.

Команда **df** отображает только информацию об активных разделах (исключая раздел подкачки). Они могут включать разделы других доступных по сети систем, как в примере, приведенном ниже, где домашние каталоги монтируются с файлового сервера по сети, такая ситуация часто встречается в корпоративных средах.

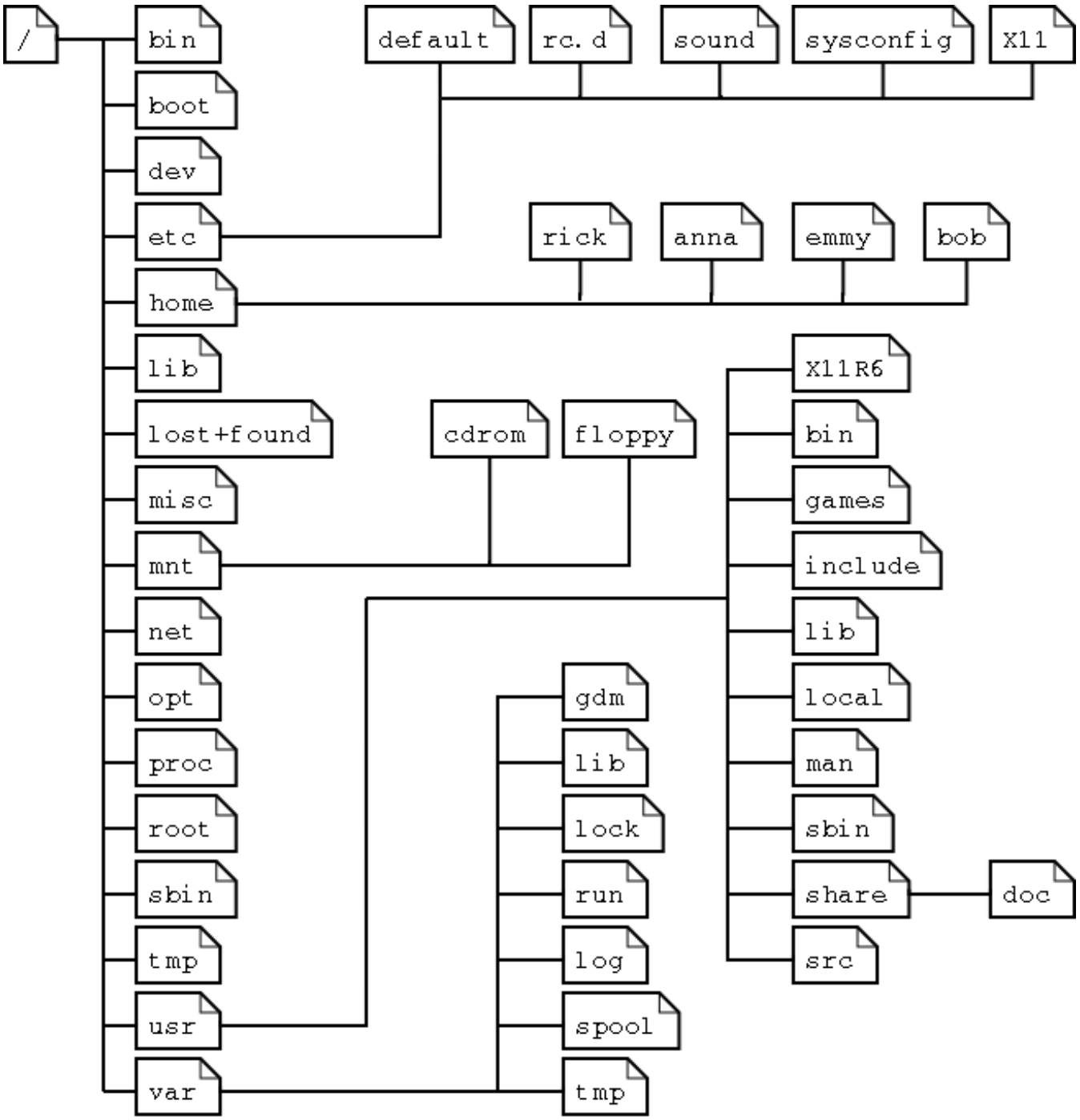
```
freddy:~> df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda8             496M  183M  288M  39% /
/dev/hda1             124M  8.4M  109M   8% /boot
/dev/hda5              19G   15G  2.7G  85% /opt
/dev/hda6             7.0G  5.4G  1.2G  81% /usr
/dev/hda7             3.7G  2.7G  867M  77% /var
fs1:/home             8.9G  3.7G  4.7G  44% /.automount/fs1/root/home
```

Подробнее о схеме файловой системы

Обзор

Для удобства файловая система Linux обычно представляется в виде древовидной структуры. В стандартной системе Linux вы обнаружите подобный перечень и расположение, как на схеме приведенной ниже.

Рисунок 3.1. Схема файловой системы Linux



Это схема из системы RedHat. В зависимости от системного администратора, операционной системы и назначения UNIX-машины, структура может меняться, и каталоги по желанию могут быть опущены или добавлены. Даже не обязательно соответствие имен, они лишь соглашение.

Дерево файловой системы начинается со *слэша*, обозначаемого наклоненной вперед косой чертой (/). Это каталог, содержащий все основные каталоги и файлы, также называется *корневой директорией* или "корнем" файловой системы.

Каталогам, которые ниже корневого каталога лишь на один уровень, зачастую предшествует слэш, для указания их положения и предотвращения путаницы с другими каталогами, которые могут иметь такое же имя. Знакомясь с новой системой, всегда хорошая идея - заглянуть в корневую директорию. Давайте посмотрим, с чем вы можете там столкнуться:

```
emmy:~> cd /
emmy:/> ls
bin/   dev/  home/    lib/         misc/  opt/     root/  tmp/  var/
boot/  etc/  initrd/  lost+found/  mnt/   proc/    sbin/  usr/
```

Таблица 3.2. Подкаталоги корневого каталога

Директория	Содержимое
/bin	Общие программы для совместного использования системой, системным администратором и пользователями.
/boot	Загрузочные файлы и ядро, vmlinuz. В некоторых последних дистрибутивах также данные grub. Grub – это большой единый загрузчик, который представляет собой попытку избавиться от многих различных загрузчиков известных нам на сегодняшний день.
/dev	Содержит ссылки на все периферийные устройства, которые представлены файлами с особыми свойствами.
/etc	Большинство важных системных файлов конфигурации находятся в /etc, этот каталог содержит данные, аналогичные тем что в Панели Управления Windows
/home	Домашние каталоги обычных пользователей.
/initrd	(в некоторых дистрибутивах) Информация для загрузки. Не удаляйте!
/lib	Файлы библиотек, включает файлы для всех разновидностей программ, необходимых системе и пользователям.
/lost+found	Каждый раздел имеет lost+found в его верхней директории. Здесь находятся файлы, которые были спасены во время сбоев.
/misc	Для разных целей.
/mnt	Стандартные точки монтирования для внешних файловых систем, например, CD-ROM'a или цифровой камеры.
/net	Стандартные точки монтирования для удаленных файловых систем
/opt	Как правило, содержит дополнительное ПО и ПО третьих сторон.

/proc	Виртуальная файловая система, содержащая информацию о системных ресурсах. Более подробная информация о назначении файлов в proc можно получить, введя команду **man** proc в окне терминала. Файл proc.txt рассматривает виртуальную файловую систему в деталях.
/root	Домашняя директория администратора. Помните о разнице между / (корневым каталогом) и /root (домашним каталогом пользователя *root*).
/sbin	Программы для использования системой и системным администратором.
/tmp	Временное место для использования системой, которое очищается после перезагрузки, так что не используйте ее под сохранение какой-нибудь работы!
/usr	Программы, библиотеки, документация и т.д. для всех пользовательских программ.
/var	Место хранения всех изменяемых и временных файлов, созданных пользователями, такие как log-файлы, почтовые очереди, the print spooler area, место для временного хранения файлов, загружаемых из Интернета, или сохранения образа CD перед записью.

Как вы можете узнать на каком разделе находится какой-нибудь каталог? Использование команды **df** с точкой (.) в качестве опции показывает раздел, которому принадлежит текущий каталог, и уведомляет о количестве используемого дискового пространства на этом разделе:

```
sandra:/lib> df -h .
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda7             980M  163M  767M  18% /
```

Как правило, каждый каталог в корневом каталоге находится на корневом разделе, за исключением когда он имеет отдельный вход в полным листинге команды **df** (или **df** -h без каких-либо других опций).

Узнайте больше в **man** hier.

Файловая система в реальности

Для большинства пользователей, и для большинства обычных задач системного администрирования, приемлемо допускать, что файлы и каталоги организованы в древовидную структуру. Однако компьютер ничего не понимает о деревьях или древовидных структурах.

Каждый раздел имеет свою собственную файловую систему. Представляя все эти файловые системы вместе, мы можем говорить о древовидной структуре всей системы, но все не так просто. В файловой системе, файл представлен с помощью *inode (индексного дескриптора)*, своего рода серийного номера, содержащего информацию о данных этого файла: кому принадлежит этот файл, и где он находится на жестком диске.

Каждый раздел имеет свой собственный набор индексных дескрипторов; на всей системе с несколькими разделами могут существовать файлы с одним и тем же номером индексного дескриптора.

Каждый inode описывает структуру данных на жестком диске, хранит информацию о свойствах файла, в том числе физическое местоположение его данных. Когда жесткий диск назначается для хранения данных (обычно во время начала процесса установки системы или при добавлении дополнительных дисков к существующей) в разделе создается определенное количество индексных дескрипторов. Этот число будет максимальным количеством файлов всех типов (в том числе каталогов, специальных файлов, ссылок и т.д.),

которые могут существовать в одно и то же время на данном разделе. Как правило, мы рассчитываем на 1 inode от 2 до 8 килобайт памяти.

Во время создания нового файла, он получает свободный inode. В этом индексном дескрипторе содержится следующая информация:

- Владельца и группа-владелец файла.
- Тип файла (обычный, каталог, ...)
- Разрешения на файл (Раздел "Права доступа: первая линия обороны Linux")
- Дата и время создания, последнего открытия и изменения.
- Дата и время, когда эта информация была изменена в индексном дескрипторе.
- Количество ссылок на этот файл (см. далее в этой главе).
- Размер файла
- Адрес, определяющий фактическое расположение данных файла.

Время создания и время изменения.
В man-страницах вы натолкнетесь на *atime* (время доступа), *ctime* (время изменения свойств файла) и *mtime* (время изменения данных (содержания) файла). Вы не всегда можете узнать время создания файла. Если вы измените права доступа на файл, ctime изменится также и больше не будет отражать реального времени создания.

Единственная информация, не включенная в индексный дескриптор, это имя файла и каталога. Они хранятся в каталогах, этих особенных файлах. Сопоставляя имена файлов и номера inode, система может составлять древовидную структуру, которая понятна пользователю. Пользователи могут отображать номера inode, используя опцию -i команды **ls**. Индексные дескрипторы имеют свои собственные отдельные места на диске.

Ориентация в файловой системе

Когда вы хотите, чтобы система выполнила команду, вам почти никогда не надо давать полный путь к этой команде. Например, мы знаем, что команда **ls** находится в каталоге /bin (проверьте командой **which** -a ls), тем не менее, мы не обязаны вводить команду **/bin/ls** в компьютер, чтобы получить список содержимого текущей директории.

Дело в том, что об этом заботится переменная окружения PATH. В ней перечисляются те каталоги системы, где могут быть найдены исполняемые файлы, что освобождает пользователя от излишнего ввода символов и запоминания местонахождения команд. Естественно PATH (путь) содержит множество каталогов, обычно содержащих где-то в своих именах bin, что показано ниже. Команда **echo** используется для отображения содержимого ("$") переменной PATH:

```
rogier:> echo $PATH
/opt/local/bin:/usr/X11R6/bin:/usr/bin:/usr/sbin/:/bin
```

В этом примере, каталоги /opt/local/bin, /usr/X11R6/bin, /usr/bin, /usr/sbin и /bin являются местами последовательного поиска нужной программы. Как только она найдена, поиск останавливается, даже если еще не каждый каталог в path проверен. Это может привести к странным ситуациям. В первом примере ниже пользователь знает, что есть программа под названием **sendsms** для отправления SMS сообщений, и другой пользователь в той же самой системе может ей пользоваться, но не первый пользователь. Разница

заключается в конфигурации переменной PATH:

```
[jenny@blob jenny]$ sendsms
bash: sendsms: command not found
[jenny@blob jenny]$ echo $PATH
/bin:/usr/bin:/usr/bin/X11:/usr/X11R6/bin:/home/jenny/bin
[jenny@blob jenny]$ su - tony
Password:
tony:~>which sendsms
sendsms is /usr/local/bin/sendsms

tony:~>echo $PATH
/home/tony/bin.Linux:/home/tony/bin:/usr/local/bin:/usr/local/sbin:\
/usr/X11R6/bin:/usr/bin:/usr/sbin:/bin:/sbin
```

Обратите внимание на использование средства **su** (переключение пользователей), которое позволяет запускать shell в среде другого пользователя, конечно при условии, что вы знаете пароль этого пользователя.

Обратная косая черта обозначает продолжение данной строки на другую и игнорирует разделение с помощью клавиши **Enter**.

В следующем примере, пользователь хочет вызвать команду **wc** (количество слов), чтобы проверить количество строк в файле, но ничего не происходит, и он вынужден прервать ее работу, используя комбинацию **Ctrl + C**:

```
jumper:~> wc -l test

(Ctrl-C)
jumper:~> which wc
wc is hashed (/home/jumper/bin/wc)

jumper:~> echo $PATH
/home/jumper/bin:/usr/local/bin:/usr/local/sbin:/usr/X11R6/bin:\
/usr/bin:/usr/sbin:/bin:/sbin
```

Использование команды **which** показывает нам, что у этого пользователя есть bin-каталог в его домашней директории, содержащей программу также под названием **wc**. Поскольку программа в домашнем каталоге находится первой при просмотре путей при вызове **wc**, выполняется эта «самодельная» программа, которая, вероятно, не понимает ввод, поэтому потребовалось ее завершить. Для решения проблем подобного рода есть несколько способов (вообще в UNIX/Linux всегда есть несколько вариантов решения проблем): одно решение — это переименовать программу **wc** пользователя, или пользователь может вводить полный путь к требуемой ему команде, который можно определить, используя опцию -a команды **which**.

Однако, если пользователь чаще использует программы из других каталогов, то может изменить последовательность просмотра каталогов на свой собственный вариант:

```
jumper:~> export PATH=/usr/local/bin:/usr/local/sbin:/usr/X11R6/bin:\
/usr/bin:/usr/sbin:/bin:/sbin:/home/jumper/bin
```

Изменения не постоянны!

Обратите внимание, что при использовании в командной строке команды **export**, изменения носят временный характер и действуют только в данной сессии (до выхода). Открытие новой сессии (даже если текущая первая по прежнему работает) не приводи к наличию в ней новых путей. Мы увидим в Разделе "Ваше текстовое окружение", как можно сделать такого рода изменения окружения постоянными, путем добавления этих строк в конфигурационные файлы оболочки.

Абсолютные и относительные пути

Путь, по которым вы должны следовать в древовидной структуре, чтобы получить заданный файл, может быть описан, начиная от ствола дерева (/ или корневого каталога). В этом случае путь начинается со слэша и называется абсолютным путем, поскольку не может быть никакого заблуждения: только один файл в системе может подойти под него.

В другом случае, путь не начинается со слэша и возможна путаница по сравнению с предыдущим вариантом между ~/bin/wc (в домашнем каталоге пользователя) и bin/wc в /usr. Пути, которые не начинаются со слэша всегда относительные.

В относительных путях мы также используем . и .., указывающие на текущий и родительский каталоги. Пара практических примеров:

- Если вы хотите компилировать исходный код, документация к установке часто инструктирует вас выполнить команду **./configure**, которая запускает программу *configure*, расположенную в текущем каталоге (и которая поставляется с новым кодом), а ни где-то в другом месте системы.
- В файлах HTML, относительных пути часто используются, чтобы сделать ряд страниц, легко переносимыми в другое место:

```
<img alt="Garden with trees" src="../images/garden.jpg">
```

- Обратите внимание на различие еще раз:

```
theo:~> ls /mp3
ls: /mp3: No such file or directory
theo:~> ls mp3/
oriental/  pop/  sixties/
```

Наиболее важные файлы и каталоги

Ядро

Ядро является сердцем системы. Оно управляет связью между основной аппаратурой и периферией. Ядро также гарантирует, что процессы и демоны (серверные процессы) начинаются и останавливаются строго в требуемое время. У ядра много других важных задач, так много, что специальная рассылка по разработки ядра существуют только по тому вопросу, где огромные объемы информации являются общими. Мы бы зашли слишком далеко, обсуждая ядро в деталях. Сейчас достаточно знать, что ядро – это самый важный файл в системе.

Оболочка (Shell)

Что такое shell?

Когда я смотрела соответствующее разъяснение по концепции оболочки, это дало мне больше хлопот, чем я

ожидала. Все доступные определения, от простого сравнения, что "оболочка - это руль автомобиля", к расплывчатому определению в руководстве Bash, где говорится, что "bash является sh-совместимым интерпретатором командного языка", до еще более неясного выражения "оболочка управляет взаимодействием между системой и ее пользователями". Shell гораздо больше, чем все это.

Самым лучшим сравнением для оболочки может быть представление о способе общения с компьютером, сравнение оболочки с языком. Большинство пользователей знают другой «язык», это язык «укажи-и-щелкни» графического рабочего стола. Но на этом языке компьютер задает беседу, а пользователю отведена пассивная роль выбора задач их тех, что представлены. Для программиста очень трудно включить в GUI-формат все опции и способы применения команд. Поэтому графические интерфейсы почти всегда менее способные, чем команды, стоящие за ним.

Оболочка, с другой стороны, это расширенный способ общения с системой, поскольку она позволяет вести двустороннюю беседу и проявлять инициативу. Оба партнера в сообщении равны, так что могут быть проверены новые идеи. Оболочка позволяет пользователю очень гибко управлять системой. Дополнительным преимуществом является то, что оболочка позволяет автоматизировать задачи.

Типы оболочек

Так же как люди знают различных языки и диалекты, компьютеру известны различные типы оболочек:

- **sh** или Bourne Shell: первая оболочка, которая до сих пор используются в системах UNIX. Это базовая оболочка, представляющая собой небольшую программу с незначительным числом функций. В POSIX-совместимом режиме bash будет эмулировать эту оболочку.
- **bash** или Bourne Again Shell: стандартная оболочка GNU, интуитивно понятная и гибкая. Наиболее предпочтительная для начинающих пользователей, и в то же время являющаяся мощным инструментом для опытных и профессиональных пользователей. На Linux **bash** – это стандартная оболочка обычных пользователей. Эта оболочка является так называемой расширенной Bourne shell, с множеством дополнений и плагинов. Это означает, что Bourne Again SHell совместима с Bourne shell: команды, которые работают в **sh**, также работают в **bash**. Однако, обратное не всегда соответствует действительности. Все примеры и упражнения в этой книге приведены с использованием **bash**.
- **csh** или C Shell: синтаксис этой оболочки напоминает синтаксис языка программирования Си. Иногда используется программистами.
- **tcsh** или Turbo C Shell: расширенный вариант обычной C Shell, обеспечивает большее удобство и скорость.
- **ksh** или оболочка Корна: иногда ценится людьми с опытом работы в UNIX. Расширение оболочки Борна; в стандартной конфигурации – кошмар для начинающих пользователей.

Файл /etc/shells дает обзор известных системе Linux оболочек:

```
mia:~> cat /etc/shells
/bin/bash
/bin/sh
/bin/tcsh
/bin/csh
```

 **Хитрость оболочки Борна.**
Обратите внимание, что /bin/sh, как правило, ссылка на Bash, которая будет при вызове по этому пути выполнять в оболочке Борна совместимый режим.

Ваша оболочка, которая используется по умолчанию, описывается в файле /etc/passwd, подобно этой строке для пользователя *mia*:

```
mia:L2NOfqdlPrHwE:504:504:Mia Maya:/home/mia:/bin/bash
```

Чтобы перейти из одной оболочки в другую, просто введите название новой оболочки в активном терминале. Система находит каталог, где встречается это имя с помощью установок PATH, и поскольку оболочка является исполняемым файлом (программой), текущая оболочка активирует ее, после чего она запустится. Обычно это отражает новое приглашение, т.к. у каждой оболочки оно имеет типичный внешний вид:

```
mia:~> tcsh
[mia@post21 ~]$
```

Какую оболочку я использую?

Если вы не знаете, какую оболочку используете, либо проверьте строку для вашей учетной записи в /etc/passwd, либо введите команду
echo $SHELL

Ваш домашний каталог

Ваш каталог является вашим местонахождением по умолчанию при подключении к системе. В большинстве случаев это подкаталог /home, хотя это можно изменить. Ваш домашний каталог может находиться на жестком диске удаленного файлового сервера, в этом случае домашнюю директорию можно найти в /nethome/your_user_name. В другом случае системный администратор может сделать выбор в пользу менее понятной схемы и ваша домашняя директория может оказаться на /disk6/HU/07/jgillard.

Каким бы ни был путь к вашей домашней директории, вы не должны слишком беспокоиться об этом. Правильный путь к вашей домашней директории хранится в переменной окружения HOME на случай, если какой-нибудь программе это потребуется. С помощью команды echo вы можете вывести на экран содержимое этой переменной:

```
orlando:~> echo $HOME
/nethome/orlando
```

В вашем домашнем каталоге вы можете делать все, что вам нравится. Вы можете поместить любое количество файлов в такое количество каталогов, как вам захочется, хотя, естественно, общий объем данных и файлов ограничивается размерами разделов и иногда тем, что системный администратор применил квоты для системы. Ограничение использования дискового пространства было обычной практикой, когда память на жестком диске оставалась дорогой. В наши дни такие ограничения встречаются почти исключительно в больших средах. Вы можете выяснить для себя, устанавливается ли ограничение с помощью команды **quota**:

```
pierre@lamaison:/> quota -v
Diskquotas for user pierre (uid 501): none
```

В случае, если квоты были установлены, вы получите список разделов, на которые распространяются ограничения, и размер последних. Превышение пределов может быть допущено во время льготного периода с меньшими ограничениями или их отсутствием. Детальную информацию можно получить с помощью команд **info** quota или **man** quota.

Нет Quota?

Если ваша система не может найти quota, то никаких ограничений в отношении использования файловой системы не применяется.

Ваш домашний каталог обозначается символом тильды (~), это сокращение для /path_to_home/user_name (домашнего каталога пользователя). Этот же путь хранится в переменной HOME, так что вам не нужно набирать лишнее для перехода в домашний каталог. Простое применение: переход из /var/music/albums/arno/2001 в images в вашем домашнем каталоге с помощью одной элегантной команды:

```
rom:/var/music/albums/arno/2001> cd ~/images

rom:~/images> pwd
/home/rom/images
```

Далее в этой главе мы поговорим о командах для управления файлами и каталогами, все это поможет сохранить порядок в вашем домашнем каталоге.

Наиболее важные конфигурационные файлы

Как мы уже отмечали ранее, подавляющее большинство конфигурационных файлов хранятся в директории /etc. Содержание файлов может быть просмотрено с помощью команды cat, которая отправляет текст файлов на стандартный вывод (обычно ваш монитор). Синтаксис прямо вперед вами:

cat file1 file2 ... fileN

В этом разделе мы постараемся дать обзор наиболее распространенных конфигурационных файлов. Это, конечно, не полный список. Добавление дополнительных пакетов может также привести к добавлению дополнительных файлов конфигурации в /etc. При чтении конфигурационных файлов, вы увидите, что они, как правило, очень хорошо прокомментированы и не требуют пояснений. У некоторых файлов также имеются man-страницы, которые содержат дополнительную документацию, например **man** group.

Таблица 3.3. Большинство распространенных конфигурационных файлов

Файл	Информация/служба
aliases	Файл почтовых псевдонимов для использования с почтовыми серверами Sendmail и Postfix. Запуск почтового сервера на каждой системе давно стал обычным делом в мире UNIX, и почти каждый дистрибутив Linux по-прежнему поставляется с пакетом Sendmail. В этом файле локальные имена пользователей сопоставляются с реальными именами, которые имеют место в E-mail адресах или других локальных адресах.
apache	Config-файлы для веб-сервера Apache.
bashrc	Общесистемный конфигурационный файл для Bourne Again Shell. Определяет возможности и псевдонимы для всех пользователей. У других оболочек могут быть свои собственные общесистемные конфигурационные файлы, к примеру cshrc.
Каталоги crontab и cron.*	Настройка задач, которые должны периодически выполняться - резервное копирование, обновления баз данных системы, очистка системы, изменяющиеся журналы и т.д
default	Параметры по умолчанию для некоторых команд, таких как **useradd**.

filesystems	Известные файловые системы: ext3, vfat, iso9660 и т.д.
fstab	Список разделов и их точек монтирования.
ftp*	Настройка FTP-сервера: кто может подключаться, какие части системы доступны и т.д.
group	Файл конфигурации для пользовательских групп. Используйте теневые утилиты **groupadd, groupmod** и **groupdel** для редактирования этого файла. Редактируйте вручную, только если точно знаете, что делаете.
hosts	Список машин, с которыми можно связаться по сети, но без использования службы доменных имен. Это не имеет ничего общего с сетевой конфигурацией системы, которая настраивается в /etc/sysconfig.
inittab	Информация для загрузки: режим, количество текстовых консолей и т.д.
issue	Информация о дистрибутиве (версия и/или информация о ядре).
ld.so.conf	Места файлов библиотек.
lilo.conf, silo.conf, aboot.conf и т.д.	Загрузочная информации для LInux LOader, системы для загрузки, которую в настоящее время постепенно вытесняет GRUB.
logrotate.*	Ротация журналов, система предотвращения накопления огромного количества лог-файлов.
mail	Каталог, содержащий инструкции для деятельности почтового сервера.
modules.conf	Конфигурация модулей, которые включают специальные функции (драйвера).
motd	Сообщение дня. Показывается каждому, кто подключается к системе (в текстовом режиме), может быть использована системным администратором для объявления о техническом обслуживании системы и т.д.
mtab	Смонтированные в данный момент файловые системы. Рекомендуется никогда не редактировать этот файл.
nsswitch.conf	Order in which to contact the name resolvers when a process demands resolving of a host name.
pam.d	Конфигурация модулей аутентификации.
passwd	Список локальных пользователей. Используйте теневые утилиты **useradd, usermod** и **userdel** для редактирования этого файла. Правьте вручную только, когда действительно знаете что делаете.
printcap	Устаревший, но по-прежнему часто используемый файл конфигурации принтера. Не изменяйте его вручную, если точно не знаете, что делаете.

profile	Система расширенной конфигурации среды shell: переменные, свойства по умолчанию новых файлов, ограничение ресурсов и т.д.
rc*	Каталоги, определяющие активные службы для каждой запущенной ступени.
resolv.conf	Последовательность, в которой связываются с DNS-серверами (только серверы доменных имен).
sendmail.cf	Главный конфигурационный файл сервера Sendmail.
services	Соединения, принятые на этой машине (открытые порты).
sndconfig или sound	Настройка звуковой карты и звуковых событий.
ssh	Каталог, содержащий конфигурационные файлы для защиты оболочки клиента или сервера.
sysconfig	Каталог, содержащий системные конфигурационные файлы: мышь, клавиатура, сеть, рабочий стол, системные часы, управление питанием т.д. (характерно для RedHat).
X11	Параметры графического сервера, X. RedHat использует XFree, что находит отражение в имени основного конфигурационного файла, XFree86Config. Также содержит общие каталоги оконных менеджеров, имеющихся в системе, например, **gdm, fvwm, twm** и т.д
xinetd.* или inetd.conf	Конфигурационные файлы для Интернет-сервисов, that are run from the system's (extended) Internet services daemon (servers that don't run an independent daemon).

В данном руководстве мы узнаем больше об этих файлах и изучим в деталях некоторые из них.

Наиболее распространенные устройства

Устройства, под которыми обычно понимается все, что принадлежит периферии ПК, что не является самим процессором, представлены в системе как запись в каталоге /dev. Одним из преимуществ этого UNIX-способа управления устройствами является то, что ни пользователю, ни системе не нужно беспокоиться о спецификации устройств.

Пользователи, которые не знакомы с Linux или UNIX в целом, часто перегруженные количеством новых имен и понятий, должны учиться. Поэтому приведем список обычных устройств, описанных в этом введении.

Таблица 3.4. Основные устройства

Имя	**Устройство**
cdrom	CD привод
console	Специальный вход для используемой в настоящее время консоли.
cua*	Последовательные порты

dsp*	Устройства для оцифровки и записи
fd*	Записи для большинства видов гибких дисков, по умолчанию это /dev/fd0, дисковод для дискет по 1,44 Мбайт.
hd[a-t][1-16]	Стандартная поддержка дисков IDE с максимальным количеством разделов для каждого.
ir*	Инфракрасные устройства
isdn*	Управление соединением ISDN
js*	Джойстики
lp*	Принтеры
mem	Память
midi*	MIDI-плеер
mixer* и music	Идеализированная модель миксера (смешивает или добавляет сигналы)
modem	Модем
mouse (также msmouse, logimouse, psmouse, input/mice, psaux)	Все виды мышей
null	Бездонный ящик для мусора
par*	Записи для поддержки параллельных портов
pty*	Псевдотерминалы
radio*	Для радиолюбителей (HAM).
ram*	Загрузочное устройство
sd*	SCSI диски с их разделами
sequencer	Для аудио приложений, использующих особенности синтезатора звуковой карты (контроллер MIDI-устройства).
tty*	Виртуальные консоли моделирования терминалов vt100.
usb*	USB карта и сканер.
video*	Для использования с графической картой, поддерживающей видео.

Наиболее распространенные временные файлы

В каталоге /var мы находим множество каталогов, предназначенных для хранения особых непостоянных данных (в отличие от конфигурационных файлов системы, которые меняют сравнительно редко или никогда вообще). Все файлы, которые часто изменяются, например, лог-файлы, почтовые ящики, заблокированные файлы, очередь печати и т.д., хранятся в подкаталоге /var.

В качестве меры безопасности эти файлы обычно хранят отдельно от основных системных файлов, поэтому мы можем закрыть на них глаза, а где нужно устанавливать более строгие разрешения. Многие из этих файлов также требуют больше прав, чем обычно, как /var/tmp, который должен быть доступен для записи для всех. Основную массу пользовательской активности можно обнаружить именно здесь, которая даже может генерироваться анонимными интернет-пользователями, подключенными к вашей системе. Это одна из причин, почему каталог /var, включая все его подкаталоги, обычно делают на отдельном разделе. Таким образом, например, пытаются избежать риска, ведь может случиться почтовая бомба, заполняющая свободную часть файловой системы, содержащей более важные данные, такие как ваши программы и файлы конфигурации.

/var/tmp и /tmp

Файлы в /tmp могут быть удалены без уведомления в процессе выполнения очередных задач системы или в результате перезагрузки. В некоторых (индивидуально настроенных) системах /var/tmp может вести себя непредсказуемо. Тем не менее, поскольку это не так по умолчанию, мы рекомендуем использовать директорию /var/tmp для сохранения временных файлов. Если есть сомнения, обратитесь к системному администратору. Если вы управляете вашей системой сами, то можете быть уверены в безопасности этого места, если целенаправленно не изменяли настройки для /var/tmp (с правами администратора, обычный пользователь этого сделать не может).
Что бы вы не делали, старайтесь придерживаться привилегий, предоставленных обычному пользователю - не сохраняйте файлы непосредственно в корневой каталог (/) файловой системы, не помещайте их в каталог /usr или или его подкаталог или в другое непредназначенное место. Это в значительной степени ограничивает ваш доступ к безопасной файловых систем.

Одной из основных систем безопасности в UNIX, и конечно в каждом дистрибутиве Linux, является журнал учета объектов, который записывает все действия пользователя, процессы, системные события и т.д. Файл конфигурации так называемого syslogdaemon определяет, какая и как долго вошедшая информация будет храниться. По умолчанию все журналы находятся в каталоге /var/log, содержащем различные файлы журналов доступа, серверов, системных сообщений и т.д.

В /var мы обычно находим серверные данные, которые сохраняются здесь с целью отделить их от критически важных данных, таких как сама программа-сервер и ее конфигурационные файлы. Типичным примером в системах Linux является /var/www, который содержит настоящие страницы HTML, скрипты и изображения, которые передал веб-сервер. FTP-дерево сервера FTP (данные, которые могут быть загружены удаленным клиентом) также лучше хранить в одном из подкаталогов /var. Эти данные общедоступны и часто изменяемы анонимными пользователями, поэтому безопаснее их держать здесь, подальше от разделов или каталогов с конфиденциальными данными.

На большинстве установленных рабочих станций /var/spool как минимум будет содержать at и каталог cron, содержащий запланированные задания. В офисных окружениях этот каталог также обычно содержит lpd, который хранит очередь(и) печати и далее файлы конфигурации принтера, а также log-файлы принтера.

На серверных системах мы будем обычно находить /var/spool/mail, содержащий входящую почту для локальных пользователей, отсортированную в один файл для каждого пользователя, "почтовый ящик пользователя". Связанный каталог mqueue, диспетчер очереди для неотправленных почтовых сообщений. Эти компоненты системы могут быть сильно задействованы на почтовых серверах с большим количеством пользователей. Новости серверов также используют пространство /var/spool из-за огромного количества

сообщений, которые они должны обработать.

Каталог /var/lib/rpm специфичен для базирующихся на RPM (RedHat Package Manager) дистрибутивов; это те, в которых информация сохраняется в RPM-пакетах. Другие менеджеры пакетов обычно также сохраняют их данные где-то в /var.

Управление файлами

Просмотр свойств файла

Дополнительная информация о ls

Помимо имени файла команда **ls** может дать много другой информации, например, тип файла (это мы уже обсуждали). Она может также показать права доступа к файлу, его размер, номер индексного дескриптора, дату и время создания, владельцев и количество ссылок на файл. Также использование **ls** вместе с опцией -a может отобразить файлы, обычно скрытые от глаз. Имена таких файлов начинается с точки. Типичный пример — это конфигурационные файлы в вашем домашнем каталоге. Когда вы проработаете в определенной системе некоторое время, то заметите, что были созданы десятки файлов и каталогов, that are not automatically listed in a directory index. Наряду с этим, каждый каталог содержит файл, именуемый просто точкой (.) и один с двумя точками (..), которые используются в сочетании с номером их индексного дескриптора для определения расположения каталога в древовидной структуре файловой системы.

Вам действительно следует прочитать страницы info о команде **ls**, так как это часто используемая команда с большим количеством полезных опций. Опции могут быть скомбинированы, как и в случае с большинством команд UNIX и их параметрами. Часто используемая комбинация **ls** -al; она отображает длинный список файлов и их свойств, а также пути, на которые указывают символические ссылки. **ls** -latr отображает те же файлы, только теперь в обратном порядке по признаку последнего изменения, так что файл, измененный в самое последнее время, находится в нижней части списка. Вот несколько примеров:

```
krissie:~/mp3> ls
Albums/  Radio/  Singles/  gene/  index.html

krissie:~/mp3> ls -a
./   .thumbs  Radio     gene/
../  Albums/  Singles/  index.html

krissie:~/mp3> ls -l Radio/
total 8
drwxr-xr-x    2 krissie krissie  4096 Oct 30  1999 Carolina/
drwxr-xr-x    2 krissie krissie  4096 Sep 24  1999 Slashdot/

krissie:~/mp3> ls -ld Radio/
drwxr-xr-x    4 krissie krissie  4096 Oct 30  1999 Radio/

krissie:~/mp3> ls -ltr
total 20
drwxr-xr-x    4 krissie krissie  4096 Oct 30  1999 Radio/
-rw-r--r--    1 krissie krissie   453 Jan  7  2001 index.html
drwxrwxr-x   30 krissie krissie  4096 Oct 20 17:32 Singles/
drwxr-xr-x    2 krissie krissie  4096 Dec  4 23:22 gene/
```

```
drwxrwxr-x   13 krissie krissie  4096 Dec 21 11:40 Albums/
```

В большинстве версий Linux **ls** является псевдонимом «цветного **ls**» по умолчанию. Это свойство позволяет увидеть тип файла без использования какой-либо опции **ls**. Для достижения этого каждому типу файла присваивается свой собственный цвет. Стандартная схема находится в /etc/DIR_COLORS:

Таблица 3.5. Цветовая схема color-ls по умолчанию

Цвет	Тип файла
Голубой	каталоги
Красный	сжатые архивы
Белый	текстовые файлы
Розовый	изображения
Голубой	ссылки
Желтый	устройства
Зеленый	исполняемые файлы
Мигающий красный	неисправные ссылки

Больше информации найдется на man-странице. В прежние дни та же информация отображалась с помощью суффиксов для каждого имени нестандартного файла. При использовании моно-цвета (например, печати списка содержимого каталога) и для общей очевидности эта схема используется до сих пор:

Таблица 3.6. Схема суффиксов по умолчанию для команды ls

Символ	Тип файла
Ничего	Обычный файл
/	Каталог
*	Исполняемый файл
@	Ссылка
=	Сокет
|	Именованный канал

Полное описание функциональности и особенностей команды **ls** можно узнать с помощью **info** coreutils ls.

Дополнительные инструменты

Чтобы узнать больше о типе данных, с которыми имеем дело, мы используем команду **file**. Применяя

определенные тесты, которые проверяют свойства файла в файловой системе, «магические числа» и делают «лингвистические пробы», **file** пытается догадаться о формате файла. Вот некоторые примеры:

```
mike:~> file Documents/
Documents/: directory

mike:~> file high-tech-stats.pdf
high-tech-stats.pdf: PDF document, version 1.2

mike:~> file Nari-288.rm
Nari-288.rm: RealMedia file

mike:~> file bijlage10.sdw
bijlage10.sdw: Microsoft Office Document

mike:~> file logo.xcf
logo.xcf: GIMP XCF image data, version 0, 150 x 38, RGB Color

mike:~> file cv.txt
cv.txt: ISO-8859 text

mike:~> file image.png
image.png: PNG image data, 616 x 862, 8-bit grayscale, non-interlaced

mike:~> file figure
figure: ASCII text

mike:~> file me+tux.jpg
me+tux.jpg: JPEG image data, JFIF standard 1.01, resolution (DPI),
            "28 Jun 1999", 144 x 144

mike:~> file 42.zip.gz
42.zip.gz: gzip compressed data, deflated, original filename,
         `42.zip', last modified: Thu Nov  1 23:45:39 2001, os: Unix

mike:~> file vi.gif
vi.gif: GIF image data, version 89a, 88 x 31

mike:~> file slide1
slide1: HTML document text

mike:~> file template.xls
template.xls: Microsoft Office Document

mike:~> file abook.ps
abook.ps: PostScript document text conforming at level 2.0

mike:~> file /dev/log
/dev/log: socket
```

```
mike:~> file /dev/hda
/dev/hda: block special (3/0)
```

Команда **file** имеет ряд опций, в частности -z, которая просматривает сжатые файлы. Чтобы получить более детальное описание, посмотрите **info** file. Имейте в виду, что результаты выполнения команды **file** не являются абсолютно достоверными, это только предположение. Другими словами, **file** может ошибиться.

Зачем весь этот шум о типах файлов и форматах?

В ближайшее время мы обсудим пару инструментов командной строки для просмотра текстовых файлов. Эти инструменты не будут работать при использовании «неправильного» типа файла. В худшем случае, они создадут аварийную ситуацию в вашем терминале и/или приведут к возникновению множества звуковых сигналов. Если у вас это случится, просто закройте сеанс терминала и начните новый. Но постарайтесь избегать этого, т.к. обычно это нервирует других людей.

Создание и удаление файлов и каталогов

Создание беспорядка...

... к этому легко придти. На сегодняшний день практически каждая система работает в сети, поэтому естественно файлы копируются с одного компьютера на другой. Особенно это происходит при работе в графической среде, где создание новых файлов часто происходит без согласия пользователя. Для иллюстрации этой проблемы здесь полное содержимое каталога нового пользователя, созданного на стандартной системе RedHat:

```
[newuser@blob user]$ ls -al
total 32
drwx------   3 user  user        4096 Jan 16 13:32 .
drwxr-xr-x   6 root     root        4096 Jan 16 13:32 ..
-rw-r--r--   1 user  user      24 Jan 16 13:32 .bash_logout
-rw-r--r--   1 user  user     191 Jan 16 13:32 .bash_profile
-rw-r--r--   1 user  user     124 Jan 16 13:32 .bashrc
drwxr-xr-x   3 user  user    4096 Jan 16 13:32 .kde
-rw-r--r--   1 user  user    3511 Jan 16 13:32 .screenrc
-rw-------   1 user  user      61 Jan 16 13:32 .xauthDqztLr
```

Также на первый взгляд содержание "используемого" домашнего каталога не выглядит так плохо:

```
olduser:~> ls
app-defaults/ crossover/   Fvwm@     mp3/      OpenOffice.org638/
articles/     Desktop/     GNUstep/  Nautilus/ staroffice6.0/
bin/          Desktop1/    images/   nqc/      training/
brol/         desktoptest/ Machines@ ns_imap/  webstart/
C/            Documents/   mail/     nsmail/   xml/
closed/       Emacs@       Mail/     office52/ Xrootenv.0
```

Но если учесть все каталоги и файлы, начинающиеся с точки, то в данном каталоге уже обнаруживается 185 объектов. Причина этого в том, что у большинства приложений есть их собственные каталоги и/или файлы в домашнем каталоге этого пользователя, содержащие пользовательские настройки. Как правило, эти файлы создаются при первом запуске приложения. В некоторых случаях вы будете уведомлены, когда несуществующая директория должна быть создана, но по большей части все будет делаться автоматически.

К тому же новые файлы создаются, по-видимому, беспрерывно, потому что пользователи хотят сохранять файлы, хранить различные версии своих работ, использовать интернет-приложения, а также загружать файлы и сопутствующие материалы на их локальный компьютер. Это не остановить. Ясно одно, безусловно потребуются инструменты для сохранения порядка.

В следующем разделе мы обсудим имеющиеся средства для поддержания порядка. Мы только обсуждаем текстовые инструменты доступные в shell, поскольку графические инструменты интуитивны, выглядят и ощущаются также как известный метод «указать-и-щелкнуть» файлового менеджера MS Windows, в том числе графические функции помощи и другие свойства, которые вы ожидаете от такого рода приложений. Ниже приведен обзор наиболее популярных файловых менеджеров для GNU/Linux. Большинство файловых менеджеров могут быть запущены из меню вашей графической среды, щелчком по иконке домашнего каталога или из командной строки с помощью приведенных команд:

- **nautilus**. Файловый менеджер Gnome (графической среды GNU) по умолчанию. Отличную документацию о работе с этим инструментом можно найти на http://www.gnome.org.
- **konqueror**. Файловый менеджер, обычно используемый в графической среде KDE. Пособие есть на http://docs.kde.org.
- **mc**. Midnight Commander, файловый менеджер Unix по образу Norton Commander. Вся документация доступна на http://gnu.org/directory/ или через зеркало, такое как http://www.ibiblio.org.

Несомненно эти приложения стоит попробовать, и они обычно впечатляют новичков Linux даже тем, что существует такой широкий выбор: это только самые популярные инструменты для управления каталогами и файлами, также многие другие проекты находятся в разработке. Теперь давайте выясним, что за всем этим стоит, и посмотрим, как эти графические инструменты используют стандартные команды UNIX.

Инструменты

Создание каталогов

Способ хранения объектов на своих местах - это помещение определенных файлов в специально для них созданные каталоги и подкаталоги (или, если хотите, папки и подпапки). Это делается с помощью команды **mkdir**:

```
richard:~> mkdir archive

richard:~> ls -ld archive
drwxrwxrwx  2 richard richard           4096 Jan 13 14:09 archive/
```

Создание каталогов и подкаталогов за один шаг делается с помощью опции -p:

```
richard:~> cd archive

richard:~/archive> mkdir 1999 2000 2001

richard:~/archive> ls
1999/  2000/  2001/

richard:~/archive> mkdir 2001/reports/Restaurants-Michelin/
mkdir: cannot create directory `2001/reports/Restaurants-Michelin/':
No such file or directory
```

```
richard:~/archive> mkdir -p 2001/reports/Restaurants-Michelin/

richard:~/archive> ls 2001/reports/
Restaurants-Michelin/
```

Если у нового файла должны быть другие права, чем те, что присваиваются по умолчанию при его создании, то новые права доступа могут быть установлены одним движением, также используя команду **mkdir**, см. страницы info для дополнительной информации. Мы собираемся обсудить режимы доступа в следующем разделе, посвященном безопасности файлов.

Имя каталога должно соответствовать тем же нормам, которые применяются для имен обычных файлов. Одним из наиболее важных ограничений является то, что у вас не может быть двух файлов с одинаковым названием в одном каталоге (но имейте в виду, операционная система Linux, как и UNIX, чувствительна к регистру). В сущности нет никаких ограничений на длину имени файла, но обычно оно содержит меньше, чем 80 символов, т. е. может поместиться в одной строке терминала. Вы можете использовать в имени файла любой символ, который вам захочется, хотя рекомендуется исключить символы, которые имеют специальное значение для оболочки. При сомнении сверьтесь с Приложением C, Особенности Shell.

Перемещение файлов

Теперь, когда мы должным образом создали структуру в нашем домашнем каталоге, настало время уборки «несекретных» файлов с использованием команды **mv**:

```
richard:~/archive> mv ../report[1-4].doc reports/Restaurants-Michelin/
```

Эта же команда используется для переименования файлов:

```
richard:~> ls To_Do
-rw-rw-r--    1 richard richard      2534 Jan 15 12:39 To_Do

richard:~> mv To_Do done

richard:~> ls -l done
-rw-rw-r--    1 richard richard      2534 Jan 15 12:39 done
```

Очевидно, что изменяется только имя файла. Все остальные свойства остаются неизменными.

Подробные сведения о синтаксисе и особенностях команды **mv** можно найти в man или info-страницах. Использование данной документации должно всегда быть вашим первым рефлексом при столкновении с проблемой. Вероятно ответ на ваш вопрос найдется в системной документации. Даже опытные пользователи читают man-страницы каждый день, поэтому начинающим пользователям следует читать их все время. Через некоторое время, вы узнаете, наиболее распространенные опции к часто используемым командам, но вам все равно будет нужна документация как основной источник информации. Обратите внимание, что информация, содержащаяся в HOWTO, FAQ, man-страницах и других источниках постепенно сливается на страницах info, которые на сегодняшний день являются наиболее современными источниками online-документации (также как и легко доступным в системе).

Копирование файлов

Копирование файлов и каталогов осуществляется с помощью команды **cp**. Есть полезная опция рекурсивного копирования (копирования всех файлов и подкаталогов), используйте опцию -R команды **cp**.

Общий синтаксис

cp [-R] fromfile tofile

В качестве примера рассмотрим случай пользователя *newguy*, который захотелось иметь такие же настройки рабочего стола Gnome, какие есть у пользователя *oldguy*. Одним из способов решения проблемы является копирование настроек *oldguy* в домашний каталог *newguy*:

```
victor:~> cp -R ../oldguy/.gnome/ .
```

Это выдает некоторые ошибки, связанные с разрешением на файлы, но все эти ошибки связаны с личными файлами, в которых *newguy* в любом случае не нуждается. В следующем разделе мы обсудим, как изменить эти разрешения, если это действительно является проблемой.

Удаление файлов

Используйте команду rm для удаления отдельных файлов, **rmdir** для удаления пустых каталогов. (Используйте **ls** -a для проверки пуст каталог или нет). Команда **rm** также имеет опции для удаления непустых каталогов со всеми их подкаталогами, но об этих опасных опциях читайте, пожалуй, info-страницы.

Как пустота может быть каталогом?
Это нормально, что каталоги . (точка) и .. (точка-точка) не могут быть удалены, т.к. они также являются необходимыми в пустой директории для определения расположения каталогов в иерархии файловой системы.

В Linux, как и в UNIX, нет мусорных ящиков, по крайней мере, для shell, хотя есть множество решений для графического использования. Так однажды удаленный, файл действительно исчезает, и, как правило, уже нет возможности вернуть его обратно, кроме случаев, когда у вас есть резервные копии, или вы очень быстры и у вас действительно хороший системный администратор. Чтобы защитить начинающих пользователей от этой напасти, интерактивное поведение команд **rm, cp** и **mv** может быть запущено с помощью опции -i. В этом случае система не будет сразу действовать после запроса. Вместо этого, появится запрос на подтверждение, что потребует дополнительного нажатия клавиши Enter для причинения вреда:

```
mary:~> rm -ri archive/
rm: descend into directory `archive'? y
rm: descend into directory `archive/reports'? y
rm: remove directory `archive/reports'? y
rm: descend into directory `archive/backup'? y
rm: remove `archive/backup/sysbup200112.tar'? y
rm: remove directory `archive/backup'? y
rm: remove directory `archive'? Y
```

Мы обсудим, как сделать это опцией по умолчанию в Главе 7, «Дом сладкий /home» , в которой обсуждается настройка окружения вашей оболочки.

Поиск файлов

Использование возможностей shell

В приведенном ранее примере перемещения файлов мы уже видели, как shell может манипулировать сразу

несколькими файлами. В том примере оболочка автоматически понимает, что пользователь имел ввиду требованием между квадратными скобками "[" и "]". Оболочка может заменить диапазоны номеров, а также прописных или строчных буквы. Она также заменяет звездочку на любое количество символов и знак вопроса на любой один.

Все виды замещения могут быть использованы одновременно; оболочка очень последовательна в этом. Оболочка Bash, например, не имеет никаких проблем с такими выражениями как **ls** dirname/*/*/*[2-3].

В других оболочках, звездочка обычно используется, чтобы свести к минимуму усилия ввода: можно ввести **cd** dir* вместо **cd** directory. В Bash однако, это не является необходимым, поскольку у оболочки GNU есть свойство, называемое автозавершением имени файла. Это означает, что вы можете набрать первые несколько символов команды или файла (в текущем каталоге), и если никакой путаницы не может быть, оболочка поймет, что вы имеете в виду. Например, в каталоге, содержащем много файлов, вы можете проверить, если ли какие-нибудь файлы, начинающиеся с буквы A просто набрав **ls** A и нажать клавишу **Tab** дважды, вместо нажатия **Enter**. Если только один файл начинается с "A", то этот файл немедленно отобразится в качестве аргумента **ls** (или любой другой команды оболочки, если на то пошло).

Which

Самый простой способ просматривать файлы — это использование команды **which** для посмотра в каталогах, перечисленных в пути поиска пользователя на требуемый файл. Конечно, путь поиска содержит только пути к каталогам, содержащим исполняемые программы, которые не работают для обычных файлов. Команда **which** полезна при выяснении причины возникновения проблемы "Команда не найдена" ("Command not Found"). В приведенном ниже примере пользователь *tina* не может использовать программу **acroread**, а ее коллега не испытывает никаких проблем на той же системе. Эта проблема аналогична проблеме PATH из предыдущей части: коллега *tina* сказал ей, что он может увидеть нужную программу в /opt/acroread/bin, но этого каталога нет в ее path:

```
tina:~> which acroread
/usr/bin/which: no acroread in (/bin:/usr/bin:/usr/bin/X11
```

Проблема может быть решена путем предоставления полного пути к команде запуска, или нужно заново экспортировать содержание переменной PATH:

```
tina:~> export PATH=$PATH:/opt/acroread/bin

tina:~> echo $PATH
/bin:/usr/bin:/usr/bin/X11:/opt/acroread/bin
```

Используя команду **which** также можно проверить, является ли команда псевдонимом другой команды:

```
gerrit:~> which -a ls
ls is aliased to `ls -F --color=auto'
ls is /bin/ls
```

Если это не сработает в вашей системе, используйте псевдоним команды:

```
tille@www:~/mail$ alias ls
alias ls='ls —color'
```

Find и locate

Кроме перечисления в path, есть инструменты, специально предназначенные для поиска. Есть очень

мощный инструмент **find**, известный из UNIX, который можно расценить, как имеющий достаточно трудный синтаксис. GNU **find**, однако, имеет дело с синтаксическими проблемами. Эта команда позволяет не только вести поиск имен файлов, она также может принимать размер файла, дату последнего изменения и другие свойства файла в качестве критериев для поиска. Наиболее часто используется поиск файлов по именам:

```
find <path> -name <searchstring>
```

Это может быть истолковано как "Просмотреть все файлы и подкаталоги, содержащихся в заданном пути, и выдать имена файлов, содержащих строку поиска в их имени" (но не в содержании).

Другое применение **find** поиск файлов по определенному размеру, как и в примере ниже, где пользователь *peter* хочет найти все файлы в текущей директории или одном из ее подкаталогов, которые больше 5 MB:

```
peter:~> find . -size +5000k
psychotic_chaos.mp3
```

Если вы покопаетесь в man-страницах, то увидите, что **find** также может совершать действия над найденными файлами. Типичным примером является удаление файлов. Лучше всего первый тест совершить без опции -exec, когда нужные файлы отобраны, после чего команда может быть повторно запущена для удаления выбранных файлов. Ниже, мы ищем файлы с расширением .tmp:

```
peter:~> find . -name "*.tmp" -exec rm {} \;

peter:~>
```

Оптимизация!

Эта команда будет вызывать **rm** так много раз, сколько подходящих к требованию файлов найдется. В худшем случае, это может быть тысячи или миллионы раз. Это довольно сильно загрузит вашу систему. Более реалистичным подходом к работе будет использование канала (|) и инструмента **xargs** с командой **rm** в качестве аргумента. Таким образом, команда **rm** вызывается только, когда командная строка заполнена, а не для каждого файла. См. Главу 5, «Перенаправление ввода/вывода (I/O)» для дополнительной информации об использовании I/O перенаправления для облегчения повседневных задач.

Позднее (в 1999 году в след за man-страницами, после 20 лет существования **find**), была разработана **locate**. Эта программа проще в использовании, но более ограничена, чем **find**, т.к. ее выдача основана на базе данных файловых индексов, который обновляется только раз в день. С другой стороны, поиск в базе данных **locate** использует меньше ресурсов, чем **find** и, следовательно, показывает результаты практически мгновенно.

В наши дни большинство дистрибутивов Linux используют **slocate** (поиск снабженный безопасностью), современная версия **locate** позволяет пользователям получать выдачу, которую они не имеют права читать. Так, например, файлы из домашнего каталога root, обычно не доступны для публичного просмотра. Пользователь, который хочет найти того, кто знает об оболочке C, может задать команду **locate** .cshrc, для просмотра всех пользователей, у которых конфигурационный файл настроен на оболочку C. Если предположить, что пользователи *root* и *jenny* находятся под управлением оболочки C, то только файл /home/jenny/.cshrc будет отображен, и не один из домашней директории *root*. На большинстве систем, **locate** является символической ссылкой на программу **slocate**:

```
billy:~> ls -l /usr/bin/locate
lrwxrwxrwx 1 root slocate  7 Oct 28 14:18 /usr/bin/locate -> slocate*
```

Пользователь *tina* могла бы использоваться **locate**, чтобы найти необходимое приложение:

```
tina:~> locate acroread
/usr/share/icons/hicolor/16x16/apps/acroread.png
/usr/share/icons/hicolor/32x32/apps/acroread.png
/usr/share/icons/locolor/16x16/apps/acroread.png
/usr/share/icons/locolor/32x32/apps/acroread.png
/usr/local/bin/acroread
/usr/local/Acrobat4/Reader/intellinux/bin/acroread
/usr/local/Acrobat4/bin/acroread
```

Каталоги, которые не содержат имя bin
, не могут содержать программы - они не содержат исполняемые файлы. Есть три возможности. Файл в /usr/local/bin
и есть то, что хотела бы *tina*: это ссылка на shell-скрипт, который запускает настоящую программу:

```
tina:~> file /usr/local/bin/acroread
/usr/local/bin/acroread: symbolic link to ../Acrobat4/bin/acroread

tina:~> file /usr/local/Acrobat4/bin/acroread
/usr/local/Acrobat4/bin/acroread: Bourne shell script text executable

tina:~> file /usr/local/Acrobat4/Reader/intellinux/bin/acroread
/usr/local/Acrobat4/Reader/intellinux/bin/acroread: ELF 32-bit LSB
executable, Intel 80386, version 1, dynamically linked (uses
shared libs), not stripped
```

Для того, чтобы сохранить путь как можно короче, и системе не приходилось искать слишком долго каждый раз, когда пользователь хочет выполнить команду, мы добавляем /usr/local/bin
к пути, а не другие каталоги, которые содержат только двоичные файлы одной конкретной программы, также /usr/local/bin содержит другие полезные программы.

Опять же, полное описание возможностей **find** и **locate** можно найти на страницах info.

Команда grep

Общая строка фильтрации

Простая, но мощная программа **grep** используется для фильтрации входных строк и выдачи подходящих к образцу на вывод. Существуют буквально тысячи применений для программы **grep**. В приведенном ниже примере, *jerry* использует **grep**, чтобы посмотреть, что он делал с **find**:

```
jerry:~> grep -a find .bash_history
find . -name userinfo
man find
find ../ -name common.cfg
```

История поиска.
Также полезна в этих случаях функция поиска в **bash**, которая сразу же запускается нажатием **Ctrl + R**, как в примере, где мы хотим проверить еще раз, что мы делали в прошлом с командой **find**:

```
thomas ~> ^R
(reverse-i-search)`find': find `/home/thomas` -name *.xml
```

Введите строку поиска для начала поиска. Чем больше символов вы вводите, тем более ограниченный поиск получаете. Здесь читается история команды для этой сессии оболочки (которая записывается в .bash_history в вашем домашнем каталоге при выходе из той сессии). Показывается самый последний случай строки вашего поиска. Если вы хотите увидеть предыдущие команды, содержащие ту же строку, нажмите **Ctrl + R** опять.

Чтобы узнать больше, изучайте info-страницы в **bash**.

Все ОС Unix имеют он-лайн словарь. Так же и Linux. Словарь представляет собой список известных слов в файле под названием words, расположенный в /usr/share/dict. Чтобы быстро проверить правильность написания слова, не графическое приложение, необходимо:

```
william:~> grep pinguin /usr/share/dict/words

william:~> grep penguin /usr/share/dict/words
penguin
penguins
```

Словарь против списка слов.
Некоторые дистрибутивы предлагают команду **dict**, которая предоставляет больше функций, чем просто поиск слов в списке.

Кто является владельцем того домашнего каталога рядом со мной? Эй, есть его номер телефона!

```
lisa:~> grep gdbruyne /etc/passwd
gdbruyne:x:981:981:Guy Debruyne, tel 203234:/home/gdbruyne:/bin/bash
```

И какой был адрес электронной почты Arno также?

```
serge:~/mail> grep -i arno *
sent-mail: To:
sent-mail: On Mon, 24 Dec 2001, Arno.Hintjens@celeb.com wrote:
```

find и **locate** часто используются в сочетании с grep для решения некоторых важных задач. Для получения дополнительной информации см. Главу 5, «Перенаправление ввода/вывода (I/O)» .

Специальные символы

Символы, имеющие специальное значение для оболочки должны быть *экранированы*. Символом экранирования в Bash является обратная косая черта, как и в большинстве оболочек; она придает специальное значение последующему символу. Оболочка знает о многих специальных символов, к числу наиболее распространенных относятся /,.,? и *. Полный список можно найти на info-страницах и документации к вашей оболочке.
Так, например, чтобы сказать, что вы хотите отобразить файл «*» вместо всех файлов в каталоге, вам придется использовать

less *

То же самое касается имен файлов, содержащих пробел:

cat This\ File

Другие способы просмотра содержимого файла

Общее

Кроме **cat**, который действительно не делать больше, чем передача файлов на стандартный вывод, есть другие инструменты для просмотра содержимого файла.

Самый простой способ, конечно, будет заключаться в использовании графических инструментов вместо инструментов командной строки. Во введении мы уже мельком видели офисное приложение OpenOffice.org. Другие примеры GIMP (запускается командой **gimp** из командной строки), GNU программа обработки изображений; **xpdf** для просмотра файлов формата Portable Document (PDF); GhostView (**gv**) для просмотра файлов PostScript; браузер Mozilla/FireFox, Konqueror, Opera и многое другое для веб-контента; XMMS, CDplay и других для мультимедийных файлов, AbiWord, Gnumeric, KOffice и т.д. для всех видов офисных приложениях и т.д. Есть тысячи приложений Linux; перечисление их всех заняло бы дни.

Вместо этого мы продолжим сосредотачиваться на приложениях текстового режима, которые создают основу для всех других приложений. Эти команды работают лучше всего в текстовом окружении для файлов, содержащих текст. В случае сомнений, сначала проверьте с помощью команды **file**.

Итак, давайте посмотрим, какие текстовые инструменты у нас есть, которые полезны, для просмотра содержимого файлов.

 **Проблемы шрифта.**
Инструменты для простого (плоского) текста, такие как те, что мы сейчас будем обсуждать, часто имеют проблемы с "плоскими" текстовыми файлами из-за кодировок шрифта, используемых в этих файлах. Специальные символы, такие как "акцентированные" буквы алфавита, китайские иероглифы и другие символы языков, использующих различные наборы символов, а не по умолчанию en_US кодировку и так далее, будут отображаться неправильно или будут заменены нечитаемым мусором. Эти проблемы обсуждаются в Разделе 7.4. "Региональные специфические настройки".

Меньше (less) значит больше

Несомненно рано или поздно вы услышите как кто-то скажет эту фразу при работе в среде UNIX. Немного истории UNIX объяснит это:

- Сначала была **cat**. Поток вывода был неконтролируемым.
- Потом была **pg**, которую все еще можно обнаружить на старых версиях Unix. Эта команда выдает текст на вывод по одной странице за раз.
- Программа **more** подправленный вариант **pg**. Эта команда по-прежнему есть в каждой системе Linux.
- **less** GNU версия **more** и включает дополнительные функции, позволяющие подсвечивать строки поиска, прокрутку назад и т.д. Синтаксис очень прост:

less name_of_file

Более подробная информация находится на info-страницах.

К настоящему времени вы уже знаете о пейджерах, поскольку они используются для просмотра man-страниц.

Команды head и tail

Эти две команды отображают соответственно n первых/последних строк файла. Чтобы посмотреть последние десять команд ввели следующее:

```
tony:~> tail -10 .bash_history
locate configure | grep bin
man bash
cd
xawtv &
grep usable /usr/share/dict/words
grep advisable /usr/share/dict/words
info quota
man quota
echo $PATH
frm
```

head работает аналогично. У команды **tail** есть удобная функция, постоянно показывающая последние n строк файла, которые меняются все время. Это опция -f, которая часто используется системными администраторами для проверки файлов журналов. Более подробная информация находится в файлах системной документации.

Связывание файлов

Типы ссылок

Поскольку мы уже знаем достаточно о файлах и их представлении в файловой системе, понимание ссылок (или ярлыков) добавит составляющую в общую картину. Ссылка — это не более чем способ сопоставления двух или более имен файлов с одним и тем же набором данных. Есть два пути достижения этой цели:

- Жесткая ссылка: Связывает два или более имени файла с одним и тем же индексным дескриптором. Жесткие ссылки определяют одни и те же блоки данных на жестком диске, в то время они продолжают вести себя как независимые файлы.

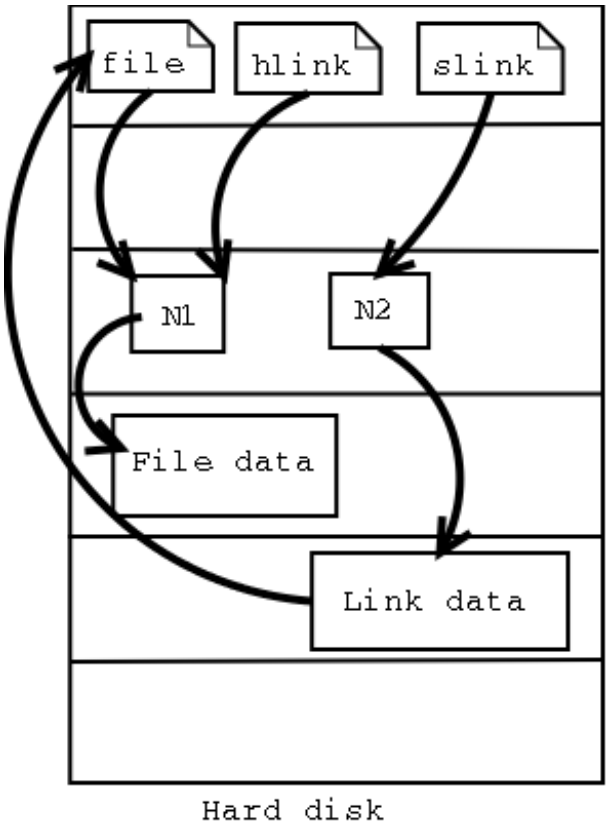
Существует очевидное неудобство: жесткие ссылки не могут распространяться на разделы, т.к. номера индексных дескрипторов уникальны только в пределах данного раздела.

- Мягкая или символическая (символьная) ссылка (или для краткости: symlink - симлинк): маленький файл, который является указателем на другой файл. Символическая ссылка содержит путь к целевому файлу, а не указание на физическое расположение данных на жестком диске. Поскольку в данном механизме не задействованы индексные дескрипторы, мягкие ссылки могут существовать между разными разделами.

Поведение этих двух типов ссылок похожее, но не одинаковое, что проиллюстрировано на схеме ниже:

Рисунок 3.2. Механизм жестких и мягких ссылок

Обратите внимание, что удаление целевого файла для символьной ссылки делает ссылку бесполезной.



Каждый обычный файл, в принципе, - жесткая ссылка. Жесткие ссылки файла не простираются на все разделы, так как они отсылают к индексным дескрипторам, а номера ID уникальны только в пределах данного раздела.

Можно утверждать, что существует третий вид ссылок, *user-space* ссылки, которые похожи на ярлык в MS Windows. Это файлы, содержащие метаданные, которые могут быть интерпретированы только графическим файловым менеджером. Для ядра и оболочки это просто обычные файлы. У них может быть в конце суффикс *.desktop* или *.lnk*; например, можно найти ~/.gnome-desktop:

```
[dupont@boulot .gnome-desktop]$ cat La\ Maison\ Dupont
[Desktop Entry]
Encoding=Legacy-Mixed
Name=La Maison Dupont
Type=X-nautilus-home
X-Nautilus-Icon=temp-home
URL=file:///home/dupont
```

Этот пример взят из графической среды KDE:

```
[lena@venus Desktop]$ cat camera
[Desktop Entry]
Dev=/dev/sda1
FSType=auto
Icon=memory
MountPoint=/mnt/camera
Type=FSDevice
X-KDE-Dynamic-Device=true
```

Создать такого рода ссылку достаточно легко путем использования особенностей вашей графической среды. Если вам нужна помощь, то системная документация должна быть вашим первым источником.

В следующем разделе мы рассмотрим создание символических ссылок в UNIX-стиле с помощью командной строки.

Создание символических ссылок

Символическая ссылка представляет особый интерес для начинающих пользователей: они достаточно очевидны для понимания и вам не нужно беспокоиться о разделах.

Команды для создания ссылок является **ln**. Для того чтобы создать символические ссылки, следует использовать опцию -s:

ln -s targetfile linkname

В приведенном ниже примере пользователь *freddy* создает ссылку в подкаталоге своей домашней папки на

каталог другой части системы:

```
freddy:~/music> ln -s /opt/mp3/Queen/ Queen

freddy:~/music> ls -l
lrwxrwxrwx  1 freddy  freddy  17 Jan 22 11:07 Queen -> /opt/mp3/Queen
```

Символические ссылки всегда очень маленькие файлы, в то время как жесткие ссылки имеют те же размеры, что и исходный файл.

Применение символических ссылок имеет широкое распространение. Они часто используются для экономии дискового пространства за счет создания «копии» файла для удовлетворения требований при установке новой программы, которая ожидает наличие файла в другом месте; они используются для исправления сценариев, которые внезапно запускаются в новых окружениях, в общем могут предотвратить большие затраты. Системный администратор может принять решение перенести домашние каталоги пользователей на новое место, disk2 например, но если он хочет, чтобы все работало, как раньше, в том числе файл /etc/passwd, с минимумом усилий он создаст символическую ссылку из /home на новое место /disk2/home.

Защита файлов

Права доступа: первая линия обороны Linux

Модель безопасности Linux основана на той, которая используется в системах UNIX, и является такой же строгой (и иногда даже более). В системе Linux каждый файл принадлежит пользователю и группе пользователей. Есть также третья категория пользователей, которые не являются пользователем-владелецем и не принадлежат группе, владеющей файлом. Для каждой категории пользователей, разрешение на чтение, запись и выполнение может быть предоставлено или отклонено.

Мы уже использовали "длинный" вариант просмотра файлов с помощью команды **ls** -l, хотя и по другим причинам. Эта команда также отображает разрешения файла (права доступа) для этих трех категорий пользователей; они указаны девятью символами, которые следуют за первым символом (индикатором типа файла). Как видно из приведенных ниже примеров, первые три символа из этой серии девяти отображают права доступа реального пользователя, который является владельцем файла. Следующие три — предназначены для группы-владельца файла, последние три — для других пользователей. Разрешения всегда описываются в одном и том же порядке: чтение, запись и исполнение (выполнение, запуск) для пользователя, группы и других. Вот некоторые примеры:

```
marise:~> ls -l To_Do
-rw-rw-r--    1 marise  users      5 Jan 15 12:39 To_Do
marise:~> ls -l /bin/ls
-rwxr-xr-x    1 root    root   45948 Aug  9 15:01 /bin/ls*
```

Первый файл является обычным (вначале прочерк). Пользователь с именем *marise* или пользователи, принадлежащие к группе *users*, могут читать и переписывать (изменять/перемещать/удалять) файл, но они не могут исполнять его (второе и третье тире). Другим пользователям разрешено только читать этот файл, но они не могут изменять или выполнять его (четвертое и пятое тире).

Второй пример — это исполняемый файл; отличие: все могут запустить данную программу, но вам нужно быть суперпользователем, чтобы изменить его.

Info-страницы подробно объясняют, каким образом команда **ls** управляет отображением прав доступа (см. в разделе *What information is listed*).

Для удобства работы с командами, как права доступа (или режимы), так и категории пользователей имеют коды. См. таблицы ниже.

Таблица 3.7. Коды режимов доступа

Код	Значение
0 или -	Права доступа, которые должны были прописаны в данном месте, не предоставляются.
4 или r	Предоставляется право на чтение для определенной категории пользователей.
2 или w	Право на запись для категории пользователей, определенных в данном месте.
1 или x	Право на выполнение для определенной категории пользователей.

Таблица 3.8. Коды группы пользователей

Код	Значение
u	Права пользователя
g	Права группы
o	Права для остальных

Использование данного механизма разделения прав обязательно; это позволяет обеспечить высокий уровень безопасности даже без сетевой защиты. Среди прочих функций такая схема безопасности заботится о доступе пользователей к программам, it can serve files on a need-to-know basis и защищает конфиденциальные данные, такие как домашние каталоги и системные конфигурационные файлы.

Вы должны знать ваше имя пользователя. Если вы не знаете, то его можно отобразить, используя команду **id**, которая также отображает группу по умолчанию, к которой вы принадлежите и, в конечном итоге, другие группы, членом которых вы являетесь:

```
tilly:~> id
uid=504(tilly) gid=504(tilly) groups=504(tilly),100(users),2051(org)
Ваше имя пользователя, также хранится в переменной окружения USER:
tilly:~> echo $USER
tilly
```

Инструменты

Команда chmod

Обычным следствием применения строгих правил доступа к файлам, а иногда и неудобством, является то, что права доступа должны быть изменены по тем или иным причинам. Чтобы это сделать, мы используем команду **chmod**, что привело к тому, что *chmod* стал почти приемлемым английским глаголом, обозначающим изменение режима доступа к файлу. Команда **chmod** может быть использована с буквенными и числовыми опциями, что вам больше всего нравится.

В приведенном ниже примере используются буквенные параметры для решения проблемы, с которой

обычно сталкиваются начинающие пользователи:

```
asim:~> ./hello
bash: ./hello: bad interpreter: Permission denied

asim:~> cat hello
#!/bin/bash
echo "Hello, World"

asim:~> ls -l hello
-rw-rw-r--    1 asim    asim    32 Jan 15 16:29 hello

asim:~> chmod u+x hello

asim:~> ./hello
Hello, World

asim:~> ls -l hello
-rwxrw-r--   1 asim    asim    32 Jan 15 16:29 hello*
```

+ и - используются для разрешения или запрещения конкретного права для определенной категории. Комбинации через запятую не допускаются. Info и man-страницы содержат полезные примеры. Здесь еще один, в котором файл из предыдущего примера становится личным файлом пользователя *asim*:

```
asim:~> chmod u+rwx,go-rwx hello

asim:~> ls -l hello
-rwx------    1 asim    asim    32 Jan 15 16:29 hello*
```

Ситуации, в которых появляются сообщения об ошибках, говорящих, что допуск запрещен, как правило, связаны с проблемой прав доступа. Также, реплики вроде: "Это работало вчера" и "Когда я запускаю это как *root*, то оно работает", скорее всего, вызваны тем, что не были учтены права доступа к файлам.

При использовании **chmod** с цифровыми аргументами, значения для каждого предоставляемого права доступа указываются в "месторасположение" категории. Таким образом, получается трехзначное число, которое представляет собой символическое обозначение того, что должна сделать команда **chmod**. В следующей таблице перечислены наиболее распространенные комбинации.

Таблица 3.9. Защита файлов с помощью chmod

Код	Значение
chmod 400 file	Защита файла от случайной перезаписи
chmod 500 directory	Чтобы защитить себя от случайного удаления, переименования или перемещения файлов из этой директории.
chmod 600 file	Личный файл, изменяемый только пользователем, который ввел эту команду.

chmod 644 file	Всеми читаемый файл, который может быть изменен только пользователем-владельцем.
chmod 660 file	Пользователи, принадлежащие вашей группе, могут изменить этот файл, другие не имеют никакого отношения к нему вообще.
chmod 700 file	Защищает файл от любого доступа посторонних, в то же время пользователь-владелец имеет полный контроль.
chmod 755 directory	Для файлов, которые должны быть читаемыми и исполняемыми всеми, но изменяемые только пользователем-владельцем.
chmod 775 file	Стандартный режим совместного использования файла группой.
chmod 777 file	Каждый может делать с этим файлом все, что хочет.

Если в качестве аргумента **chmod** вы вводите число, состоящее менее чем из трех цифр, опущенные символы заменяются на нули, начиная слева. В системах Linux на самом деле существует четвертая цифра, которая предшествует первому из трех и устанавливает специальные режимы доступа. Все об этом и многом другом находится в info-страницах.

Вступление в другую группу

Когда вы введете **id** в командной строке, то получите список всех групп, к которым вы можете принадлежать; перед этим будет ваше имя пользователя и ID, а также название группы и ее ID, с которой вы в настоящее время связаны. Однако, во многих системах Linux можно активно входить только в одну группу в одно и то же время. По умолчанию, это активная или первичная группа является первой, которая вам устанавливается из файла /etc/passwd. Четвертое поле в этом файле содержит ID основной группы пользователей, что можно также увидеть в файле /etc/group. Например:

```
asim:~> id
uid=501(asim) gid=501(asim) groups=100(users),501(asim),3400(web)

asim:~> grep asim /etc/passwd
asim:x:501:501:Asim El Baraka:/home/asim:/bin/bash

asim:~> grep 501 /etc/group
asim:x:501:
```

В приведенном выше примере четвертое поле в строке из /etc/passwd содержит значение "501", которое представляет собой группу asim. Из /etc/group мы можем получить имя, соответствующие этому идентификатору группы. При первом подключении к системе, это та группа, к которой будет принадлежать *asim*.

"Механизм" индивидуальной группы пользователя.

В целях обеспечения большей гибкости, большинство систем Linux преследуют так называемую индивидуальную групповую схему пользователей, которая, в первую очередь, присваивает каждому пользователю его собственную группу. Этой группе принадлежит только данный пользователь, отсюда и название "личная группа". Обычно эта группа имеет такое же имя как название логина пользователя, что

может приводить к небольшой путанице.

Помимо своей собственной группы, пользователь *asim* также может быть в группах *users* и *web*. Т.к. это вторые группы для данного пользователя, ему придется использовать команду **newgrp**, чтобы войти в любую из этих групп (сначала используйте **gpasswd** для установки пароля для группы). В этом примере asim необходимо создать файлы, которые принадлежат группе *web*.

```
asim:/var/www/html> newgrp web

asim:/var/www/html> id
uid=501(asim) gid=3400(web) groups=100(users),501(asim),3400(web)
```

Теперь, когда *asim* создает новые файлы, они будут находиться в собственности группы *web*, вместо того, чтобы принадлежать группе *asim*:

```
asim:/var/www/html> touch test

asim:/var/www/html> ls -l test
-rw-rw-r--  1 asim web   0 Jun 10 15:38 test
```

Вход в новую группу освобождает вас от необходимости использовать chown (см. Раздел "Смена владельцев и групп") или вызывать системного администратора для смены владельцев для вас.

См. man-страницу для **newgrp** для получения дополнительной информации.

Маска файла

Прежде чем новый файл куда-то сохраняется, он подвергаться стандартной процедуре защиты. В Linux не существуют файлы без установленных прав доступа. Стандартное разрешение на файл определяется маской при его создании. Значение этой маски может быть получено с помощью команды **umask**:

```
bert:~> umask
0002
```

Вместо добавления символических значений друг к другу, как с **chmod**, для выяснения разрешения на новый файл, необходимо вычесть из суммарной возможности прав доступа. Однако в примере выше мы видим четыре значения, но есть только три категории разрешений: для пользователя, группы и других. Первый ноль — это установленные специальные файловые атрибуты, которые мы будем обсуждать в Разделе "Смена владельцев и групп" и в Разделе "SUID и SGID". Возможно, с тем же успехом, в вашей системе этот первый ноль не отображается при вводе команды **umask**, и тогда вы видите только три числа, представляющих маску по умолчанию, «накладываемую» на файл.

Каждая UNIX-подобная система имеет системную функцию для создания новых файлов, которая вызывается каждый раз, когда пользователь использует программу, которая создает новые файлы, например, при загрузке файлов из Интернета, при сохранении нового текстового документа и т.д. Эта функция создает как новые файлы, так и директории. Полные права на чтение, запись и выполнение предоставляются всем при создании нового каталога. При создании нового файла, эта функция создаст право на чтение и запись для всех, но прав на выполнение отсутствует для всех категорий пользователей. Таким образом, первоначально применяется маска для каталога с разрешением *777* или *rwxrwxrwx*, а для обычного файла *666* или *rw-rw-rw-*.

Значение **umask** вычитается из этих разрешений по умолчанию, определенных инструментом, создающим новый файл или каталог. Таким образом, каталог будет иметь права доступа *775* по умолчанию, а файл *664*,

если значение маски *(0)002*. Это демонстрируется на примере ниже:

```
bert:~> mkdir newdir

bert:~> ls -ld newdir
drwxrwxr-x    2 bert    bert  4096 Feb 28 13:45 newdir/

bert:~> touch newfile

bert:~> ls -l newfile
-rw-rw-r--    1 bert    bert     0 Feb 28 13:52 newfile
```

 **Файлы по сравнению с каталогами.**
Каталог получает больше разрешений по умолчанию: он всегда имеет разрешение на выполнение. Если бы не это, они не были бы доступны. Исследуйте это, изменив режим доступа к каталогу на 644!

Если вы войдете в другую группу, используя команду **newgrp**, маска остается неизменной. Таким образом, если она установлена на *002*, файлы и каталоги, которые вы создаете, находясь в новой группе, также будут доступны для других членов этой группы; вам не придется использовать команду **chmod**.

Пользователь *root* обычно имеет более строгую маску по умолчанию:

```
[root@estoban root]# umask
022
```

Эти значения по умолчанию — общесистемная установка в конфигурационном файле shell, например /etc/bashrc или /etc/profile. Вы можете изменить их на ваш собственный файл конфигурации shell, см. в Главе 7, “Дом сладкий /home” о настройках окружения оболочки.

Смена владельцев и групп

Если файл находится в собственности не того пользователя или группы, то недочет может быть устранен с помощью команд **chown** (смена владельца) и **chgrp** (смена группы). В среде, где файлы должны находиться в коллективном доступе группы, изменение владельца является частой задачей системного администрирования. Обе команды достаточно гибкие, о чем можно узнать с помощью опции --help.

Команда **chown** может использоваться как для изменения владельца-пользователя файла, так и группы, а **chgrp** изменяет только группу. Конечно, система проверит, имеет ли пользователь, использующий эти команды, достаточно прав на файл(ы), чтобы изменять владельцев.

Чтобы изменить лишь владельца-пользователя файла, используйте следующий синтаксис:

chown *newuser* file

Если вы используете двоеточие после имени пользователя (см. info-страницы), группа-владелец будет изменена также на основную группу пользователя, запускающего команду. В системе Linux каждый пользователь имеет свою собственную группу, так что эта особенность может использоваться, чтобы делать файлы личными:

```
jacky:~> id
uid=1304(jacky) gid=(1304) groups=1304(jacky),2034(pproject)
```

```
jacky:~> ls -l my_report
-rw-rw-r--  1 jacky   project       29387 Jan 15 09:34 my_report

jacky:~> chown jacky: my_report

jacky:~> chmod o-r my_report

jacky:~> ls -l my_report
-rw-rw----  1 jacky   jacky         29387 Jan 15 09:34 my_report
```

Если *jacky* захотел бы поделиться этим файлом, без предоставления всем разрешения на запись, то он может использовать команду **chgrp**:

```
jacky:~> ls -l report-20020115.xls
-rw-rw---- 1 jacky   jacky   45635 Jan 15 09:35 report-20020115.xls

jacky:~> chgrp project report-20020115.xls

jacky:~> chmod o= report-20020115.xls

jacky:~> ls -l report-20020115.xls
-rw-rw---- 1 jacky   project 45635 Jan 15 09:35 report-20020115.xls
```

Таким образом, пользователи из группы *project* смогут работать с этим файлом. У пользователей не из этой группы нет никакого доступа к нему вообще.

Как **chown**, так и **chgrp** могут быть использованы для рекурсивной смены владельца с помощью опции -r. В этом случае, все вложенные файлы и подкаталоги данного каталога будет принадлежать указанному пользователю и/или группе.

 **Ограничения.**
На большинстве систем, использование команд **chown** и **chgrp** ограничено для непривилегированных пользователей. Если вы не администратор системы, то не можете изменить пользователя или группу по соображениям безопасности. Если использование этих команд не будет ограничено, то злостные пользователи могут назначать владельцами файлов других пользователей и/или другие группы, и изменить поведение окружения пользователей, и даже повредить чужие для них файлы.

Специальные режимы

Чтобы все время не беспокоить системного администратора решением проблем, связанных с правами, специальные права доступа могут быть предоставлены для целых каталогов или отдельных программ. Есть три специальных режима:

- Режим "липкого бита": После выполнения задания, команда находится в оперативной памяти. Первоначально это была функция, которая использовалась в основном для сохранения памяти: большие рабочие части загружались в память только один раз. Но в наши дни память не такая дорогая и есть технологии, которые управляют ей лучше, поэтому это больше не используются для оптимизации возможностей на отдельных файлах. По отношению к целому каталогу, однако, у липкого бита есть другое значение. В этом случае, пользователь может изменять файлы в данной директории, если является владелец файла или файл имеет соответствующие разрешения. Эта возможность используется для таких каталогов как /var/tmp, которые должны быть доступны для всех, но где нельзя пользователям изменять или удалять данные друг друга. Липкий бит указывает в конце поля прав доступа к файлу:

```
mark:~> ls -ld /var/tmp
drwxrwxrwt   19 root     root         8192 Jan 16 10:37 /var/tmp/
```

Липкий бит установливается с помощью команды **chmod** o+t *directory*. Историческое происхождение “t” - сохранение возможности *Text access* (текстовый доступ) в UNIX.

- SUID (установка ID пользователя) и SGID (установка ID группы): представлены символом s в поле прав доступа пользователя или группы. Когда этот режим установлен на исполняемый файл, то он будет запускаться с правами пользователя и группы, имеющих разрешения на файл, а не под тем пользователем, который ввел команду, таким образом, предоставляется доступ к системным ресурсам. Мы обсудим это далее в Главе 4, Процессы.
- SGID (установка ID группы) на каталог: в этом конкретном случае все файлы, созданные в данном каталоге, будут иметь туже группу-владельца, что и сам каталог (в то время как нормальным поведением является то, что новые файлы принадлежат пользователям, которые их создают). Таким образом, пользователям не нужно беспокоиться о собственности файла при совместном использовании папок:

```
mimi:~> ls -ld /opt/docs
drwxrws---  4 root    users          4096 Jul 25 2001 docs/

mimi:~> ls -l /opt/docs
-rw-rw----  1 mimi    users        345672 Aug 30 2001-Council.doc
```

Это стандартный способ совместного доступа к файлам в UNIX.

 **Имеющиеся файлы остаются без изменений!**
Файлы, которые перемещаются в SGID-каталог, но были созданы в других местах, сохраняют их первоначальные пользователя-владельца и группу. Это может привести к путанице.

Резюме

В UNIX, как и в Linux, все объекты, так или иначе, представлены в системе в виде файлов с соответствующими свойствами. Использование (предопределенных) адресных путей позволяет пользователям и системному администратору находить, читать файлы и управлять ими.

Мы сделали первые шаги, чтобы стать экспертом: мы обсудили реальную и "поддельную" структуру файловой системы, и мы узнали о модели файловой безопасности Linux, а также ряде других мер предосторожности, существующих на каждой системе по умолчанию.

Оболочка (shell) – самый важный инструмент для взаимодействия с системой. В этой главе мы

познакомились с некоторыми командами, которые перечислены в таблице ниже.

Таблица 3.10. Новые команды из Главы 3: Файлы и файловая система

Команда	Значение
bash	Программная оболочка проекта GNU
cat file(s)	Отправляет содержимое файла(ов) на стандартный вывод
cd directory	Переход в directory. **cd** является встроенной командой **bash**
chgrp *newgroup* file(s)	Изменение группы владельцев file(s) на *newgroup*
chmod *mode* file(s)	Изменение прав доступа к file(s)
chown *newowner***[:[newgroup]]** file(s)	Изменение собственника файла и группы владельцев
cp sourcefile targetfile	Копирование sourcefile в targetfile
df file	Отчеты о использовании дискового пространства раздела, на котором находится файл
echo string	Отображение строки текста
export	Часть **bash**, которая объявляет переменные и их значения в системе
file filename	Определить тип файла filename
find path expression	Поиск файлов в иерархии файловой системы
grep PATTERN file	Вывод строк файла, содержащих шаблон поиска
head file	Отправить первую часть файла на стандартный вывод
id	Вывод имени пользователя и названий групп
info command	Чтение документации о **command**
less file	Просмотр файла с помощью специальной программы
ln targetfile linkname	Создание ссылки с именем linkname на targetfile
locate searchstring	Вывод всех доступных файлов, соответствующих шаблону поиска
ls file(s)	Вывод содержимого директории
man command	Форматирует и отображает страницы системного руководства для **command**

mkdir newdir	Создает новый пустой каталог
mv oldfile newfile	Переименовывает или перемещает oldfile
newgrp groupname	Вход в новую группу
pwd	Печать текущего рабочего каталога
quota	Показывает используемое дисковое пространство и ограничения
rm file	Удаление файлов и директорий
rmdir file	Удаление директорий
tail file	Вывод последней части file
umask [value]	Отображает или изменяет режим создания новых файлов
wc file	Считает строки, слова и символы в file
which command	Показывает полный путь к команде

Мы также подчеркнули тот факт, что вы должны ЧИТАТЬ СТРАНИЦЫ MAN. Эта документация является вашей первой помощью и содержит ответы на многие вопросы. Упомянутый выше список содержит основные команды, которые вы будете использовать ежедневно, но они могут делать гораздо больше, чем выполнение тех задач, о которых мы говорили здесь. Чтение документации предоставит вам необходимый контроль.

Последнее, но не менее значимое. Удобный обзор файловых разрешений:

Таблица 3.11. Файловые разрешения

Кто\Что	r(ead)	w(rite)	(e)x(ecute)
u(ser)	4	2	1
g(roup)	4	2	1
o(ther)	4	2	1

Упражнения

Просто войдите под вашим обычным пользовательским ID.

Разделы

- На каком разделе находится ваша домашняя директория?
- Сколько разделов в вашей системе?
- Каков общий размер вашего установленного Linux?

Пути

- Отобразите ваши пути поиска программ.
- Экспортируйте бессмысленный адрес, введя, например, **export** PATH=*blah* и попытайтесь получить список содержимого каталога.
- Каков путь к домашней директории? Как бы другой пользователь достиг вашего домашнего каталога, начиная от своего домашнего каталога и используя относительные пути?
- Перейти в каталог tmp, находящийся в /var.
- Теперь перейдите в каталог share, находящийся в /usr, используя только одну команду. Измените на doc. Каков ваш текущий рабочий каталог?

Экскурсияпо системе

- Перейдите в каталог /proc.
- На каком процессоре(ax) работает система?
- Сколько на текущий момент используется оперативной памяти?
- Каким объемом пространства подкачки вы располагаете?
- Какие драйверы загружены?
- Сколько часов уже работает система?
- С какими файловыми системами «знакома» ваша операционная система?
- Перейдите в /etc/rc.d | /etc/init.d | /etc/runlevels и выберите каталог соответствующий вашему запущенному режиму.
- Какие сервисы должны быть запущены на этом уровне?
- Какие сервисы запускаются в графическом режиме из тех, которые не запускаются в текстовом?
- Перейдите в /etc.
- Как долго в системе хранятся лог-файл с мониторингом входов пользователя.
- Какой релиз вы используете?
- Есть какие-либо вопросы или сообщения дня?
- Сколько пользователей определены в вашей системе? Не считайте их, пусть компьютер сделает это за вас!
- Сколько групп?
- Где хранится информация о часовом поясе?
- В вашей системе установлены какие-нибудь HOWTO?
- Перейдите в /usr/share/doc.
- Назовите три программы, которые поставляются с пакетом GNU *coreutils*.
- Какая версия bash установлена в данной системе?

Манипуляции над файлами

- Создайте новый каталог в домашней директории.
- Можно ли переместить эту папку на тот же уровень, что и ваш домашний каталог?

- Скопируйте все XPM файлы из /usr/share/pixmaps в новый каталог. Что значит XPM?
- Отобразите список файлов в обратном алфавитном порядке.
- Перейдите в ваш домашний каталог. Создайте новый каталог и скопируйте в него все файлы из каталога /etc. Убедитесь, что вы также скопировали файлы и каталоги, которые находятся в подкаталогах /etc! (рекурсивное копирование)
- Перейдите в новый каталог и создайте папку для файлов, начинающихся с большой буквы, и одну — для файлов, начинающихся с символов нижнего регистра. Переместите все файлы в предназначенные для них каталоги. Используйте как можно меньше команд, насколько это возможно.
- Удалите остальные файлы.
- Удалить каталог и все его содержимое с помощью одной команды.
- Используйте **grep**, чтобы выяснить, какой сценарий запускает сервер шрифтов в графическом режиме.
- Где находится программа *sendmail*?
- Создайте символическую ссылку в вашем домашнем каталоге на /var/tmp. Убедитесь, что она действительно работает.
- Сделайте еще символическую ссылку в вашем домашнем каталоге на эту ссылку. Также проверьте ее работоспособность. Удалите первую ссылку и отобразите содержимое каталога. Что случилось со второй ссылкой?

Разрешения файлов

- Вы можете поменять права доступа к файлам в /home?
- Какой у вас стандартный режим создания файлов?
- Измените собственника /etc на вашего личные пользователя и группу.
- Измените права доступа к файлу ~/.bashrc так, чтобы только вы и ваша основная группа могли читать его.
- Введите команду **locate** root. Вы заметили что-нибудь особенное?
- Создайте символическую ссылку на /root. Можно ли ее использовать?

Глава 4. Процессы

Аннотация

Наряду с файлами, процессы представляют собой наиболее важных объекты в системах UNIX/Linux. В этой главе мы присмотримся к этим процессам. Мы узнаем больше о том, что такое:

- Многопользовательская работа и многозадачность
- Типы процессов
- Контроль процессов различными сигналами
- Свойства процессов
- Жизненный цикл процесса
- Системный запуск и остановка

- SUID и SGID
- Системная скорость и ответная реакция
- Планирование процессов
- Система Vixie cron
- Как получить максимум от вашей системы

Процессы изнутри

Многопользовательская работа и многозадачность

Теперь, когда мы уже привыкли к нашему окружению и способны немного взаимодействовать с нашей системой, самое время для более детального изучения процессов. Не все команды запускают единственный процесс. Некоторые из них начинают целый ряд процессов, например, **mozilla**; другие, как **ls**, выполняются как единственная команда.

Кроме того, Linux основан на UNIX, где это является обычной практикой при многочисленных пользователях, запускающих множество команд в одно и то же время в одной и той же системе. Очевидно, что были приняты меры, чтобы процессор управлял всеми этими процессами, и обеспечивалась функциональность, когда пользователи могут переключаться между процессами. В некоторых случаях процессы должны продолжать работать, даже если пользователь, который их запустил, вышел из системы. С другой стороны, пользователи нуждаются в средствах для прерывания процессов.

В следующих разделах мы расскажем об устройстве процессов в Linux.

Типы процессов

Интерактивные процессы

Интерактивные процессы инициализируется и контролируется через терминальную сессию. Иными словами, кто-то должен быть подключен к системе, чтобы эти процессы запустились; они не запускаются автоматически, как часть функциональности системы. Эти процессы могут работать на переднем плане, занимая терминал, который запустил программу, и вы не сможете запускать другие приложения до тех пор, как этот процесс работает таким образом. Иначе они могут работать в фоновом режиме, таким образом, терминал, в котором вы запустили программу, может принимать новые команды, пока та программа тоже работает. До сих пор мы в основном уделяли внимание программам, работающим на переднем плане - продолжительность времени, необходимого для их работы, была слишком короткой, чтобы заметить - но просмотр файлов командой less - хороший пример, когда команда занимает терминал сессии. В этом случае, активная программа ожидает от вас какого-нибудь действия. Программа по-прежнему связана с терминалом, откуда она была запущена, а терминал пригоден только для ввода команд для этой программы, только их он может понять. Другие команды просто приведут к ошибкам или зависанию системы.

Несмотря на то, что процесс выполняется в фоновом режиме, пользователь все равно не имеет возможности делать другие вещи в терминале, в котором он запустил программу, пока программа работает.

Оболочка предлагает функцию управления заданиями, которая позволяет легко обрабатывать множество процессов. Этот механизм переключает процессы между передним и задним фоном. Используя эту систему, программы также могут быть запущены в фоновом режиме сразу же.

Запуск процесса в фоновом режиме, имеет смысл только для программ, которые не нуждаются в

пользовательском контроле (через shell). Перевод задания в фоновый режим делается, как правило, когда для его выполнения, как ожидается, потребует длительного времени. Для того, чтобы освободить выдачу терминала после ввода команды, добавляется завершающий амперсанд. В примере, используя графический режим, мы открываем дополнительное окно терминала из имеющегося:

```
billy:~> xterm &
[1] 26558

billy:~> jobs
[1]+  Running                 xterm &
```

Подробное описание функции контроля заданий есть в info-страницах **bash**, так что здесь перечислены только наиболее часто используемые способы управления заданиями:

Таблица 4.1. Управление процессами

Команды (часть из существующих)	Значение
regular_command	Запуск данной команды на переднем плане.
command &	Запуск команды в фоновом режиме (в версии терминала).
jobs	Показывает команды запущенные в фоновом режиме.
Ctrl+Z	Приостановить (остановить, но не выйти) процесс, выполняемый на переднем плане (засыпание).
Ctrl+C	Прервать (прекратить и выйти) процесс, выполняемый на переднем плане.
%n	Каждый процесс, выполняемый в фоновом режиме, получает присвоенный ему номер. Используя выражение %, указание может быть отнесено к процессу путем использования его номера, например, **fg** %2.
bg	Возобновить приостановленную программу в фоновом режиме.
fg	Поместить фоновое задание на передний план.
kill	Завершение процесса (см. также Shell Builtin Commands на info-страницах **bash**).

Подробные практические примеры можно найти в упражнениях.

Большинство UNIX систем, вероятно, могут быть способны запустить экран, который полезен, когда вам действительно нужна еще одна оболочка для выполнения команд. После вызова экрана, новая сессия создается с сопутствующей оболочкой и/или командами, которые указаны и которые вы можете отложить в сторону. В этой новой сессии вы можете делать все, что вы хотите. Все программы и операции будут работать независимо от выдачи оболочки. Затем вы можете отделить эту сессию, пока программы, которые вы запустили в ней, продолжат работать, даже когда вы выйдите из оболочки давшей начало, и выберите экран снова в любое удобное для вас время.

Эта программа берет свое начало со времени, когда виртуальные консоли еще не придумали, и все должно

было быть сделано с помощью одного текстового терминала. Для увлеченных это все еще имеет значение в Linux, хотя уже почти 10 лет у нас есть виртуальные консоли.

Автоматические процессы

Автоматические или пакетные процессы не связаны с терминалом. Скорее, это такие задачи, которые могут быть поставлены в очередь области диспетчера печати, где они ожидают выполнения по принципу FIFO (первый вошел, первый вышел). Такие задачи могут быть выполнены с помощью одного из двух критериев:

- В определенную дату и время: выполнение, используя команду **at**, которую мы обсудим во второй части этой главы.
- В периоды, когда общая загруженность системы достаточно низка, чтобы принять дополнительные работы: делается с помощью команды **batch**. По умолчанию задачи ставятся в очередь, где они ожидают выполнения в период, когда загрузка системы будет меньше, чем 0,8. В больших средах системный администратор может отдать предпочтение пакетной обработке, когда должны быть обработаны большие объемы данных или когда выполнение задач требуют много системных ресурсов в уже загруженной системе. Пакетная обработка используется также для оптимизации производительности системы.

Демоны

Демоны — это серверные процессы, которые работают постоянно. В большинстве случаев они инициализируются при запуске системы, и затем ожидают в фоновом режиме до того, как их услуги потребуются. Типичным примером является сетевой демон *xinetd*, которая запускается почти при каждой процедуре загрузки. После того как система загрузится, сетевой демон просто сидит и ждет клиентскую программу, которую необходимо подключить, такую как, например, FTP-клиент.

Свойства процессов

Процесс имеет ряд особенностей, которые могут быть просмотрены с помощью команды **ps**:

- Идентификатор процесса или PID: уникальный идентификационный номер, используемый для обращения к процессу.
- ID родительского процесса или PPID: номер процесса (PID), который запустил данный процесс.
- Nice-число: степень дружелюбия этого процесса по отношению к другим процессам (не путать с приоритетом процесса, который рассчитывается на основе этого милого числа и последнего использования процессора этим процессом).
- Терминал или TTY: терминал, к которому привязан данный процесс.
- Имя реального и эффективного пользователя (RUID и EUID): владелец процесса. Реальным владельцем является пользователь, который ввел команду, эффективный пользователь — тот, кто определяет порядок доступа к системным ресурсам. RUID и EUID обычно это одно и то же, и процесс имеет те же права доступа, которые есть у пользователя его запустившего. Поясним это на примере: браузер **mozilla**, находящийся в /usr/bin, принадлежит пользователю *root*:

```
theo:~> ls -l /usr/bin/mozilla
-rwxr-xr-x  1 root   root      4996 Nov 20 18:28 /usr/bin/mozilla*

theo:~> mozilla &
[1] 26595
```

```
theo:~> ps -af
UID      PID  PPID C STIME TTY       TIME CMD
theo   26601 26599 0 15:04 pts/5 00:00:00 /usr/lib/mozilla/mozilla-bin
theo   26613 26569 0 15:04 pts/5 00:00:00 ps -af
```

Когда пользователь *theo* запускает эту программу, сам процесс и все процессы, запущенные им, будут принадлежать пользователю *theo*, а не системному администратору. Когда **mozilla** будет требовать доступ к определенным файлам, то доступ будет зависеть от разрешений *theo*, а не администратора.

- Реальная и эффективная группы владельца (RGID и EGID): реальной группой- владельцем процесса является основная группа пользователя, запустившего процесс. Эффективная группа-владелец - обычно то же самое, за исключением случаев, когда режим доступа SGID применяется к файлу.

Просмотр информации о процессе

Команда **ps** является одним из инструментов для визуализации процессов. Эта команда имеет несколько опций, которые могут быть объединены для отображения различных свойств процесса.

При отсутствии конкретных опций **ps** выдает только информацию о текущей оболочке и возможных процессах:

```
theo:~> ps
  PID TTY          TIME CMD
 4245 pts/7    00:00:00 bash
 5314 pts/7    00:00:00 ps
```

Поскольку это не дает достаточно информации - как правило, по меньшей мере сотня процессов запущены в вашей системе – мы обычно будем выбирать отдельные процессы из списка всех процессов, используя команду **grep** в конвейере , см. Раздел "Выход перенаправления с > и | ", как в этой строке, в которой будут выбраны и отображены все процессы, принадлежащие определенному пользователю:

ps -ef | grep username

Этот пример показывает все процессы с процессорным именем **bash**, наиболее распространенной оболочкой в системах Linux:

```
theo:> ps auxw | grep bash
brenda   31970  0.0  0.3  6080 1556 tty2   S  Feb23   0:00 -bash
root     32043  0.0  0.3  6112 1600 tty4   S  Feb23   0:00 -bash
theo     32581  0.0  0.3  6384 1864 pts/1  S  Feb23   0:00 bash
theo     32616  0.0  0.3  6396 1896 pts/2  S  Feb23   0:00 bash
theo     32629  0.0  0.3  6380 1856 pts/3  S  Feb23   0:00 bash
theo      2214  0.0  0.3  6412 1944 pts/5  S  16:18   0:02 bash
theo      4245  0.0  0.3  6392 1888 pts/7  S  17:26   0:00 bash
theo      5427  0.0  0.1  3720  548 pts/7  S  19:22   0:00 grep bash
```

В этих случаях команда **grep** находит строки, содержащие строку *bash*, которые множество раз отображаются также в системах, которые имеют много простоев. Если вы не хотите, чтобы это произошло, используйте команду **pgrep**.

Bash-оболочки представляют собой особый случай: этот процессорный список также показывает, какие из них являются оболочками входа (где вы должны ввести ваше имя пользователя и пароль, например, когда

вы входите в текстовом режиме или же совершаете удаленный вход, в отличие от оболочек не требующих логина, например, нажав значок окна терминала). Такие оболочки, требующие логин, начинаются с дефиса (-).


|?

Мы расскажем об операторе | в следующей главе, см. Главу 5, "Перенаправление ввода/вывода".

Дополнительная информация может быть найдена как обычно: **ps** --help или **man** ps. GNU **ps** поддерживает различные форматы опций; приведенные выше примеры не содержат ошибок.

Заметьте, что **ps** выдает только сиюминутное состояние активных процессов, это одномоментный срез. Программа **top** дает более точный обзор путем обновления результатов, определенных **ps** (с множеством опций), один раз в 5 секунд, создавая новый список процессов, что периодически вызывает большие нагрузки, поэтому используется подключение дополнительной информации о файле подкачки и состоянии процессора; из файла proc системы:

```
 12:40pm up 9 days, 6:00, 4 users, load average: 0.21, 0.11, 0.03
89 processes: 86 sleeping, 3 running, 0 zombie, 0 stopped
CPU states:  2.5% user,  1.7% system,  0.0% nice, 95.6% idle
Mem:   255120K av, 239412K used, 15708K free, 756K shrd, 22620K buff
Swap: 1050176K av, 76428K used, 973748K free, 82756K cached

  PID USER  PRI NI SIZE  RSS SHARE STAT %CPU %MEM TIME COMMAND
 5005 root  14  0 91572  15M 11580 R    1.9  6.0  7:53 X
19599 jeff  14  0  1024 1024   796 R    1.1  0.4  0:01 top
19100 jeff   9  0  5288 4948  3888 R    0.5  1.9  0:24 gnome-terminal
19328 jeff   9  0 37884  36M 14724 S    0.5 14.8  1:30 mozilla-bin
    1 root   8  0   516  472   464 S    0.0  0.1  0:06 init
    2 root   9  0     0    0     0 SW   0.0  0.0  0:02 keventd
    3 root   9  0     0    0     0 SW   0.0  0.0  0:00 kapm-idled
    4 root  19 19     0    0     0 SWN  0.0  0.0  0:00 ksoftirqd_CPU0
    5 root   9  0     0    0     0 SW   0.0  0.0  0:33 kswapd
    6 root   9  0     0    0     0 SW   0.0  0.0  0:00 kreclaimd
    7 root   9  0     0    0     0 SW   0.0  0.0  0:00 bdflush
    8 root   9  0     0    0     0 SW   0.0  0.0  0:05 kupdated
    9 root  -1-20     0    0     0 SW<  0.0  0.0  0:00 mdrecoveryd
   13 root   9  0     0    0     0 SW   0.0  0.0  0:01 kjournald
   89 root   9  0     0    0     0 SW   0.0  0.0  0:00 khubd
  219 root   9  0     0    0     0 SW   0.0  0.0  0:00 kjournald
  220 root   9  0     0    0     0 SW   0.0  0.0  0:00 kjournald
```

Первая строка **top** содержит такую же информацию, которая отображается командой **uptime**:

```
jeff:~> uptime
  3:30pm, up 12 days, 23:29, 6 users, load average: 0.01, 0.02, 0.00
```

Данных для этих программ хранятся среди других в /var/run/utmp (информация о текущих подключенных пользователях) и в виртуальной файловой системе /proc, например, /proc/loadavg (обычная загрузочная информация). Есть разные виды графических приложений для просмотра этой информации, такие как Gnome System Monitor и *lavaps*. На FreshMeat и SourceForge и вы найдете десятки приложений, которые централизируют эту информацию наряду с другими серверными данными и журналами со множества

серверов на один (web) сервер, что позволяет осуществлять мониторинг всей ИТ-инфраструктуры с одной рабочей станции.

Отношения между процессами могут быть отображены с использованием команды **pstree**:

```
sophie:~> pstree
init-+-amd
     |-apmd
     |-2*[artsd]
     |-atd
     |-crond
     |-deskguide_apple
     |-eth0
     |-gdm---gdm-+-X
     |           `-gnome-session-+-Gnome
     |                           |-ssh-agent
     |                           `-true
     |-geyes_applet
     |-gkb_applet
     |-gnome-name-serv
     |-gnome-smproxy
     |-gnome-terminal-+-bash---vim
     |                |-bash
     |                |-bash---pstree
     |                |-bash---ssh
     |                |-bash---mozilla-bin---mozilla-bin---3*[mozilla-bin]
     |                `-gnome-pty-helper
     |-gpm
     |-gweather
     |-kapm-idled
     |-3*[kdeinit]
     |-keventd
     |-khubd
     |-5*[kjournald]
     |-klogd
     |-lockd---rpciod
     |-lpd
     |-mdrecoveryd
     |-6*[mingetty]
     |-8*[nfsd]
     |-nscd---nscd---5*[nscd]
     |-ntpd
     |-3*[oafd]
     |-panel
     |-portmap
     |-rhnsd
     |-rpc.mountd
     |-rpc.rquotad
     |-rpc.statd
     |-sawfish
     |-screenshooter_a
```

```
|-sendmail
|-sshd---sshd---bash---su---bash
|-syslogd
|-tasklist_applet
|-vmnet-bridge
|-xfs
`-xinetd-ipv6
```

Опции -u и -a дают дополнительную информацию. Для знакомства с другими опциями, а также тем, что они делают, направляем к info-страницам.

В следующем разделе мы увидим, как один процесс может создавать другой.

Жизнь и смерть процесса

Создание процесса

Новый процесс появляется вследствие того, что существующий процесс создает точную копию самого себя. У этого дочернего процесса тоже окружение, что и у родителя, отличается только номер ID процесса. Эта процедура называется *forking* (порождение).

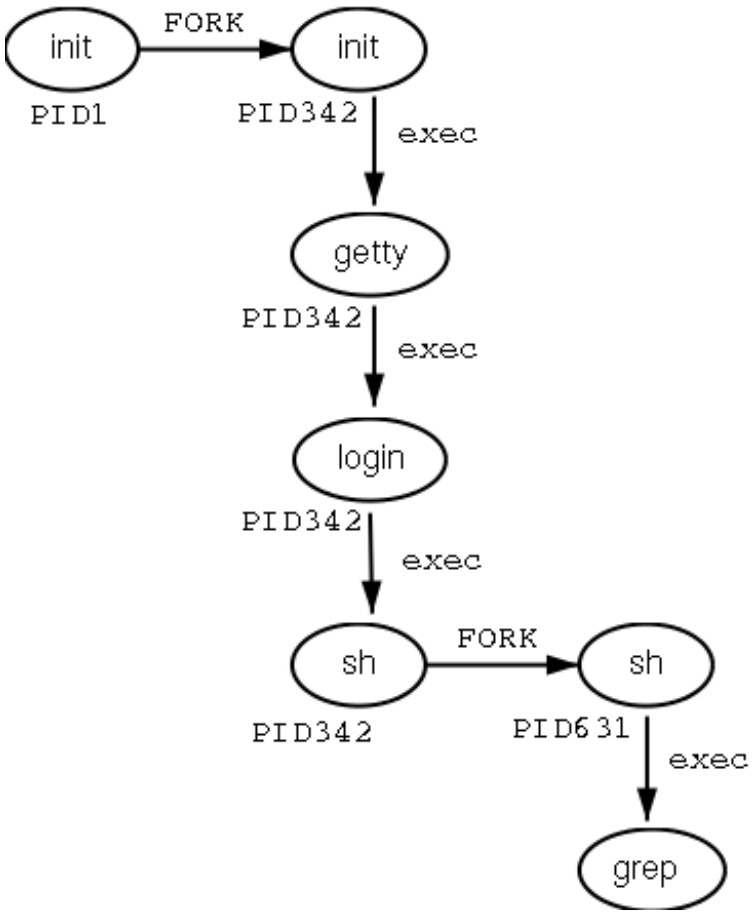
После рождения, адресное пространство дочернего процесса переписывается с новыми данными процесса. Это делается посредством вызова *exec* в системе.

Таким образом, механизм *fork-and-exec* переключает старые команды на новые, пока окружение, в котором выполняется новая программа остается таким же, в том числе конфигурация устройств ввода/вывода, переменные окружения и приоритет. Этот механизм используется для создания всех процессов UNIX, также это относится и к операционной системе Linux. Даже первый процесс, **init**, с процессорным ID 1, порождается во время загрузки в так называемой процедуре *bootstrapping* (начальной загрузки).

Эта схема иллюстрирует механизм *fork-and-exec*. ID процесса изменяется после процедуры порождения:

Рисунок 4.1. Механизм fork-and-exec

Есть несколько случаев, в которых **init** становится родителем процесса, хотя процесс не был запущен **init**, как мы уже видели в примере **pstree**. Многие программы, например, *daemonize* - их дочерние процессы, так они могут продолжать работать, когда родитель завершается или его завершают. Оконный менеджер является типичным примером; он запускается процессом **xterm**, который создает оболочку (shell), принимающую команды. Затем оконный менеджер отстраняется от какой-либо дальнейшей ответственности и передает дочерний процесс **init**. С помощью этого механизма можно менять оконные менеджеры, не прерывая запущенных приложений.



Время от времени что-то идет не так даже в хороших семьях. В исключительных случаях процесс может закончиться, в то время как родитель не ждет завершения этого процесса. Такой непогребенный процесс называется зомби-процессом.

Завершение процесса

Когда процесс завершается нормально (его не завершили или иным способом неожиданно не прервали), программа возвращает его *exit status* (статус выхода) родителю. Этот *exit status* является числом, возвращаемым программой, предоставившей результаты своего выполнения. Система передачи информации об исполненной работе берет свое начало с языка программирования Си, на котором был написан UNIX.

Коды возврата могут быть интерпретированы родителем или скриптами. Значения кодов возврата специфичны для программ. Эту информацию обычно можно найти в man-страницах указанной программы, например команда **grep** возвращает -1, если не найдено совпадений, при этом в строке может быть выведено сообщение: "Нет найденных файлов". Другим примером является встроенная команда Bash **true**, которая не делает ничего, кроме возвращения статуса выхода 0, что означает успех.

Сигналы

Процессы заканчиваются, если они получают сигнал. Есть несколько сигналов, которые вы можете отправить процессу. Используйте команду **kill**, чтобы послать сигнал процессу. Команда **kill** -l отображается список сигналов. Большинство сигналов предназначены для внутреннего использования системой или для программистов, когда они пишут код. Как пользователю вам потребуются следующие сигналы:

Таблица 4.2. Часто используемые сигналы

Имя сигнала	Номер сигнала	Значение
SIGTERM	15	Завершить процесс надлежащим образом.
SIGINT	2	Прерывание процесса. Процесс может игнорировать этот сигнал.

SIGKILL	9	Прерывание процесса. Процесс не может игнорировать этот сигнал.
SIGHUP	1	Для демонов: перечитать конфигурационный файл.

Вы можете прочитать больше о действиях по умолчанию, которые предпринимаются при передаче сигнала процессу, в **man** 7 *signal*.

SUID и SGID

Как и было обещано в предыдущей главе, теперь мы будем обсуждать специальные режимы SUID и SGID более подробно. Эти режимы существуют, чтобы обеспечить обычным пользователям возможность выполнять задачи, которые они, как правило, не могут осуществить из-за строгой схемы управления доступом к файлам, используемой в UNIX-системах. В идеальной ситуации специальные режимы используются так редко, насколько это возможно, так как они включают в себя риски для безопасности. Разработчики Linux, как правило, стараются также избегать их по возможности. Версия **ps** в Linux, например, использует информацию, хранящуюся в файловой системе /proc, которая доступна каждому, что позволяет избегать демонстрации важных системных данных и ресурсов для широкой общественности. До этого (и до сих пор на старых системах UNIX) программа **ps** нуждалась в доступе к файлам, таким как /dev/mem и /dev/kmem, что имело свои недостатки из-за разрешений и владельцев этих файлов:

```
rita:~> ls -l /dev/*mem
crw-r-----    1 root     kmem       1,   2 Aug 30 22:30 /dev/kmem
crw-r-----    1 root     kmem       1,   1 Aug 30 22:30 /dev/mem
```

В более ранних версиях **ps** не было возможности для запуска программы под обычным пользователем без применения специальных режимов.

Хотя мы обычно стараемся избегать применения каких-либо специальных режимов, иногда бывает необходимо использовать SUID. Примером может служить механизм смены пароля. Очевидно, что пользователи хотят делать это самостоятельно вместо того, чтобы их пароль устанавливал системный администратор. Как мы знаем, имена пользователей и пароли перечислены в файле /etc/passwd, который имеет следующие права доступа и владельцев:

```
bea:~> ls -l /etc/passwd
-rw-r--r--    1 root     root     1267 Jan 16 14:43 /etc/passwd
```

Тем не менее, у пользователей должна быть возможность изменять свою информацию в этом файле. Это достигается путем предоставления программой **passwd** специальных разрешений:

```
mia:~> which passwd
passwd is /usr/bin/passwd

mia:~> ls -l /usr/bin/passwd
-r-s--x--x    1 root     root    13476 Aug  7 06:03 /usr/bin/passwd*
```

При вызове команда passwd выполняется с использованием прав доступа *root*, что позволяет обычному пользователю редактировать файл паролей, который принадлежит системному администратору.

SGID режимы на файл не распространяются так же часто, как SUID, потому что SGID зачастую включает в себя создание дополнительных групп. В некоторых случаях, однако, мы должны пройти через эту неприятность, с тем чтобы выполнить элегантное решение (слишком об этом не беспокойтесь - необходимые группы обычно создаются при установке). Это происходит для программ **write** и **wall**, которые

используются для отправки сообщений другим пользовательским терминалами (ttys). Команда **write** выводит сообщение одному пользователю, а **wall** пишет всем подключенным пользователям.

Отправка текста иным пользовательским терминалам или на графический дисплей, как правило, не допускается. Для того чтобы обойти эту проблему, была создана группа, которая является владельцем всех терминальных устройств. Когда команды **write** и **wall** получают SGID разрешения, команды запустятся с помощью прав доступа, применимых к этой группе, *tty* в примере. Поскольку у этой группы есть право на запись в назначенный терминал, то пользователь, не имеющий разрешения на использование этого терминала, в любом случае может отправлять туда сообщения.

В приведенном ниже примере пользователь *joe* в начале узнает, с каким терминалом связан его корреспондент, для этого он использует команду **who**. Затем он отправляет ей сообщение с помощью команды **write**. Кроме того, иллюстрируются права доступа на программу **write** и какие терминалы занимает принимающий пользователь: очевидно, что отличные от пользователя-владельца не имеют разрешения на устройство, за исключением владельца-группы, которая может писать в него.

```
joe:~> which write
write is /usr/bin/write

joe:~> ls -l /usr/bin/write
-rwxr-sr-x    1 root     tty      8744 Dec  5 00:55 /usr/bin/write*

joe:~> who
jenny     tty1     Jan 23 11:41
jenny     pts/1    Jan 23 12:21 (:0)
jenny     pts/2    Jan 23 12:22 (:0)
jenny     pts/3    Jan 23 12:22 (:0)
joe       pts/0    Jan 20 10:13 (lo.callhost.org)

joe:~> ls -l /dev/tty1
crw--w----    1 jenny   tty  4,     1 Jan 23 11:41 /dev/tty1

joe:~> write jenny tty1
hey Jenny, shall we have lunch together?
^C
```

Пользователь *jenny* получает это на ее экране:

```
Message from joe@lo.callhost.org on ptys/1 at 12:36 ...
hey Jenny, shall we have lunch together?
EOF
```

После получения сообщения, терминал может быть очищен помощью комбинации клавиш **Ctrl + L**. Для того, чтобы не получать никакие сообщения вообще (кроме тех, что от системного администратора), используйте команду **mesg**. Чтобы узнать, какие подключенные пользователи принимают сообщения от других, используют **who** -w. Все особенности полностью разъяснены на info-страницах для каждой команды.

Названия групп могут быть различными.
Групповая схема специфична для дистрибутива. Разные дистрибутивы могут использовать разные имена или различные решения.

Процесс загрузки, инициализация и завершение работы

Введение

Одна из наиболее мощных сторон Linux связана с ее открытым методом запуска и остановки операционной системы; Linux загружает определенные программы, используя их особую конфигурацию, что позволяет вам изменять эти конфигурации для контроля процесса загрузки и останавливать работу системы элегантно и организовано.

Помимо задач контроля загрузки и процесса завершения работы, открытый характер Linux облегчает в большинстве случаев точное определение источников проблем, связанных с запуском и остановкой системы. Понимание основ этого является весьма полезным для всех, кто использует Linux.

Множество систем Linux используют **lilo** (LInux LOader) для загрузки операционных систем. Мы обсудим только GRUB, т. к. он проще в использовании и является более гибким. Если вам необходима информация о **lilo**, обратитесь к man-страницам и HOWTO. Оба загрузчика поддерживают двойные загрузки; мы ссылаемся на HOWTO по этому вопросу для практических примеров и справочной информации.

Загрузочный процесс

Когда загружается компьютер x86, процессор просматривает конец системной памяти на обнаружение BIOS (Basic Input/Output System) и запускает его. (Программа BIOS записывается в постоянной памяти только для чтения, и всегда доступна для использования.) BIOS обеспечивает низкоуровневый интерфейс для периферийных устройств и контролирует первый шаг процесса загрузки.

BIOS тестирует систему, ищет и проверяет периферию, и затем ищет устройство, чтобы использовать его для загрузки системы. Обычно BIOS проверяет устройство для дискет (или на многих современных системах дисковод CD-ROM) на наличие загрузочного диска, если их там нет, то смотрит на жестком диске. Последовательность устройств, используемых для загрузки, обычно контролируется индивидуальными для системы настройками BIOS. После того, как Linux установлен на жесткий диск, BIOS ищет главную загрузочную запись (Master Boot Record - MBR), начиная с первого сектора первого жесткого диска, загружает его содержимое в память и передает ему управление.

MBR содержит инструкции о том, как загрузить загрузчик GRUB (или LILO), использующий предварительно заданные операционные системы. MBR загружает загрузчик, который принимает на себя процесс (если загрузчик установлен в MBR). В стандартной конфигурации Red Hat Linux GRUB использует настройки в MBR для отображения параметров загрузки в меню. Как только GRUB получил корректные инструкции для запуска операционной системы либо из его командной строки или из файла конфигурации, он находит необходимые загрузочные файлы и удаляется от управления машиной в данной операционной системе.

Особенности GRUB

Этот метод загрузки называется *прямой загрузкой*, поскольку инструкции используются для непосредственной загрузки операционной системы, без какого-либо промежуточного кода между загрузчиком и основными файлами операционной системы (например, ядром). Процесс загрузки, используемый в других операционных системах, может незначительно отличаться от вышеуказанного. Например, операционные системы Microsoft (DOS и Windows) полностью переписывают MBR, когда они устанавливаются без учета какой-либо текущей конфигурации MBR. Все они стирают любую другую информацию, хранящуюся в MBR от других операционных систем, таких как Linux. Операционные системы Microsoft, также как и другие разные проприетарные операционные системы, загружаются с помощью метода цепной загрузки. При использовании этого метода, MBR указывает на первый сектор раздела, на котором

находится операционная система, где находит специальные файлы, необходимые для фактической загрузки данной операционной системы.

GRUB поддерживает оба загрузочных метода, что позволяет использовать его практически с любой операционной системой, наиболее популярными файловыми системами, и практически любым жестким диском, который сможет распознать ваш BIOS.

GRUB содержит ряд других особенностей. Из них наиболее важные:

- GRUB обеспечивает истинно командное, предшествующее ОС, окружение на машинах x86 для обеспечения максимальной гибкости при загрузке операционной системы с определенными параметрами или сбора информации о системе.
- GRUB поддерживает режим логической адресации блоков (LBA), необходимый для доступа ко многим IDE и всем SCSI жестким дискам. До LBA жесткие диски могли столкнуться с 1024-цилиндровый пределом, когда BIOS после этого места не может найти файл.
- Конфигурационный файл GRUB читается с диска каждый раз при загрузке системы, освобождая вас от необходимости писать поверх MBR при каждом изменении параметров загрузки.

Полное описание GRUB можно найти, выполнив команду **info** grub или на сайте GRUB. В Linux Documentation Project есть мини-HOWTO по мультизагрузке GRUB.

Init

Ядро, как только оно загружено, находит **init** в sbin и выполняет его.

Когда **init** запускается, то становится родителем и прародителем всех процессов, которые запускаются автоматически в вашей системе Linux. Первое, что делает **init**, читает его файл инициализации /etc/inittab. Он инструктирует **init** прочитать первоначальный сценарий конфигурации для окружения, в котором определены пути, запускается своппинг, проверяются файловые системы и так далее. В принципе, этот шаг позаботится обо всем, что ваша система должна сделать при своей инициализации: установка часов, инициализация последовательных портов и т.д.

Затем **init** продолжает читать файл /etc/inittab, который описывает, каким образом система должна подниматься на каждом уровне запуска и устанавливает *уровень выполнения* по умолчанию. *Уровень выполнения* определяет конфигурацию процессов. Все UNIX-подобные системы могут работать в различных конфигурациях, среди них есть однопользовательский режим, который называется уровнем выполнения 1 или S (или s). В этом режиме только системный администратор может подключиться к системе. Он используется для выполнения задач по обслуживанию без риска повреждения системы или пользовательских данных. Естественно, в такой конфигурации нам не следует предоставлять пользовательские службы, так что все они будут отключены. Другой уровень выполнения — это режим перезагрузки, или уровень 6, который завершает все запущенные службы, выполнив надлежащие процедуры, и перезагружает систему.

Используйте **who** для проверки, каким является ваш текущий уровень выполнения:

```
willy@ubuntu:~$ who -r
 run-level 2 2006-10-17 23:22  last=S
```

Подробнее об уровнях выполнения в следующем разделе, см. Раздел "Уровни выполнения Init".

После того, как для вашей системы определяется уровень выполнения по умолчанию, **init** запускает все фоновые процессы, необходимые для запуска системы, просматривая соответствующую для данного

уровня выполнения директорию rc. **init** выполняет каждый kill-сценарий (их файловые имена начинаются с K) с параметром stop. Затем он проходит все запускающие сценарии (их имена файлов начинаются с S) в соответствующем уровню выполнения каталоге, с тем, чтобы все службы и приложения запустились правильно. На самом деле вы вручную можете выполнить те же сценарии после того, как система завершит загрузку, командами подобными **/etc/init.d/httpd** stop или **service** httpd stop, войдя в систему с правами *root*; в этом случае останавливается веб-сервер.

 **Особый случай.**
Обратите внимание, что при запуске операционной системы скрипты в rc2.d и rc3.d, как правило, выполняются. В этом случае, никакие службы не останавливаются (по крайней мере, не навсегда). Службы только запускаются.

Ни один из сценариев, которые фактически запускают и останавливают службы, не расположен в /etc/rc.d. Вернее, все файлы в /etc/rc.d. являются символическими ссылками, указывающими на реальные сценарии, находящихся в /etc/init.d. Символическая ссылка это не более чем файл, который указывает на другой файл; в данном случае они используются, т. к. их можно создавать и удалять без ущерба для фактических сценариев, завершающих или запускающих службы. Символические ссылки на различные сценарии нумеруются в определенном порядке, в таком порядке они и запускаются. Вы можете изменить порядок запуска и остановки служб, изменив названия символических ссылок, которые ссылаются на сценарий, который фактически управляют службами. Вы можете использовать один и тот же номер множество раз, если хотите, чтобы определенный сервис запускался или останавливался непосредственно перед или после другого; в примере ниже перечислено содержимое /etc/rc5.d, в котором у **crond** и **xfs** имя ссылки начинается с "S90". В этом случае, скрипты запускаются в алфавитном порядке.

```
[jean@blub /etc/rc5.d] ls
K15httpd@     K45named@    S08ipchains@  S25netfs@       S85gpm@
K16rarpd@     K46radvd@    S08iptables@  S26apmd@        S90crond@
K20nfs@       K61ldap@     S09isdn@      S28autofs@      S90xfs@
K20rstatd@    K65identd@   S10network@   S30nscd@        S95anacron@
K20rusersd@   K74ntpd@     S12syslog@    S55sshd@        S95atd@
K20rwalld@    K74ypserv@   S13portmap@   S56rawdevices@ S97rhnsd@
K20rwhod@     K74ypxfrd@   S14nfslock@   S56xinetd@      S99local@
K25squid@     K89bcm5820@  S17keytable@  S60lpd@
K34yppasswdd@  S05kudzu@    S20random@    S80sendmail@
```

Затем **init** проходит через уровень выполнения, чтобы задать уровень по умолчанию, сценарий /etc/inittab разветвляется процессом **getty** для каждой виртуальной консоли (приглашение на вход в текстовом режиме). **getty** открывает строки tty, устанавливает их режимы, печатает строку входа, получает имя пользователя, и затем начинает процесс входа для данного пользователя. Это позволяет пользователям представится системе и использовать ее. По умолчанию, большинство систем предлагают 6 виртуальных консолей, но, как вы можете увидеть из файла inittab, это можно изменить.

/etc/inittab может также указать **init**, как тот должен обрабатывать пользовательское нажатие **Ctrl + Alt + Delete** в консоли. Поскольку система должна надлежащим образом выключаться и перезапускаться, а не прекращать немедленно работу, **init** сообщается выполнить команду /sbin/shutdown -t3 -r now , например, когда пользователь нажимает эти клавиши. Кроме того, /etc/inittab устанавливает, что **init** должен делать в случае отключения электропитания, если у вашей системы есть UPS unit, прикрепленный к ней.

В большинстве систем, основанных на RPM, графический экран для входа запускается на уровне выполнения 5, где /etc/inittab запускает скрипт под названием /etc/X11/prefdm. Скрипт prefdm преимущественно работает в графическом менеджере X, основываясь на содержимом каталога

/etc/sysconfig/desktop. Это, как правило, **gdm**, если вы работаете на GNOME или **kdm**, если вы используете KDE, но они могут быть смешаны, кроме того, есть **xdm**, который предоставляется при стандартной установке X.

Но есть и другие возможности. В Debian, например, есть initscript для каждого менеджера дисплея, и содержание /etc/X11/default-display-manager используется для определения того, какой из них запустить. Подробнее о графическом интерфейсе можно прочитать в Разделе 7.3. "Графическая среда". В конечном счете, ваша системная документация разъяснит подробности высокоуровневых аспектов **init**.

Каталоги /etc/default и/или /etc/sysconfig содержат записи для целого ряда функций и служб, все это читается во время загрузки. Местонахождение директории, содержащей системные настройки по умолчанию, может несколько различаться в зависимости от вашего дистрибутива Linux.

Кроме того, графическая среда пользователя, ряд других служб могут быть запущены также. Но если все пойдет хорошо, то когда процесс загрузки завершается, вы видите приглашение для входа или графический экран для входа.

 **Другие процедуры.**
Мы объяснили, как SysV **init** работ на базе x86. Запуск процедур может различаться на других архитектурах и дистрибутивах. Другие системы могут использовать BSD-стиль **init**, когда загрузочные файлы не разделяются на множество каталогов /etc/rc.d. Также может оказаться, что ваша система использует /etc/rc.d/init.d вместо /etc/init.d.

Уровни выполнения Init

Идея функционирования различных служб на разных уровнях выполнения по существу вращается вокруг того факта, что различные системы могут быть использованы по-разному. Некоторые службы не могут быть использованы, пока система находится в определенном состоянии или режиме, таком как быть готовой к более чем одному пользователю или доступной сети.

Бывают периоды, в которые вы можете работать с системой в низкоуровневом режиме. Примеры: решение проблем испорченного диска на уровне выполнения 1, так чтобы у других пользователей не было возможности быть в системе, или оставить сервер на уровне выполнения 3 (без работающей X сессии). В этих случаях, функционирование сервисов, которые зависят от высокоуровневого режима системы, не имеет смысла, т. к. они не будут работать правильно в любом случае. Уже имеющимся службам назначается их запуск, когда будет достигнут определенный уровень выполнения; вы обеспечиваете последовательный запуск процессов, и вы можете быстро изменить режим машины, не беспокоясь о том, что вручную придется запускать и останавливать службы.

Доступные уровни выполнения, как правило, описаны в файле /etc/inittab, содержимое которого частично показано ниже:

```
#
# inittab   This file describes how the INIT process should set up
#           the system in a certain run-level.

# Default run level. The run levels are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS
# (The same as 3, if you do not have networking)
```

```
#    3 - Full multiuser mode
#    4 - unused
#    5 - X11
#    6 - reboot (Do NOT set initdefault to this)
#
id:5:initdefault:
<--cut-->
```

Вы можете настроить неиспользованные уровни выполнения (обычно уровень 4) как вы считаете нужным. Многие пользователи настраивают эти уровни выполнения, таким образом, который наиболее подходит для них, оставляя стандартные уровни, поскольку они существуют по умолчанию. Это позволяет им быстро входить в и выходить из их пользовательской конфигурации, не нарушая нормальной установки свойств на стандартных уровнях запуска.

Если ваша машина оказывается в ситуации, когда она не может загрузиться из-за испорченного /etc/inittab или не позволят вам войти, т.к. вы повредили файл /etc/passwd (или если вы просто забыли свой пароль), загрузитесь в однопользовательском режиме.

 **Без графики?**
Когда вы работаете в текстовом режиме, т. к. не был предоставлен экран графического входа на вашей машине, обычно вы можете переключаться на консоль 7. Если это не так, проверить текущий режим работы с помощью команды **who** -r. Если он установлен на что-нибудь другое, чем есть оригинальные установки по умолчанию в /etc/inittab, то вероятно это может быть причина того, что система не запускается в графическом режиме по умолчанию. В таком случае обратитесь к системному администратору или прочитайте **man** init. Заметим, что переключение режимов работы осуществляется преимущественно с использованием команды **telinit**; переключение из текстовой в графическую консоль или наоборот не подразумевает изменение режима выполнения.

Мы стараемся в данном руководстве обсуждать режимы выполнения, скрипты и конфигурации в общих чертах. Существует множество вариаций. Например, Gentoo Linux хранит скрипты в уровнях /etc/run. Другие системы могут запускаться через первый нижний режим выполнения и выполнить все сценарии в нем до прибытия конечного режима выполнения и исполнения его сценариев. Обратитесь к системной документации для получения дополнительной информации. Вы также можете прочитать скрипты, которые относятся к /etc/inittab, чтобы лучше понять, что происходит в вашей системе.

Инструменты

Если в вашей системе установлены утилиты **chkconfig** или **update-rc.d**, то они обеспечат простой инструмент командной строки для поддержания иерархии каталога /etc/init.d. Это освобождает системных администраторов от необходимости напрямую управлять многочисленные символическими ссылками в каталогах в /etc/rc[x].d.

Кроме того, некоторые системы предлагают инструмент **ntsysv**, который предоставляет текстовый интерфейс; вы можете сочти его легче в использовании, чем интерфейс командной строки **chkconfig**. В SuSE Linux, вы обнаружите инструменты **yast** и **insserv**. В Mandriva есть Mandriva Linux Control Center (Центр управления Mandriva Linux).

Большинство дистрибутивов обеспечивают графический пользовательский интерфейс для настройки процессов; обратитесь к вашей системной документации.

Все эти утилиты должны быть запущен под root. Системный администратор также может вручную создать

соответствующие ссылки в каждом каталоге режима выполнения для того, чтобы запустить или остановить службы определенного режима.

Shutdown – выключение

UNIX была создана как система, которая не останавливается, но если вам это действительно надо, используйте команду **shutdown**. Опция -h будет остановить систему, а -r будет перезагружать ее.

Команды **reboot** и **halt** теперь способны вызывать **shutdown**, когда система находится в режимах выполнения 1-5, и таким образом обеспечить надлежащее завершение ее работы, но это плохая привычка, поскольку не во всех версиях UNIX/Linux есть такая возможность.

Пока ваш компьютер сам не выключится, вы не должны выключать его, пока не увидите сообщение с указанием, что система остановилась или все процессы завершились, т. к. система должна отключить все разделы. Нетерпеливость может привести к потере данных.

Управление процессами

Работа для системного администратора

Хотя управление системными ресурсами, включая процессы, является задачей местного системного администратора, но и обычному пользователю не повредит кое-что узнать об этом, особенно в той части, которая касается оптимального выполнения его или ее собственных процессов.

Мы немного расскажем на теоретическом уровне о производительности системы, но не будем касаться оптимизации оборудования и других продвинутых процедур. Вместо этого мы изучим насущные проблемы, с которыми сталкивается обычный пользователь, и действия, которые пользователь может предпринять для обеспечения оптимального использования имеющихся ресурсов. Как мы узнаем в следующем разделе, следует поразмыслить, прежде чем действовать.

Рисунок 4.2. Ты не мог бы идти побыстрее?

Сколько времени это займет?

Bash предоставляет встроенную команду **time**, которая отображает, сколько времени занимает выполнение команды. Она рассчитывает время с высокой точностью и может быть использована с любой командой. В примере ниже, проходит около полторы минуты to make this book:

```
tilly:~/xml/src> time make
Output written on abook.pdf (222 pages, 1619861 bytes).
Transcript written on abook.log.

real 1m41.056s
user 1m31.190s
sys 0m1.880s
```

Команда GNU **time** в /usr/bin (в отличие от версии, встроенной в shell) отображает больше информации, которая может быть отформатирована различными способами. Она также показывает статус завершения команды и общее истекшее время. Такая же команда, как выше, использующая независимый **time**, дает следующий вывод:

```
tilly:~/xml/src> /usr/bin/time make
Output written on abook.pdf (222 pages, 1595027 bytes).
Transcript written on abook.log.
```

```
Command exited with non-zero status 2
88.87user 1.74system 1:36.21elapsed 94%CPU
    (0avgtext+0avgdata 0maxresident)k
0inputs+0outputs (2192major+30002minor)pagefaults 0swaps
```

Для получения полной информации снова обратитесь к info-страницам.

Исполнение

Для пользователей эффективность означает быстрое выполнение команд. Для системного администратора, с другой стороны, это означает гораздо больше: администратор оптимизирует производительность всей системы, включая пользователей, все программы и демоны. Производительность системы может зависеть от тысячи незначительных вещей, которые не учитываются командой **time**:

- выполняющаяся программа плохо написана или использует компьютер надлежащим образом
- доступ к дискам, контроллерам, дисплею и всем другим видам интерфейсов
- достижимость удаленных систем (производительность сети)
- количество пользователей в системе, количество пользователей, одновременно работающих на самом деле.
- время дня
- ...

Нагрузка

Короче говоря, нагрузка зависит от того, что является нормальным для вашей системы. На моем старом P133 работает брандмауэр, SSH сервер, файловый сервер, служба маршрутизации, sendmail сервер, прокси сервер и некоторые другие службы, все они не жалуются на семь подключенных пользователей; нагрузка по-прежнему 0 или в пределах нормы. Некоторые (многопроцессорные) системы, которые я видела, были вполне счастливы с нагрузкой в 67. Существует только один способ узнать, что нагрузка нормальна, - это проверять ее регулярно. Если так не делать, вы только сможете оценить загрузку системы по времени отклика в командной строке, что является очень грубым измерением, поскольку эта скорость зависит от сотни других причин.

Имейте в виду, что разные системы будут вести себя по-разному с одной и той же средней нагрузкой. Например, системе с видеокартой, поддерживающей аппаратное ускорение, не будет иметь никаких проблем при прорисовке 3D изображений, в то же время в такой же системе с VGA видеокартой будет чрезвычайно замедленное время рендеринга. Мой старый P133 будет довольно неудобен, когда я запущу X-сервер, но на современной системе вы вряд ли заметите разницу в нагрузке на систему.

Я могу что-нибудь сделать как пользователь?

Большое окружение замедляет работу. Если у вас установлено множество переменных окружения (вместо переменных оболочки), длинных путей поиска, которые не оптимизированы (ошибки в настройках переменной окружения PATH), и присутствует много настроек, которые обычно делаются "на лету", системе будет требоваться больше времени на поиск и чтение данных.

В X, оконные менеджеры и среды рабочего стола могут действительно «съедать» процессор. На самом деле приходится расплачиваться за фантастический рабочий стол, даже если вы можете скачать его бесплатно, поскольку большинство десктопов снабжаются дополнениями до бесконечности. Скромность

окажется добродетелью, если вы не покупаете новый компьютер каждый год.

Приоритет

Приоритет или важность работы определяется ее числом *nice*. Программа с большим nice-числом дружественна к другим программам, другим пользователям и системе; т.е. эта программа - не важное дело. Низкое nice-число обозначает более важное задание, которое потребует больше ресурсов, не делясь ими.

Улучшение выполнения задания за счет увеличения его nice-числа полезно только для процессов, которые используют много процессорного времени (компиляторы, математические приложения и т.п.). Процессы, которые постоянно обращаются к вводу/выводу имеют более высокий приоритет (низкое nice-число), например, ввод с клавиатуры всегда получает наивысший приоритет в системе.

Определение приоритетов программ осуществляется с помощью команды **nice**.

Большинство систем также предоставляют команду BSD **renice**, которая позволяет изменять дружелюбность выполняющейся команды. Опять же, читайте man-страницы для получения системно-специфичной информации.

 **Интерактивные программы.**
Плохая идея, выполнять nice или renice по отношению к интерактивной программе или заданию, работающему на переднем плане.

Использование этих команд, как правило, является задачей системного администратора. Читайте man-страницы для дополнительной информации о дополнительной функциональности, доступной для системного администратора.

Ресурсы процессора

На каждой системе Linux множество программ одновременно нуждаются в ресурсах процессора(ов), даже в том случае, если вы являетесь единственным пользователем в системе. Каждая программа для работы нуждается в определенном количестве циклов центрального процессора. Бывают случаи, когда циклов процессора недостаточно, поскольку он слишком занят. Команда **uptime** чрезвычайно неточна (отображает только средние, вы должны знать, что это нормально), но далеко не бесполезна. Есть некоторые действия, которые можно предпринять, если вы думаете, что ваш процессор является причиной зависания вашей системы:

- Выполняйте тяжелые программы, когда нагрузка невелика. Это можно делать в вашей системе в ночное время. См. следующий раздел про планирование.
- Предотвращайте систему от выполнения ненужной работы: остановите демонов и программы, которые вы не используете, используйте **locate** вместо тяжелой **find**, ...
- Выполняйте больше заданий с низким приоритетом.

Если ни одно из этих решений не вариант в вашей конкретной ситуации, вы можете обновить процессор. На машине UNIX это работа для системного администратора.

Ресурсы памяти

Когда текущие работающие процессы ожидают больше памяти, чем доступно физически в системе, Linux не разрушится; она начнет подкачку или своппинг, смысл в том, что процесс использует память на диске или из пространства подкачки, перемещая содержимое физической памяти (части работающих программ или

целых программ в случае своппинга) на диск, тем самым давая физической памяти обрабатывать несколько процессов. Это в достаточной степени замедляет работу системы, поскольку доступ к диску намного медленнее, чем доступ к памяти. Команда **top** может быть использована для отображения того, насколько память и подкачка используются. Системы, использующие glibc, предлагают команды **memusage** и **memusagestat** для визуализации использования памяти.

Если вы обнаружите, что используется много памяти и пространства подкачки, вы можете попробовать:

- Прерывание, остановку или переопределение nice для тех программ, которые используют большую часть памяти.
- Добавление дополнительной памяти (а в некоторых случаях больше пространства подкачки) системе.
- Настройка производительности системы, что выходит за рамки этого документа. См. список в Приложении A, «Куда идти дальше?» за дополнением.

Ресурсы I/O (ввода/вывода)

Хотя ограничения ввода-вывода являются основной причиной стресса системных администраторов, системы Linux предлагает довольно бедные утилиты для измерения I/O производительности. Инструменты **ps**, **vmstat** и **top** дают некоторое представление о том, сколько программ ждут ввода-вывода; **netstat** отображает интерфейс сетевой статистики, но в сущности нет инструментов нормального измерения I/O отклика при загрузке системы, а команда **iostat** дает краткий обзор общего I/O использования. Существуют различные графические приложения для отображения вывода этих команд в понятном виде.

Каждое устройство имеет свои собственные проблемы, но пропускная способность сетевых интерфейсов и пропускная способность дисков два основных узких места в производительности ввода-вывода.

Сетевые I/O проблемы:

- Сетевые перегрузки:

Количество данных, передаваемых по сети больше возможностей сети, в результате медленное выполнение каждой связанной с сетью задачи для всех пользователей. Это может быть решено путем очистки сети (которая в основном заключается в отключении протоколов и услуг, которые вам не нужны), либо путем реконфигурации сети (например, использование подсетей, замена узлов с переключателями, модернизация интерфейсов и оборудования).

- Проблемы сетевой целостности:

Случается, когда данные передаются неправильно. Решение такого рода проблем может быть выполнено только путем изоляции неисправного элемента и его заменой.

Дисковые I/O проблемы:

- для каждого процесса скорость передачи слишком низкая:

Скорость чтения и записи для определенного процесса не является достаточной.

- совокупная скорость передачи слишком низкая:

Максимум общей пропускной способности, которые система может обеспечить для всех выполняющихся программ, недостаточна.

Обнаружить такого рода проблемы сложнее, и обычно это требует дополнительное оборудование для того,

чтобы разделить потоки данных по шинам, контроллерам и дискам, если причина проблемы — это перегрузка оборудования. Одним из решений этого является оптимизированная конфигурация RAID-массива для действий ввода-вывода. Так у вас появляется возможность остаться при том же оборудовании. Перед обновлением пропускной способности шин, контроллеров и дисков всегда можно попробовать другие варианты.

Если перегрузка не является причиной, может быть, ваше оборудование постепенно портится, или неправильно подключено к системе. Проверьте контакты, соединители и разъемы для начала.

Пользователи

Пользователи могут быть разделены на несколько классов в зависимости от их отношения к использованию ресурсов:

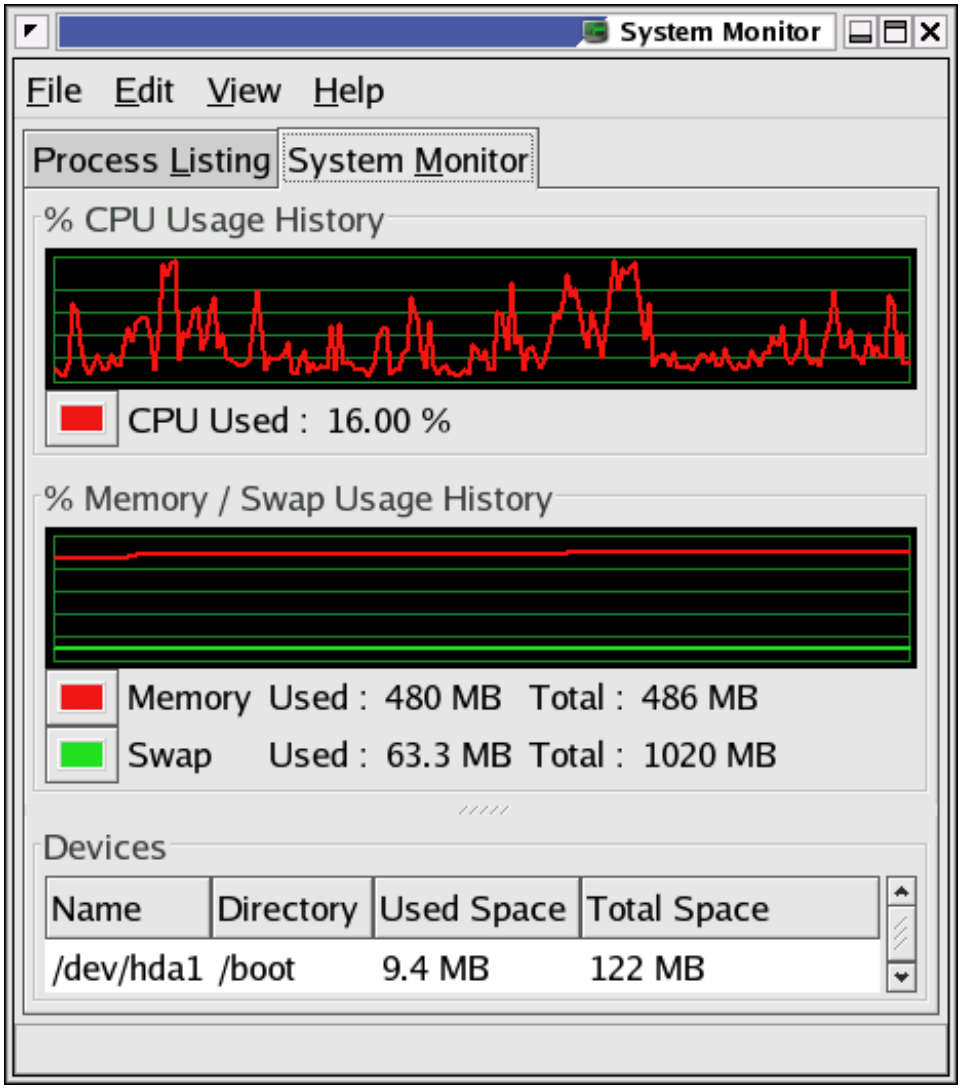
- Пользователи, которые запускают большое число маленьких заданий: вы, начинающий пользователь Linux, например.
- Пользователи, которые запускают относительно немного, но больших заданий: пользователи, работающие с моделированием, расчетами, эмуляторами или другими программами, которые съедают много памяти, и обычно этих пользователей сопровождает большой объем файлов данных.
- Пользователи, которые запускают немного заданий, но используют много процессорного времени (разработчики и т.п.).

Вы можете видеть, что системные требования могут отличаться для каждого класса пользователей, и поэтому сложно удовлетворить каждого. Если вы работаете на многопользовательской системе, полезно (и приятно) узнать привычки других пользователей системы, с тем чтобы получить максимальную отдачу от ее для выполнения ваших конкретных задач.

Графические инструменты

Для графической среды доступен целый букет инструментов мониторинга. Ниже приведен снимок экрана Системного Монитора Gnome, который способен отображать и искать информацию о процессах и контролировать системные ресурсы:

Рисунок 4.3. Системный Монитор Gnome



Также существует несколько полезных иконок, которые вы можете установить на панели задач, таких как мониторы диска, памяти и загруженности. **Xload** - это другое небольшое приложение X для мониторинга загрузки системы. Определитесь сами со своими предпочтениями.

Прерывание ваших процессов

Как непривилегированный пользователь, вы можете влиять только на собственные процессы. Мы уже видели, как вы можете отображать процессы и отфильтровывать процессы, относящиеся к конкретному пользователю, а также знаете, какие могут произойти возможные ограничения. Когда вы видите, что один из ваших процессов потребляет слишком много системных ресурсов, есть две вещи, которые вы можете сделать:

- Заставить процесс использовать меньше ресурсов, не прерывая его;
- Остановить процесс вообще.

В случае, если вы хотите, чтобы процесс продолжил работу, но также хотите дать шанс другим процессам в системе, вы можете переопределить *nice* процесса. Наряду с использованием команд **nice** или **renice**, **top** является простым способом обнаружения процесса(ов), причиняющего беспокойство, и снижения приоритета.

Узнайте процесс в столбце "NI", у него, скорее всего, будет отрицательный приоритет. Наберите r и введите ID процесса, для которого вы хотите переопределить число *nice*. Затем введите значение nice, например,

"20". Это означает, что отныне этот процесс будет занимать самое большое 1/5 циклов процессора.

Примеры процессов, которые вы захотите сохранить в рабочем состоянии, — это эмуляторы, виртуальные машины, компиляторы и т.д.

Если вы хотите остановить процесс, потому что он висит или собирается полностью вывести из рабочего состояния ввод-вывод, создание файла или использование других ресурсов системы, используйте команду **kill**. Если у вас есть возможность, попробуйте сначала завершить процесс "по тихому", отправив ему сигнал SIGTERM. Это указание к прекращению любой активности в соответствии с процедурами, описанными в коде программы:

```
joe:~> ps -ef | grep mozilla
joe    25822 1  0 Mar11 ? 00:34:04 /usr/lib/mozilla-1.4.1/mozilla-

joe:~> kill -15 25822
```

В приведенном выше примере, пользователь *joe* остановил его браузер Mozilla, поскольку он подвис.

От некоторых процессов избавиться немного труднее. Если у вас есть время, вы можете отправить им сигнал SIGINT, прервав их. Если этот трюк не сработает, используйте самый сильный сигнал SIGKILL. В приведенном ниже примере *joe* останавливается Mozilla, которая повисла окончательно:

```
joe:~> ps -ef | grep mozilla
joe    25915 1  0 Mar11 ? 00:15:06 /usr/lib/mozilla-1.4.1/mozilla-

joe:~> kill -9 25915

joe:~> ps -ef | grep 25915
joe 2634 32273 0 18:09 pts/4   00:00:00 grep 25915
```

В таких случаях вы можете захотеть проверить, что этот процесс действительно прерван, используя фильтр **grep** опять же по PID. Если это возвращает только процесс **grep**, вы можете быть уверены, что вам удалось остановить тот процесс.

Среди процессов есть те, которые сложно прервать в вашей оболочке. И это хорошо: если бы их можно было легко прервать, то вы бы теряли вашу оболочку каждый раз при случайном вводе **Ctrl-C** в командной строке, т.к. это равносильно отправке SIGINT.

 **UNIX без конвейеров почти немыслим.**
Использование вертикальной черты (|) при использовании выводов одной команды в качестве входных данных другой объясняется в следующей главе, Глава 5, "Перенаправление ввода-вывода".

В графической среде, программа **xkill** очень проста в использовании. Просто введите имя команды, затем нажмите Enter и выберите в окне приложение, которое вы хотите остановить. Это довольно опасно, потому что программа отправляет SIGKILL по умолчанию, поэтому используйте ее, только когда приложение зависает.

Планирование процессов

Используйте незанятое время!

Система Linux может выдерживать сильную нагрузку, но обычно перегружается только в рабочее время.

Будь это офис, серверная комната или дом, большинство систем Linux тратят попусту время утром, вечером, ночью и в выходные. Использование этого незанятого времени может быть намного дешевле, чем покупка компьютеров, которые вам точно потребуются, если вы захотите все делать в одно и то же время.

Существует три типа для задержки выполнения работ:

- Небольшое ожидание, а затем возобновление выполнения задания, это делается командой **sleep**. Время запуска задания зависит от системного времени в момент подачи команды.
- Выполнение команды в определенное время, используя команду **at**. Выполнение задания(ий) зависит от системного времени, а не от времени подачи команды.
- Регулярный запуск команд по месяцам, неделям, дням или часам основано на использовании возможностей **cron**.

В следующих разделах описывается каждая возможность.

Команда sleep

Info-страница о сне, вероятно, одна из самых коротких. Все, что **sleep** делает, это ждет. По умолчанию время ожидания выражается в секундах.

Так зачем же она нужна? Некоторые практические примеры:

Кто-то звонит вам по телефону, вы говорите "Да, мы встретимся через полчаса", но вы тонете в работе и забываете про свой ланч:

(sleep *1800*; echo *"Lunch time.."*) &

Если вы не можете по какой-то причине использовать команду **at**, 5:00 часов, и вы хотите идти домой, но еще есть работа, которую предстоит сделать, а сейчас кто-то использует системные ресурсы:

(sleep *10000*; myprogram) &

Убедитесь, что в вашей системе есть авто-выход, и что вы выходите из системы или блокируется ваш рабочий стол при подаче такого рода задания, или запустите ее в сессии.

Когда вы запускаете на распечатку ряд больших файлов, но вы хотите, чтобы другие пользователи имели возможность печати между ними:

lp lotoftext; sleep *900*; lp hugefile; sleep *900*; lp anotherlargefile

Печать файлов обсуждается в Главе 8, Принтеры и печать.

Программисты часто используют команду **sleep** для остановки выполнения скрипта или программы на определенное время.

Команда at

Команда **at** выполняет команды в данный момент времени, используя вашу оболочку по умолчанию, если вы не укажете команде иное (см. man-страницу).

Опции **at** достаточно понятны, что продемонстрировано в примере ниже:

```
steven@home:~> at tomorrow + 2 days
```

```
warning: commands will be executed using (in order) a) $SHELL
        b) login shell c) /bin/sh
at>  cat reports | mail myboss@mycompany
at>
job 1 at 2001-06-16 12:36
```

Нажатие **Ctrl + D** завершает работу утилиты **at** и создает сообщение "EOT".

Пользователь *steven* делает здесь странные вещи, объединяя две команды; мы будем изучать подобную практику в Главе 5, Перенаправление ввода-вывода.

```
steven@home:~> at 0237
warning: commands will be executed using (in order) a) $SHELL
        b) login shell c) /bin/sh
at>  cd new-programs
at>  ./configure; make
at>
job 2 at 2001-06-14 02:00
```

Опция -m отправляет сообщение пользователю, когда задание выполнено, или разъяснение, когда работа не может быть сделана. Команда **atq** создает список заданий; выполните эту команду перед отправкой заданий в целях предотвращения их от запуска одновременно с другими. Командой **atrm** можно удалить запланированные задания, если вы изменили свою точку зрения.

Хорошая идея выбрать необычное время выполнения, потому что системные задания часто работают "круглые" часы, как вы можете увидеть в Разделе "Cron и crontab" далее. Например, задания часто запускаются ровно в 1:00 утра (например, системы, индексирующие обновление стандартной локальной базы данных), поэтому ввод времени 0100 легко может замедлить работу системы, а не увеличить ее. Чтобы предотвратить задания от запуска всех в одно и то же время, вы можете также использовать команду **batch**, которая ставит процессы в очередь и задает работу системе сбалансированным образом, предотвращая чрезмерные очереди использования системных ресурсов. Смотрите info-страницы для получения дополнительной информации.

Cron и crontab

Система **cron** находится под управлением демона cron. Он получает информацию из записей crontab системы и пользователей о том, какие программы и когда должны работать. Только пользователь root имеет доступ к системным crontab, в то время как у каждого пользователя должен быть доступ только к собственным crontab. На некоторых системах у (некоторых) пользователей отсутствует доступ к объекту cron.

При запуске системы демон cron ищет /var/spool/cron/ на наличие записей crontab, названных в честь аккаунтов в /etc/passwd, он ищет /etc/cron.d/ и /etc/crontab, а затем ежеминутно использует эту информацию для проверки, есть ли что-то что нужно сделать. Он выполняет команды от пользователя, который является собственником файла crontab и отправляет владельцу на mail все выданные команды.

В системах, использующих хрон Vixie, задания, которые выполняются ежечасно, ежедневно, еженедельно и ежемесячно хранятся в отдельной директории в /etc чтобы было легче просматривать. Это отличается от стандартной функции хрон UNIX, где все задачи вводятся в один большой файл.

Пример файла crontab Vixie:

```
[root@blob /etc]# more crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
# commands to execute every hour
01 * * * * root run-parts /etc/cron.hourly
# commands to execute every day
02 4 * * * root run-parts /etc/cron.daily
# commands to execute every week
22 4 * * 0 root run-parts /etc/cron.weekly
commands to execute every month
42 4 1 * * root run-parts /etc/cron.monthly
```

 **Альтернатива.**
Вы можете также использовать команду **crontab -l** для отображения всех файлов crontab.

После установки некоторых переменных, появляется текущее расписание, по одной строке на задание, начинающиеся с 5-ти полей даты и времени. Первое поле содержит минут (от 0 до 59), второе - определяет час исполнения (0-23), третее - день месяца (1-31), и, наконец, номер месяца (1-12) , последнее - день недели (0-7, где 0 и 7 - это воскресенье). Звездочка в этих полях обозначает общий предел для поля. Списки допускаются; для выполнения заданий с понедельника по пятницу введите 1-5 в последнем поле, чтобы выполнить работу в понедельник, среду и пятницу введите 1,3,5.

Затем указывается пользователь, который должен выполнить процессы, которые перечислены в последнем столбце. Пример выше из конфигурации хрона Vixie, где root запускает программу **run-parts** на регулярной основе с соответствующими каталогами в качестве опций. В этих каталогах фактические задания для выполнения в назначенное время хранятся в виде скриптов, как этот маленький скрипт, который выполняется ежедневно, чтобы обновить базу данных, используемую командой **locate**:

```
billy@ahost cron.daily]$ cat slocate.cron
#!/bin/sh
renice +19 -p $$ >/dev/null 2>&1
/usr/bin/updatedb -f "nfs,smbfs,ncpfs,proc,devpts" -e \
"/tmp,/var/tmp, /usr/tmp,/afs,/net"
```

Пользователи должны отредактировать для своих crontab безопасный способ использования команды **crontab -e**. Это позволит предотвратить пользователя от случайного открытия более чем одной копии его/ее файла crontab. По умолчанию предоставлен редактор **vi** (см. Главу 6, Текстовые редакторы), но вы можете использовать любой текстовый редактор, такой как **gvim** или **gedit**, если вам более комфортно с редактором GUI.

При выходе система сообщит вам, что установлен новый crontab.

Эта запись crontab напоминает *billy* пойти в его спортивный клуб каждый вечер четверга:

```
billy:~> crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.20264 installed on Sun Jul 20 22:35:14 2003)
```

```
# (Cron version -- $Id: chap4.xml,v 1.28 2007/09/19 12:22:26 tille Exp $)
38 16 * * 3 mail -s "sports evening" billy
```

После добавления нового запланированного задания, система сообщит вам, что новый crontab установлен. Вам не нужно перезапустить демон **cron**, чтобы изменения вступили в силу. В примере *billy* добавил новую строку, указывающую на сценарий резервного копирования:

```
billy:~> crontab -e
45 15 * * 3 mail -s "sports evening" billy
4 4 * * 4,7 /home/billy/bin/backup.sh

<--write and quit-->

crontab: installing new crontab

billy:~>
```

Скрипт backup.sh запускается каждые четверг и воскресенье. См. Раздел "Сценарии оболочки" для введения в написание скриптов для оболочки. Имейте в виду, что вывод команд, если таковые имеются, отправляются по почте владельцу файла crontab. Если почтовая служба не настроена, можно найти вывод ваших команд в вашем локальном почтовом ящике, /var/spool/mail/, в виде обычного текстового файла.

 **Кто запускает мои команды?**
Вы не можете устанавливать пользователей, которые должны выполнять команды. Они выполняются с полномочиями пользователя по умолчанию.

Резюме

Linux является многопользовательской и многозадачной операционной системой с UNIX-подобным способом управления процессами. Скорость выполнения команд может зависеть от тысячи незначительных факторов. Среди прочего мы узнали много новых команд для отображения процессов и их управления. Вот список:

Команда	Значение
at	Очередь заданий для последующего выполнения.
atq	Списки ожидающих заданий пользователей.
atrm	Удаление заданий, определяется их номера.
batch	Выполнение команд, когда позволяет уровень загруженности системы
crontab	Сохранение crontab'ов для индивидуальных пользователей
halt	Остановка системы
init run level	Контроль процесса инициализации

jobs	Списки текущих выполняющихся работ
kill	Завершение процесса
mesg	Управление доступом на запись в вашем терминале.
netstat	Отображение сетевых подключений, таблиц маршрутизации, статистики интерфейсов и др.
nice	Запуск программы с измененным приоритетом.
pgrep	Отображение процессов
ps	Отчет о статусе процесса
pstree	Отображение дерева процессов
reboot	Остановка системы
renice	Изменение приоритета запущенных процессов
shutdown	Отключение системы
sleep	Задержка в течение определенного времени.
time	Команда времени или отчет использования ресурсов
top	Отображение процессов главного центрального процессора
uptime	Отображает, как долго система работает
vmstat	Отчет статистики виртуальной памяти
w	Показывает, кто вошел в систему, и что они делают.
wall	Отправка сообщения всем терминалам
who	Показывает, кто вошел в систему
write	Отправка сообщения другому пользователю

Упражнения

Основное

Вот некоторые упражнения, которые помогут вам ощутить процессы, запущенные в вашей системе.

- Выполните **top** в одном терминале, пока вы делаете упражнения в другом.
- Выполнить команду **ps**.

- Прочитайте man-страницы, чтобы узнать, как отобразить все ваши процессы.
- Выполните команду **find /**. Какой эффект это окажет на загруженность системы? Остановите эту команду.
- В графическом режиме запустите программу **xclock** на переднем плане. Затем отправьте ее работать в фоновый режим. Остановите программу, используя команду **kill**.
- Выполните **xcalc** сразу в фоновом режиме для того, чтобы приглашение терминала было доступно снова.
- Что делает **kill -9 -1**?
- Откройте два терминала или терминальных окна снова и используйте write для отправки сообщения с одного на другой.
- Выдайте команду **dmesg**. Что она сообщает?
- Сколько времени занимает выполнение **ls** в текущем каталоге?
- На основе записей процессов в /proc, принадлежащих вашей UID, чтобы вы сделали с целью узнать, какие из этих процессов действительно представлены?
- Как долго ваша система была запущена?
- Какой ваш текущий TTY?
- Название 3-х процессов, которые не могут иметь **init** в качестве первоначального родителя.
- Название 3-х команд, которые используют режим SUID. Объясните, почему это так.
- Название команд, которые, как правило, приводят к самой высокой загруженности вашей системы.

Загрузка, инициализация и т.д.

- Вы можете перезагрузить систему как обычный пользователь? Почему это так?
- В соответствии с вашим текущим уровнем выполнения перечислите шаги завершения работы.
- Как вы измените системный уровень выполнения? Перейдите с вашего уровня выполнения по умолчанию на уровень 1 и обратно.
- Составьте список всех служб и демонов, которые запущены, когда ваша система загрузилась.
- Какое ядро в текущий момент загружается при старте?
- Предположим, что вы должны запустить некие экзотические службы во время загрузки. До этого вы заходили в систему после ее загрузки и запускали эти службы вручную, используя сценарий под названием deliver_pizza в вашем домашнем каталоге. Что вы сделаете для того, чтобы запустить службы автоматически на уровне выполнения 4, который вы определили только для этой цели?

Планирование

- Используйте **sleep** для создания напоминания, что ваша паста готова через 10 минут.
- Задайте **at** задание, которое копирует все файлы в домашней директории в /var/tmp в течение получаса. Вы можете создать подкаталог в /var/tmp.
- Создайте cronjob, который решает эту задачу с понедельника по пятницу в обед.
- Убедитесь, что он работает.
- Сделать ошибку в записи crontab как выдача несуществующей команды **coppy** вместо **cp**. Что произойдет при выполнении задачи?

Глава 5. Перенаправление ввода-вывода (I/O)

Аннотация

В этой главе подробно описываются мощные механизмы UNIX, связанные с перенаправлением ввода, вывода и ошибок. Темы включают:

- Стандартные ввод, вывод и ошибки
- Операторы перенаправления
- Как использовать выходные данные одной команды в качестве входных данных другой
- Как отправить вывод команды в файл для последующего использования
- Как добавить вывод множества команд в файл
- Перенаправления ввода
- Обработка сообщений стандартных ошибок
- Комбинирование перенаправления потоков ввода, вывода и ошибок
- Выходные фильтры

Простые перенаправления

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Что такое стандартный ввод и стандартный вывод?

Большинство команд Linux считывают с ввода, которым может быть файл или другой атрибут команды, и записывают в вывод. По умолчанию ввод принимается с клавиатуры, а вывод отображается на экране. Клавиатура является вашим стандартным устройством *ввода* (stdin), а экран или конкретное окно терминала — стандартным устройством *вывода* (stdout).

Однако, поскольку Linux - гибкая система, эти настройки по умолчанию не обязательны к применению. Стандартный вывод, например, на сильно загруженном сервере может быть направлен на принтер.

Операторы перенаправления

Перенаправление вывода с > и |

Когда-нибудь вам захочется отправить вывод команды в файл, или вам может потребоваться применить другую команду к выводу первой. Это явление известно как перенаправление вывода. Перенаправление производится с использованием либо ">" (символ "больше чем"), либо с помощью оператора "|" (канал), который направляет стандартный вывод одной команды к другой команде в качестве стандартного ввода.

Как мы видели раньше, команда **cat** объединяет файлы и отправляет их все вместе на стандартный вывод. При перенаправлении этого вывода в файл он будет создан или перезаписан, если он уже существует, так что будьте аккуратнее.

```
nancy:~> cat test1
some words
```

```
nancy:~> cat test2
some other words

nancy:~> cat test1 test2 > test3

nancy:~> cat test3
some words
some other words
```

 **Не перезаписывайте!**
Будьте осторожны, чтобы не перезаписать существующие (важные) файлы при перенаправлении вывода. Многие оболочки, в том числе и Bash, имеют встроенную функцию для защиты вас от этого риска: **noclobber**. См. info-страницы для получения дополнительной информации. В Bash возможно вам захочется добавить команду **set** *-o noclobber* в ваш файл конфигурации .bashrc в целях предотвращения случайной перезаписи файлов.

Если перенаправить в файл «ничего», то он станет пустым:

```
nancy:~> ls -l list
-rw-rw-r--    1 nancy   nancy     117 Apr  2 18:09 list

nancy:~> > list

nancy:~> ls -l list
-rw-rw-r--    1 nancy   nancy       0 Apr  4 12:01 list
```

Этот процесс называется усечением.

Такое же перенаправление на несуществующий файл создаст новый пустой файл с заданным именем:

```
nancy:~> ls -l newlist
ls: newlist: No such file or directory

nancy:~> > newlist

nancy:~> ls -l newlist
-rw-rw-r--  1 nancy   nancy     0 Apr  4 12:05 newlist
```

В Главе 7, Дом, сладкий /home дается еще несколько примеров использования подобного способа переадресации.

Далее идут некоторые примеры использования конвейера команд.

Чтобы найти слово в каком-то тексте, отображаются все строки, соответствующие "pattern1", а также исключаются строки, соответствующие "pattern2":

grep *pattern1* file | grep -v *pattern2*

Отображение вывода каталога с перелистыванием одной страницы за раз:

ls -la | less

Нахождение файла в каталоге:

ls -l | grep *part_of_file_name*

Перенаправление ввода

В другом случае вам может понадобиться файл, который послужит вводом для команды, которая обычно не принимает файл в качестве параметра. Такое перенаправление ввода осуществляется с помощью оператора "<" (символ "меньше, чем").

Ниже приведен пример отправки кому-то файла с использованием перенаправления ввода.

```
andy:~> mail mike@somewhere.org < to_do
```

Если пользователь mike существует в данной системе, вам не нужно вводить полный адрес. Если вы хотите достать кого-то в Интернете, введите полный адрес в качестве аргумента **mail**.

По началу это читается немного сложнее, чем **cat** file | **mail** someone, но это, без сомнения, гораздо более удобный способ использования имеющихся инструментов.

Объединение перенаправлений

В следующем примере объединяются входное и выходное перенаправление. Файл text.txt сначала проверяется на наличие орфографических ошибок, а вывод перенаправляется в файл журнала ошибок:

spell < text.txt > error.log

Следующая команда выводит список всех команд, которые вы можете передать на рассмотрение другому файлу при использовании **less**:

```
mike:~> less --help | grep -i examine
  :e [file]      Examine a new file.
  :n          *  Examine the (N-th) next file from the command line.
  :p          *  Examine the (N-th) previous file from the command line.
  :x          *  Examine the first (or N-th) file from the command line.
```

Опция -i используется для поиска без учета регистра - вспомните, что системы UNIX очень чувствительны к регистру.

Если вы хотите сохранить вывод этой команды на будущее, перенаправьте вывод в файл:

```
mike:~> less --help | grep -i examine > examine-files-in-less

mike:~> cat examine-files-in-less
  :e [file]      Examine a new file.
  :n          *  Examine the (N-th) next file from the command line.
  :p          *  Examine the (N-th) previous file from the command line.
  :x          *  Examine the first (or N-th) file from the command line.
```

Вывод одной команды может быть передан другой команде фактически столько раз, сколько вам потребуется, но только до тех пор, пока эти команды читают со стандартного ввода и отправляют результат на стандартный вывод. Иногда этого не происходит, в таком случае могут существовать специальные опции, которые отдают распоряжение таким командам, делая их поведение соответствующим стандартным установкам; так что читайте документацию (man и info-страницы) используемым вами командам, если вам приходится сталкиваться с ошибками.

Опять же, убедитесь, что вы не используете имена существующих файлов, которые вам все еще нужны. Перенаправление вывода в существующий файл заменит его содержимое.

Оператор >>

Вместо перезаписи данных файла, вы можете также добавить текст в существующий файл с помощью двух следующих друг за другом символов "больше, чем".

Пример:

```
mike:~> cat wishlist
more money
less work

mike:~> date >> wishlist

mike:~> cat wishlist
more money
less work
Thu Feb 28 20:23:07 CET 2002
```

Команда **date**, как правило, выводит результат на экран, но теперь добавляется к файлу wishlist.

Расширенные свойства перенаправления

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Использование дескрипторов файла

Существует три типа I/O (ввода-вывода), которые имеют свои собственные идентификаторы, называемые дескрипторами файла:

- стандартный ввод: 0
- стандартный вывод: 1
- стандартная ошибка: 2

В следующих описаниях, если номер дескриптора файла не указан, и первый символ перенаправления оператор <, перенаправление ссылается на стандартный ввод (дескриптор файла 0). Если первый символ перенаправления оператор >, перенаправление ссылается на стандартный вывод (дескриптор файла 1).

Некоторые практические примеры внесут больше ясности:

ls > dirlist 2>&1

направит как стандартный вывод, так и стандартную ошибку в файл dirlist, в то время как команда

ls 2>&1 > dirlist

направит только стандартный вывод в dirlist. Это может быть полезным вариантом для программистов.

Существует сложность: не следует путать использование амперсанда здесь с использованием его в разделе

"Интерактивные процессы", где амперсанд используется для запуска процесса в фоновом режиме. В данном случае он просто служит признаком того, что число, которое следует за ним, является не именем файла, а местом, на которое указывается потоку данных. Также отметим, что символ "больше, чем" не должен быть отделен пробелами от номера дескриптора файла. Если он будет отделен, мы снова укажем на вывод в файл. Пример ниже демонстрирует это:

```
[nancy@asus /var/tmp]$ ls 2> tmp

[nancy@asus /var/tmp]$ ls -l tmp
-rw-rw-r--  1 nancy nancy 0 Sept  7 12:58 tmp

[nancy@asus /var/tmp]$ ls 2 > tmp
ls: 2: No such file or directory
```

Первая команда, которую выполняет nancy, правильна (даже несмотря на отсутствие ошибок, и поэтому файл, в который перенаправлялась стандартная ошибка пуст). Вторая команда предполагает, что 2 - это имя файла, которого в данном случае нет, поэтому отображается ошибка.
Все эти особенности подробно описаны в info-страницах Bash.

Примеры

Анализ ошибок

Если ваш процесс порождает много ошибок, есть способ тщательно их изучить:

command 2>&1 | less

Такое часто используется при создании нового программного обеспечения с использованием команды **make**, так как здесь:

```
andy:~/newsoft> make all 2>&1 | less
--output ommitted--
```

Отделение стандартного вывода от стандартной ошибки

Конструкции подобные этим часто используется программистами, в результате вывод отображается в одном окне терминала, а ошибки в другом. Сначала выясняется, какой псевдо терминал вы используете выдачей команды **tty**.

```
andy:~/newsoft> make all 2> /dev/pts/7
```

Одновременная запись в вывод и в файлы

Вы можете использовать команду **tee** для копирования ввода на стандартный вывод и один или несколько выходных файлов за один раз. Использование опции -a с **tee** в результате добавляет ввод в файл(ы). Эта команда полезна, если вы хотите как увидеть, так и сохранить вывод. Операторы > и >> не позволяют выполнить оба действия одновременно.

Этот инструмент обычно называется подачей на конвейер (|), что демонстрируется в следующем примере:

```
mireille ~/test> date | tee file1 file2
Thu Jun 10 11:10:34 CEST 2004
```

```
mireille ~/test> cat file1
Thu Jun 10 11:10:34 CEST 2004

mireille ~/test> cat file2
Thu Jun 10 11:10:34 CEST 2004

mireille ~/test> uptime | tee -a file2
 11:10:51 up 21 days, 21:21, 57 users,  load average: 0.04, 0.16, 0.26

mireille ~/test> cat file2
Thu Jun 10 11:10:34 CEST 2004
 11:10:51 up 21 days, 21:21, 57 users,  load average: 0.04, 0.16, 0.26
```

Фильтры

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Когда программа выполняет операции с вводом и записывает результат в стандартный вывод, она называется фильтром. Одним из наиболее распространенных видов использования фильтров является использование по реогранизованному выводу. Ниже мы обсудим несколько наиболее важных фильтров.

Дополнительная информация о grep

Как мы уже видели в разделе "Команда grep", **grep** сканирует строку вывода через строку поиска по определенным шаблонам. Все строки, содержащие шаблон будут распечатаны на стандартном выводе. Такое поведение может быть отменено путем использования опции -v.

Некоторые примеры: допустим, мы хотим знать, какие файлы в определенной директории были изменены в феврале:

```
jenny:~> ls -la | grep Feb
```

Команда **grep**, как и большинство команд, учитывает регистр. Используйте опцию -i, чтобы не учитывать различие между верхним и нижним регистром. Также доступны многие расширения GNU, например --color, который полезен для подсветки терминов поиска в длинных строках, и --after-context, который печатает число строк после последней совпавшей строки. Вы можете оформить рекурсивных **grep**, который ищет во всех подкаталогах встречающихся каталогов, с помощью опции -r. Как обычно, опции могут быть объединены.

Регулярные выражения могут быть использованы для более подробной детализации совпадений, которые вы хотите выбрать из всех строк ввода. Лучшим способом познакомиться с регулярными выражениями является чтение документации по **grep**. Отличная глава включена в info-страницу **grep**. Поскольку обсуждение вводов и выводов регулярных выражений завело бы нас слишком далеко, настоятельно рекомендуется начать с документации, если вы хотите знать о них больше.

Поиграйте немного с **grep**, это займет у вас какое-то время для изучения этой самой базовой, но очень мощной команды фильтрации. Упражнения в конце этой главы помогут вам начать работу, см. раздел 5.5. "Упражнения".

Фильтрация вывода

Команда **sort** по умолчанию сортирует строки в алфавитном порядке:

```
thomas:~> cat people-I-like | sort
Auntie Emmy
Boyfriend
Dad
Grandma
Mum
My boss
```

Но есть множество другого, что **sort** может делать. Просмотр размера файла, например. С помощью этой команды, содержимое каталога сортируется от наименьших файлов в начале, до самых больших в конце:

ls -la | sort -nk 5

 **Старый синтаксис sort.**
Вы можете получить тот же результат с **ls** -la | **sort** +4n, но это старая форма, которая не соответствует текущим стандартам.

Команда **sort** также используется в сочетании с программой **uniq** (или **sort** -u) для сортировки вывода и отфильтровывания двойных записей:

```
thomas:~> cat itemlist
1
4
2
5
34
567
432
567
34
555

thomas:~> sort itemlist | uniq
1
2
34
4
432
5
555
567
```

Резюме

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

В этой главе мы узнали, как команды могут быть связаны друг с другом, и как вывод одной команды может быть использован в качестве ввода для другой.

Перенаправление ввода/вывода — обычная задача на машинах UNIX и Linux. Этот мощный механизм позволяет гибко использовать блоки, из которых сделан UNIX.

Наиболее часто используемыми перенаправлениями являются > и |. См. Приложение C, "Особенности Shell" для обзора команд перенаправления и других конструкций shell.

Команда	Значение
date	Отображает дату и время
set	Настройка опций оболочки
sort	Сортирует строки текста
uniq	Удаляет повторяющиеся строки из отсортированного файла

Упражнения

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Эти упражнения представляют дополнительные примеры того, как объединять команды. Основная цель - попытаться использовать клавишу Enter как можно меньше.

Все упражнения выполняются под обычным пользовательским ID, так что появляются некоторые ошибки. Когда вы их видите, не забывайте прочитать man-страницы!

- Используйте команду **cut** на вывод длинного списка каталога, чтобы отобразить только права доступа к файлам. Затем отправьте в конвейере этот вывод на **sort** и **uniq**, чтобы отфильтровать все повторяющиеся строки. Потом с помощью **wc** посчитайте различные типы разрешений в этом каталоге.
- Отправьте вывод **date** в файл. Добавьте выход **ls** в этот же файл. Отправьте этот файл на ваш локальный почтовый ящик (не указывайте ничего вроде «@домен», достаточно просто имени пользователя). В случае успеха при использовании Bash вы увидите новое почтовое уведомление.
- Отобразите список устройств в /dev, которые в настоящее время используются вашим UID. Организуйте конвейер через **less**, чтобы посмотреть их должным образом.
- Выполните следующие команды под непривилегированным пользователем. Определите стандартный ввод, вывод и ошибку для каждой команды.

  Теперь проверьте ваши результаты выдачи команд снова, теперь перенаправляя стандартный вывод в файл /var/tmp/output и стандартную ошибку в файл /var/tmp/error.

  - cat nonexistentfile
  - file /sbin/ifconfig
  - grep root /etc/passwd /etc/nofiles > grepresults
  - /etc/init.d/sshd start > /var/tmp/output
  - /etc/init.d/crond start > /var/tmp/output 2>&1
- Как много процессов у вас сейчас работает?

- Сколько скрытых файлов в вашем домашнем каталоге?
- Используйте locate, чтобы найти документацию по ядру.
- Узнайте, какой файл содержит следующую запись:

```
root:x:0:0:root:/root:/bin/bash
```

А эту:

```
system:  root
```

- Посмотрите, что произойдет после выполнения этой команды:

> time; date >> time; cat < time

- Какие команды вы бы использовали, чтобы проверить, какие сценарии в /etc/init.d запускают данный процесс?

Глава 6. Текстовые редакторы

Аннотация

В этой главе мы обсудим насколько важно освоение редактора. В основном уделим внимание Улучшенному редактору **vi**.

После завершения этой главы вы будете уметь:

- Открытывать и закрывать файлы в текстовом режиме
- Редактировать файлы
- Искать текст
- Отменять ошибки
- Объединять файлы
- Восстановить потерянные файлы
- Найти программу или пакет для использования в офисе

Текстовые редакторы

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Зачем мне может потребоваться редактор?

Очень важно уметь пользоваться хотя бы одним редактором в текстовом режиме. Знание того, как использовать редактор в вашей системе, является первым шагом к независимости.

В этой главе нам необходимо освоить редактор, чтобы научиться редактировать файлы, оказывающие влияние на наше окружение. Как опытный пользователь, вы можете начать писать скрипты или книги, разрабатывать веб-сайты или новые программы. Освоение редактора позволит значительно повысить производительность, а заодно и ваши возможности.

Какой редактор лучше использовать?

Мы делаем упор на текстовые редакторы, которые помимо прочего могут быть использованы в системах без графической среды и в терминальных окнах. Дополнительное преимущество освоения текстового редактора заключается в использовании его на удаленной машине. Поскольку вам не нужно передавать всю графическую среду по сети, работа с текстовыми редакторами чрезвычайно повышает скорость работы сети.

Как обычно, существует несколько способов решения проблемы. Давайте посмотрим, какие редакторы обычно доступны:

GNU Emacs

Emacs является расширяемым, настраиваемым, самодокументируемым, отображающим в реальном времени редактором, известным во многих UNIX и других системах. Редактируемый текст отображается на экране и обновляется автоматически при вводе команд. Это и есть редактор реального времени, т.к. экран обновляется очень часто, обычно после каждого символа или пары введенных вами символов. Это минимизирует то количество информации, которую вы должны держать в голове при редактировании. Emacs называется продвинутым, т.к. он предоставляет возможности, выходящие за рамки простой вставки и удаления: контроль субпроцессов; автоматический отступ в программном коде; просмотр двух или более файлов за один раз; редактирование форматированного текста; и "разбирается" в значениях символов, слов, строк, предложений, абзацев и страниц, также как в выражениях и комментариях ряда языков программирования.

Самодокументируемый означает, что в любой момент вы можете ввести специальную комбинацию **Ctrl + H**, чтобы узнать, какие у вас есть опции. Вы можете также использовать это для выяснения, что делает какая-нибудь команда, или найти все команды, которые относятся к какой-нибудь теме. *Настраиваемый* означает, что вы достаточно легко можете изменить определения команд Emacs. Например, если вы используете язык программирования, в котором комментарии начинаются с "<**" и заканчиваются "**>", то можете сообщить Emacs команды манипуляции с комментарием при использовании этих строк. Другим видом настройки является перестройка набора команд. Например, если вы предпочитаете четыре основные команды движения курсора (вверх, вниз, влево и вправо) на клавишах клавиатуры, то существует способ переопределить клавиши.

Расширяемый означает, что вы можете выйти за рамки простой настройки и писать совершенно новые команды, программы на языке Lisp; интерпретатор Lisp входит в состав Emacs. Emacs является онлайн-расширяемой системой, это означает, что он разделен на множество функций, которые вызывают друг друга, и любая из которых может быть переопределена в середине сеанса редактирования. Почти любая часть Emacs может быть заменена без создания отдельной копии всего Emacs. Большинство команд редактирования Emacs уже написаны на Lisp; за некоторыми исключениями могли бы быть написаны в Lisp, но написанный на C для повышения эффективности. Хотя только программист может написать расширение, кто-нибудь может использовать его позже.

При запуске под X Window System (запускается как **xemacs**) Emacs предоставляет собственные меню и удобные привязки к кнопкам мыши. Но Emacs может обеспечить многие из преимуществ оконной системы исключительно в текстовом терминале. Например, вы можете просматривать или редактировать несколько файлов одновременно, перемещать текст между файлами и редактировать файлы во время работы команд оболочки.

Vi(m)

Vim означает "Vi IMproved" ("Vi Улучшенный"). Раньше считалось "Vi IMitation" ("Vi Подражание"), но существует так много улучшений, что изменение названия стало необходимостью. Vim является текстовым редактором, который включает в себя почти все команды от UNIX программы **vi** и множество новых.

Команды в редакторе **vi** вводятся только с помощью клавиатуры; в этом есть преимущество, т.к. ваши пальцы могут оставаться на клавиатуре, а глаза на экране, вместо того, чтобы перемещать руку несколько раз на мышь. Для тех, кому это нравится, могут быть активированы поддержка мыши и GUI версия с полосами прокрутки и меню.

В этой книге при редактировании файлов мы будем ссылаться на **vi** или **vim**, а вы, конечно, можете использовать редактор по вашему выбору. Тем не менее, мы рекомендуем, по крайней мере, изучить основы **vi**, потому что это стандартный текстовый редактор почти на всех системах UNIX, в то время как **emacs** может быть дополнительным пакетом. Могут быть небольшие различия между различными компьютерами и терминалами, но главным остается то, что если вы можете работать с **vi**, то сможете выжить в любой системе UNIX.

Помимо команды **vim**, пакеты vIm могут также предоставлять **gvim**, Gnome-версию **vim**. Начинающие пользователи могут найти, что он проще в использовании, так как в меню предоставляется помощь, на случай, если вы забыли или не знаете, как выполнять определенную задачу редактирования, используя стандартные команд **vim**.

Использование редактора Vim

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Два режима

Редактор **vi** очень мощный инструмент и содержит очень обширное встроенное руководство, которое можно открыть используя команду **:help**, когда программа запущена (вместо использования **man** или **info**, которые не содержат так много информации). Здесь мы обсудим только самые основы, чтобы вы смогли начать.

Тем, что вводит начинающих пользователей **vi** в заблуждение, является то, что он может работать в двух режимах: режиме команд и режиме вставки. Редактор всегда запускается в командном режиме. Команды перемещают вас по тексту, осуществляют поиск, замену, маркировку блоков и выполняют другие задачи редактирования, некоторые команды переключают редактор в режим вставки.

Все это означает, что у каждой клавиши есть не одно, а, вероятно, два значения: она может либо представлять команду для редактора в командном режиме, или символ, который вам требуется в тексте в режиме вставки.

 **Произношение.**
Это произносится как "vee-eye".

Основные команды

Перемещение по тексту

Обычно перемещение по тексту возможно с помощью клавиш навигации. Если нет, попробуйте:

- **h** для перемещения курсора влево

- **i** для его перемещения в право
- **k** для движения вверх
- **j** для движения вниз

Shift-G быстро переместить в конец документа.

Основные операции

Вот наиболее используемые команды **vi**:

- **n dd** удалит n линий, начиная с текущей позиции курсора.
- **n dw** удалит n слов с правой стороны от курсора.
- **x** удалит символ, на котором стоит курсор
- **:n** переход к линии n файла.
- **:w** сохранит (запишет) файл.
- **:q** осуществит выход из редактора.
- **:q!** принудительный выход, когда требуется выйти из файла, содержащего несохраненные изменения.
- **:wq** сохранение и выход.
- **:w** newfile сохранение текста в newfile.
- **:wq!** переопределяет разрешение «только для чтения» (если у вас есть разрешение на переопределение разрешений, например, когда вы используете учетную запись *root*).
- **/astring** будет искать строку в файле и установит курсор на первое соответствие ниже его позиции.
- **/** будет выполнять тот же поиск снова, перемещения курсор к следующему соответствию.
- **:1, $s/word/anotherword/g** заменит word на anotherword во всём файле.
- **yy** скопирует блок текста.
- **n p** вставит его n раз.
- **:recover** восстановит файл после внезапного сбоя.

Команды, которые переключаются редактор в режим вставки

- a добавление: перемещает курсор на одну позицию вправо перед включением режима вставки.
- i вставка.
- o вставка пустой строки под текущей позицией курсора и перемещение курсора в эту строку.

Нажатие клавиши **Esc** переключает обратно в командный режим. Если вы не уверены, в каком режиме находитесь по причине использования очень старой версии, где **vi** не отображает сообщение "INSERT", нажмите **Esc**, и вы будете уверены, что вернулись в командный режим. Вполне возможно, если вы уже находитесь в командном режиме, система даст небольшое предупреждение при нажатии **Esc** с помощью звукового сигнала или визуально (вспышка на экране). Это нормальное поведение.

Простой способ

Вместо чтения текста, что довольно скучно, вы можете использовать vimtutor для изучения ваших первых команд Vim. Это тридцати минутный учебник, который рассказывает о самой основной функциональности

Vim в восьми простых упражнениях. Хотя вы и не узнаете все о Vim за полчаса, это руководство предоставит описание достаточного количества команд, после чего вы сможете с легкостью использовать Vim в качестве универсального редактора.

В UNIX и MS Windows, если Vim был правильно установлен, вы можете запустить эту программу из shell или командой строки, введя команду **vimtutor**. Это создаст копию файла учебника, так что вы можете редактировать его без риска повреждения оригинала. Есть несколько версий перевода руководства. Чтобы узнать, доступен ли ваш язык, используйте двухбуквенный код языка. Для французского это будет **vimtutor** fr (если он установлен в системе).

Linux в офисе

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

История

На протяжении последнего десятилетия в сфере офиса чаще доминирует MS Office, и, будем откровенны, форматы Microsoft Word, Excel и PowerPoint являются отраслевыми стандардартами, с которыми, рано или поздно, вы будете иметь дело.

Это монопольное положение Microsoft негативно сказалось на появлении новых пользователей Linux; поэтому группа немецких разработчиков запустила проект StarOffice, который существует до сих пор, он был направлен на создание клона MS Office. Их компания StarDivision была приобретена компанией Sun Microsystems к концу 1990-х годов, незадолго до релиза 5.2. Sun продолжила развитие, но ограничила доступ к исходным кодам. Тем не менее, развитие оригинального пакета исходников продолжается в сообществе Open Source, которые переименовали проект в OpenOffice. В настоящее время OpenOffice доступен для различных платформ, включая MS Windows, Linux, MacOS и Solaris. Скриншот в разделе 1.3.2 "Пятнадцать лет опыта к вашим услугам".

Почти одновременно было запущено несколько других весьма известных проектов. Также очень распространенной альтернативой использования MS Office является KOffice, офисный пакет, которые раньше был популярен среди пользователей SuSE. Этот клон включает совместимые с MS Word и Excel программы и многое другое.

С отдельными программами MS боролись небольшие проекты, например, такие пакеты как Abiword и MS Wordview для совместимости с документами MS Word, Gnumeric для просмотра и создания Excel-совместимых таблиц.

Пакеты и программы

Сегодняшние дистрибутивы обычно поставляются со всеми необходимыми инструментами. Поскольку они снабжены превосходными руководствами и поисковыми указателями в меню **Справка**, мы не будем обсуждать их более подробно. Если нужны ссылки, посмотрите вашу системную документацию или веб-сайты таких проектов как

- http://www.openoffice.org
- http://www.koffice.org
- Freshmeat и SourceForge для различных других проектов.

Замечания

Главное в использовании офисных документов

Постарайтесь использовать офисные приложения исключительно для целей, к которым они предназначены: офис.

Например, большинство пользователей Linux сводит с ума, если вы отправите им почту, в теле которого говорится что-то вроде: "Здравствуйте, я хочу сказать тебе что-то, смотри вложение", а затем приложение к письму оказывается MS Word совместимым документом с примерно таким содержанием: "Привет, мой друг, как идет ваша новая работа, будет ли у тебя время, чтобы пообедать со мной завтра?" Также плохой идеей является вложение вашей подписи в такой файл. Если вы хотите подписывать сообщения или файлы, использовать GPG, PGP-совместимых GNU Privacy Guard или SSL (Secure Socket Layer) сертификаты.

Подобное раздражает пользователей, т.к. они не в состоянии прочитать эти документы или обеспокоены тем, что эти форматы обычно генерируют много больших файлов, а скорее из-за причастности к MS Windows, и, возможно, из-за дополнительной работы, связанной с запуском каких-то дополнительных программ.

Системные и пользовательские файлы конфигурации

В следующей главе мы приступить к настройке нашего окружения, что может включать редактирование всех видов файлов, определяющих поведение программы.

Не используйте для изменения этих файлов любые офисные программы!

Спецификация формата файла по умолчанию заставила бы программу добавить несколько строк кода, определяющих формат файла и используемые шрифты. Эти строки не будут правильно интерпретированы программой, для которой они предназначены, в результате произойдут ошибки в работе или крах программы, читающей такой файл. В некоторых случаях вы можете сохранить файл в виде простого текста, но столкнетесь с проблемами при формировании такой привычки.

Но я хочу графический текстовый редактор!

Если вы и правда настаиваете, попробуйте **gedit**, **kedit**, **kwrite** или **xedit**; эти программы создают только текстовые файлы, в чем мы и будем нуждаться. Если вы собираетесь делать что-нибудь серьезное, то все-таки лучше использовать **vim** или **emacs**.

Приемлемой альтернативой является **gvim**, версия **vim** для Gnome. Для этого необходимо использовать команду **vi**, но если вы затрудняетесь, то можете посмотреть их в меню.

Резюме и упражнения

Резюме

В этой главе мы научились пользоваться редактором. Что использовать, зависит от ваших личных предпочтений, но необходимо знать, по крайней мере, как пользоваться одним редактором.

Редактор **vi** есть в любой системе UNIX.

Большинство дистрибутивов Linux включают офисный пакет и графический текстовый редактор.

Упражнения

В этой главе есть только одно упражнение: запустить учебник Vim, введя **vimtutor** в терминале, и начать работу.

Как альтернативу вы можете запустить **emacs** и нажать **Ctrl + H**, а затем **T** для вызова отдельного учебника Emacs.

Единственный путь - практика!

Глава 7. Дом, сладкий /home

Аннотация

Эта глава о настройке вашего окружения. Теперь, когда мы знаем, как использовать редактор, можем изменить все виды файлов, что заставит нас чувствовать себя дома лучше. После изучения этой главы вы будете больше знать о следующем:

- Обустройство среды
- Общие установочные файлы shell
- Конфигурация shell
- Настройка приглашения
- Настройка графической среды
- Звуковые и видео приложения
- Дисплей и оконные менеджеры
- Как работает клиент-серверная система X
- Язык и настойки шрифта
- Установка нового программного обеспечения
- Обновление существующих пакетов

Важные вещи в домашнем хозяйстве

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Введение

Как мы уже упоминали, навести беспорядок в системе достаточно легко. Поэтому не следует пренебрегать тем, что позволяет сохранять рабочее место опрятным. Если вы научитесь этому с самого начала, то приобретете хорошую привычку, которая поможет вам сэкономить время при программировании в системе Linux (UNIX) или при решении задач управления системой. Здесь приведены несколько способов сделать себе жизнь легче:

- Создайте каталог bin для ваших файлов программ и скриптов.

- Организуйте неисполняемые файлы в подходящие каталоги и создайте так много каталогов, как вам требуется. Например, могут быть отдельные каталоги для изображений, документов, проектов, загруженных файлов, электронных таблиц, личных файлов, и так далее.
- Делайте каталоги личными с помощью команды **chmod** 700 dirname.
- Давайте вашим файлам осмысленные имена, такие как Жалоба премьер-министру 050302, а не письмо1.

Создайте пространство

На некоторых системах установленная квота может заставлять вас время от времени выполнять очистку, также физические ограничения жесткого диска могут заставить вас оставлять больше места без запуска каких-либо программ мониторинга. В этом разделе рассматриваются несколько способов освобождения места на диске, помимо использования команды **rm**.

Выполните команду **quota** -v, чтобы увидеть, сколько места осталось.

Опустошение файлов

Иногда содержание файла может вас не интересовать, но вам нужно имя файла в качестве маркера (например, вам просто необходим файл-метка, напоминание о том, что файл был там или должен быть через некоторое время в будущем). Перенаправление вывода "нулевой" команды в оболочках Bourne и Bash как раз это сделает:

```
andy:~> cat wishlist > placeholder

andy:~> ls -la placeholder
-rw-rw-r--    1 andy   andy       200 Jun 12 13:34 placeholder

andy:~>  > placeholder

andy:~> ls -la placeholder

-rw-rw-r--    1 andy   andy         0 Jun 12 13:35 placeholder
```

Процесс уменьшения существующего файла в файл с тем же именем, но размер которого равен нулю байтов, называется усечением.

Также создать новый пустой файл можно с помощью команды **touch**. Использование **touch** на существующий файл обновит только его метку. Для выяснения деталей посмотрите info-страницу для **touch**.

Для получения "почти" пустого файла используйте команду **tail**. Допустим, список пожеланий пользователя *andy* становится довольно длинным, т.к. он всегда добавляет материал в конец, но никогда не удаляет то, что он уже получил. Теперь он хочет только сохранить последние пять пунктов:

```
andy:~> tail -5 wishlist > newlist

andy:~> cat newlist > wishlist

andy:~> rm newlist
```

Подробнее о log-файлах

Некоторые программы Linux настойчиво требуют записи различных выходных данных в файл журнала (log-файл). Обычно существуют установки только для журналов ошибок, или журналировать минимальное количество информации, например, установка уровня отладки программы. Но даже в этом случае, вы можете не волноваться о файле журнала. Здесь несколько способов, чтобы избавиться от них или, по крайней мере, несколько ограничить их размер:

- Если вы уверены, что log-файл вам больше не потребуется, попробуйте удалить его, когда программа не запущена. Некоторые программы могут даже обнаружить после перезагрузки, что файла журнала нет и не будут создавать его снова.
- Если после удаления log-файла программа создает его снова, прочитайте документацию к этой программе в поисках варианта команды, которая исключит создание лог-файла.
- Попробуйте сделать log-файлы меньше за счет того, что будет вводится только информация, имеющая отношение к вам, или только имеющая значение.
- Попробуйте заменить файл журнала символической ссылкой /dev/null; если вам повезет, программа не будет жаловаться. Не делайте такого с лог-файлами программ, которые запускаются при загрузке системы, или программами, которые работают от хрона (см. Главу 4, Процессы). Эти программы могут заменить символическую ссылку небольшим файлом, который начнет расти снова.

Почта

Регулярно очищайте ваш почтовый ящик, создавайте подкаталоги и автоматически перенаправляйте используемый **procmail** (см. info-страницы) или фильтры от предпочитаемого вами приложения для чтения почты. Если у вас есть папка-корзина, то регулярно очищайте ее.

Для перенаправления почты используйте файл .forward в вашем домашнем каталоге. Почтовый сервис Linux ищет именно этот файл для локальной доставки. Содержание файла определяет то, что почтовая система должна делать с вашей почтой. Он может содержать одну строку, содержащую полный адрес электронной почты. В таком случае система будет отправлять все письма на этот адрес. Например, при аренде места для сайта, вы можете направлять почту, предназначенную для веб-мастера на свой аккаунт, чтобы не использовать место на диске. Файл .forward вебмастера может выглядеть следующим образом:

```
webmaster@www ~/> cat .forward
mike@pandora.be
```

Использование пересылки почты также полезно, т.к. освобождает вас от необходимости проверять несколько почтовых ящиков. Вы можете сделать так, чтобы каждый адрес указывал на основную почту, которую вам удобней использовать.

Вы можете обратиться к системному администратору, чтобы он определил направления для вас в локальный файл почтовых псевдонимов, например, когда аккаунт закрывается, но электронная почта пока остается активной.

Экономия пространства ссылкой

Когда нескольким пользователям необходимо иметь доступ к одному и тому же файлу или программе, когда исходное имя файла слишком длинное или его слишком трудно запомнить, используйте символическую ссылку вместо создания отдельной копии для каждого пользователя или назначения.

Множественные символические ссылки могут иметь различные названия, например, ссылку можно назвать monfichier в одной пользовательской директории и mylink в другой. Несколько ссылок (с разными именами) на один и тот же файл может быть в одном и том же каталоге. Часто это можно наблюдать в каталоге /lib с помощью команды

ls -l /lib

Вы увидите, что в этом каталоге очень много ссылок, указывающих на одинаковые файлы. Они создаются для программ, ищущих какое-нибудь определенное название, в результате ссылки указывают на правильные/текущие название библиотек, которые требуются под другими именами.

Ограничение размера файлов

Оболочка содержит встроенную команду для ограничения размера файлов, **ulimit**, которая также может быть использована для отображения ограничений на системные ресурсы:

```
cindy:~> ulimit -a
core file size (blocks)     0
data seg size (kbytes)      unlimited
file size (blocks)          unlimited
max locked memory (kbytes)  unlimited
max memory size (kbytes)    unlimited
open files                  1024
pipe size (512 bytes)       8
stack size (kbytes)         8192
cpu time (seconds)          unlimited
max user processes          512
virtual memory (kbytes)     unlimited
```

Cindy не разработчик и не заботится об основных дампах, которые содержат информацию об отладке программы. Если вам нужны основные дампы, вы можете установить их размер с помощью команды **ulimit**. В info-страницах bash дается более подробное объяснение.

 **Core файл?**
Файл core или основной дамп иногда создается, когда не все так хорошо с программой во время ее выполнения. Файл core содержит копию памяти системы во время того, когда произошла ошибка.

Сжатые файлы

Сжатые файлы весьма полезны, поскольку они занимают меньше места на жестком диске. Еще одним преимуществом является то, что они занимают меньше пропускной способности при отправке их по сети. Многие файлы, такие как man-страницы, в системе хранятся в сжатом виде. Тем не менее, чтобы получить информацию их надо распаковать; как и сжатие это занимает довольно много времени. Вряд ли вам захочется распаковывать страницу, например, описывающую опции команды, а затем сжимать ее снова. Большинство людей, кроме того, забывают удалить лишнее после того, как они нашли необходимую им информацию.

Поэтому есть инструменты, которые создают сжатые файлы, но распаковывают их только в память. Фактически сжатый файл остается на диске как есть. Большинство систем поддержки **zgrep**, **zcat**, **bzless** и других членов z-семейства предотвращают ненужные действия по распаковке/сжатию. Посмотрите в ваш

каталог исполняемых файлов системы, а также info-страницы.

См. Главу 9, Основные методы резервного копирования для дополнительной информации по существующему сжатию файлов и примеры создания архивов.

Ваше текстовое окружение

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Переменные окружения

Главное

Мы уже упоминали несколько переменных окружения, таких как PATH и HOME. До сих пор мы видели только примеры, в которых они используются в оболочке для определенной цели. Но в Linux существует множество других утилит, для нормальной работы которых необходима информация о вас.

Какая еще информация нужна программам, кроме путей и домашних каталогов?

Многим программам необходима информация о том, какой терминал вы используете; эта информация хранится в переменной TERM. В текстовом режиме, это будет эмуляция терминала *linux*, в графическом режиме, скорее всего, вы используете *xterm*. Многим программам требуется информация о предпочитаемом вами редакторе на случай, если они должны будут запускать редактор как дочерний процесс. Оболочка, которую вы используете, хранится в переменной SHELL, тип операционной системы - в OS и так далее. Список всех текущих переменных, определенных для вашей сессии, можно посмотреть, введя команду **printenv**.

Переменные окружения управляются оболочкой. В отличие от регулярных переменных оболочки, переменные окружения наследуются любой программой, которую вы запускаете, в том числе другой оболочкой. Новым процессам присваиваются копии этих переменных; процессы могут их читать, изменять и, в свою очередь, передавать собственным дочерним процессам.

В именах переменных нет ничего особенного, за исключением того, что по соглашению используют прописные буквы. Вы можете придумать любое имя, какое захотите, хотя существуют стандартные переменные, которые являются достаточно важными, поэтому они одинаковы во всех системах Linux, например PATH и HOME.

Экспорт переменных

Содержимое отдельных переменной обычно отображается с помощью команды **echo**, как в этих примерах:

```
debby:~> echo $PATH
/usr/bin:/usr/sbin:/bin:/sbin:/usr/X11R6/bin:/usr/local/bin

debby:~> echo $MANPATH
/usr/man:/usr/share/man/:/usr/local/man:/usr/X11R6/man
```

Если вы хотите изменить содержимое переменной таким, чтобы она была полезна для других программ, вам следует экспортировать новое значение из вашего окружения в окружение, которое запускают эти программы. Типичным примером является экспорт переменной PATH. Вы можете объявить ее добавлением очередного пути, например, с тем чтобы иметь возможность играть в симулятор полетов, который находится

в /opt/FlightGear/bin:

```
debby:~> PATH=$PATH:/opt/FlightGear/bin
```

Такая запись указывает оболочке искать программ не только в текущем пути $PATH, но и в дополнительном каталоге /opt/FlightGear/bin.

Но если новое значение переменной PATH не известно окружению, то работать ничего не будет:

```
debby:~> runfgfs
bash: runfgfs: command not found
```

В Bash мы обычно делаем это с помощью одного элегантного шага:

export VARIABLE=value

Тот же самый способ используется для переменной MANPATH, которая сообщает команде **man**, где искать сжатые man-страницы. Когда новые программы устанавливаются в новые или необычные каталоги, документация для них, вероятно, также будет в нестандартных каталогах. Если вы хотите прочитать справочные страницы этих новых программ, расширьте переменную MANPATH:

```
debby:~> export MANPATH=$MANPATH:/opt/FlightGear/man

debby:~> echo $MANPATH
/usr/man:/usr/share/man:/usr/local/man:/usr/X11R6/man:/opt/FlightGear/man
```

Вы можете избежать повторного ввода этой команды в каждом окне, которое вы открываете, добавив эту строку в один из ваших установочных файлов оболочки, см. Раздел "Установочные файлы shell".

Зарезервированные переменные

Следующая таблица дает обзор наиболее распространенных предопределенных переменных:

Таблица. Стандартные переменные окружения

Имя переменной	Хранимая информация
DISPLAY	используется системой X Window для определения сервера, который отображает на дисплей
DOMAIN	доменное имя
EDITOR	хранит предпочитаемый вами редактор
HISTSIZE	размер файла истории shell (в количестве строк)
HOME	адресный путь вашего домашнего каталога
HOSTNAME	имя локального хоста
INPUTRC	определяет местоположение файлов устройств ввода, например, клавиатуры
LANG	предпочитаемый язык

LOGNAME	логин
MAIL	местонахождение вашей папки для входящей почты
MANPATH	пути поиска для man-страниц
OS	строка, описывающая операционную систему
OSTYPE	дополнительные сведения о версии и т.п.
PAGER	используется программами, такими как **man**, которым требуется знать, что делать в случае, если вывод больше, чем окно терминала
PATH	поиск путей для команд
PS1	первоначальное приглашение
PS2	вторичное приглашение
PWD	текущий рабочий каталог
SHELL	текущая оболочка
TERM	тип терминала
UID	ID пользователя
USER(NAME)	имя пользователя
VISUAL	ваш предпочитаемый полноэкранный редактор
XENVIRONMENT	местонахождение ваших личных настроек поведения X
XFILESEARCHPATH	пути поиска графических библиотек

Многие переменные не только предопределены, но также их значения устанавливаются заранее, при этом используются конфигурационные файлы. Мы обсудим это в следующем разделе.

Установочные файлы shell

При вводе команды **ls -al** для получения расширенного списка всех файлов, в том числе начинающихся с точки, содержащихся в вашем домашнем каталоге, вы увидите один или несколько файлов, начинающихся с . и заканчивающихся на *rc*. Для Bash это будет .bashrc. Это копия общесистемного конфигурационного файла /etc/bashrc.

При входе логина в shell, **login** выполнит идентификацию, установит окружение и запустит вашу оболочку. В случае **bash** следующим шагом будет чтение общего профиля из /etc, если такой файл существует. Затем **bash** ищет ~/.bash_profile, ~/.bash_login и ~/.profile в указанном порядке, читает и выполняет команды из первого, который существует и который можно прочитать. Если таких файлов нет, то используется /etc/bashrc.

Когда происходит выход их оболочки, **bash** считывает и выполняет команды из файла ~/.bash_logout, если он существует.

Эта процедура подробно описана в man-страницах **login** и **bash**.

Стандартный набор установочных файлов

Пример /etc/profile

Давайте посмотрим на некоторые из этих конфигурационных файлов. Первый для чтения — это /etc/profile, в котором устанавливаются такие важные переменные, как PATH, USER и HOSTNAME:

```
debby:~> cat /etc/profile
# /etc/profile

# System wide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc


# Path manipulation
if [ `id -u` = 0 ] && ! echo $PATH | /bin/grep -q "/sbin" ; then
    PATH=/sbin:$PATH
fi

if [ `id -u` = 0 ] && ! echo $PATH | /bin/grep -q "/usr/sbin" ; then
    PATH=/usr/sbin:$PATH
fi

if [ `id -u` = 0 ] && ! echo $PATH | /bin/grep -q "/usr/local/sbin"
    then
    PATH=/usr/local/sbin:$PATH
fi

if ! echo $PATH | /bin/grep -q "/usr/X11R6/bin" ; then
    PATH="$PATH:/usr/X11R6/bin"
fi
```

Эти строки проверяют устанавливаемые пути: если root открывает оболочку (ID пользователя 0), проверяется пути /sbin, /usr/sbin и /usr/local/sbin. Если нет, то они добавляются. Для всех проверяется путь /usr/X11R6/bin.

```
# No core files by default
ulimit -S -c 0 > /dev/null 2>&1
```

Все, что не нужно отправляется в /dev/null, если пользователь не меняет эту настройку.

```
USER=`id -un`
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"

HOSTNAME=`/bin/hostname`
HISTSIZE=1000
```

Здесь общим переменным назначаются их стандартные значения.

```
if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
    INPUTRC=/etc/inputrc
fi
```

Если переменная INPUTRC не установлена, и нет .inputrc в домашнем каталоге пользователя, то будет загружен файл, отвечающий за ввод по умолчанию.

```
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC
```

Все переменные экспортируются, поэтому они доступны для других программ, запрашивающих информацию о вашем окружении.

Каталог profile.d

```
for i in /etc/profile.d/*.sh ; do
    if [ -r $i ]; then
     . $i
    fi
done
unset i
```

Все читаемые скрипты shell из каталога /etc/profile.d читаются и выполняются. Это то, что делает возможным *color-ls*, похожесть **vi** на **vim**, локальные настройки и т.д. Временная переменная *i* не устанавливается для предотвращения неадекватного поведения оболочки позже.

Пример .bash_profile

Затем bash ищет .bash_profile в домашнем каталоге пользователя:

```
debby:~> cat .bash_profile
#####################################################
#                                                   #
#   .bash_profile file                              #
#                                                   #
#   Executed from the bash shell when you log in.   #
#                                                   #
#####################################################

source ~/.bashrc
source ~/.bash_login
```

В файле содержаться указания оболочке сначала прочитать ~/.bashrc, потом ~/.bash_login. При работе с окружением оболочки вы будете сталкиваться с встроенной в shell командой **source** достаточно часто: она используется для применения изменений конфигурации к текущему окружению.

Пример .bash_login

Файл ~/.bash_login определяет по умолчанию защиту файлов, устанавливая значение **umask**, см. Раздел "Вступление в другую группу". Файл ~/.bashrc используется для определения группы пользовательских псевдонимов, функции и личные переменных окружения. Сначала читается /etc/bashrc, который описывает

стандартное приглашение (PS1) и значение по умолчанию umask. После этого вы можете добавить собственные настройки. Если нет ~/.bashrc, то по умолчанию читается /etc/bashrc.

Пример /etc/bashrc

Ваш файл /etc/bashrc может выглядеть следующим образом:

```
debby:~> cat /etc/bashrc
# /etc/bashrc

# System wide functions and aliases
# Environment stuff goes in /etc/profile

# by default, we want this to get set.
# Even for non-interactive, non-login shells.
if [ `id -gn` = `id -un` -a `id -u` -gt 99 ]; then
 umask 002
else
 umask 022
fi
```

Эти строки устанавливают значение **umask**. Затем, в зависимости от типа shell, устанавливается приглашение:

```
# are we an interactive shell?
if [ "$PS1" ]; then
  if [ -x /usr/bin/tput ]; then
    if [ "x`tput kbs`" != "x" ]; then
# We can't do this with "dumb" terminal
      stty erase `tput kbs`
    elif [ -x /usr/bin/wc ]; then
      if [ "`tput kbs|wc -c `" -gt 0 ]; then
# We can't do this with "dumb" terminal
        stty erase `tput kbs`
      fi
    fi
  fi
  case $TERM in
 xterm*)
 if [ -e /etc/sysconfig/bash-prompt-xterm ]; then
  PROMPT_COMMAND=/etc/sysconfig/bash-prompt-xterm
 else
   PROMPT_COMMAND='echo -ne "\033]0;${USER}@${HOSTNAME%%.*}:\
${PWD/$HOME/~}\007"'
 fi
    ;;
 *)
   [ -e /etc/sysconfig/bash-prompt-default ] && PROMPT_COMMAND=\
/etc/sysconfig/bash-prompt-default
     ;;
    esac
```

```
    [ "$PS1" = "\\s-\\v\\$ " ] && PS1="[\u@\h \W]\\$ "

    if [ "x$SHLVL" != "x1" ]; then # We're not a login shell
        for i in /etc/profile.d/*.sh; do
     if [ -x $i ]; then
         . $i
     fi
 done
    fi
fi
```

Пример .bash_logout

Во время выхода из системы, выполняются команды из ~/.bash_logout, которые могут, например, очищать терминал, так что у вас появляется чистое окно при входе из удаленного сеанса или при выходе из системной консоли:

```
debby:~> cat .bash_logout
# ~/.bash_logout

clear
```

Давайте ближе рассмотрим, как эти скрипты работают в следующем разделе. Храните **info bash** под рукой.

Приглашение Bash

Введение

Приглашение Bash может делать гораздо больше, чем отображение такой простой информации, как имя пользователя, имя машины и некоторое указание на текущий рабочий каталог. Мы можем добавить другую информацию, такую как дата и время, количество подключенных пользователей и т.д.

Однако, прежде чем мы начнем, сохраним наше текущее приглашение в другую переменную окружения:

```
[jerry@nowhere jerry]$ MYPROMPT=$PS1

[jerry@nowhere jerry]$ echo $MYPROMPT
[\u@\h \W]\$

[jerry@nowhere jerry]$
```

Когда мы теперь будем изменять строку приглашения, например, с помощью команды PS1="->", то всегда сможем получить нашу оригинальную строку обратно с помощью команды PS1=$MYPROMPT. Конечно, вы получите ее обратно, и когда подключитесь снова; так будет до тех пор, пока вы возитесь с приглашением в командной строке и не вставляете его в файл конфигурации оболочки.

Некоторые примеры

Для того, чтобы понять эти приглашения и используемые управляющие последовательности, обратитесь к info- и man-страницам Bash.

- **export PS1="[\t \j] "**

Отображает время и количество выполняемых заданий.

- **export PS1="[\d][\u@\h \w] : "**

Отображает дату, имя пользователя, название хоста и текущую рабочую директорию. Заметьте, что \W отображает только конечные имена текущего каталога.

- **export PS1="{\!} "**

Отображает номер каждой команды в истории.

- **export PS1="\[\033[1;35m\]\u@\h\[\033[0m\] "**

Делает user@host розовым.

- **export PS1="\[\033[1;35m\]\u\[\033[0m\] \[\033[1;34m\]\w\[\033[0m\] "**

Раскрашивает имя пользователя в розовый цвет, а рабочий каталог в голубой.

- **export PS1="\[\033[1;44m\]$USER is in \w\[\033[0m\] "**

Приглашение для людей, которым трудно видеть разницу между приглашением и тем, что они вводят.

- **export PS1="\[\033[4;34m\]\u@\h \w \[\033[0m\]"**

Подчеркнутое приглашение.

- **export PS1="\[\033[7;34m\]\u@\h \w \[\033[0m\] "**

Белые символы на синем фоне.

- **export PS1="\[\033[3;35m\]\u@\h \w \[\033[0m\]\a"**

Розовая строка иным шрифтом, которая предупреждает вас, когда ваши команды закончили работу.

- **export PS1=...**

Переменные экспортируются, так что впоследствии выполняющиеся команды также будут знать об окружении. Для конфигурации строки приглашения лучше изменить конфигурационный файл оболочки, ~/.bashrc.

Если вам захочется, то приглашения могут выполнять shell-скрипты и вести себя по-разному в зависимости от условий. У вас может даже строка приглашения играть мелодию при выполнении команды, хотя это вам быстро надоест. Более подробную информацию можно найти в Bash-Prompt HOWTO.

Сценарии оболочки

Что такое сценарии?

Скрипт оболочки, как мы видели в примерах конфигурации shell, представляет собой текстовый файл, содержащий команды оболочки. Когда такой файл используется в качестве первого аргумента без опции при вызове Bash, и без опций -c или -s, Bash считывает и выполняет команды из файла, затем выходит. Этот режим работы создает неинтерактивную оболочку. Когда Bash запускает shell-скрипт, он устанавливает специальный параметр 0 к имени файла, а не имя к оболочке, и позиционные параметры (все, что следует за именем скрипта) устанавливаются в остальные аргументы, если таковые задаются. Если

дополнительные аргументы не задаются, позиционные параметры не включаются.

Скрипт оболочки можно сделать исполняемым с помощью команды **chmod** за счет включения бита исполнения. Когда Bash находит такой файл при поиске в PATH по отношению к команде, он порождает подоболочку для его выполнения. Другими словами, выполнение

filename *ARGUMENTS*
эквивалентно выполнению
bash *filename ARGUMENTS*

если "filename" представляет собой исполняемый скрипт. Эта подоболочка инициализирует себя, так что эффект такой, как если бы новая оболочка интерпретировала сценарий, с тем исключением, что места команд, запомненные родителем (см. **hash** в info-страницах), сохраняются ребенком.

Большинство версий UNIX наделяют эту часть команд операционной системы исполняемым механизмом. Если первая строка сценария начинается с двух символов "#!", оставшаяся часть строки указывает интерпретатор для программы. Таким образом, вы можете указать **bash, awk, perl** или другой интерпретатор или оболочку и писать остальную часть файла сценария на этом языке.

Аргументы для интерпретатора состоят из одного дополнительного аргумента, который следует за именем интерпретатора в первой строке файла сценария, после следует имя файла сценария, а затем остальные аргументы. Bash будет выполнять это действие на операционных системах, которые не обрабатывают это самостоятельно.

Сценарии Bash часто начинаются с

#! /bin/bash

(предполагая, что Bash была установлена в /bin), так как это гарантирует использование Bash для интерпретации сценариев, даже если скрипт выполняется под другой оболочкой.

Некоторые простые примеры

Очень простой скрипт, состоящий только из одной команды, которая приветствует пользователя, выполняется так:

```
[jerry@nowhere ~] cat hello.sh
#!/bin/bash
echo "Hello $USER"
```

Сценарий фактически состоит только из одной команды **echo**, которая использует значение (*$*) переменной окружения USER, чтобы напечатать строку, настроенную на пользователя, который вызывает скрипт.

Другой однострочный пример, используемый для отображения подключенных пользователей:

```
#!/bin/bash
who | cut -d " " -f 1 | sort -u
```

Вот скрипт, состоящий из несколько строк, которые я использую, чтобы создать резервные копии всех файлов в каталоге. Сначала скрипт создает список всех файлов в текущем каталоге и помещает его в переменную LIST. Затем он устанавливает имя копии для каждого файла, и затем копирует файл. Для каждого файла выводится сообщение:

```
tille:~> cat bin/makebackupfiles.sh
```

```
#!/bin/bash
# make copies of all files in a directory
LIST=`ls`
for i in $LIST; do
 ORIG=$i
 DEST=$i.old
 cp $ORIG $DEST
 echo "copied $i"
done
```

Просто ввод строки подобной **mv * *.old** не будет работать, как вы заметите при попытке выполнить это на ряд тестовых файлов. Команда **echo** была добавлена для отображения какой-либо деятельности. **echo** обычно полезна, когда сценарий не работает: постепенно просматривая код, вы найдете ошибку в кратчайшие сроки.

Каталог /etc/rc.d/init.d содержит множество примеров. Давайте посмотрим на этот сценарий, который управляет фиктивным сервером IcanSeeYou:

```
#!/bin/sh
# description: ICanSeeYou allows you to see networked people

# process name: ICanSeeYou
# pidfile: /var/run/ICanSeeYou/ICanSeeYou.pid
# config: /etc/ICanSeeYou.cfg

# Source function library.
. /etc/rc.d/init.d/functions

# See how (with which arguments) we were called.
case "$1" in
 start)
  echo -n "Starting ICanSeeYou: "
  daemon ICanSeeYou
  echo
  touch /var/lock/subsys/ICanSeeYou
  ;;
 stop)
  echo -n "Shutting down ICanSeeYou: "
  killproc ICanSeeYou
  echo
  rm -f /var/lock/subsys/ICanSeeYou
  rm -f /var/run/ICanSeeYou/ICanSeeYou.pid
  ;;
 status)
  status ICanSeeYou
  ;;
 restart)
  $0 stop
  $0 start
  ;;
 *)
```

```
  echo "Usage: $0 {start|stop|restart|status}"
  exit 1
esac

exit 0
```

Во-первых, размещается . (точка), которая управляет множеством функций оболочки и используется практически во всех скриптах в /etc/rc.d/init.d. Затем размещается команда **case**, которая определяет 4 различных способа выполнения сценария. Например, это может быть **ICanSeeYou** start. Решением в этом случае будет чтение (первого) аргумента сценария с выражением *$1*.

Когда дается не совместимый вход, то срабатывает код, отмеченный звездочкой, в котором скрипт выдает сообщение об ошибке. Список **case** заканчивается инструкцией **esac**. В случае *start* программа сервера запускается как демон, и назначаются ID процесса и шлюз. В случае *stop* процесс сервера сходит на нет и останавливается, шлюз и PID удаляются. Опции, такие как опции daemon и функции как killproc определены в файле /etc/rc.d/init.d/functions. Эта установка специфична для дистрибутива, используемого в этом примере. Запускающие скрипты в вашей системе могут использовать другие функции, определенные в других файлах, или вообще никакие.

В случае успеха сценарий возвращает код выхода нуля к его родителю.
Этот сценарий является прекрасным примером использования функций, которые позволяют легче читать сценарий и делать работу быстрее. Заметьте, что здесь используется **sh** вместо **bash**, чтобы сделать их полезными для более широкого круга систем. В системе Linux, вызывая **bash** как **sh**, результаты в оболочке работают в POSIX-совместимом режиме.

Man-страницы **bash** содержат больше информации об объединении команд, **for-** и **while**-петлях, регулярных выражений, а также примеры. Понятный Bash для системных администраторов и опытных пользователей, с упражнениями, от того же автора, как "Введение в руководство Linux", находится на http://tille.garrels.be/training/bash/. Подробное описание особенностей Bash и приложений есть в справочном руководстве "Продвинутые сценарии Bash".

Графическая среда

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Введение

Обычный пользователь может особо не заботиться о настройках входа в систему, однако Linux предлагает широкий спектр красивых менеджеров рабочего стола для использования под X, различные графические среды. Использование и настройка оконных менеджеров и графических сред достаточно простая задача, они напоминают стандартные среды MS Windows, Apple или UNIX CDE, хотя многие пользователи Linux предпочитают роскошные настольные ПК и фантастические оконные менеджеры. Мы не будем обсуждать здесь специфичные настройки пользователя. Простой эксперимент и чтение документации из встроенной **Справки** менеджеров дадут больше информации для того, чтобы настройка стала легким делом.

Мы, однако, внимательно рассмотрим лежащую в основе систему.

Система X Window

X Window System представляет собой явно сетевое окно системы, которая работает на широком спектре

вычислительных и графических машин. Серверы X Window System работают на компьютерах с растровыми дисплеями. X сервер распределяет пользовательский ввод и принимает запросы на вывод от нескольких клиентских программ через различные межпроцессорные каналы связи. Хотя самый распространенный случай – клиентские программы работают на той же машине, что и сервер, но клиенты могут также прозрачно работать от других машин (включая машины с различными архитектурами и операционными системами). Мы узнаем, как это делается в Главе 10, Сеть, в которой рассказывается о сетях и удаленных приложениях.

X поддерживает параллельные иерархические окна, текст и графические операции как на монохромных, так и цветных дисплеях. Число клиентских программ X, которые используют X-сервер достаточно велико. Некоторые программы, внедренные в ядро распределением X Consortium, включают:

- **xterm**: эмулятор терминала
- **twm**: минималистичный оконный менеджер
- **xdm**: менеджер дисплеев
- **xconsole**: программа перенаправления консоли
- **bitmap**: растровый редактор
- **xauth, xhost** и **iceauth**: программы контроля доступа
- **xset, xmodmap** и многие другие: предпочитаемые пользователем настройки программ
- **xclock**: часы
- **xlsfonts** и другие: экранный шрифт, утилиты для просмотра информации о шрифтах, окна и отображения
- **xfs**: сервер шрифтов
- ...

Мы опять отсылаем к man-страницам этих команд за подробной информацией. Подробнее разъяснения доступных функций можно найти в руководстве *Xlib - C language X Interface*, которое поставляется с вашим X дистрибутивом, спецификацией *X Window System Protocol*, а также различные руководства и документация по X. Каталог /usr/share/doc содержит ссылки на эти и многие другие документы.

Многие другие утилиты, оконные менеджеры, игры, пособия и гаджеты включены распределением X Consortium как дополнительное пользовательское программное обеспечение, или доступны с помощью анонимного ftp по Интернету. Хорошими местами для начала являются http://www.x.org и http://www.xfree.org.

Кроме того, все ваши графические приложения, такие как браузер, программа для электронной почты, программа для просмотра изображений, инструменты для проигрывания звуков и так далее являются клиентами X сервера. Отметим, что в обычных условиях эксплуатации графического режима X клиенты и X сервер Linux работают на одной и той же машине.

Имена дисплеев

С точки зрения пользователя, у каждого X сервера есть отображаемое имя в такой форме:

hostname:displaynumber.screennumber

Эта информация используется приложением для определения того, как ему следует соединиться с X сервером и какой экран он должен использовать по умолчанию (на настольных компьютерах с несколькими мониторами):

- *hostname*: Имя хоста указывает имени компьютера-клиента, какой дисплей физически подключен. Если имя хоста не задано, будет использоваться наиболее целесообразный способ общения к серверу на одной и той же машине.
- *displaynumber*: Фраза "display", обычно используется для обозначения коллекции мониторов, которые разделяют общие клавиатуру и указатель (мышь, планшет, и т.д.). У большинства рабочих станций, как правило, только одна клавиатура, и, следовательно, только один дисплей. Большие многопользовательские системы, однако, часто имеют несколько дисплеев, так что группа людей сразу могут выполнять графическую работу. Чтобы избежать путаницы, каждому дисплею компьютера присваивается номер (начиная с 0), когда запускается X сервер для этого дисплея. Номер дисплея должен всегда быть приведен в отображаемом имени.
- *screen number*: Некоторые дисплеи разделяют одну клавиатуру и указатель между двумя или более мониторами. Так как у каждого монитора есть собственный ряд окон, каждому экрану присваивается номер (начиная с 0), когда запускается X сервер для данного дисплея. Если номер экрана не указан, будет использоваться экран 0.

В POSIX-системах имя дисплея по умолчанию хранится в переменной окружения DISPLAY. Эта переменная автоматически устанавливается эмулятором терминала **xterm**. Однако, когда вы по сети входите на другой компьютер, может потребоваться установить DISPLAY вручную, чтобы указать на ваш дисплей, смотрите Раздел "Telnet и X".

Более подробную информацию можно найти в man-страницах для X.

Менеджеры окон и рабочего стола

Расположение окон на экране находится под контролем специальных программ, которые называются оконными менеджерами. Хотя многие оконные менеджеры соблюдают предоставленные геометрические характеристики экрана, существуют такие, которые могут их игнорировать (требуя от пользователя точного указания области окна на экране с помощью указателя, например).

Поскольку оконные менеджеры обычные (хотя и сложные) клиентские программы, могут быть построены разнообразные пользовательские интерфейсы. От X Consortium предоставляется оконный менеджер под названием **twm**, но большинство пользователей предпочитают что-то более изощренное, когда позволяют ресурсы системы. Sawfish и Enlightenment являются популярными примерами, которые позволяют каждому пользователю иметь рабочий стол под соответствующее настроение и стиль.

Менеджер рабочего стола использует один или другой оконный менеджер для организации вашего графического экрана в удобном виде, с панелями, выпадающими меню, информационными сообщениями, часами, менеджерами программ, файловыми менеджерами и т.д. Среди наиболее популярных менеджеров рабочего стола выделяют Gnome и KDE, оба работают почти на любом дистрибутиве Linux и многих других UNIX-системах.

Приложения KDE в Gnome/приложений Gnome в KDE
Вам не обязательно запускать рабочий стол KDE для того, чтобы иметь возможность запускать приложения KDE. Если у вас установлены библиотеки KDE (пакет kdelibs), вы можете запустить эти приложения из меню Gnome или его терминала.
Запуск приложений Gnome в среде KDE немного сложнее, потому что нет единого набора базовых библиотек в Gnome. Однако при запуске или установке таких приложений станут ясны зависимости и, следовательно, дополнительные пакеты, которые вам, возможно, придется установить.

Конфигурация X сервера

Распределение X, которое поставляется с Linux, XFree86, использует конфигурационный файл XF86Config для ее начальной установки. Этот файл настраивает вашу видеокарту и может быть найден в ряде мест, хотя обычно это /etc/X11.

Если вы видите, что файл /etc/X11/XF86Config присутствует в вашей системе, полное описание можно найти в info- или man-страницах о XF86Config.

Из-за проблем с лицензированием XFree86, более новые системы обычно поставляются с вариантом X сервера и инструментов от *X.Org*. Главный конфигурационный файл здесь xorg.conf, обычно находится также в /etc/X11. Файл состоит из нескольких разделов, которые могут находиться в любом порядке. Разделы содержат информацию о вашем мониторе, видеоадаптере, конфигурации экрана, клавиатуры и т.д. В качестве пользователя, вам не надо слишком беспокоиться о том, что в этом файле, так как все обычно определяется на момент, когда система установлена.

Однако, если вам потребуется изменить настройки графического сервера, вы можете запустить инструменты для настройки или редактировать конфигурационный файл, которые обеспечивает инфраструктуру для использования сервера XFree86. Смотрите man-страницы для получения дополнительной информации; у вашего дистрибутива могут быть собственные инструменты. Поскольку неверная конфигурация может привести к нечитаемой чуши в графическом режиме, вам следует сделать резервную копию конфигурационного файла, чтобы обезопасить себя, прежде, чем приступать к его изменению.

Разные специфичные настройки

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Настройка клавиатуры

Настройка раскладки клавиатуры для текстовых консолей производится с помощью команды **loadkeys**. Используйте ваши локальные инструменты конфигурации X или изменяйте раздел *Keyboard* в XF86Config вручную, чтобы настроить раскладку для графического режима. XkbdLayout является тем, что вам требуется установить:

```
XkbLayout       "us"
```

Это по умолчанию. Измените на ваши локальные настройки путем замены значения в кавычках на любые имена, перечисленные в подкаталогах папки keymaps. Если вы не можете найти keymaps, попробуйте отобразить их местонахождение в вашей системе с помощью команды

locate keymaps

Можно комбинировать настройки раскладки, как показано в этом примере:

```
Xkblayout      "us,ru"
```

Сделайте резервную копию файла /etc/X11/XF86Config перед его редактированием! Чтобы сделать это, вам придется использовать аккаунт суперпользователя.

Выйдите и снова войдите в систему, чтобы перезагрузить настройки X.
Апплет клавиатуры Gnome позволяет в режиме реального времени переключаться между раскладками, для использования этой программы специальных разрешений не требуется. KDE также имеет подобный инструмент для переключения между раскладками клавиатуры.

Шрифты

Используйте инструмент **setfont** для загрузки шрифтов в текстовом режиме. Большинство систем поставляются со стандартным файлом inputrc, который позволяет сочетание символов, таких как французский "?" (метасимволы). Для этого системный администратор должен добавить строку

```
export INPUTRC="/etc/inputrc"
```

к файлу /etc/bashrc.

Зона даты и времени

Настройка информации о времени обычно происходит во время установки. После этого, она постоянно соответствует текущей дате за счет использования клиента *NTP* (Network Time Protocol). Большинство Linux системы запускают **ntpd** по умолчанию:

```
debby:~> ps -ef | grep ntpd
ntp      24678     1  0  2002 ?        00:00:33 ntpd -U ntp
```

Вы можете запустить **ntpdate** вручную, чтобы установить время, при условии, что можете подсоединиться к серверу времени. Демон **ntpd** не должен быть запущен, когда вы устанавливаете время с использованием **ntpdate**. Используйте сервер времени в качестве аргумента для команды:

```
root@box:~# ntpdate 10.2.5.200
26 Oct 14:35:42 ntpdate[20364]: adjust time server 10.2.5.200 offset
 -0.008049 sec
```

Смотрите ваши системные руководства и документацию, которые поставляются с пакетом NTP. Большинство менеджеров рабочего стола включают инструменты для установки системного времени при условии, что у вас есть доступ к учетной записи системного администратора.

Для установки правильного часового пояса, вы можете использовать команды **tzconfig** или **timezone**. Обычно информация о часовом поясе задается во время установки вашей операционной системы. Многие системы имеют специфичные для дистрибутива инструменты для настройки; обратитесь к документации по системе.

Язык

Если вы предпочитаете получать сообщения от системы на голландском или французском языке, то можете установить переменные окружения LANG и LANGUAGE, что позволит поддерживать нужный язык и, в конечном итоге, шрифты, связанные с символами на этом языке.

В большинстве графических систем входа, таких как **gdm** или **kdm**, у вас есть возможность настроить параметры языка до входа в учетную запись.

Обратите внимание, сейчас на большинстве систем по умолчанию имеется тенденция к использованию *en_US.UTF-8*. Это не проблема, потому что системы, где такое существует по умолчанию, будут также поставляться со всеми программами, поддерживающими эту кодировку. Таким образом, **vi** может редактировать все файлы на вашей системе, а **cat** не будет вести себя странно и т.п.

Проблема начинается тогда, когда вы подключаетесь к более старой системе, не поддерживающей эту кодировку шрифта, или при открытии файла, закодированного с помощью *UTF-8*, на системе,

поддерживающей шрифты, у которых символы кодируются 1 байтом. Может пригодиться утилита **recode** для конвертирования файлов из одного ряда символов в другой. Читайте man-страницы для обзора возможностей и их использования. Другим решением может быть временная работа в другой кодировке, настроив для этого переменную окружения LANG:

```
debby:~> acroread /var/tmp/51434s.pdf
Warning: charset "UTF-8" not supported, using "ISO8859-1".
Aborted

debby:~> set | grep UTF
LANG=en_US.UTF-8

debby:~> export LANG=en_US

debby:~> acroread /var/tmp/51434s.pdf
<--new window opens-->
```

Установка нового ПО

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Общее

Большинство людей с удивлением видит, что у них после установки Linux оказывается рабочий, пригодный для использования компьютер; в большинстве дистрибутивов поддерживается большинство видео и сетевых карт, мониторов и других внешних устройств, так что, как правило, не требуется установка дополнительных драйверов. Кроме обычных инструментов, таких как офисные пакеты, веб-браузеры, электронная почта и другие программы в наиболее распространенные дистрибутивы включены клиентские сетевые программы. Тем не менее, начальные установки могут не соответствовать вашим требованиям.

Если вы не можете найти то, что вам нужно, то может быть, это просто не установлено в системе. Также может случиться, что у вас есть необходимое программное обеспечение, но оно делает не то, что должно делать. Помните, что Linux быстро развивается, и программное обеспечение улучшается ежедневно. Не стоит тратить время на устранение проблем, которые могут быть уже решены.

Вы можете обновить вашу систему или добавить пакеты в нее в любое время. Большинство программного обеспечения поставляется в пакетах. Дополнительное программное обеспечение может быть найдено на установочных компакт-дисках или в Интернете. Веб-сайт вашего дистрибутива Linux является хорошим местом для начала поиска дополнительного программного обеспечения и содержит инструкции о том, как установить что-то на ваш Linux, см. Приложение A, Куда идти дальше?. Всегда читайте документацию, которая поставляется с новой программой, т.к. там могут находиться какие-нибудь руководства по установке пакета. Все программное обеспечение поставляется с файлом README, который вам очень настоятельно рекомендуется к прочтению.

Форматы пакетов

Пакеты RPM

Что такое RPM?

RPM, RedHat Package Manager, представляет собой мощный менеджер пакетов, который вы можете использовать для установки, обновления и удаления пакетов. Он позволяет вам находить пакеты и отслеживает файлы, которые поставляются с каждым пакетом. Система построена таким образом, что вы можете проверить подлинность пакетов, скаченных из Интернета. Опытные пользователи могут создавать свои собственные пакеты RPM.

Пакет RPM состоит из архива файлов и метаданных, используемых для установки и удаления файлов архива. Метаданные включают вспомогательные скрипты, атрибуты файлов, и информацию со сведениями о пакете. Пакеты выпускаются в двух вариантах: бинарные, используется для упаковки программного обеспечения для установки, и пакеты с исходным кодом и предписанием, необходимым для создания бинарных пакетов.

Многие другие дистрибутивы поддерживают пакеты RPM, среди популярных - RedHat Enterprise Linux, Mandriva (бывший Mandrake), Fedora Core и SuSE Linux. Помимо советов для вашего дистрибутива, вы захотите прочитать **man** rpm.

Примеры работы с RPM

Большинство пакетов просто устанавливаются при использовании опции обновления -U, и не важно установлен уже пакет или нет. Пакет RPM содержит полную версию программы, которая перезаписывает существующую версию или инсталлируется как новый пакет. Типичное использование выглядит следующим образом:

rpm -Uvh /path/to/rpm-package(s)

Опция -v генерирует более подробный вывод, -h заставляет **rpm** выводить индикатор прогресса установки:

```
[root@jupiter tmp]# rpm -Uvh totem-0.99.5-1.fr.i386.rpm
Preparing...                ########################################### [100%]
   1:totem                  ########################################### [100%]
[root@jupiter tmp]#
```

При этом новые пакеты ядра, устанавливаются с опцией -i, которая не перезаписывает существующую версию пакета. Таким образом, вы по-прежнему сможете загрузить систему со старым ядром, если новое окажется не рабочим.
Вы также можете использовать rpm для проверки, установлен ли пакет в вашей системе:

```
[david@jupiter ~] rpm -qa | grep vim
vim-minimal-6.1-29
vim-X11-6.1-29
vim-enhanced-6.1-29
vim-common-6.1-29
```

Или вы можете узнать, какой пакет содержит определенный файл или исполняемый файл:

```
[david@jupiter ~] rpm -qf /etc/profile
setup-2.5.25-1

[david@jupiter ~] which cat
cat is /bin/cat
```

```
[david@jupiter ~] rpm -qf /bin/cat
coreutils-4.5.3-19
```

Заметьте, вам не нужно иметь доступ к административным привилегиям, чтобы использовать rpm для запроса к базе данных RPM. Аккаутн root требуется только при добавлении, изменении или удалении пакетов.
Ниже приведен еще один пример, демонстрирующий, как удалить пакет при помощи **rpm**:

```
[root@jupiter root]# rpm -e totem
[root@jupiter root]#
```

Обратите внимание, что удаление по умолчанию происходит без подробностей, это нормально, что вы не видите многое из того, что происходит. Если вы сомневаетесь, используйте **rpm** -qa опять, чтобы убедиться, что пакет был удален.

RPM может делать гораздо больше, чем те несколько основных функций, которые мы привели здесь.

Пакеты DEB (.deb)

Что из себя представляют пакеты Debian?

Этот формат пакетов используется по умолчанию в Debian GNU/Linux, где **dselect**, и, в настоящее время все более распространенный, **aptitude** являются стандартными инструментами для управления пакетами. Они используются для выбора пакетов, которые вы хотите установить или обновить, но также работают во время установки системы Debian и помогают вам определить метод доступа для использования, список доступных пакетов и их настройки.

Веб-сайт Debian содержит всю необходимую информацию.

Формат пакетов Debian становится все более и более популярным. Также **apt-get** становится популярным на не-DEB системах.

Примеры с инструментами DEB

Проверка, установлен ли пакет, осуществляется с помощью команды **dpkg**. Например, если вы хотите узнать, какая версия Gallery установлена на вашем компьютере, то следует выполнить такую команду:

```
nghtwsh@gorefest:~$ dpkg -l *gallery*
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name           Version        Description
+++-==============-==============-============================================
ii  gallery        1.5-1sarge2    a web-based photo album written in php
```

Префикс "ii" означает, что пакет установлен. Если вы увидите префикс "un", это означает, что пакет известен в списке, который поддерживает ваша система, но не установлен.

Поиск, какому пакету принадлежит файл, выполняется путем использования -S для **dpkg**:

```
nghtwsh@gorefest:~$ dpkg -S /bin/cat
```

```
coreutils: /bin/cat
```

Больше информации можно найти на info-страницах **dpkg**.

Пакеты с исходным кодом

Наибольшая часть программ Linux является Free/Open Source, что означает, что пакеты с исходными кодами доступны для этих программ. Исходники необходимы для компиляции собственного варианта программы. Истодные коды программы можно скачать с ее веб-сайта, обычно в виде сжатого архива (program-version.tar.gz или аналогичного). Для дистрибутивов, основанных на RPM, исходники часто поставляются как program-version.src.rpm. Debian и большинство дистрибутивов на его основе, предоставляют для себя адаптированные исходники, который можно получить с помощью **apt-get source**.

Особые требования, зависимости и инструкции по установке приведены в файле README. Вам, вероятно, понадобится компилятор с языка C **gcc**. Этот компилятор GNU входит в большинство систем Linux и портирован на множество других платформ.

Автоматизация управления пакетами и обновлениями

Общие замечания

Первое, что вы делаете после установки новой системы, - это обновляете ее; это относится ко всем операционным системам и Linux не исключение.

Обновления для большинства систем Linux обычно могут быть найдены на ближайшем сайте-зеркале вашего дистрибутива. Списки сайтов, предлагающих такую возможность, можно найти на веб-сайте вашего дистрибутива, см. Приложение A, Куда идти дальше?.

Обновления следует применять регулярно, если это возможно, то ежедневно, но можно начать с того, чтобы проводить обновление каждые несколько недель. Вы действительно должны стремиться к тому, чтобы иметь самую последнюю версию дистрибутива, так как Linux изменяется постоянно. Как было прежде отмечено, появление новых возможностей, улучшений и исправлений ошибок происходит регулярно, иногда решаются важные проблемы безопасности.

Хорошей новостью является то, что большинство дистрибутивов Linux предоставляют инструменты, поэтому вам не придется ежедневно обновлять десятки пакетов вручную. Следующие разделы дают обзор менеджеров пакетов. Это достаточно обширная тема, даже регулярные обновления пакетов с исходными кодами управляются автоматически; мы перечислим лишь наиболее широко известных системы. Всегда обращайтесь к документации для вашего конкретного дистрибутива для выяснения действий.

APT

Advanced Package Tool является системой управления пакетами программного обеспечения. Команда инструмента для обработки пакетов – **apt-get**, который поставляется с превосходной man-страницей, описывающей как установить и обновить пакеты или весь дистрибутив. APT зародился в дистрибутиве Debian GNU/Linux, где он по умолчанию является менеджером пакетов. Также APT был портирован для работы с пакетами RPM. Главное преимущество APT заключается в том, что он является свободным и гибким в использовании. Это позволит вам создать системы, подобные конкретным дистрибутивам (и в некоторых случаях коммерческие), перечисленные в следующих разделах.

Как правило, при первом использовании **apt-get**, вы должны будете получить индекс доступных пакетов. Это

делается с помощью команды

apt-get update

После этого, вы можете использовать **apt-get** для обновления вашей системы:

apt-get upgrade

Делайте это часто, это легкий способ сохранять вашу систему современной и, следовательно, безопасной.

Помимо этого общего пользования, **apt-get** также очень быстр для установки отдельных пакетов. Вот как это работает:

```
[david@jupiter ~] su - -c "apt-get install xsnow"
Password:
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  xsnow
0 packages upgraded, 1 newly installed, 0 removed and 3 not upgraded.
Need to get 33.6kB of archives.
After unpacking 104kB of additional disk space will be used.
Get:1 http://ayo.freshrpms.net redhat/9/i386/os xsnow 1.42-10 [33.6kB]
Fetched 33.6kB in 0s (106kB/s)
Executing RPM (-Uvh)...
Preparing...                ########################################### [100%]
   1:xsnow                  ########################################### [100%]
```

Обратите внимание на опцию -c команды **su**, которая указывает, чтобы shell администратора выполнила только эту команду, а затем вернуться к среде пользователя. Таким образом, у вас не получится забыть осуществить выход из учетной записи суперпользователя.

Если существуют какие-либо зависимости от других пакетов, **apt-get** загрузит и установит эти вспомогательные пакеты.

Системы, использующие пакеты RPM

Update Agent, который изначально поддерживал только пакеты RPM RedHat, теперь перенесен на более широкий набор программного обеспечения, в том числе не-RedHat репозитории. Этот инструмент предоставляет полную систему для обновления RPM пакетов на системах RedHat или Fedora Core. Для обновления вашей системы в командной строке следует ввести **up2date**. На рабочем столе по умолчанию активируется маленький значок, говорящий вам, нет ли обновлений для вашей системы.

Yellowdog's Updater Modified (**yum**) является еще одним инструментом, который в последнее время стал достаточно популярным. Это интерактивная, но автоматически работающая программа для инсталляции, обновления или удаления пакетов RPM в системе. Этот инструмент выбран на системах Fedora.

В SuSE Linux, все это делается с YaST, Yet another Setup Tool, который поддерживает широкий спектр задач системного администрирования, среди которых и обновление RPM пакетов. Начиная с SuSE Linux 7.1 вы можете производить обновление также с помощью веб-интерфейса и YOU, Yast Online Update.

Mandrake Linux и Mandriva предоставлять так называемые URPMI инструменты, ряд программ-оберток, которые делают установку нового программного обеспечения проще для пользователя. Эти инструменты в

сочетании с RPMDrake и MandrakeUpdate обеспечить все необходимое для упрощения установки и удаления пакетов программного обеспечения. MandrakeOnline предлагает расширенный спектр услуг и может автоматически уведомлять администраторов, когда обновления доступны для вашей конкретной системы. Помимо прочего посмотрите **man** urpmi для дополнительной информации.

Также среды KDE и Gnome имеют собственные (графический) варианты пакетных менеджеров.

Обновление ядра

Большинство установок в Linux протекает без проблем, если вы периодически обновляете ваш дистрибутив. Процедура обновления при необходимости установит новое ядро и сделает все необходимые изменения в вашей системе. Вам придется компилировать или устанавливать новое ядро вручную, если вам нужны функции ядра, которые не включены в ядро вашего дистрибутива Linux по умолчанию.

Компилируете ли вы собственное оптимизированное ядро или используете предварительно скомпилированный пакет ядра, установите его так, чтобы старое и новое ядра существовали вместе, пока вы не убедитесь, что все работает как надо.

Затем создайте двойную загрузку системы, что позволит вам выбрать ядро для загрузки, обновив загрузочный конфигурационный файл grub.conf. Вот простой пример:

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making config changes.
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, e.g.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/hde8
#          initrd /initrd-version.img
#boot=/dev/hde
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title Red Hat Linux new (2.4.9-31)
 root (hd0,0)
 kernel /vmlinuz-2.4.9-31 ro root=/dev/hde8
 initrd /initrd-2.4.9-31.img
title old-kernel
        root (hd0,0)
        kernel /vmlinuz-2.4.9-21 ro root=/dev/hde8
        initrd /initrd-2.4.9-21.img
```

После того, как новое ядро окажется рабочим, вы можете удалить строки для загрузки старого из конфигурационного файла GRUB, хотя лучше подождать пару дней, просто чтобы быть уверенным.

Установка дополнительных пакетов с установочных CD

Монтирование CD

В основном это происходит так же, как и установка пакетов вручную, кроме того, вы должны добавить файловую систему CD к файловой системе вашего компьютера, чтобы сделать ее доступной. На

большинстве систем, это будет сделано автоматически при вставке компакт-диска в привод, потому что во время загрузки запускается демон automount. Если ваш компакт-диск не становится доступным автоматически, выполните команду mount в окне терминала. В зависимости от вашей конкретной системной конфигурации, строка, подобная этой, обычно делает свое дело:

mount /dev/cdrom /mnt/cdrom

На некоторых системах только суперпользователь может монтировать сменные носители, это зависит от конфигурации.

Для того, чтобы все происходило автоматически, устройство для чтения CD обычно имеет запись в /etc/fstab, в котором перечислены файловые системы и их точки монтирования, в результате чего получается дерево файловой системы. Вот подобная строка:

```
[david@jupiter ~] grep cdrom /etc/fstab
/dev/cdrom /mnt/cdrom iso9660  noauto,owner,ro 0 0
```

Это означает, что система будет понимать команду **mount** /mnt/cdrom. Опция noauto означает, что на этой системе компакт-диски не монтируются во время загрузки.

Вы даже можете попробовать щелкнуть правой кнопкой мыши на значке компакт-диска на рабочем столе для монтирования CD, если ваш файловый менеджер не сделает это за вас. Вы можете проверить, сработало ли это, с помощью команды mount без аргументов:

```
[david@jupiter ~] mount | grep cdrom
/dev/cdrom on /mnt/cdrom type iso9660 (ro,nosuid,nodev)
```

Использование CD

После монтирования CD, вы можете перемещаться по его каталогам, как правило, в точке монтирования /mnt/cdrom, где вы можете получить доступ к содержимому CD-ROM. Используйте те же команды для работы с файлами и каталогами, какие вы бы использовали для файлов на жестком диске.

Извлечение CD

Для того, чтобы вытащить компакт-диск из дисковода после того, как вы закончили использовать его, файловая система CD не должна использоваться. Даже нахождение в одном из подкаталогов точки монтирования, /mnt/cdrom в нашем примере, будет рассматриваться как "использование файловой системы", поэтому вам следует выйти оттуда. Сделайте это, например, введя **cd** без аргументов, эта команда вернет вас обратно в домашний каталог. После этого, вы можете использовать команду

umount /mnt/cdrom

или

eject cdrom

Заблокированные диски

Никогда не применяйте силу к дисководу. Трюк со скрепкой является плохой идеей, и хотя это, в конечном итоге, выдернет CD, но ваша система будет думать, что CD по-прежнему там, т.к. обычные процедуры не были соблюдены. Скорее всего, вам придется перезагрузиться, чтобы привести систему в согласованное

состояние.
Если вы продолжаете получать сообщения "устройство занято", сначала проверьте, что все сессии shell вышли из файловой системы CD, а также графические приложения не используют его больше. Если есть сомнения, используйте инструмент **lsof**, чтобы убрать процесс(ы), которые продолжают использовать ресурсы компакт-диска.

Резюме

Если вы дошли до этого места, то это значит, что половина работы уже сделана.

Хотя поддержание порядка имеет важное значение, не менее важно, чтобы вы чувствовали себя как дома в вашей среде, будь она текстовой или графической. Текстовое окружение контролируется с помощью установочных файлов оболочки. Графическая среда, в первую очередь, зависит от конфигурации X сервера, от которого зависит ряд других приложений, например, окна, менеджеры рабочего стола и графические приложения, и у каждого из них есть собственный конфигурационный файл. Вам следует почитать документацию к системе и программам, чтобы узнать о том, как их настраивать.

Региональные настройки, такие как настройку клавиатуры, установку соответствующих шрифтов и поддержку языка, лучше сделать во время установки.

Программное обеспечение управляется либо автоматически, либо вручную с помощью системы пакетов.

В этой главе описывались следующие команды:

Таблица 7.2. Новые команды главы 7: Создай себе дом

Команда	Значение
aptitude	Управление пакетами в стиле Debian
automount	Автоматическое подключение вновь добавленной файловой системы
dpkg	Пакетный менеджер Debian
dselect	Управление пакетами в стиле Debian
loadkeys	Загрузка конфигурации клавиатуры
lsof	Определяет процессы
mount	Подключает новые файловые системы к существующему дереву каталогов
ntpdate	Устанавливает системное время и дату использования времени сервера
quota	Отображает информацию о разрешенном к использованию дисковом пространстве
recode	Конвертирует файлы в другие таблицы символов
rpm	Управление RPM-пакетами
setfont	Выбор шрифтов

timezone	Установка временной зоны
tzconfig	Установка временной зоны
ulimit	Установка или отображение ограничения ресурсов
up2date	Управление RPM-пакетами
urpmi	Управление RPM-пакетами
yum	Управление RPM-пакетами

Упражнения

Окружение Shell

- Выведите ваши настройки окружения. Какую переменную можно использовать для хранения типа процессора вашей машины?
- Создайте скрипт, который может выдать строку типа "Hello, World". Дайте ему соответствующие разрешения, чтобы его можно было запустить. Проверьте свой сценарий.
- Создайте папку в вашем домашнем каталоге и переместите скрипт в новую директорию. Добавить эту новую директорию в путь поиска на постоянной основе. Проверьте, что сценарий может быть выполнен без указания пути его фактического нахождения.
- Создайте подкаталоги в домашнем каталоге для хранения различных файлов, например, папку music для хранения аудио файлов, папку documents для заметок, и т.д. И используйте их!
- Создайте собственную строку приглашения.
- Отобразите ограничения на использование ресурсов. Вы можете изменить их?
- Попробуйте прочитать сжатые man-страницы без предварительной их распаковки.
- Сделать псевдоним **lll** для команды **ls -la**.
- Почему команда **tail testfile > testfile** не работает?
- Подключите CD с данными, например, установочный компакт-диск с Linux, и посмотрите, что на нем. Не забудьте отключить его, когда он вам больше не будет нужен.
- Скрипт из Раздела "Некоторые простые примеры" не совсем корректный. Он выдает ошибки для файлов, которые являются каталогами. Измените сценарий так, чтобы он выбирал только текстовые файлы для копирования. Чтобы делать выбор, используйте **find**. Не забудьте сделать скрипт исполняемым, прежде чем пытаться запустить его.

Графическое окружение

- Попробуйте все кнопки мыши в различных местах (терминале, фоне, панели задач).
- Исследуйте меню.
- Настройте окно вашего терминала.
- Используйте кнопки мыши для копирования и вставки текста из одного терминала в другой.

- Узнайте, как настроить ваш оконный менеджер; попробуйте различные рабочие столы (виртуальные экраны).
- Добавьте на панель задач апплет монитора загрузки.
- Примените другую тему оформления.
- Включите так называемый мокрый фокус, когда окно активизируется всего лишь перемещением мыши над ним, так что вам не приходится нажимать на него, чтобы оно стало активным.
- Переключитесь на другой менеджер окон.
- Выйдите из системы и выберите другой тип сессии, например, KDE, если вы использовали Gnome до этого. Повторите предыдущие шаги.

Глава 8. Принтеры и печать

Аннотация

В этой главе мы узнаем о принтерах и печати файлов. После прочтения этой части, вы сможете:

- Оформлять документы
- Предварительно просматривать документы перед отправкой их на принтер
- Выбрать хороший принтер, который работает в вашей системе Linux
- Печатать файлы и проверять состояние принтера
- Устранять проблемы при печати
- Находить необходимую документацию о том, как установить принтер

Печать файлов

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Команда для печати

Получение файла для принтера

Очень легко распечатать документ из приложения, выбрав в меню пункт **Печать**.

Для печати из командной строки используйте команду **lp** или **lpr**.
lp file(s)
lpr file(s)

Эти команды могут читать из канала, так что вы можете распечатать вывод какой-либо команды вот так command | lp

Существует множество опций, осуществляющих настройку макета страницы, установку количества копий, выбор принтер, на который вы хотите отправить печать, если у вас их больше, чем один, размер бумаги, одностороннюю или двустороннюю печать, если принтер поддерживает эту функцию, поля и т.д. Читайте man-страницы для полного обзора.

Статус вашего задания на печать

После того как файл принят в очередь печати, данному заданию на печать присваивается идентификационный номер:

```
davy:~> lp /etc/profile
request id is blob-253 (1 file(s))
```

Для просмотра очереди печати используются команды **lpq** или **lpstat**. Когда они вводятся без аргументов, то отображается по умолчанию содержимое очереди печати.

```
davy:~> lpq
blob is ready and printing
Rank Owner Job File(s)  Total Size
active davy 253 profile  1024 bytes
davy:~> lpstat
blob-253 davy 1024 Tue 25 Jul 2006 10:20_01 AM CEST
```

Статус принтера

Какой принтер установлен по умолчанию в системе, которая имеет доступ к нескольким принтерам?

```
davy:~> lpstat -d
system default destination: blob
```

Какой статус у моего принтера(ов)?

```
davy:~> lpstat -p
printer blob now printing blob-253. enabled since Jan 01 18:01
```

Удаление заданий из очереди печати

Если вам не нравится, что вы видите из состояния команд, используйте **lprm** или **cancel** для удаления заданий.

```
davy:~> lprm 253
```

В графической среде вы можете увидеть всплывающее окно, сообщающее вам, что задание было отменено.

В больших системах **lpc** может быть использована для управления несколькими принтерами. Смотрите info- или man-страницы для каждой команды.

Существует множество GUI инструментов для печати, которые используются как интерфейс для **lp**, и у большинства графических приложений есть функции печати, которые используют **lp**. См. встроенную справку по функциям и специальную документацию к программам для получения дополнительной информации.

Почему существует две команды для каждой задачи, связанной с печатью?
Печать на UNIX и подобных системах имеет длинную историю. Когда-то использовались два весьма различных подхода: BSD-стиль печати и SystemV-стиль печати. Для совместимости Linux CUPS поддерживает команды обоих способов. Также следует отметить, что **lp** не ведет себя так же, как **lpr**, у **lpq**

есть несколько различных опций, почти **lpstat** и **lprm**, но не совсем, как **cancel**. Не важно какую вы используете, просто используйте команды, которые наиболее удобный для вас, или с которыми у вас есть опыт работы в других UNIX-подобных системах.

Форматирование

Инструменты и языки

Если мы хотим получить что-то разумное от принтера, в первую очередь файл должен быть отформатирован. Помимо того, что существует обилие программного обеспечения для оформления, Linux поставляется с базовыми UNIX-инструментами и языками для форматирования.

Современные системы Linux поддерживают прямую печать, без форматирования пользователем целого ряда типов файлов: текст, PDF, PostScript и несколько графических форматов, таких как PNG, JPEG, BMP и GIF.

Для тех форматов файлов, которым необходимо форматирование, Linux поставляется с большим количеством соответствующих инструментов, таких как команды **pdf2ps**, **fax2ps** и **a2ps**, которые преобразуют другие форматов в PostScript. Эти команды могут создавать файлы, которые затем могут использоваться в других системах, в которых не установлены все инструменты для преобразования формата.

Помимо этих инструментов командной строки существует много графических программ для обработки текста. Доступно несколько офисных комплектов, многие из которых свободны. Они выполняют оформление автоматически после подачи задания на печать. Вот некоторые: OpenOffice.org, KOffice, AbiWord, WordPerfect и т.д.

Ниже приведены распространенные языки по отношению к печати:

- **groff**: GNU версия команды **roff** UNIX. Является интерфейсом к groff-системе форматирования документов. Обычно она запускает команду **troff**, а для выбранного устройства назначается пост-процессор. Это позволяет создавать файлы PostScript.
- TeX и макропакет LaTeX: один из наиболее широко используемых языков разметки в системах UNIX. Обычно вызывается как **tex**, который форматирует файлы и выводит соответствующее аппаратно-независимое представление набранного документа.
  Технические работы по-прежнему часто создаются в LaTeX из-за его поддержки математических формул, хотя усилия, чтобы включить эту функцию в другие приложениях, предпринимаются W3C (World Wide Web Consortium).
- SGML и XML: Свободные парсеры, доступные для UNIX и Linux. XML является следующим поколением SGML, он образует основу для DocBook XML — системы документов (эта книга написана на XML, например).

Печать документации

Man-страницы содержат предварительно отформатированные **troff** данные, которые должны быть приведены к соответствующему виду, прежде чем они могут быть распечатаны на принтере. Печать выполняется с использованием опции -t команды **man**:
man -t *command* > man-command.ps
После чего распечатывается файл PostScript. Если место назначения печати настроено для вашей системы/аккаунта по умолчанию, вы можете просто выполнить команду **man** -t *command* для отправки напрямую отформатированных страниц на принтер.

Предварительный просмотр отформатированных файлов

Все, что вы можете отправить на принтер, может с таким же успехом быть отображено на экране. В зависимости от формата файла, можно использовать одну из следующих команд:

- Файлы PostScript: с помощью команды **gv** (GhostView).
- TeX dvi файлов: **xdvi**, или с помощью варианта для KDE - **kdvi**.
- PDF файлы: **xpdf, kpdf, gpdf** или программа для просмотра Adobe, **acroread**, которая также доступна для бесплатного использования, но не является свободным программным обеспечением.
- В таких приложениях как Firefox или OpenOffice обычно можно выбрать предварительный просмотр печати из меню.

Серверная сторона

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Основное

Когда-то выбор для пользователей Linux был прост: все использовали один и тот же старый LPD от принадлежащего BSD кода Net-2. В то время LPR был более популярным, однако в настоящее время современные дистрибутивы Linux используют CUPS, Common UNIX Printing System (общую систему печати UNIX). CUPS представляет собой реализацию Internet Printing Protocol (IPP), подобного HTTP стандарта RFC, заменяющего почтенный (и неуклюжий) протокол LPD. CUPS распространяется под GNU Public License. CUPS также является системой печати по умолчанию в MacOS X.

Графическая конфигурация принтера

Большинство дистрибутивов поставляются с графическим интерфейсом для настройки сетевых и локальных (через параллельный порт или USB) принтеров. Они позволяют выбрать принтер из списка и проверить его работоспособность. Вам не придется беспокоиться о содержимом и местонахождении конфигурационных файлов. Перед установкой принтера смотрите системную документацию.

CUPS можно также настроить с помощью веб-интерфейса, который работает на порте 631 вашего компьютера. Чтобы проверить, что такая возможность включена, попробуйте просмотреть localhost:631/help или localhost:631/.

Покупка принтера для Linux

Поскольку все больше и больше поставщиков принтеров делают драйверы доступными для CUPS, последний позволяет легко подключить практически любой принтер, который вы можете вставить в последовательный, параллельный или USB порт, а также любой принтер в сети. CUPS будет обеспечивать единообразное представление для вас и ваших приложений всех различных типов принтеров.

Принтеры, которые поставляются только с драйверами для Windows, могут вызывать проблемы, если у них нет иной поддержки.

В прошлом, вашим самым лучшим выбором был бы принтер с поддержкой PostScript в прошивке, так как почти все программное обеспечение для печати UNIX или Linux производит печатную продукцию на PostScript, лучшем языке для печати на принтере. Принтеры PostScript, как правило, немного дороже, но

PostScript - аппаратно-независимый, открытый язык программирования, и вы всегда на 100% уверены, что принтер с такой прошивкой будет работать. Однако в наши дни это уже почти не актуально.

Проблемы печати

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

В этом разделе мы обсудим, что может сделать пользователь, когда что-то идет не так. Мы не будем обсуждать все проблемы, имеющие отношение к демонам служб печати, так как это задача для системных администраторов.

Не тот файл

При печати не того файла, задание может быть отменено с помощью команды **lprm** jobID, где jobID указывается в формате *printername-printjobnumber* (получить такую информацию можно с помощью команд **lpq** или **lpstat**). Это будет работать, когда другие задания ожидают печати в очереди этого принтера. Однако, вы должны действовать очень быстро, если единолично используете принтер, так как задания, как правило, становятся в очередь и отправляются на принтер в считанные секунды. Как только они появляются на принтере, то становится уже слишком поздно удалять задания инструментами Linux.

В подобных случаях, а также когда неправильно настроен драйвер печати, и выходит только какой-то мусор, вы можете попробовать выключить принтер. Однако, это может быть не лучшим путем решения проблемы, так как может привести к замятию бумаги и прочим нарушениям.

Не получается распечатать

С помощью команды **lpq** вы можете увидеть статус вашего задания:

```
elly:~> lpq
Printer: lp@blob
 Queue: 2 printable jobs
 Server: pid 29998 active
 Unspooler: pid 29999 active
 Status: waiting for subserver to exit at 09:43:20.699
 Rank   Owner/ID            Class Job Files          Size Time
1      elly@blob+997          A   997 (STDIN)         129 09:42:54
2      elly@blob+22           A    22 /etc/profile    917 09:43:20
```

В наши дни многие принтеры имеют веб-интерфейсы, которые могут отображать информацию о состоянии, если ввести IP-адрес принтера в веб-браузере:
Рисунок 8.1. Состояние принтера через веб-интерфейс

Веб-интерфейс CUPS в сравнении с веб-интерфейсом принтера

Заметьте, что это не веб-интерфейс CUPS и работает только для принтеров, поддерживающих такую функцию. Проверьте документацию вашего принтера.

Если ID вашего задания нет в системе и нет на принтере, обратитесь к системному администратору. Если ID вашего задания указан в выходных данных, убедитесь, что принтер в настоящее время печатает. Если это так, просто подождите, ваше задание будет выполнено в свое время.
Если принтер не печатает, проверьте, что там есть бумага, физическое подключение к электроэнергии и сети передачи данных. Если все нормально, принтер может нуждаться в перезагрузке. Узнайте мнение системного администратора.

В случае сетевого принтера, попробуйте выполнить печать с другого хоста. Если принтер доступен с вашего собственного хоста (см. Главу 10, Сеть), вы можете попробовать отправить файл в его формате, file.ps в случае принтера PostScript, используя клиент FTP. Если это сработает, ваша система печати неправильно сконфигурирована. Если не сработает, возможно принтер не понимает формат, который вы ему

скармливаете.

Резюме

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Служба печати Linux поставляется с рядом инструментов, основанных на стандартных инструментах UNIX LPD, будь то варианты от SystemV или BSD. Ниже приведен список команд, связанных с печатью.

Таблица 8.1. Новые команды в главе 8: Печать

Команда	Значение
lpr или **lp**	Печать файла
lpq или **lpstat**	Запрос очереди печати
lprm или **cancel**	Удаление задания на печать
acroread	Просмотр документов pdf
groff	Инструмент форматирования
gv	Просмотр файлов PostScript
printconf	Настройка принтеров
xdvi	Просмотр документов dvi
xpdf	Просмотр документов pdf
***2ps**	Конвертация файлов в PostScript

Упражнения

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Настройка и тестирование принтеров означает быть под одним пользователем и иметь доступ к учетной записи суперпользователя. Если это так, то вы можете попробовать:

- Установку принтера с помощью графического интерфейса вашей системы.
- Печать тестовой страницы с помощью GUI.
- Печать тестовой страницы с помощью команды **lp**.
- Печать из приложения, например, Mozilla или OpenOffice, выбрав **Файл ? Печать** в меню.
- Отключите принтер от сети или локального компьютера/принт-сервера. Что происходит при попытке что-нибудь распечатать?

Следующие упражнения можно делать без принтера или доступа к аккаунту root.

- Попробуйте создать файлы PostScript из различных исходных файлов (например, HTML, PDF, man-страниц). Посмотрите результат с помощью вьювера **gv**.
- Убедитесь, что демон печати работает.
- Как бы ни было попробуйте распечатать файл. Что происходит?
- Создайте файл PostScript с помощью Mozilla. Проверьте его с помощью **gv**.
- Конвертируйте его в формат PDF. Посмотрите с помощью **xpdf**.
- Как бы вы распечатали файл GIF из командной строки?
- Используйте **a2ps** для печати файл /etc/profile к выходному файлу. Проверьте опять с **gv**. Что произойдет, если вы не уточните выходной файл?

Глава 9. Основные методы резервного копирования

Аннотация

Аварии рано или поздно случаются. В этой главе мы обсудим, как сохранить данные в безопасном месте с помощью других компьютеров, дискет, компакт-дисков и лент. Мы также рассмотрим наиболее популярные команды для сжатия и архивирования.

По завершении этой главы вы будете знать, как:

- Создавать и распаковывать файловые архивы, а также получать о них сведения
- Обходиться дискетой и создавать загрузочный диск для вашей системы
- Записывать CD-диски
- Создавать обновляемые резервные копии
- Создавать Java-архивы
- Находить документацию по использованию других устройств и программ резервного копирования
- Как шифровать ваши данные

Введение

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Хотя Linux является одной из самых безопасных операционных систем существующих на данный момент, и даже если он проектировался, чтобы быть еще более надежным, данные могут быть потеряны. К потере данных чаще всего приводят ошибки пользователей, но иногда причиной является неисправность системы, такая как сбой питания или выход из строя диска, так что неплохо всегда иметь дополнительные копии личных и/или важных данных.

Подготовка данных

Архивирование с помощью tar

В большинстве случаев, сначала принято собирать все данные для резервного копирования в один архивный файл, который затем сжимают. Сам процесс архивирования представляет собой объединение всех перечисленных файлов и устранение бесполезных пустот. В Linux для этого часто используют команду **tar**. Эта команда была первоначально создана для архивирования данных на ленты, но она также может создавать архивы, известные как тарболы (tarballs).

У **tar** есть множество опций, наиболее важные из них приведены ниже:

-v: подробный вывод
-t: тест, показывающий содержимое архива
-x: извлечение архива
-c: создание архива
-f archivedevice: использование archivedevice в качестве источника/назначения для архива, устройство по умолчанию — первый накопитель на магнитной ленте (обычно /dev/st0 или что-то подобное)
-j: фильтр через **bzip2**, см. Раздел "Обновление бэкапов с помощью tar"

Перед опциями **tar** обычно пропускают штрих-префикс, что видно в примерах ниже.

 **Использование GNU tar для совместимости.**
Архивы, созданные с помощью проприетарной версии **tar** в одной системе, могут быть несовместимы с **tar** на другой проприетарной системе. Это может доставить головную боль, например, если архив может быть восстановлен только в системе, которая больше не существует. Используйте GNU-версию **tar** на всех системах, чтобы ваш системный администратор не рыдал. Linux всегда использует GNU tar. При работе на других машинах UNIX, введите **tar –help**, чтобы выяснить, какую версию вы используете. Свяжитесь с системным администратором, если вы не заметите слова GNU.

В приведенном ниже примере архив создается и распаковывается.

```
gaby:~> ls images/
me+tux.jpg  nimf.jpg

gaby:~> tar cvf images-in-a-dir.tar images/
images/
images/nimf.jpg
images/me+tux.jpg

gaby:~> cd images

gaby:~/images> tar cvf images-without-a-dir.tar *.jpg
me+tux.jpg
nimf.jpg

gaby:~/images> cd

gaby:~> ls */*.tar
images/images-without-a-dir.tar

gaby:~> ls *.tar
images-in-a-dir.tar

gaby:~> tar xvf images-in-a-dir.tar
```

```
images/
images/nimf.jpg
images/me+tux.jpg

gaby:~> tar tvf images/images-without-dir.tar
-rw-r--r-- gaby/gaby  42888 1999-06-30 20:52:25 me+tux.jpg
-rw-r--r-- gaby/gaby   7578 2000-01-26 12:58:46 nimf.jpg

gaby:~> tar xvf images/images-without-a-dir.tar
me+tux.jpg
nimf.jpg

gaby:~> ls *.jpg
me+tux.jpg  nimf.jpg
```

Этот пример также иллюстрирует разницу между заархивированными каталогом и группой файлов. Желательно сжимать только каталоги, так как файлы могут не распаковаться вместе (архив может быть на другой системе, и вы можете не знать, какие файлы уже существовали, а какие те, которые из архива).

Если магнитный накопитель подключен к вашей машине и настроен системным администратором, имена файлов, оканчивающиеся на .tar заменяются именем устройства, например:

tar cvf /dev/tape mail/

Каталог почты и все файлы, которые он содержит, сжимаются в файл, который записывается сразу на носитель. Содержание листинга отображается, потому что мы использовали соответствующую опцию.

Обновление бэкапов с помощью tar

Программа **tar** поддерживает создание дополняющихся резервных копий, с помощью опции -N. С помощью этой опции можно указать дату, и **tar** будет проверять время изменения всех указанных файлов от этой даты. Если файлы будут изменены позже, они будут включены в бэкап. В приведенном ниже примере используется метка на предыдущий архив как значение даты. Сначала создается начальный архив и метка указывает на этот архивный файл. Затем создается новый файл, который принимает новую резервную копию, содержащую только этот новый файл:

```
jimmy:~> tar cvpf /var/tmp/javaproggies.tar java/*.java
java/btw.java
java/error.java
java/hello.java
java/income2.java
java/income.java
java/inputdevice.java
java/input.java
java/master.java
java/method1.java
java/mood.java
java/moodywaitress.java
java/test3.java
java/TestOne.java
java/TestTwo.java
```

```
java/Vehicle.java

jimmy:~> ls -l /var/tmp/javaproggies.tar
-rw-rw-r-- 1 jimmy   jimmy   10240 Jan 21 11:58 /var/tmp/javaproggies.tar

jimmy:~> touch java/newprog.java

jimmy:~> tar -N /var/tmp/javaproggies.tar \
-cvp /var/tmp/incremental1-javaproggies.tar java/*.java 2> /dev/null
java/newprog.java

jimmy:~> cd /var/tmp/

jimmy:~> tar xvf incremental1-javaproggies.tar
java/newprog.java
```

Стандартные ошибки перенаправляются в /dev/null. Если вы не сделаете этого, **tar** будет выводить сообщение для каждого неизмеренного файла, сообщая вам, что он не будет сбрасываться.

У такого способа работы есть неудобство, т.к. просматриваются метки на файлы. Скажем, вы загрузили архив в каталог, содержащий ваши бэкапы, и этот архив содержит файлы, которые были созданы два года назад. При сверке меток этих файлов с метками на начальный архив, новые файлы действительно окажутся старым для **tar** и не будут добавлены в резервную копию с помощью опции -N.

Лучшим выбором будет опция -g, которая создаст список файлов для резервного копирования. При обновлении резервной копии, файлы сверяются с этим списком. Вот как это работает:

```
jimmy:~> tar cvpf work-20030121.tar -g snapshot-20030121 work/
work/
work/file1
work/file2
work/file3

jimmy:~> file snapshot-20030121
snapshot-20030121: ASCII text
```

На следующий день, пользователь *jimmy* немного поработал с file3 и создал file4. В конце дня он создает новый бэкап:

```
jimmy:~> tar cvpf work-20030122.tar -g snapshot-20030121 work/
work/
work/file3
work/file4
```

Это очень простые примеры, но вы также можете использовать этот вид команды в работе хрона (см. Раздел "Cron и crontab"), который указывает, например, снимок файла для еженедельного резервного копирования и один для ежедневного резервного копирования. Снимки файлов должны быть заменены, в случае, когда создается полная резервная копия.

Более подробную информацию можно найти в документации к **tar**.

 **Важно знать.**

Как вы могли, вероятно, заметить, **tar** хорошо работает, когда речь идет о простом каталоге с набором лежащих вместе файлов. Однако есть инструменты, которые легче в управлении, когда вам нужен архив целых разделов, дисков или больших проектов. Мы же рассказываем про tar потому, что это очень популярный инструмент для распространения архивов. Довольно часто бывает ситуация, когда вам необходимо установить программное обеспечение, входящее в так называемый "сжатый тарбол". См. Раздел "Использование rsync", чтобы узнать более простой способ выполнять регулярное резервное копирование.

Сжатие и распаковка с помощью gzip и bzip2

Данные, в том числе архивы, могут быть сжаты с использованием инструментов zip. Команда **gzip** добавит суффикс .gz к имени файла и удаляет исходный файл.

```
jimmy:~> ls -la | grep tar
-rw-rw-r-- 1 jimmy  jimmy    61440 Jun  6 14:08 images-without-dir.tar

jimmy:~> gzip images-without-dir.tar

jimmy:~> ls -la images-without-dir.tar.gz
-rw-rw-r-- 1 jimmy  jimmy    50562 Jun  6 14:08 images-without-dir.tar.gz
```

Распаковываются сжатые файлы с помощью опции -d.

bzip2 работает аналогично, но использует улучшенный алгоритм сжатия, поэтому файлы получаются меньшего размера. За дополнительной информацией обратитесь к info-страницам **bzip2**.

Пакеты софта для Linux часто распространяются как сжатые архивы. После распаковки такого архива разумно найти README и прочитать его. Он обычно содержат руководство для установки пакета.

Команда GNU tar понимает сжатые файлы. Используйте команду

tar zxvf file.tar.gz

для распаковки файлов tar.gz или .tgz. Используйте

tar jxvf file.tar.bz2

для распаковки tar-архивов, которые были сжаты с помощью **bzip2**.

Архивы Java

Проект GNU предоставляет также **jar** для создания Java-архивов. Это приложение Java, которая объединяет несколько файлов в один архивный файл JAR. Хотя он поддерживает обычное архивирование и сжатие, основанное на ZIP и ZLIB форматах сжатия, **jar** в основном предназначен для облегчения упаковки кода Java, апплетов и/или приложений в один файл. Скомбинированные в одном архиве, компоненты приложения Java, могут быть скачаны гораздо быстрее.

В отличие от **tar, jar** сжимает по умолчанию, независимо от других инструментов - потому что это в основном java-версия zip. Кроме того, **jar** позволяет на отдельные записи в архиве ставить подпись автора, что дает возможность проверять происхождение.
Синтаксис практически идентичен команде **tar**. Чтобы найти различия обратитесь к info-страницам **jar**.

 **tar, jar и символические ссылки.**
То, на что следует обратить внимание в упомянутой документации, **jar** следует по символьным ссылкам. Данные, к которым эти ссылки указывают, будут включены в архив. По умолчанию **tar** делает резервную копию самой символьной ссылки, но это поведение можно изменить с помощью опции -h для **tar**.

Транспортировка данных

Сохранение копии ваших данных на другом компьютере является наиболее надежным способом создания резервных копий. См. Главу 10, Сеть для дополнительной информации по **scp, ftp** и др.

В следующем разделе мы обсудим иные устройства для резервного копирования.

Перемещение данных на устройство резервного копирования

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Создание копии на дискете

Форматирование дискеты

На большинстве систем Linux, пользователи имеют доступ к дисководу. Название устройства может варьировать в зависимости от размера и количества гибких дисков. На некоторых системах, вероятно, это будет ссылка /dev/floppy, указывающая на нужное устройство, возможно /dev/fd0 (при автоматическом обнаружении дисковода гибких дисков) или /dev/fd0H1440 (для дискет в 1,44 Мб).

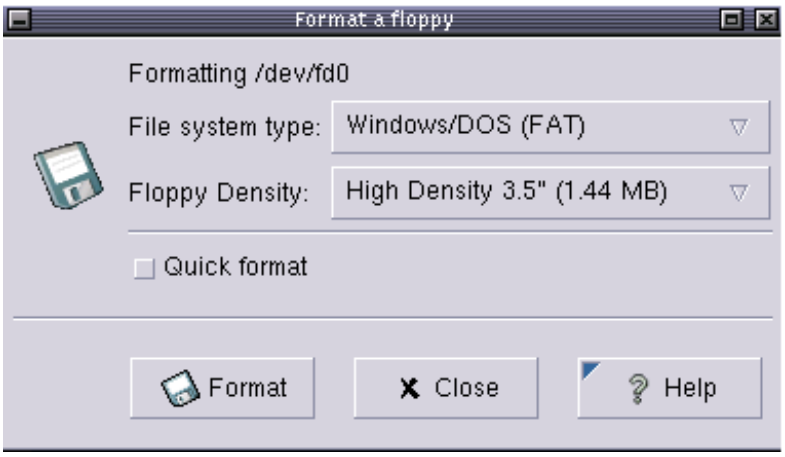
fdformat представляет собой инструмент низкоуровневого форматирования дискеты. Команда принимает имя устройства в качестве опции. **fdformat** сообщит об ошибке, если дискета защищена от записи.

```
emma:~> fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
emma:~>
```

Команда **mformat** (из пакета mtools) используется для создания DOS-совместимых дискет, которые затем могут быть доступны через **mcopy, mdir** и другие m-команды.

Кроме того, доступны графические инструменты.

Рисунок 9.1. Форматирование дискеты



После того как дискета отформатирована, она может быть примонтирована к файловой системе и доступна как обычный, хотя маленький, каталог, обычно через /mnt/floppy.

Если вам это нужно, установите утилиту **mkbootdisk**, которая создает дискету, с которой может загрузиться текущая система.

Использование команды dd для сбрасывания данных

Команда **dd** может использоваться для перемещения данных на диск, или обратно на дискету, в зависимости от заданных устройств ввода и вывода. Например:

```
gaby:~> dd if=images-without-dir.tar.gz of=/dev/fd0H1440
98+1 records in
98+1 records out

gaby~> dd if=/dev/fd0H1440 of=/var/tmp/images.tar.gz
2880+0 records in
2880+0 records out

gaby:~> ls /var/tmp/images*
/var/tmp/images.tar.gz
```

Заметьте, что сброс производится на демонтированное устройство. Дискеты, созданные с помощью этого метода не монтируются к файловой системе, это способ для создания загрузочных или спасательных дисков. Для получения дополнительной информации о возможностях **dd** читайте man-страницы.

Этот инструмент является частью пакета GNU *coreutils*.

 **Сбрасывание дисков.**
Команда **dd** также может быть использована для создания "сырого" дампа всего жесткого диска.

Создание копии на CD

На некоторых системах пользователи могут использовать записывающие CD-устройства. Сначала ваши данные должны быть отформатированы. Используйте команду **mkisofs**, чтобы сделать это в каталоге, содержащем файлы, для которых вы хотите сделать резервную копию. Проверьте командой **df**, что на диске достаточно свободное места, так как новый файл будет примерно такого же размера, как вся текущая директория:

```
[rose@blob recordables] df -h .
Filesystem            Size  Used Avail Use% Mounted on
/dev/hde5              19G   15G  3.2G  82% /home

[rose@blob recordables] du -h -s .
325M    .

[rose@blob recordables] mkisofs -J -r -o cd.iso .
<--snap-->
making a lot of conversions
<--/snap-->
98.95% done, estimate finish Fri Apr  5 13:54:25 2002
Total translation table size: 0
Total rockridge attributes bytes: 35971
Total directory bytes: 94208
Path table size(bytes): 452
Max brk space used 37e84
166768 extents written (325 Mb)
```

Опции -J и -r используются для монтирования CD-ROM на различных системах, см. man-страницы. После этого, компакт-диск можно создать с помощью инструмента **cdrecord** с соответствующими опциями:

```
[rose@blob recordables] cdrecord -dev 0,0,0 -speed=8 cd.iso
Cdrecord 1.10 (i686-pc-linux-gnu) (C) 1995-2001 Joerg Schilling
scsidev: '0,0,0'
scsibus: 0 target: 0 lun: 0
Linux sg driver version: 3.1.20
Using libscg version 'schily-0.5'
Device type    : Removable CD-ROM
Version        : 0
Response Format: 1
Vendor_info    : 'HP      '
Identification : 'CD-Writer+ 8100 '
Revision       : '1.0g'
Device seems to be: Generic mmc CD-RW.
Using generic SCSI-3/mmc CD-R driver (mmc_cdr).
Driver flags   : SWABAUDIO
Starting to write CD/DVD at speed 4 in write mode for single session.
Last chance to quit, starting real write in 0 seconds.
Operation starts.
```

В зависимости от скорости вашего CD-дисковода, у вас появится время, чтобы перекусить. Когда задание будет выполнено, вы получите подтверждающее сообщение:

```
Track 01: Total bytes read/written: 341540864/341540864
          (166768 sectors).
```

Существуют некоторые графические инструменты, облегчающие создание дисков. Одним из популярных является **xcdroast**, который находится в свободном доступе на сайте http://www.xcdroast.org и включен во многие системы и в каталог GNU. Менеджеры рабочего стола KDE и Gnome имеют собственные средства для создания компакт-дисков.

Бэкапы на/с jazz дисков, USB-устройств и других съемных носителей

Эти устройства обычно монтируются к файловой системе. После процедуры монтирования они доступны как обычные каталоги, так что вы можете использовать стандартные команды для работы с файлами.

В примере, приведенном ниже, подготавливается копирование изображений с камеры на жесткий диск:

```
robin:~> mount /mnt/camera

robin:~> mount | grep camera
/dev/sda1 on /mnt/camera type vfat (rw,nosuid,nodev)
```

Если у камеры есть только USB-устройство хранения, которое вы когда-нибудь подключали к вашей системе, это безопасно. Но имейте в виду, что USB-устройства присваиваются записям в /dev, как только они подключены к системе. Таким образом, если вы сначала подключите USB-устройство к системе, оно окажется на /dev/sda входе, и если вы подключите камеру после этого, ей будет назначен в /dev/sdb - при условии, что у вас нет каких-либо SCSI-дисков, которые также монтируются на /dev/sd*. На новых системах, где ядро 2.6, автоопределение системы под названием HAL (Hardware Abstraction Layer) гарантирует, что пользователи не должны иметь дело с этим бременем. Если вы хотите проверить, где устройство, введите **dmesg** после его вставки.

Теперь можно копировать файлы:

```
robin:~> cp -R /mnt/camera/* images/

robin:~> umount /mnt/camera
```

Кроме того, джаз-диск может быть смонтирован в /mnt/jazz.
Чтобы это произошло, соответствующие строки должны быть добавлены в /etc/modules.conf и /etc/fstab.
Обратитесь к специальным HOWTO аппаратного обеспечения для получения дополнительной информации.
В системах с ядром 2.6.x или выше, вы также можете посмотреть man-страницы **modprobe** и modprobe.conf.

Резервирование данных с помощью ленточных устройств

Это делается с помощью **tar** (см. ранее). Инструмент **mt** используется для контроля магнитного ленточного устройства, подобно /dev/st0. Целые книги были написаны о резервном копировании на ленту, поэтому обратитесь к Приложению B, Сравнение команд DOS и Linux. Имейте в виду, что базы данных могут потребовать другие процедуры резервного копирования из-за своей архитектуры.

Соответствующие команды резервного копирования обычно помещают в один из каталогов cron с тем, чтобы они выполнялись регулярно.

Инструменты вашего дистрибутива

Большинство дистрибутивов Linux предлагают свои собственные инструменты для облегчения жизни. Вот краткий перечень:

- SuSE: YaST теперь включает расширенные модули для резервного копирования и восстановления.
- RedHat: инструмент File Roller обеспечивает визуальное управление сжатыми архивами. Похоже, они предпочитают X-CD-Roast для перемещения бэкапов на внешние устройства.
- Mandrake: X-CD-Roast.

- Большинство дистрибутивов поставляются с утилитами BSD **dump** и **restore** для создания резервных копий файловых систем *ext2* и *ext3*. Этот инструмент может записывать на различные устройства и буквально сбрасывать файл(ы) или файловую систему бит за битом на указанное устройство. Подобно **dd**, это позволяет создавать бэкапы для специальных типов файлов, подобных тем, что находятся в /dev.

Использование rsync

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Введение

Программа **rsync** представляет собой быстрый и гибкий инструмент для удаленного резервного копирования. Она часто встречается на UNIX и UNIX-подобных системах, легко настраивается и используется в сценариях. Хотя *r* в **rsync** означает "удаленный", вы не должны понимать это слишком буквально. Ваше "удаленное" устройство может быть просто устройством USB или другим разделом жесткого диска, вам не обязательно иметь две отдельные машины.

Пример: применение rsync к устройству USB

Как уже говорилось в Разделе 3.1.2.3 "Точки монтирования", мы сначала должны смонтировать устройство. Возможно, это должно быть сделано под *root*'ом:

```
root@theserver# mkdir /mnt/usbstore

root@theserver# mount -t vfat /dev/sda1 /mnt/usbstore
```

 **Удобство для пользователя**
Все больше и больше дистрибутивов предоставляют доступ к съемным устройствам для непривилегированных пользователей и монтируют USB-устройства, CD-диски и другие съемные устройства автоматически.

Заметим, что такая директива требует USB поддержку, которая устанавливается на вашей системе. Проверьте с **dmesg**, что /dev/sda1 действительно является устройством для монтирования.

Затем вы можете непосредственно начать создавать резервную копию, например, каталога /home/karl:

```
karl@theserver:~> rsync -avz /home/karl/ /mnt/usbstore
```

Как обычно, за дополнительной информацией обращайтесь к man-страницам.

Шифрование

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Общие замечания

Почему вам следует шифровать данные?

Шифрование - это синоним секретности. Совместно с резервным копированием, шифрование может быть очень полезно, например, если вам нужно оставить зарезервированные данные в месте, где вы не можете контролировать доступ, допустим, на сервере вашего провайдера.

Кроме того, шифрование может быть также применено к электронной почте: обычно, почта не шифруется и часто пересылается в открытом виде по местной сети или Интернет. Если ваше сообщение содержит конфиденциальную информацию, лучше его зашифровать.

GNU Privacy Guard

В операционных системах Linux вы найдете GnuPG, GNU Privacy Guard, который представляет собой набор программ, совместимых с инструментами PGP (Pretty Good Privacy), который имеется в продаже.

В этом руководстве мы обсудим только очень простое использование инструментов шифрования и покажем, что вам следует делать, чтобы генерировать ключ шифрования и использовать его для шифрования данных, после чего вы сможет безопасно хранить их в общественном месте. Более сложные способы использования можно найти на man-страницах различных команд.

Генерирование ключа

Перед началом шифрования данных, необходимо создать пару ключей. Пара состоит из закрытого и открытого ключа. Вы можете отправить открытый ключ корреспондентам, которые могут использовать его для шифрования данных для вас, которые вы будете расшифровывать вашим закрытым ключом. Вы всегда должны хранить закрытый ключ, никогда не делиться им с кем-то еще, иначе другие будут иметь возможность расшифровать данные, предназначенные только для вас. Просто чтобы была уверенность, что ничего не случится, закрытый ключ защищается паролем. Пара ключей создается с помощью следующей команды:

```
willy@ubuntu:~$ gpg --gen-key
gpg (GnuPG) 1.4.2.2; Copyright (C) 2005 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions.  See the file COPYING for details.

gpg: directory `/home/willy.gnupg' created
gpg: new configuration file `/home/willy/.gnupg/gpg.conf' created
gpg: WARNING: options in `/home/willy/.gnupg/gpg.conf' are not yet
 active during this run
gpg: keyring `/home/willy/.gnupg/secring.gpg' created
gpg: keyring `/home/willy/.gnupg/pubring.gpg' created
Please select what kind of key you want:
    (1) DSA and Elgamal (default)
    (2) DSA (sign only)
    (5) RSA (sign only)
Your selection? 1
DSA keypair will have 1024 bits.
ELG-E keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
         0 = key does not expire
```

```
        = key expires in n days
      w = key expires in n weeks
      m = key expires in n month
      y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

You need a user ID to identify your key; the software constructs the
user ID from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) "

Real name: Willy De Wandel
Email address: wdw@mvg.vl
Comment: Willem
You selected this USER-ID:
    "Willy De Wandel (Willem) "

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
You need a Passphrase to protect your secret key.

Passphrase:
```

Теперь введите свой пароль. Это может быть фраза любой длины и содержания, единственным условием является то, что вы должны быть в состоянии запомнить ее навсегда. Для проверки необходимо ввести ту же фразу еще раз.

Теперь пара ключей генерируется программой, которая порождает случайные числа, и эта деятельность, среди прочих факторов, обеспечивается текущей активностью системы. Так что хорошей идеей будет запуск в это время каких-нибудь программ, перемещение курсора мыши или ввод некоторых случайных символов в окне терминала. Это повысит шансы генерирования числа, которое содержит множество различных цифр, и ключ будет труднее взломать.

О вашем ключе

Когда ваш ключ будет создан, вы получите сообщение об "отпечатках пальцев". Это последовательность из 40 шестнадцатеричных чисел, которая представляет собой достаточно длинную последовательность, ее почти невозможно создать дважды на любом компьютере. Вы можете быть в достаточной степени уверены, что это уникальная последовательность. Краткая форма этого ключа состоит из вашего имени, потом следуют последние 8 шестнадцатеричных чисел.

Вы можете получить информацию о вашем ключе следующим образом:

```
willy@ubuntu:~$ gpg --list-keys
/home/willy/.gnupg/pubring.gpg
------------------------------
pub     1024D/BF5C3DBB 2006-08-08
uid                    Willy De Wandel (Willem)
sub     4096g/A3449CF7 2006-08-08
```

Для данного ключа *key ID* имеет значение "BF5C3DBB". Вы можете отправить ваши *key ID* и имя на сервер ключей, таким образом другие люди смогут получить эту информацию о вас и использовать ее для

шифрования данных для вас. Кроме того, вы можете непосредственно отправить ваш открытый ключ людям, которым он нужен. Открытая часть ключа представляет собой длинный ряд цифр, которые вы можете увидеть при использовании опции --export команды **gpg**:

gpg --export -a

Однако, мы считаем, что в пределах данного руководства вам потребуется только ключ для шифрования и дешифрования данных для себя. Читайте man-страницы команды **gpg**, если хотите знать больше.

Шифрование данных

Теперь вы можете шифровать архивы .tar или сжатые архивы, перед сохранением их на резервном носителе или транспортировке на сервер, предназначенный для бэкапов. Используйте команду **gpg** вот так: gpg -e -r (part of) uid archive

Опция -e говорит **gpg** шифровать данные, опция -r указывает, для кого выполняется шифрование. Имейте в виду, что только пользователи, чьи имена следуют после опции -r будут в состоянии расшифровать данные. Например:

```
willy@ubuntu:~$ gpg -e -r Willy /var/tmp/home-willy-20060808.tar
```

Расшифровка файлов

Используя опцию -d, вы можете расшифровать файлы, которые были зашифрованы для вас. Данные будут прокручиваться на экране, но зашифрованная копия останется на диске. Таким образом, для файлов, которые имеют отличный от обычного текста формат, вы захотите сохранить расшифрованные данные, чтобы посмотреть их соответствующей программой. Это делается с помощью опции -o команды **gpg**:

```
willy@ubuntu:~$ gpg -d -o /var/tmp/home-willy-decrypt.tar /var/tmp/home-willy-
20060808.tar.gpg

You need a passphrase to unlock the secret key for
user: "Willy De Wandel (Willem) "
4096 ELG-E key, ID A3449CF7, created 2006-08-08 (main key ID BF5C3DBB)

gpg: encrypted with 4096-bit ELG-E key, ID A3449CF7, created 2006-08-08
        "Willy De Wandel (Willem) "
```

Нет пароля = нет данных

Если вы не сможете вспомнить свой пароль, данные будут потеряны. И даже системный администратор не сможет расшифровать данные. Именно поэтому копии важных ключей иногда хранят в запечатанном сейфе в банке.

Резюме

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Здесь список команд, так или иначе связанных с созданием резервной копии файла:

Таблица 9.1. Новые команды из главы 9: Резервное копирование

Команда	Значение
bzip2	Блок-сортирующий компрессор файлов.
cdrecord	Запись владельцем CD с аудио и данными.
dd	Конвертация и копирование файла.
fdformat	Низкоуровневое форматирование дискеты.
gpg	Шифрование и дешифрование данных.
gzip	Сжатие или распаковка файлов.
mcopy	Копирование файлов MSDOS в/из UNIX.
mdir	Отображение каталога MSDOS.
mformat	Добавление файловой системы MSDOS к низкоуровнево-отформатированной дискете.
mkbootdisk	Создание загрузочной дискеты для текущей системы.
mount	Монтирование файловой системы (включение ее в текущую файловую систему, подключение к точке монтирования).
rsync	Синхронизация каталогов.
tar	Утилита архивирования на ленту, также используется для создания архивов на дисках.
umount	Размонтирование файловых систем.

Упражнения

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

- Создайте резервную копию вашей домашней директории в /var/tmp, используя команду **tar**. После чего сожмите файл с помощью **gzip** или **bzip2**. Делайте это так, чтобы получить архивный файл, который не создает беспорядок при распаковке.
- Отформатируйте дискету и запишите на нее некоторые файлы из вашей домашней директории.
- Отформатируйте дискету в формате DOS. Используйте *mtools*, чтобы размещать и удалять файлы на ней.
- Что происходит с неформатированной дискетой, когда вы хотите смонтировать ее с вашей файловой системой?
- Если у вас есть устройство хранения USB, попробуйте положить файл на него.
- Используя **rsync**, создайте копию вашего домашнего каталога на другой локальной или удаленной файловой системе.

- Оставляя файлы на сетевом сервере, лучше зашифровать их. Создайте архив **tar** своего домашнего каталога и зашифруйте его.

Глава 10. Сеть

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Аннотация

Когда речь идет о сети, то часто выбор падает на операционные системы на базе Linux. Это связано не только с тем, что сеть тесно интегрирована в самой ОС и существует большой набор бесплатных инструментов и приложений. Также это связано с надежностью системы при высоких нагрузках, что было достигнуто после многих лет ее отладки и тестирования в проекте Open Source.

Книжные полки полны информацией, написанной о Linux и сетях, мы же постараемся в этой главе дать краткий обзор. После ее завершения, вы будете знать больше о

- Поддерживаемых сетевых протоколах
- Конфигурационных файлах сети
- Командах настройки и исследования сети
- Демонах и клиентских программах, запускающих различные сетевые приложения
- Разделении доступа к файлам и печати
- Удаленном выполнении команд и приложений
- Основах сетевого взаимодействия
- Безопасном выполнении удаленных приложений
- Брандмауэрах и обнаружении вторжений

Беглый обзор сети

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Модель OSI

Протокол - это, проще говоря, набор правил для общения.

Для того, чтобы передать данные по сети, например, электронную почту с вашего компьютера на какой-нибудь компьютер в другом конце мира, должны вместе работать множество различных аппаратных и программных средств.
Все эти элементы железа и различного ПО "говорят на разных языках". Представьте, вашу программу для E-mail: она в состоянии обмениваться информацией с операционной системой компьютера через специальный протокол, но не в состоянии "говорить" с компьютерной техникой. Нам нужны специальные программы в операционной системе, которые выполняли бы эту функцию. В свою очередь, компьютер должен быть в состоянии соединяться с телефонной линией или другой аппаратурой для подключения к Интернет. А за кулисами сетевое оборудование должно быть в состоянии соединяться, чтобы передать вашу почту от одного устройства к другому, вплоть до конечного компьютера.

Все эти разные виды протоколов связи подразделяются на 7 уровней, которые известны как модель Open Systems Interconnection Reference (модель взаимодействия открытых систем), или, сокращенно, модель OSI. Для облегчения понимания, эта модель сводится к четырёхуровневому описанию протоколов, как показано в таблице ниже:

Таблица. Упрощенная модель OSI

Название уровня	Протоколы уровня
Уровень приложений	HTTP, DNS, SMTP, POP, ...
Транспортный	TCP, UDP
Сетевой	IP, IPv6
Уровень доступа к сети	PPP, PPPoE, Ethernet

Каждый уровень может использовать функциональность только уровня, расположенного ниже; каждый уровень может предоставлять функциональность уровню, расположенному выше. Другими словами: уровни взаимодействуют только со своими соседями. Возьмем пример сообщения электронной почты: вы вводите его через прикладной уровень. В вашем компьютере оно спускается до транспортного и сетевого уровня. Ваш компьютер выкладывает его в сеть через уровень доступа к сети. Это уровень, который будет перемещать сообщение по всему миру. В месте назначения принимающий компьютер примет сообщение через свой сетевой уровень, и далее оно будет предоставлено получателю через транспортный и прикладной уровни.

 **В действительности все гораздо сложнее.**
Вам рано или поздно придется столкнуться с некоторыми сетевыми терминами; эти разделы дадут вам ряд отправных точек, когда вам захочется узнать о деталях.

Некоторые популярные сетевые протоколы

Linux поддерживает много разных сетевых протоколов. Перечислим только наиболее важные:

TCP/IP

Transport Control Protocol и *Internet Protocol* являются наиболее популярными способами соединения в Интернете. Множество приложений, таких как браузер и программа для электронной почты, построены в первую очередь на этом наборе протоколов.

Если уж совсем просто говорить, то IP обеспечивает отправку пакетов информации от одной машины к другой, в то время как TCP гарантирует правильную последовательность пакетов, так что пакеты различных приложений не путаются, и будут доставлены получателю в правильном порядке.

Хорошей отправной точкой для получения дополнительной информации о TCP и IP являются следующие документы:

- **man** 7 ip: описание реализации протокола IPv4 на Linux (версия 4 в настоящее время наиболее широко распространенный вариант протокола IP).
- **man** 7 tcp: реализация протокола TCP.

- RFC793, RFC1122, RFC2001 для TCP, и RFC791, RFC1122 и RFC1112 для IP.

Документы содержат описания сетевых стандартов, протоколов, приложений и реализации. Эти документы находятся под управлением Internet Engineering Task Force, международной общественности, заинтересованной в бесперебойной работе Интернет и развитии его архитектуры.

TCP/IPv6

Никто не ожидал, что Интернет разрастется так быстро. В IP нашлись некоторые недостатки, связанные с большим количеством компьютеров в сети, наиболее серьезный из которых - наличие уникальных адресов, назначаемых каждому компьютеру в сети. Таким образом, IP версии 6 был способен удовлетворить потребности современного Интернет.

К сожалению, не все приложения и службы пока поддерживают IPv6. В настоящее время миграция осуществляется во многих средах, которые могут получить выгоду от обновления до IPv6. Для некоторых приложений, старый протокол все еще используется, для приложений, которые были переработаны, новая версия уже действует. Так проверка конфигурации сети иногда может быть немного запутанной, так как могут быть приняты всевозможные меры, чтобы скрыть один протокол от другого, чтобы не перепутать два соединения.

Более подробную информацию можно найти в следующих документах:

- **man** 7 ipv6: реализация протокола Linux IPv6.
- RFC1883 описании протокола Ipv6.

PPP, SLIP, PLIP, PPPOE

Ядро Linux имеет встроенную поддержку PPP (Point-to-Point-Protocol), SLIP (Serial Line IP), PLIP (Parallel Line IP) и PPPP Over EThernet. PPP - наиболее популярный способ для индивидуальных пользователей получать доступ к их ISP (Internet Service Provider), хотя в густонаселенных районах часто используется PPPOE, протокол, используемый для ADSL (Asymmetric Digital Subscriber Line) соединений.

Большинство дистрибутивов Linux предоставляют легкий в использовании инструмент для настройки Интернет-соединения. Единственное, что вам понадобится - это имя пользователя и пароль для подключения к вашему провайдеру (ISP), и номер телефона в случае PPP. Эти данные вводятся в окно для настройки, которое, вероятно, также позволит запускать и разрывать соединение с вашим провайдером.

ISDN

Ядро Linux имеет встроенные возможности ISDN. Isdn4linux управляет картами ISDN PC и может эмулировать модем с набором команд Hayes ("AT" команды). Возможности от просто использования терминальной программы до полного подключения к Интернет.

Посмотрите системную документацию.

AppleTalk

Appletalk - это название межсетевого стека Apple. Он делает возможным сетевую модель peer-to-peer, которая обеспечивает основные функциональные возможности, такие как к разделение файлов и принтеров. Каждая машина может одновременно выступать в качестве клиента и сервера, а необходимые

программы и аппаратное обеспечение включены в каждый компьютер Apple.

Linux полностью поддерживает сети AppleTalk. Netatalk является реализация на уровне ядра набора протоколов AppleTalk, первоначально это было только для BSD-подобных систем. Включает поддержку маршрутизации AppleTalk, обслуживание файловых систем UNIX и AFS, использующих AppleShare, а также обслуживание принтеров UNIX и доступ к принтерам AppleTalk.

SMB/NMB

Для совместимости с MS Windows пакет Samba, включающий поддержку протоколов NMB и SMB, может быть установлен на любой UNIX-подобной системе. Протокол Server Message Block (также называемый Session Message Block, NetBIOS или протокол LanManager) используется на MS Windows 3.11, NT, 95/98, 2K и XP для доступа к дискам и принтерам.
Основными функциями пакета Samba являются: разделение дисков Linux с машинами Windows, доступ к SMB частям из машин Linux, совместное использование принтеров Linux с машин Windows и совместное использование принтеров Windows с машин Linux.

Большинство дистрибутивов Linux поставляются с пакетом samba, который делает большую часть работы при установке сервера и запускает smbd, сервер Samba, и nmbd, netbios имени сервера, загрузочное время по умолчанию. Samba можно настроить графически, с помощью веб-интерфейса или через командную строку, а также текстовые конфигурационные файлы. Демоны заставляют машину Linux представляться как хост Windows в сетевом окружении Windows; разделяемые ресурсы компьютера с Linux будут неотличимы от любого другого компьютера в среде MS Windows.

Более подробную информацию можно найти в следующих местах:

- **man** smb.conf: описывает формат основного конфигурационного файла Samba.
- Samba Project Documentation (или проверьте локальные зеркала samba.org) содержит подробное руководство об установке и тестировании, в котором также объясняется, как настроить сервер Samba в качестве основного контроллера домена. Все man-страницы также доступны здесь.

Смешанные протоколы

Linux также имеет поддержку для Amateur Radio, сетей WAN (X25, Frame Relay, ATM), InfraRed и других беспроводных соединений, но, поскольку эти протоколы обычно требуют специального оборудования, мы не будем обсуждать их в данном документе.

Конфигурация сети и информация о ней

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Конфигурация сетевых интерфейсов

Все большие дружелюбные дистрибутивы Linux поставляются с различными графическими инструментами, которые позволяют легко подключить компьютер к локальной сети, поставщику услуг Интернета или настроить беспроводное соединение. Эти инструменты могут быть запущены из командной строки или из меню:

- Настроить Ubuntu можно так: **Система ? Администрирование ? Сеть**.

- RedHat Linux поставляется с **redhat-config-network**, который имеет как графический интерфейс, так и работает в текстовом режиме.
- YAST Suse или YaST2 представляет собой инструмент настройки "все-в-одном".
- Mandrake/Mandriva поставляется с мастером настройки сети и Интернет, который обычно запускается из **Центра управления Mandriva**.
- В системах Gnome - **gnome-network-preferences**.
- В системах KDE - **knetworkconf**.

Системная документация содержит множество советов и информации о наличии и использовании этих программ.

Информация, которую вам необходимо предоставить:

- Для подключения к локальной сети, например, с домашних компьютеров или на работе: имена хоста, домена и IP-адрес. Если вы хотите установить собственную сеть, лучше всего сначала почитать что-нибудь еще. На работе эту информацию компьютер может получать автоматически при загрузке. Если вы сомневаетесь, то лучше ничего не указывать, чем прописывать неизвестно что.
- Для подключения к Интернет: имя пользователя и пароль для вашего провайдера, номер телефона при использовании модема. Ваш провайдер обычно автоматически назначает вам IP-адрес и все остальное, что необходимо для работы ваших интернет-приложений.

Сетевые конфигурационные файлы

Графические программы-помощники редактируют определенный набор конфигурационных файлов сети, используя несколько основных команд. Точные имена конфигурационных файлов и их расположение в файловой системе в значительной степени зависит от вашего дистрибутива и его версии. Тем не менее, несколько сетевых конфигурационных файлов можно найти на всех UNIX-системах:

/etc/hosts

Файл /etc/hosts всегда содержит локальный IP-адрес 127.0.0.1, который используется для связи между процессами. Никогда не удаляйте эту строку! Иногда содержит адреса дополнительных хостов, с которыми можно соединиться без использования внешней службы имен, такой как DNS (Domain Name Server).

Пример файла хостов для небольшой домашней сети:

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain   localhost
192.168.52.10 tux.mylan.com  tux
192.168.52.11 winxp.mylan.com  winxp
```

Дополнительная информация содержится в **man** hosts.

/etc/resolv.conf

В файле /etc/resolv.conf настраивается доступ к DNS-серверу. Этот файл содержит ваши доменные имена и имена сервера(ов) для соединения:

```
search mylan.com
```

```
nameserver 193.134.20.4
```

Дополнительную информацию можно почерпнуть на man-страницах resolv.conf.

/etc/nsswitch.conf

Файл /etc/nsswitch.conf определяет порядок, в котором происходит связь с различными службами имен. Для использования Интернет, важно, чтобы dns появилась в строке "hosts":

```
[bob@tux ~] grep hosts /etc/nsswitch.conf
hosts: files dns
```

Это указывает компьютеру на поиск имен хостов и IP-адресов сначала в файле /etc/hosts, и связь с сервером DNS, если данный хоста нет в локальном файле hosts. Другими возможными именами сервисов для соединения являются LDAP, NIS и NIS +.

Подробнее в **man** nsswitch.conf.

Сетевые конфигурационные команды

Команда ip

Специфичные для дистрибутива скрипты и графические инструменты — это надстройки к **ip** (или **ifconfig** и **route** на старых системах) для просмотра и настройки сетевой конфигурации ядра.

Команда **ip** используется для присвоения IP-адресов интерфейсам, установки маршрутов к Интернет и другим сетям, отображения конфигураций TCP/IP и так далее.

Следующие команды показывают IP-адрес и информацию о маршрутизации:

```
benny@home benny> ip addr show
1: lo:  mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
    inet6 ::1/128 scope host
2: eth0:  mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:50:bf:7e:54:9a brd ff:ff:ff:ff:ff:ff
    inet 192.168.42.15/24 brd 192.168.42.255 scope global eth0
    inet6 fe80::250:bfff:fe7e:549a/10 scope link

benny@home benny> ip route show
192.168.42.0/24 dev eth0  scope link
127.0.0.0/8 dev lo  scope link
default via 192.168.42.1 dev eth0
```

На что стоит обратить внимание:

- Два сетевых интерфейса даже на системе, которая имеет только одну карту с сетевым интерфейсом: "lo" является локальным узлом, используемым для внутренней сетевой коммуникации; "eth0" является общим названием для реального интерфейса. Никогда не изменяйте конфигурацию локального узла, иначе ваша машина перестанет нормально работать! Беспроводные интерфейсы, как правило, определяется как "wlan0"; модемные интерфейсы как "ppp0", хотя могут быть и другие имена.

- IP-адреса, отмеченные как "inet": у локального узла всегда 127.0.0.1, физический интерфейс может иметь любую другую комбинацию.
- Аппаратный адрес вашего интерфейса, который может потребоваться как часть процедуры проверки подлинности для подключения к сети, отмечен как "ether". У локального узла имеется 6 пар из нулей, физический узел состоит из 6 пар шестнадцатеричных символов, из которых первые 3 пары зависят от конкретного производителя.

Команда ifconfig

Хотя **ip** является самым новым способом настроить систему Linux, по-прежнему популярным является **ifconfig**. Используйте его без опции для вывода информации о сетевом интерфейсе:

```
els@asus:~$ /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:70:31:2C:14
          inet addr:60.138.67.31  Bcast:66.255.255.255  Mask:255.255.255.192
          inet6 addr: fe80::250:70ff:fe31:2c14/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:31977764 errors:0 dropped:0 overruns:0 frame:0
          TX packets:51896866 errors:0 dropped:0 overruns:0 carrier:0
          collisions:802207 txqueuelen:1000
          RX bytes:2806974916 (2.6 GiB)  TX bytes:2874632613 (2.6 GiB)
          Interrupt:11 Base address:0xec00
          lo        Link encap:Local Loopback
   inet addr:127.0.0.1  Mask:255.0.0.0
   inet6 addr: ::1/128 Scope:Host
   UP LOOPBACK RUNNING  MTU:16436  Metric:1
   RX packets:765762 errors:0 dropped:0 overruns:0 frame:0
   TX packets:765762 errors:0 dropped:0 overruns:0 carrier:0
   collisions:0 txqueuelen:0
   RX bytes:624214573 (595.2 MiB)  TX bytes:624214573 (595.2 MiB)
```

Также отметим наиболее важные аспекты конфигурации интерфейса:

- IP-адрес отмечен как "inet addr".
- Аппаратный адрес тегом "HWaddr".

Обе команды (**ifconfig** и **ip**) отображают более подробные сведения о конфигурации и ряд статистических данных о каждом интерфейсе и, наверное, самое главное, является ли он "UP" и "RUNNING".

Команды PCMCIA

На вашем ноутбуке, который вы обычно подключаете к корпоративной сети с помощью Ethernet-соединения, но которое вы сейчас настраиваете для dial-in дома или в отеле, возможно необходимо активировать карту PCMCIA. Это делается с помощью утилита управления **cardctl**, или **pccardctl** на новых дистрибутивах.

Пример использования:

cardctl insert

Теперь карта может быть настроена с помощью графического или интерфейса командной строки. До

извлечения карты используйте следующую команду:

cardctl eject

Тем не менее, хороший дистрибутив должн обеспечить поддержку PCMCIA в инструментах конфигурации сети, предотвращая пользователей от необходимости выполнять команды PCMCIA вручную.

Дополнительная информация

Дальнейшее обсуждение настройки сети выходит за рамки этого документа. Вашим основным источником дополнительной информации будут man-страницы для служб, которые вы хотите установить. Дополнительное чтение:

- Модем-HOWTO (tldp.org/HOWTO/Modem-HOWTO.html). Поможет с выбором, соединением, настройкой, поиском неисправностей и пониманием аналоговых модемов для ПК.
- LDP HOWTO Index, раздел 4.4 (tldp.org/HOWTO/HOWTO-INDEX/networking.html). Список HOWTO по категориям об обычных сетях, протоколах, dial-up, DNS, VPNs, мостах, маршрутизации, безопасности и многом другом.
- На большинстве систем есть версия файла ip-cref (найти его можно с помощью команды **locate**); формат PS этого файла можно просматривается, например, с помощью **gv**.

Названия сетевых интерфейсов

На Linux-машине, название устройства *lo* или локальный узел связано с внутренним адресом 127.0.0.1. Компьютер будет иметь сложности, заставляя ваши приложения работать, если это устройство не будет существовать; так всегда даже на тех компьютерах, которые не объединены в сеть.

Первое ethernet-устройство, *eth0*, в случае стандартной сетевой картой, указывает на ваш локальный адрес IP-сети. Обычные клиентские машины имеют только одну сетевую карту. Маршрутизаторы (роуторы), связующие сети между собой, имеют по одному сетевому устройству на каждую сеть, которую они обслуживают.

Если вы используете модем для подключения к Интернет, ваше сетевое устройство, скорее всего, будет называться *ppp0*.

Существует намного больше имен, например, для интерфейса Virtual Private Network (VPN), и множество интерфейсов могут быть активны одновременно, так что вывод команд **ifconfig** или **ip** может быть довольно длинным, если не использовать опции. Даже могут быть активны несколько интерфейсов одного и того же типа. В этом случае, они нумеруются последовательно: первый получит номер 0, второй получит суффикс 1, третий 2, и так далее. Так обстоит дело на многих серверных приложениях, на машинах с отказоустойчивой конфигурацией, роутерах, брандмауэрах и многом другом.

Проверка конфигурации хоста с помощью netstat

Кроме команды **ip** для отображения конфигурации сети, есть команд **netstat**, которая имеет много опций и, как правило, полезна в любой системе UNIX.

Маршрутизация информация может быть отображена с помощью опции -nr команды **netstat**:

```
bob:~> netstat -nr
Kernel IP routing table
```

```
Destination  Gateway      Genmask       Flags MSS Window irtt Iface
192.168.42.0 0.0.0.0      255.255.255.0 U      40 0         0 eth0
127.0.0.0    0.0.0.0      255.0.0.0     U      40 0         0 lo
0.0.0.0      192.168.42.1 0.0.0.0       UG     40 0         0 eth0
```

Это обычный клиентский компьютер в сети IP. У него только одно сетевое устройство, *eth0*. Интерфейс *lo* является локальным узлом.


Современный способ
Новый способ получить эту же информацию от вашей системы – использовать команду **ip**:
ip route show

Когда эта машина пытается соединиться с хостом, находящимся в другой сети, а не ее собственной, обозначаемый линией, начиная с 0.0.0.0, она будет посылать запросы на подключение к маршрутизатору с IP-адресом 192.168.42.1, и будет использовать основной интерфейс, *eth0*, чтобы сделать это.

Хосты, которые находятся в той же сети, имеют линию, начиная с 192.168.42.0, также будут соединяться через основной сетевой интерфейс, но нет необходимости в маршрутизаторе, данные просто выкладываются в сеть.

Компьютеры могут иметь гораздо более сложные таблицы маршрутизации, с большим количеством различных пар "адресат-шлюз" для подключения к различным сетям. Если у вас есть возможность подключиться к серверу приложений, например, на работе, то ради интереса проверьте информацию о маршрутизации.

Другие хосты

Внушительное количество инструментов ориентировано на управление сетью и удаленное администрирование машин Linux. Ваше местное зеркало программного обеспечения для Linux предложит множество из них. Это увело бы нас слишком далеко, если обсуждать их в этом документе, поэтому, пожалуйста, обратитесь к документации конкретных программ.

В этом разделе мы обсудим только некоторые общие UNIX/Linux текстовые инструменты.

Команда host

Для отображения информации о хостах или доменах используется команда **host**:

```
[emmy@pc10 emmy]$ host www.eunet.be
www.eunet.be. has address 193.74.208.177

[emmy@pc10 emmy]$ host -t any eunet.be
eunet.be. SOA dns.eunet.be. hostmaster.Belgium.EU.net.
  2002021300 28800 7200 604800 86400
eunet.be. mail is handled by 50 pophost.eunet.be.
eunet.be. name server ns.EU.net.
eunet.be. name server dns.eunet.be.
```

Аналогичная информация может быть отображена с помощью команды **dig**, которая предоставляет дополнительную информацию о том, каким образом записи хранятся на сервере имен.

Команда ping

Чтобы проверить, доступен ли хост, используйте **ping**. Если ваша система настроена отправлять более одного пакета, **ping** можно прервать комбинацией клавиш **Ctrl + C**:

```
[emmy@pc10 emmy]$ ping a.host.be
PING a.host.be (1.2.8.3) from 80.20.84.26: 56(84) bytes of data.
64 bytes from a.host.be(1.2.8.3):icmp_seq=0 ttl=244 time=99.977msec
--- a.host.be ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/mdev = 99.977/99.977/99.977/0.000 ms
```

Команда traceroute

Чтобы проверить маршрут, по которому пакеты следуют к сетевому узлу, используйте команду **traceroute**:

```
[emmy@pc10 emmy]$ /usr/sbin/traceroute www.eunet.be
traceroute to www.eunet.be(193.74.208.177),30 hops max,38b packets
1 blob (10.0.0.1)
       0.297ms  0.257ms  0.174ms
2 adsl-65.myprovider.be (217.136.111.1)
       12.120ms 13.058ms 13.009ms
3 194.78.255.177 (194.78.255.177)
       13.845ms 14.308ms 12.756ms
4 gigabitethernet2-2.intl2.gam.brussels.skynet.be (195.238.2.226)
       13.123ms 13.164ms 12.527ms
5 pecbru2.car.belbone.be (194.78.255.118)
       16.336ms 13.889ms 13.028ms
6 ser-2-1-110-ias-be-vil-ar01.kpnbelgium.be (194.119.224.9)
       14.602ms 15.546ms 15.959ms
7 unknown-195-207-939.eunet.be (195.207.93.49)
       16.514ms 17.661ms 18.889ms
8 S0-1-0.Leuven.Belgium.EU.net (195.207.129.1)
       22.714ms 19.193ms 18.432ms
9 dukat.Belgium.EU.net (193.74.208.178) 22.758ms * 25.263ms
```

На некоторых системах **traceroute** может быть переименована в **tracepath**.

Команда whois

Специфичная информация о доменных именах может быть получена с помощью команды **whois**, которая разъясняется множеством **whois** серверов, как показано ниже:

```
[emmy@pc10 emmy]$ whois cnn.com
[whois.crsnic.net]

Whois Server Version 1.3

 $<--snap server message-->
```

```
   Domain Name: CNN.COM
   Registrar: NETWORK SOLUTIONS, INC.
   Whois Server: whois.networksolutions.com
   Referral URL: networksolutions.com
   Name Server: TWDNS-01.NS.AOL.COM
   Name Server: TWDNS-02.NS.AOL.COM
   Name Server: TWDNS-03.NS.AOL.COM
   Name Server: TWDNS-04.NS.AOL.COM
   Updated Date: 12-mar-2002
>>> Last update of whois database: Fri, 5 Apr 2002 05:04:55 EST <<<

The Registry database contains ONLY .COM, .NET, .ORG, .EDU domains
and Registrars.

[whois.networksolutions.com]

        $<--snap server message-->

Registrant:
Turner Broadcasting (CNN-DOM)
   1 CNN Center
   Atlanta, GA 30303

   Domain Name: CNN.COM

   Administrative Contact:
        $<--snap contactinfo-->
   Technical Contact:
        $<--snap contactinfo-->
   Billing Contact:
 $<--snap contactinfo-->
   Record last updated on 12-Mar-2002.
   Record expires on 23-Sep-2009.
   Record created on 22-Sep-1993.
   Database last updated on 4-Apr-2002 20:10:00 EST.

   Domain servers in listed order:

   TWDNS-01.NS.AOL.COM  149.174.213.151
   TWDNS-02.NS.AOL.COM  152.163.239.216
   TWDNS-03.NS.AOL.COM  205.188.146.88
   TWDNS-04.NS.AOL.COM  64.12.147.120
```

Для других доменных имен, отличных от .com, .net, .org и .edu, возможно, потребуется указать сервер whois, например, как для этого домена .be:
whois domain.be@whois.dns.be

Internet/Intranet приложения

Introduction to Linux. A Hands on Guide —

Система Linux является отличной платформой для предоставления сетевых сервисов. В этом разделе мы попытаемся дать краткий обзор наиболее распространенных сетевых серверов и приложений.

Типы серверов

Автономный сервер

Предоставляемые пользователям сервисы могут быть доступны двумя разными способами. Демон или служба могут работать или в автономном режиме, или зависеть от активации другого сервиса.

Сервисы сети, которые сильно и/или постоянно используются, как правило, работают в автономном режиме: они представляют собой независимые программы-демоны, которые работают всегда. Они чаще всего запускаются во время загрузки системы и ожидают запросы по специальным точкам соединениям или портам, которые они прослушивают. Когда приходит запрос, он обрабатывается, и прослушивание продолжается до следующего запроса. Веб-сервер представляет собой типичный пример: вы хотите, чтобы он был доступен 24 часа в сутки, поэтому при большой загрузке следует создать больше прослушивающих служб, обслуживающих одновременно работающих пользователей. Другими примерами являются большие архивы программного обеспечения или зеркала, которое должно обрабатывать тысячи FTP-запросов в день.

Примером автономного сетевого сервиса на домашнем компьютере может быть **named** (демон named), кэширующий сервер имен. У автономных сервисов есть собственные запущенные процессы, вы можете проверить это в любое время с помощью **ps**:

```
bob:~> ps auxw | grep named
named   908  0.0  1.0 14876 5108 ?   S  Mar14  0:07 named -u named
```

При этом существуют некоторые службы, которые вы можете использовать на вашем ПК, даже если не будет существовать запущенного для него серверного процесса. В качестве примера можно привести службы FTP, сервис безопасного копирования. У таких служб есть интернет-демон (**inetd**) прослушивающий их на месте.

(x)inetd

На вашем домашнем компьютере обычно немного спокойнее. У вас может быть небольшая сеть, и вы передаете файлы с одного ПК на другой время от времени, используя FTP или Samba (для связи с машинами MS Windows). В таких случаях все службы, которые вам нужны, запускаются время от времени, работать с ними все время было бы пустой тратой ресурсов. Поэтому в небольших установках вы обнаружите, что необходимые демоны зависит от центральной программы, которая прослушивает все порты служб, за которые она несет ответственность.

Этот супер-сервер, демон сетевых сервисов, запускается во время системной инициализации. Существует две распространенные реализации: **inetd** и **xinetd** (расширенный демон интернет-услуг). Тот или иной обычно запущен на каждой системе Linux:

```
bob:~> ps -ef | grep inet
root  926   1 0 Mar14 ?   00:00:00 xinetd-ipv6 -stayalive -reuse \
-pidfile /var/run/xinetd.pid
```

Службы, за которые отвечает интернет-демон, перечислены в его конфигурационном файле /etc/inetd.conf для **inetd** и в каталоге /etc/xinetd.d для **xinetd**. Обычно управляемые службы включают сервисы разделения

файлов и печати, SSH, FTP, telnet, конфигурационный демон Samba, опрашивающие и временные службы.

Как только запрос на соединение получен, центральный сервер запускает необходимую службу. Так в приведенном ниже примере, когда пользователь bob запускает FTP-сессию к локальному хосту, FTP-демон работает, пока сессия активна:

```
bob:~> ps auxw | grep ftp
bob     793  0.1  0.2  3960 1076 pts/6    S    16:44   0:00 ncftp localhost
ftp     794  0.7  0.5  5588 2608 ?        SN   16:44   0:00 ftpd:
localhost.localdomain: anonymous/bob@his.server.com: IDLE
```

Конечно, то же самое происходит при открытии соединения с удаленными хостами: либо демон отвечает непосредственно, либо удаленный **(x)inetd** запускает нужную вам службу, и останавливает ее, когда вы выходите.

Почта

Серверы

Sendmail является стандартной почтовой серверной программой или агентом почтовой пересылки для платформ UNIX. Она надежна, масштабируема и при правильной настройке соответствующего оборудования без проблем поддерживает тысячу пользователей. Более подробную информацию о настройке Sendmail входит в пакеты sendmail и sendmail-cf, вы можете прочитать файлы README и README.cf в /usr/share/doc/sendmail. Также полезны будут **man sendmail** и **man aliases**.

Qmail другой почтовый сервер, который приобретает все большую популярность, поскольку утверждается, что он более безопасен, чем Sendmail. Хотя Sendmail является монолитной программой, Qmail состоит из меньшего числа взаимодействующих частей программы, что может обеспечивать лучшую безопасность. Postfix и Exim являются примерами других почтовых серверов, которые набирают популярность.

Эти серверы обрабатывают списки рассылки, выполняют фильтрацию, сканируют на вирусы и делают многое другое. Бесплатные и коммерческие сканеры доступны для использования в Linux. Примеры программного обеспечения для рассылки - Mailman, Listserv, Majordomo и EZmlm. Посмотрите веб-страницу предпочитаемого вами антивирусного сканера для получения информации о клиентской и серверной поддержкеа на Linux. Amavis и Spamassassin свободные реализации сканеров на вирусы и спам.

Удаленные почтовые серверы

Наиболее популярных протоколы для доступа к почте удаленно POP3 и IMAP4. Оба, IMAP и POP, позволяет офлайновую работу, удаленный доступ к новым письмам, и оба зависят от сервера SMTP для отправки почты.

Хотя POP представляет собой простой протокол, легкий в реализации и поддерживает практически любой почтовый клиент, IMAP является предпочтительным, потому что:

- Он может управлять метками статуса сообщений.
- Он может как хранить, так и получать сообщения.
- Он может получать доступ и управлять несколькими почтовыми ящиками.
- Он поддерживает одновременное обновление и общие почтовые ящики.
- Он также подходит для доступа к сообщениям Usenet и другим документам.

- IMAP работает как в он-лайн так и офф-лайн.
- Он оптимизирован для работы в он-лайн, особенно по низкоскоростным ссылкам.

Mail-агенты пользователя

Есть много текстовых и графических почтовых клиентов, мы назовем несколько из наиболее распространенных. Выберите наилучший для вас.

Команда UNIX **mail** использовалась в течение многих лет еще до того, как появились сети. Она представляет собой простой интерфейс для отправки сообщений и небольших файлов другим пользователям, которые затем могут сохранить сообщение, перенаправить или ответить на него.

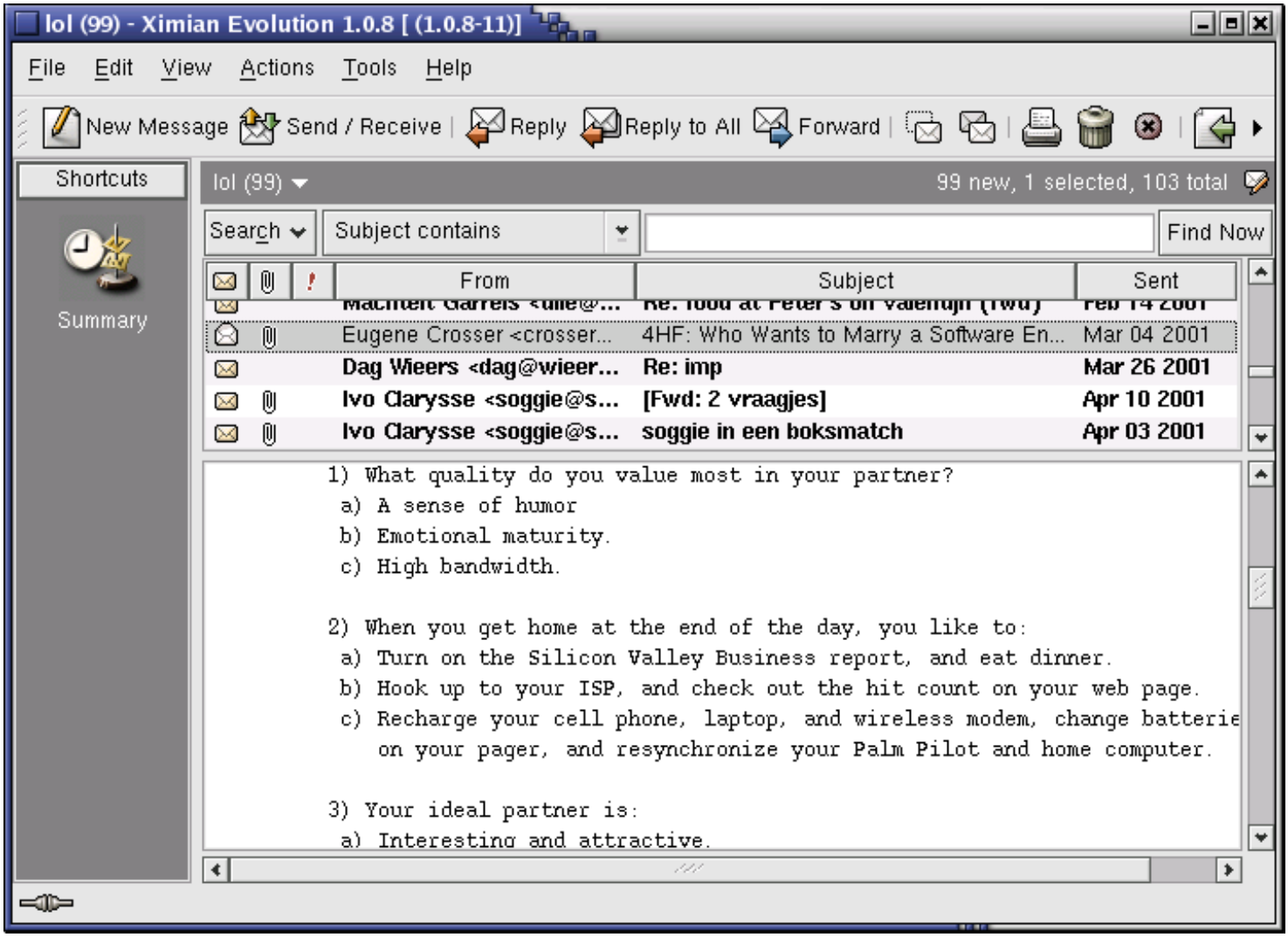
Хотя эта программа уже почти не используется в качестве клиента, она все еще может быть полезна, например, если на почту кому-то отправить вывод команды:

mail < cv.txt

Читатель почты **elm** — очень важное улучшение для **mail**, и также есть **pine** (Pine Is Not ELM). Более новый читатель почты mutt предлагает такие возможности как многопоточность.

Для тех пользователей, которые предпочитают графический интерфейс, существуют сотни вариантов. Наиболее популярны для новых пользователей Mozilla Mail/Thunderbird, который легко настраивается против спама, а Evolution представляет собой клон MS Outlook. Kmail пользуется популярностью среди пользователей KDE.

Рисунок 10.1. Почтовая программа Evolution



Существуют также десятки web-mail приложений, таких как Squirrelmail, Yahoo! mail, gmail от Google and Hotmail.

Большинство дистрибутивов Linux включают **fetchmail**, утилиту для восстановления и отправки почты. Она забирает почту с удаленных почтовых серверов (POP, IMAP и некоторые другие) и направляет ее к вашей локальной системе доставки. После этого можно обрабатывать полученную почту с помощью обычных почтовых клиентов. Эта программа может быть запущена в режиме демона для многократного опроса одного или нескольких систем в указанный промежуток времени. Информация и примеры использования можно найти в info-страницах; каталог /usr/share/doc/fetchmail[-] содержит полный список возможностей и FAQ для новичков.

Фильтр **procmail** может быть использован для фильтрации входящей почты, создания списков рассылки, предварительной обработки почты, выборочно пересылать электронную почту и многое другое. Сопутствующая ему программа **formail** среди прочего позволяет генерировать автоответы и разделение почтовых ящиков. Procmail работал в течение многих лет на машинах UNIX и Linux и является надежной системой, предназначенной для работы даже в наихудших обстоятельствах. Более подробную информацию можно найти в каталоге /usr/share/doc/procmail[-] и на справочных страницах.

Заметка о почтовом этикете

Некоторые люди в наши дни, возможно, думают, что сообщение электронной почты не должна быть слишком формальными. Конечно, все зависит от обстоятельств. Если вы пишете тому, кого вы не знаете, лучше держать на некотором расстоянии, как вы бы делали в традиционном письме. И не забывайте:

человек, которого вы не знаете, может оказаться как мужчиной, так и женщиной ...

Web

Веб-сервер Apache

Apache на сегодняшний день является самым популярным веб-сервером, используемом на более чем половине всех серверов в сети Интернет. Большинство Linux дистрибутивов включают Apache. Преимущества Apache заключаются в модульной разработке, поддержке SSL, стабильности и скорости. При наличии соответствующих аппаратных средств и их настройке он может выдерживать высокие нагрузки.

В системах Linux конфигурация сервера, как правило, делается в каталоге /etc/httpd. Наиболее важным конфигурационным файлом является httpd.conf; он пожалуй достаточно очевидный. Если вам нужна помощь, вы можете найти ее на man-странице **httpd** или на веб-сайт Apache.

Веб-броузеры

Для платформы Linux существует большое количество как бесплатных, так и коммерческих веб-браузеров. Netscape Navigator как единственный достойный вариант уже давно ушел в прошлое, так Mozilla/Firefox предлагает конкурентоспособную альтернативу, работающую также хорошо на многих других операционных системах, таких как MS Windows и MacOS X.

Amaya является W3C браузером. Opera - коммерческих браузер, компактный и быстрый. Многие менеджеры рабочего стола включают возможность просмотра веб-страниц в их файловом менеджере.

Популярными текстовыми браузерами являются **lynx** и **links**. Возможно, вам придется определить прокси-серверов в вашей оболочке, установив соответствующие переменные. Текстовые браузеры быстры и удобны, когда не установлена графическая среда, также как и при использовании в скриптах.

Proxy-серверы

Что такое proxy-сервер?

Компании и организации часто хотят, чтобы их пользователи могли использовать прокси-сервер. Особенно в среде с большим количеством пользователей прокси-сервер может позволить быструю загрузку веб-страниц. Прокси-сервер сохраняет веб-страницы. Когда пользователь запрашивает веб-страницу, которая уже была запрошена ранее, прокси-сервер выдаст пользователю ту страницы напрямую, без обращения к сети Интернет, что занимает меньше времени. Конечно, такой ход событий может быть приемлем, когда прокси-сервер делает быструю проверку и всегда предоставляет самую последнюю версию страницы. В некоторых средах использование прокси-сервера является обязательным, в других средах может быть выбор, следует ли его использовать.

Конфигурация прокси

Если у вас есть имя и порт прокси-сервера, это должно быть достаточно очевидным, чтобы скормить эту информацию в вашем браузере. Однако многие (командная строка) приложения зависят от переменных http_proxy и ftp_proxy для правильного функционирования. Для вашего удобства вы можете добавить строку подобной следующей в файл ~/.bashrc:

export http_proxy=http://username:password@proxy_server_name:port_number

Например:

export http_proxy=http://willy:Appelsi3ntj3@proxy:80

Если вам не требуется задавать имя пользователя и пароль, просто не вписывайте их до знака "@", а сам знак следует оставить.

File Transfer Protocol

FTP-серверы

На системах Linux сервер FTP, как правило, запускается от **xinetd**, используя сервер WU-ftpd, хотя FTP-сервер может быть настроен как автономный на системах с огромным FTP-трафиком. Смотрите упражнения.

Другие серверы FTP включают среди прочих vsftpd, Ncftpd и Proftpd.

Большинство дистрибутивов содержат пакет anonftp, который устанавливает анонимное дерево FTP-сервера и сопутствующие файлы конфигурации.

FTP-клиенты

Большинство дистрибутивов Linux включают **ncftp**, улучшенную версию распространенной команды UNIX **ftp**, которая может быть вам также знакома по командной строки Windows. Программа **ncftp** предлагает дополнительные функции, такие как лучший и понятный интерфейс пользователя, завершение имени файла, добавление и возобновление функций, закладки, управление сессиями и т.д.:

```
thomas:~> ncftp blob
NcFTP 3.0.3 (April 15, 2001) by Mike Gleason (ncftp@ncftp.com).
Connecting to blob...
blob.some.net FTP server (Version wu-2.6.1-20) ready.
Logging in...
Guest login ok, access restrictions apply.
Logged in to blob.
ncftp / > help
Commands may be abbreviated.  'help showall' shows hidden and
unsupported commands.
'help ' gives a brief description of .

ascii      cat      help      lpage     open     quote    site
bgget      cd       jobs      lpwd      page     rename   type
bgput      chmod    lcd       lrename   pdir     rhelp    umask
bgstart    close    lchmod    lrm       pls      rm       version
binary     debug    lls       lrmdir    put      rmdir
bookmark   dir      lmkdir    ls        pwd      set
bookmarks  get      lookup    mkdir     quit     show
ncftp / >
```

Отличную помощь с большим количеством примеров можно найти в документации. И опять же, существует

ряд приложений с GUI.

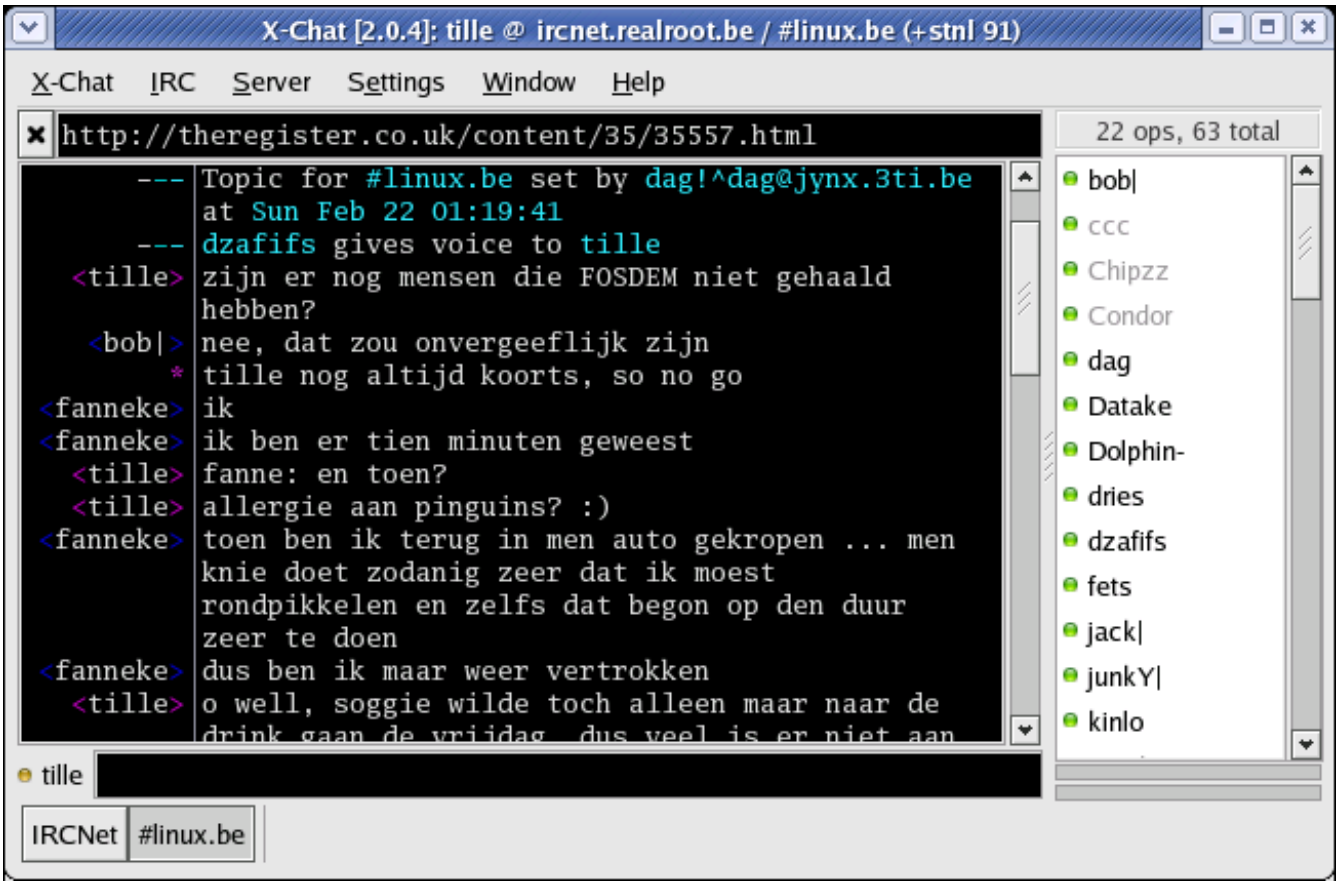
FTP не безопасен!
Не используйте протокол передачи файлов для неанонимного логина, если вы не уверены в том, что вы делаете. Ваше имя пользователя и пароль могут быть захвачены злонамеренными пользователями сети! Вместо этого используйте защищенное FTP; программа sftp поставляется с пакетом Secure SHell, см. Раздел "Безопасное удаленное копирование".

Чаты и конференции

В каждом дистрибутиве доступны различные клиенты и системы, которые заменяют старый IRC-стиль текстовых чатов. Вот короткий и неполный перечень наиболее популярных программ:

- **pidgin**: мульти-протокольный клиент обмена мгновенными сообщениями под Linux, Windows и Mac, совместимый с MSN Messenger, ICQ, IRC и многими другими; смотрите info-страницы или сайт этой программы.
- **xchat**: IRC клиент для X Window System:

Рисунок 10.2. X-Chat



- **aMSN**: клон MSN.
- **Konversation, kopete, KVIrc** и многие другие К-инструменты из пакета KDE.
- **gnomemeeting**: программы видеоконференции для UNIX (в настоящее время Ekiga).
- **jabber**: программа для обмена мгновенными сообщениями с открытым исходным кодом, совместимая с ICQ, AIM, Yahoo, MSN, IRC, SMTP и другими.

- **psi**: клиент для jabber.
- **skype**: программа для совершения телефоноподобных звонков через Интернет другим пользователям Skype. Skype является бесплатным, но не открытым.
- **Gizmo**: бесплатный (но не открытый) телефон для вашего компьютера

.

Службы новостей

Запуск сервера Usenet предполагает наличие специальных знаний и тонкой настройки.

Есть несколько интересных новостей в comp.* иерархии, которые можно получить с помощью различных текстовых и графических клиентов. Многие почтовые клиенты также поддерживают просмотр новостей, проверьте вашу программу или обратитесь к локальным зеркалам открытого программного обеспечения за такими текстовыми клиентами, как **tin, slrnn** и **mutt**, или загрузите Mozilla или любой другой графический клиент.

Deja.com хранит архив всех новостей с возможностью поиска, его работа обеспечивается Google. Это очень мощный инструмент для получения помощи: есть большие шансы, что кто-то уже сталкивался с вашей проблемой, нашел решение и отправил его в одну из групп новостей.

Система доменных имен

Все эти приложения пользуются службой DNS для сопоставления IP-адреса с именами хоста и обратно. Сервер DNS не знает всех IP-адресов в мире, но он может отправлять запросы на другие сервера DNS для поиска неизвестного адреса. Большинство систем UNIX могут запускать **named**, который является частью пакета BIND (Berkeley Internet Name Domain) распространяемого Internet Software Consortium. Он может работать как автономный кэширующий *nameserver*, что часто делается на системах Linux для ускорения доступа к сети.

Ваш основной клиентский файл конфигурации — это /etc/resolv.conf, который определяет порядок, в котором происходит соединение с Domain Name Servers:

```
search somewhere.org
nameserver 192.168.42.1
nameserver 193.74.208.137
```

Более подробную информацию можно найти в info-страницах для **named**, в файлах /usr/share/doc/bind[-] и на домашней странице проекта Bind. DNS HOWTO описывает использование BIND в качестве сервера DNS.

DHCP

DHCP - это протокол динамической настройки хоста, который постепенно заменяет старый добрый **bootp** в больших средах. Он используется для контроля жизненно важных параметров сетей, таких как IP-адресов и серверов имен хостов. DHCP имеет обратную совместимость с **bootp**. Для настройки сервера следует прочитать HOWTO.

DHCP клиентских машин, как правило, конфигурируется с помощью графического интерфейса, который настраивает dhcpcd, клиентский демон DHCP. Проверьте системную документацию, если вам нужно настроить компьютер в качестве клиента DHCP.

Службы проверки подлинности

Традиционный

Традиционно пользователи проходят проверку подлинности локально, используя информацию, хранящуюся в /etc/passwd и /etc/shadow на каждой системе. Но даже при использовании сетевой службы для проверки подлинности, локальные файлы всегда будут присутствовать для настройки системных аккаунтов для административного использования, такого как учетная запись суперпользователя, аккаунты демонов и других программ и целей.

Эти файлы часто являются первыми кандидатами на изучение хакерами, поэтому убедитесь, что права доступа и владельцы установлены именно так, как должно быть:

```
bob:~> ls -l /etc/passwd /etc/shadow
-rw-r--r--    1 root     root         1803 Mar 10 13:08 /etc/passwd
-r--------    1 root     root         1116 Mar 10 13:08 /etc/shadow
```

PAM

Linux может использовать PAM, Pluggable Authentication Module, гибкий метод аутентификации в UNIX. Преимущества PAM:

- Обычная схема аутентификации, которая может использоваться с большим рядом приложений.
- PAM может быть реализован с различными приложениями без необходимости перекомпиляции приложений специально для поддержки PAM.
- Большая гибкость и контроль над аутентификациями администратора и разработчика приложений.
- Разработчикам приложений не требуется создавать свои программы, чтобы использовать определенную схему аутентификации. Вместо этого, они могут сосредоточиться исключительно на деталях своей программы.

Каталог /etc/pam.d содержит конфигурационные файлы PAM (используемые /etc/pam.conf). Каждое приложение или служба имеют свой собственный файл. Каждая строка в файле состоит из четырех элементов:
Модуль:

- auth: обеспечивает проверку подлинности (запрашивая и сверяя пароль) и устанавливает учетные данные, такие как членство в группе или тикеты Kerberos.
- account: проверяет для уверенности, что доступ разрешен для пользователя (время учетной записи еще не истекло, что пользователь может войти в это время суток, и так далее).
- password: используется для установки паролей.
- session: используется после того как пользователь прошел аутентификацию. Этот модуль выполняет дополнительные задачи, которые необходимы для разрешения доступа (например, монтирование домашней директории пользователя или имеющегося в наличии почтового ящика).

Порядок, в котором расставляются модули такой, чтобы могли использоваться несколько модулей; это является важным.

- *Control Flags*: сообщают PAM, какие действия предпринять в случае ошибки или удачи. Значения могут быть required, requisite, sufficient или optional.
- *Module Path*: путь к подключаемый модуль, который будет использоваться, обычно в /lib/security.

- *Arguments*: информация для модулей.

Файлы теневых паролей автоматически обнаруживаются PAM.
Более подробную информацию можно найти на man-страницах pam или домашней странице проекта.

LDAP

Lightweight Directory Access Protocol является клиент-серверной системой для доступа к глобальной или локальной службам каталогов по сети. В Linux используется реализация OpenLDAP. Она включает в себя **slapd**, отдельный сервер; **slurpd**, автономный сервер репликации LDAP; библиотеки, реализующие протокол LDAP, а также ряд утилит, инструментов и примеров клиентов.

Главным преимуществом использования LDAP является консолидация отдельных видов информации в пределах организации. Например, все разные списки пользователей в вашей организации могут быть объединены в один LDAP-каталог. Этот каталог может быть запрошен любым LDAP-приложением, которому необходима эта информация. Она также может быть доступна пользователям.

Другие преимущества LDAP или X.500 Lite заключаются в его простоте реализации (по сравнению с X.500) и его четко определенном Application Programming Interface (API), что означает, что число LDAP-приложений с поддержкой LDAP и шлюзов увеличиться в будущем.

Однако есть и другая сторона. Если вы хотите использовать LDAP, вам потребуются LDAP-приложения или возможность использования LDAP-шлюзов. Хотя использование LDAP должно только возрастать, в настоящее время существует не так много LDAP-приложений для Linux. Кроме того, хотя LDAP поддерживает некоторый контроль доступа, он не обладает теми многими функциями безопасности, как X.500.

Так как LDAP является открытым и настраиваемый протоколом, он может быть использован для хранения практически любой информации, относящейся к конкретной организационной структуре. Типичными примерами являются поиск почтовых адресов, централизованная аутентификация в сочетании с PAM, телефонные справочники и базы данных конфигурации компьютера.

Посмотрите вашу системную информацию и man-страницы связанных команд, таких как **ldapmodify** и **ldapsearch** для выяснения деталей. Более подробную информацию можно найти в LDAP Linux HOWTO, в котором обсуждается установка, настройка, запуск и обслуживание LDAP-сервера на Linux.

Удаленное выполнение приложений

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Введение

Существует несколько различных способов выполнения команд или запуска программ на удаленном компьютере, а также получение результатов их работы на рабочей станции, будь то текст или графика. Связь может быть безопасной или нет. Хотя советуют использовать защищенное соединение, а не транспортировать пароль по сети в открытом виде, мы обсудим некоторые практические приложения, использующие более старые (небезопасные) механизмы, т.к. они все еще полезны в современной сетевой среде, например, для устранения неполадок или работы экзотических программ.

Rsh, rlogin и telnet

Команды для удаленного входа и выполнения команд **rlogin** и **rsh** пришли из UNIX. Хотя они редко используются, т.к. это небезопасно, они по-прежнему поставляются почти с каждым дистрибутивом для обратной совместимости с UNIX-программами.

Кроме того, **telnet** по-прежнему широко используется системными и сетевыми администраторами. Telnet является одним из наиболее мощных инструментов для удаленного доступа к файлам и удаленного администрирования, позволяющий устанавливать связь из любой точки мира через Интернет. В сочетании с X-сервером, удаленные графические приложения могут быть отображены на локальной машине. В результате стираются границы между работой на локальном компьютере и с помощью удаленной машины.

По причине того, что всё соединение не зашифровано, позволяя **telnet** соединяться, вы берете на себя высокие риски, связанные с безопасностью. Для нормального удаленного выполнения программ рекомендуется использовать Secure Shell, или **ssh**. Далее мы будем обсуждать безопасный метод связи.

Однако, во многих случаях **telnet** до сих пор используется. Ниже приведены некоторые примеры, в которых почтовый сервер и веб-сервер тестируются на ответы.

Проверка того, что почтовый сервер работает:

```
[jimmy@blob ~] telnet mailserver 25
Trying 192.168.42.1...
Connected to mailserver.
Escape character is '^]'.
220 m1.some.net ESMTP Sendmail 8.11.6/8.11.6; 200302281626
ehlo some.net
250-m1.some.net Hello blob.some.net [10.0.0.1], pleased to meet you
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-SIZE
250-DSN
250-ONEX
250-ETRN
250-XUSR
250 HELP
mail from: jimmy@some.net
250 2.1.0 jimmy@some.net... Sender ok
rcpt to: davy@some.net
250 2.1.5 davy@some.net... Recipient ok
data
354 Enter mail, end with "." on a line by itself
test
.
250 2.0.0 g2MA1R619237 Message accepted for delivery
quit
221 2.0.0 m1.some.net closing connection
Connection closed by foreign host.
```

Проверка того, что веб-сервер отвечает на основные запросы:

```
[jimmy@blob ~] telnet www.some.net 80
Trying 64.39.151.23...
Connected to www.some.net.
```

```
Escape character is '^]'.
HEAD / ;HTTP/1.1

HTTP/1.1 200 OK
Date: Fri, 22 Mar 2002 10:05:14 GMT
Server: Apache/1.3.22 (UNIX) (Red-Hat/Linux)
 mod_ssl/2.8.5 OpenSSL/0.9.6
 DAV/1.0.2 PHP/4.0.6 mod_perl/1.24_01
Last-Modified: Fri, 04 Jan 2002 08:21:00 GMT
ETag: "70061-68-3c3565ec"
Accept-Ranges: bytes
Content-Length: 104
Connection: close
Content-Type: text/html

Connection closed by foreign host.

[jimmy@blob ~]
```

Это совершенно безопасно, потому что вы нигде не вводите имя пользователя и/или пароль для получения данных, которые вам требуются, поэтому никто не сможет стащить какую-нибудь важную информацию с кабеля.

X Window System

Особенности X

Как мы уже объясняли в Главе 7, система X Window поставляется с X-сервером, который обслуживает графические приложения клиентов.

Важно понимать различие между X-сервероми и клиентскими X-приложениями. X-сервер управляет дисплеем напрямую и несет ответственность за все входящие и выходящие данные, которые проходят через клавиатуру, мышь и монитор. X-клиент, со своей стороны, не имеет доступа к устройствам ввода и вывода напрямую. Он связывается с X-сервером, который обрабатывает ввод и вывод. X-клиент выполняет реальную работу, такую как вычисления значений, обеспечивает работу приложений и т.д. X-сервер только открывает окна для обработки ввода и вывода для конкретного клиента.

При обычной работе (в графическом режиме), каждая рабочая станция Linux является X-сервером для себя самой. Все приложения, которые вы используете (например, Gimp, окно терминала, браузер, офисное приложение, инструмент проигрывания CD и т.д.) являются клиентами вашего X-сервера. Получается, что сервер и клиент работают на одной машине.

Такая клиент/серверная особенность системы X делает ее идеальной средой для удаленного выполнения приложений и программ. Поскольку процесс реально выполняется на удаленной машине, требуется мало мощности процессора на локальном хосте. Такие машины, которые работают только как сервера для X, называют X-терминалами и когда-то они были очень популярны.

Telnet и X

Если бы вы захотели использовать **telnet** для отображения графических приложений, работающих на удаленном компьютере, необходимо сначала дать удаленному компьютеру доступ к вашему дисплею

(теперь уже X-серверу!) используя команду **xhost**, набрав ее как в примере ниже в окне терминала на вашем локальном компьютере:

```
davy:~> xhost +remote.machine.com
```

После этого, надо подключиться к удаленному хосту и сообщить ему об отображении графики на локальной машине, установив переменную окружения:

```
[davy@remote ~] export DISPLAY="local.host.com:0.0"
```

После завершения этого шага, любое приложение, запущенное в этом окне терминала будет отображаться на вашем локальном компьютере, используя удаленные ресурсы для вычислений, но ваши местные графические ресурсы (Ваш X-сервер) для отображения приложений.

Эта процедура предполагает, что у вас на компьютере уже установлены какие-нибудь X-сервера (XFree86, X.org, Exceed, Cygwin), и вам есть с помощью чего отображать изображения. Архитектура и операционная система клиентского компьютера не важны, если они позволяют запускать на нем X-сервер.

Помните, что отображение окна терминала с удаленной машины также считается прорисовкой изображения.

Пакет SSH

Введение

Большинство систем UNIX и Linux в настоящее время запускают Secure SHell, чтобы избежать рисков для безопасности, которые давал **telnet**. На большинстве систем Linux будет работать версия OpenSSH, открытая реализация протокола SSH, обеспечивающая безопасную зашифрованную связь между непроверенными хостами в ненадежной сети. X-соединения при обычной установке автоматически передаются, но произвольные TCP/IP порты могут также быть переданы с использованием защищенного канала.

Клиент **ssh** подключается и регистрируется на указанном имени хоста. Пользователь должен предоставить данные о себе на удаленной машине, как указано в файле sshd_config, который обычно можно найти в /etc/ssh. Содержимое конфигурационного файла говорит само за себя, и по умолчанию включает наиболее используемы возможности. Если вам нужна помощь, вы сможете найти ее в man-страницах **sshd**.

Когда личность пользователя становится идентифицированной сервером, он либо выполняет данную команду, либо заходит на компьютер и предоставляет пользователю обычный shell на удаленной машине. Вся связь с удаленной командой или оболочкой будет автоматически зашифрована.

Сессия заканчивается, когда команда или оболочка на удаленной машине завершает работу, и все X11 и TCP/IP соединения оказываются закрытыми.

При подключении к хосту в первый раз, используя любую из программ, которые включены в коллекцию SSH, необходимо установить подлинность этого хоста и подтвердить, что вы хотите подключиться:

```
lenny ~> ssh blob
The authenticity of host 'blob (10.0.0.1)' can't be established.
RSA fingerprint is 18:30:50:46:ac:98:3c:93:1a:56:35:09:8d:97:e3:1d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'blob,192.168.30.2' (RSA) to the list of
known hosts.
```

```
Last login: Sat Dec 28 13:29:19 2002 from octarine
This space for rent.

lenny is in ~
```

Важно, чтобы вы ввели "yes", из трех символов, а не только "y". Все это изменяет ваш файл ~/.ssh/known_hosts, см. Раздел "Проверка подлинности сервера".

Если вы просто хотите проверить что-то на удаленной машине, а затем вернуться обратно к оболочке на локальном компьютере, вы можете передать команды, которые вы хотите выполнить удаленно, в качестве аргументов **ssh**:

```
lenny ~> ssh blob who
jenny@blob's password:
root     tty2         Jul 24 07:19
lena  tty3         Jul 23 22:24
lena     0:        Jul 25 22:03

lenny ~> uname -n
magrat.example.com
```

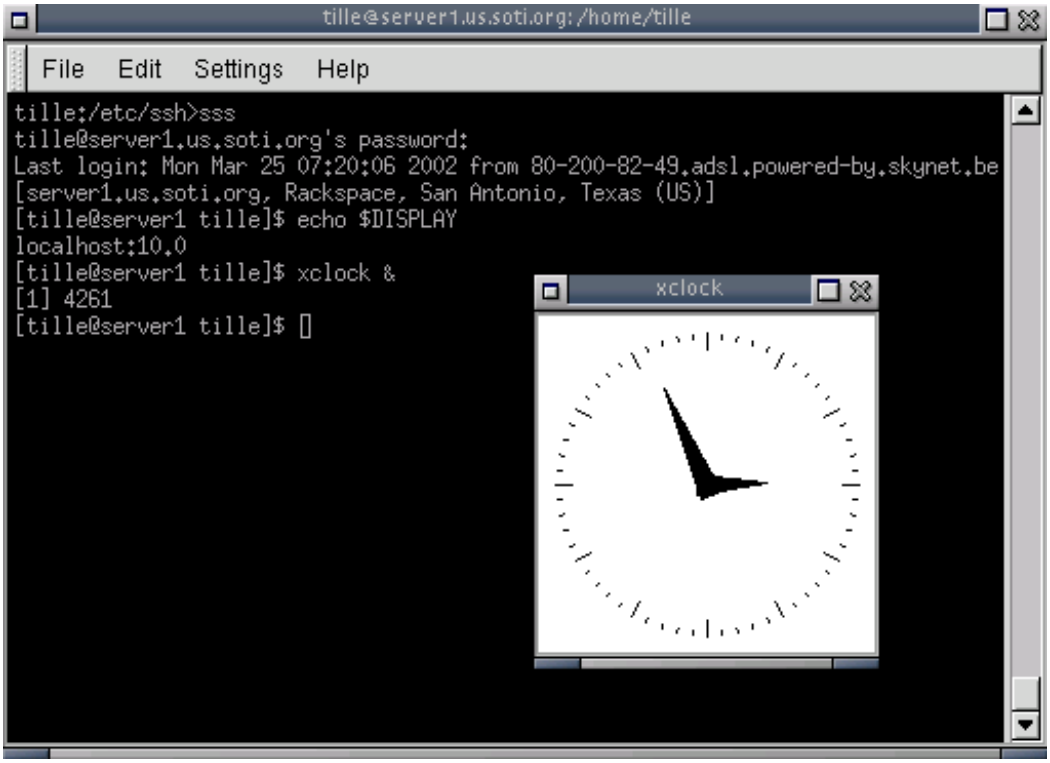
Пересылка X11 и TCP

Если параметру X11Forwarding присвоено значение yes на требуемом компьютере и пользователь использует X-приложения, устанавливается переменная среды DISPLAY, подключение к X11-дисплею автоматически пересылаются на удаленный сервер таким образом, что любая X11-программа, запускаемая из оболочки, будет проходить через шифрованный канал, и связь с реальным X-сервером будет создана из локальной машине. Пользователю. не надо вручную устанавливать DISPLAY. Отправка X11-соединений может быть сконфигурирована в командной строке или в конфигурационном файле **sshd**.

Значение для DISPLAY, установленное **ssh**, будет указывать на сервер, но с номером дисплея больше нуля. Это нормально и происходит потому, что ssh создает proxy X сервер на сервере (который запускает X-приложение клиента) для пересылки соединений через зашифрованный канал.

Все это делается автоматически, так что, когда вы набираете название графического приложения, оно отобразиться на вашем локальном компьютере, а не на удаленном хосте. В примере мы используем **xclock**, так как это небольшая программа, которая обычно установлена и идеально подходит для тестирования:

Рисунок 10.3. SSH X11 перенаправление



SSH также автоматически настроит X-полномочия на сервере. Для этой цели будут сгенерированы случайные авторизационные куки, SSH сохранит их на сервере и проверит, что какое-либо соединение несет эти куки и заменит их реальными, когда соединение будет установлено. Реальные аутентификационные куки никогда не передается на сервер (и никакие куки не отправляются так как есть).

Пересылка произвольных TCP/IP-соединений через защищенный канал может быть определена либо в командной строке, либо в файле конфигурации.

X-сервер

Эта процедура предполагает, что у вас работает X-сервер на стороне клиента, где вы отображаете приложение от удаленного хоста. Клиент может иметь иную архитектуру и операционную систему, чем удаленный хост в том случае, если он может запустить X-сервер, например, Cygwin (который реализует X.org сервер для клиентов MS Windows и др.) или Exceed, должна быть возможность настроить удаленное соединение с любым компьютером Linux или Unix.

Проверка подлинности сервера

Клиент/серверная система **ssh** автоматически поддерживает и проверяет базу данных, содержащую идентификационную информация для всех хостов, которые когда-либо использовались. Ключи хостов хранятся в $HOME/.ssh/known_hosts, который находится в домашнем каталоге пользователя. Кроме того, автоматически проверяется файл /etc/ssh/ssh_known_hosts на наличие известных хостов. Любые новые хосты, автоматически добавляются в файл пользователя. Если идентификация хоста когда-либо изменяется, **ssh** предупреждает об этом и отключает аутентификацию по паролю, чтобы какой-нибуль троянский конь не получил пароля пользователя. Другая цель этого механизма заключается в предотвращении "человек-в-середине" атаки, которые могут быть использованы для обхода шифрования. В средах, где необходим высокий уровень безопасности, **sshd** даже может быть настроен для предотвращения входов на машины, чьи ключи хоста были изменены или неизвестны.

Безопасное удаленное копирование

Пакет SSH предоставляет **scp** в качестве безопасной альтернативой команды **rcp**, которая была популярна, когда только она и существовала. **scp** использует **ssh** для передачи данных, использует ту же самую проверку подлинности и обеспечивает такую же безопасность как **ssh**. В отличие от **rcp**, **scp** будет запрашивать пароль или парольную фразу, если они необходимы для аутентификации:

```
lenny /var/tmp> scp Schedule.sdc.gz blob:/var/tmp/
lenny@blob's password:
Schedule.sdc.gz  100% |*****************************| 100 KB 00:00

lenny /var/tmp>
```

Любое имя файла может содержать хост и спецификации пользователя для указания, что файл должен быть скопирован на/с этого хоста. Копии между двумя удаленными хостами не допускаются. См. info-страницы для получения дополнительной информации.

Если вы предпочитаете использовать FTP-подобный интерфейс, используйте **sftp**:

```
lenny /var/tmp> sftp blob
Connecting to blob...
lenny@blob's password:

sftp> cd /var/tmp

sftp> get Sch*
Fetching /var/tmp/Schedule.sdc.gz to Schedule.sdc.gz

sftp> bye

lenny /var/tmp>
```

Безопасное копирование или FTP GUI
Еще не чувствуйте себя комфортно в командной строке? Попробуйте возможности Konqueror для безопасного удаленного копирования, или установите Putty.

Проверка подлинности ключей

Команда **ssh-keygen** создает, управляет и преобразует ключи аутентификации для **ssh**. Она может создавать ключи RSA для использования протоколом SSH версии 1 и ключи RSA или DSA для использования протоколом SSH версии 2.

Обычно каждый пользователь, желающий использовать SSH с аутентификацией RSA или DSA, выполняет команду единожды для создания ключей аутентификации в $HOME/.ssh/identity, id_dsa или id_rsa. Кроме того, системный администратор может использовать ее для генерации ключей хоста системы.

Обычно эта программа генерирует ключ и опрашивает файл, в котором он хранится. Открытый ключ хранится в файле с тем же именем, но добавляется расширение *.pub*. Также программа запрашивает пароль. Пароль может быть пустым, что указывает на то, что его нет (ключи хостов должно быть с пустым паролем), или это может быть строка произвольной длины.
Не существует способа восстановить потерянный пароль. Если фраза утеряна или забыта, новый ключ должен быть создан и скопирован в соответствующие открытые ключи.

Мы изучим ключи SSH в упражнениях. Вся информация может быть найдена в man- или info-страницах.

VNC

VNC или Virtual Network Computing, на самом деле система удаленного дисплея, которая позволяет просматривать рабочую среду не только на локальной машине, на которой он запущен, но и из любой точки Интернет и с самых разных машин и архитектур, в том числе MS Windows и некоторых дистрибутивов UNIX. Можно, например, запустить MS Word на компьютере с Windows NT и отобразить результ на вашем рабочем столе Linux. VNC обеспечивает серверы также как клиентов, поэтому противоположное также работает, что можно использовать для отображения Linux программ для клиентов Windows. VNC, вероятно, самый простой способ получить X соединения на ПК. Следующие характеристики показывают, чем VNC отличается от обычного X-сервера или коммерческих реализаций:

- Ни одно состояние не хранится на стороне зрителя: вы можете покинуть свой стол и возобновить работу с другого компьютера с того места, где вы остановились. Когда вы запускаете X-сервер, а также происходят сбои или перезагрузки ПК, то все удаленные приложения, которые вы запустили, прекращают работать. А вот VNC они продолжат свое выполнение.
- Это маленькая и простая программа, не требующая установки; если потребуется, может быть запущена с дискеты.
- Платформенно независима с Java-клиентом, работает в виртуальной среде, которая поддерживает X.
- Разделяема: один рабочий стол может отображаться на множество зрителей.
- Свободна.

Более подробную информацию можно найти на man-страницах (**man vncviewer**) или на веб-сайте VNC.

Протокол rdesktop

Для того, чтобы облегчить управление хостами MS Windows, новые дистрибутивы Linux поддерживают Remote Desktop Protocol (RDP), который реализован в клиенте **rdesktop**. Протокол используется в ряде продуктов Microsoft, включая Windows NT Terminal Server, Windows 2000 Server, Windows XP и Windows 2003 Server.

Руководство **man rdesktop** предоставляет больше информации.

Cygwin

Cygwin обеспечивает существенную функциональность UNIX на системах MS Windows. Помимо предоставления утилит командной строки UNIX и графических приложений, она также может быть использована для отображения рабочего стола Linux на компьютере с MS Windows, с помощью удаленного X. Из оболочки Bash Cygwin, введите команду

/usr/X11R6/bin/XWin.exe -query your_linux_machine_name_or_IP

Соединение по умолчанию запрещено. Вы должны изменить конфигурацию X Display Manager (XDM) и, возможно, конфигурацию X Font Server (XFS), чтобы включить этот тип соединения, где вы получите экран входа на удаленной машине. В зависимости от вашего менеджер рабочего стола (Gnome, KDE, другие), вам, возможно, придется изменить некоторые настройки и там.

Если вам не нужно отображать весь десктоп, вы можете использовать SSH в Cygwin, так же, как описано в Разделе "Пакет SSH", без всякой суеты по редактированию конфигурационных файлов.

Безопасность

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Введение

Как только компьютер оказывается подключен к сети, то становится возможным его злонамеренное использование независимо от операционной системы, установленной на нем. Много слов написано на эту тему, и это завело бы нас слишком далеко, если бы мы стали обсуждать вопросы безопасности в деталях. Есть, однако, несколько довольно логичных вещей, которые может сделать даже начинающий пользователь, чтобы получить достаточно безопасную систему, потому что большинство взломов является результатом невежественности или беспечности пользователей.

Возможно, вы не знаете, относится ли это к вашему домашнему или рабочему компьютеру. Вопросы, которые вам следует задать себе, должны быть следующими:

- Вы хотите иметь контроль над собственной системой?
- Хотите ли вы (невольно) участвовать в преступной деятельности?
- Вы хотите, чтобы ваше оборудование использовалось кем-то другим?
- Вы хотите иметь возможность потерять подключение к Интернет?
- Вам нравится восстанавливать вашу систему каждый раз, когда она была взломана?
- Вы хотите рисковать потерять личные или иные данные?

Предположим, что вы всего этого не хотите; мы распишем несколько шагов, которые вы можете предпринять для защиты вашей машины.

Службы

Цель состоит в том, чтобы работало как можно меньше служб. Если количество портов, открытых для внешнего мира, было сведено к минимуму, то это облегчит слежение. Если службы не могут быть отключены в локальной сети, постарайтесь хотя бы отключить их для внешних подключений.

Эмпирическое правило таково, что если вы не узнаете конкретную службу, вы, вероятно, не нуждаетесь в ней. Также имейте в виду, что некоторые службы на самом деле не предназначены для использования через Интернет. Не полагайтесь на то, что запущено все, что должно быть запущено, проверьте, какие сервисы прослушиваются на TCP-портах, используя команду **netstat**:

```
[elly@mars ~] netstat -l | grep tcp
tcp     0      0 *:32769                 *:*       LISTEN
tcp     0      0 *:32771                 *:*       LISTEN
tcp     0      0 *:printer               *:*       LISTEN
tcp     0      0 *:kerberos_master       *:*       LISTEN
tcp     0      0 *:sunrpc                *:*       LISTEN
tcp     0      0 *:6001                  *:*       LISTEN
tcp     0      0 *:785                   *:*       LISTEN
tcp     0      0 localhost.localdom:smtp *:*       LISTEN
tcp     0      0 *:ftp                   *:*       LISTEN
tcp     0      0 *:ssh                   *:*       LISTEN
tcp     0      0 ::1:x11-ssh-offset      *:*       LISTEN
```

Чего следует избегать:

- **exec, rlogin, rsh** и **telnet** просто на всякий случай.
- X11 на серверах.
- Отсутствие lp, если ни один принтер физически подключен.
- Нет MS Windows хостов в сети, не требуется Samba.
- Не допускайте FTP, если сервер FTP не требуется.
- Не позволяйте NFS и NIS через Интернет, отключите все сопутствующие службы на автономной установке.
- Не запускать MTA, если вы не находитесь на почтовом сервере.
- ...

Остановите запущенные службы с помощью команды **chkconfig**, скриптов init или путем редактирования конфигурационного файла **(x)inetd**.

Регулярное обновление

Способность быстро адаптироваться в постоянно меняющейся среде позволяет Linux процветать. Но это также создает вероятность того, что обновления безопасности выпускаются после того, как вы установили новую версию дистрибутива; поэтому первое, что нужно сделать (и это относится к любой ОС) после установки — получить и установить все доступные обновления.

Некоторые обновления могут потребоваться новые конфигурационные файлы, а старые файлы могут быть заменены. Проверьте документацию и убедитесь, что все работает нормально после обновления.

Большинство Linux дистрибутивов предлагают сервисы списков рассылки для уведомления об обновлении безопасности и инструменты для применения обновлений к системе.

Обновление представляет собой непрерывный процесс, поэтому происходит почти ежедневно.

Брандмауэры и политики доступа

Что такое брандмауэр (фаервол)?

В предыдущем разделе мы уже упоминали возможности брандмауэра (межсетевого экрана) в Linux. Хотя управление им является одной из задач вашего сетевого администратора, вы должны знать некоторые вещи о фаерволах.

Брандмауэр - не четко определенный термин, он может означать все, что действует как защитный барьер между нами и внешним миром, как правило, Интернет. Брандмауэр может быть встроенным в систему или отдельным приложением, которое обеспечивает подобную функциональность. Или это может быть сочетание компонентов, в том числе различные комбинации аппаратного и программного обеспечения. Межсетевые экраны построены из "правил", которые используются для определения того, чему позволено входить и/или выходить на данной системе или в сети.

После отключения ненужных служб, мы должны ограничить запущенные службы таким образом, чтобы позволить им только минимально необходимые соединения. Прекрасным примером является работа на дому: должно быть разрешено только строго определенное соединение между вашими офисом и дом, соединения с другими машинами в сети Интернет должны быть заблокированы.

Пакетные фильтры

Первой линией обороны является пакетный фильтр, который может просматривать содержимое IP-пакетов и принимать решения исходя из их содержания. Самым популярным является пакет Netfilter, предоставляющий команду **iptables**, следующее поколение пакетных фильтров для Linux.

Один из самых примечательных усовершенствований в новых ядрах является способность проверять сохранность состояния, при этом сообщается не только о том, что находится внутри пакета, но и обнаруживается, если пакет принадлежит или связан с новым или существующим соединением.

Shoreline Firewall или Shorewall представляет собой интерфейс для доступа к стандартной функциональности брандмауэра в Linux.

Обертки TCP

Упаковка TCP обеспечивает во многом такие же результаты, как и фильтры пакетов, но работает по-другому. Обертка принимает попытки подключения, а затем проверяет файлы конфигурации и решает, следует ли принять или отклонить запрос на соединение. Она контролирует соединения на уровне приложений, а не на сетевом уровне.

TCP-обертки, как правило, используются с **xinetd** для предоставления имени хоста и основанном на IP-адресе управлении доступа. Кроме того, эти инструменты включают регистрацию и использование возможностей управления, которые легко настроить.

Преимущества оболочек TCP в том, что подключившийся клиент не знает, что используются обертки, и что они работают отдельно от приложений, которые они защищают.

На хосте доступ контролируется в файлах hosts.allow и hosts.deny. Более подробную информацию можно найти в файлах документации обертки TCP в /usr/share/doc/tcp_wrappers[-/] или /usr/share/doc/tcp и в man-страницах для файлов контроля доступа хоста, там же содержатся примеры.

Прокси

Прокси-серверы могут выполнять различные задачи, и не все из них имеют непосредственное отношение к безопасности. Однако эти службы являются промежуточным звеном, которое делает прокси прекрасным местом для применения политики управления доступом, ограничения прямого соединения через брандмауэр и контроля того, как сеть после прохождения через прокси выглядит в Интернет.

Обычно в сочетании с пакетным фильтром, но иногда и сами по себе, прокси-серверы обеспечивают дополнительный уровень контроля.

Доступ к отдельным приложениям

Некоторые серверы могут иметь свои собственные средства контроля доступа. Например, Samba, X Window, Bind, Apache и CUPS. Для каждой службы используются определенные файлы конфигурации.

Файлы журналов

Во всяком случае UNIX-путь фиксации всех видов деятельности на всех видах файлов подтверждает, что "это что-то делает". Конечно, файлы журналов должны регулярно проверяться вручную или автоматически.

Межсетевые экраны и другие средства контроля доступа, как правило, создают огромное количество log-файлов, так что проблема заключается в том, чтобы попытаться найти в них только ненормальную активность.

Обнаружение вторжений

Системы обнаружения вторжений (Intrusion Detection Systems - IDS) предназначены поймать то, что, возможно, пропустил брандмауэр. Они могут также быть предназначены для улавливания активной попытки взлома или для обнаружения уже свершившегося вторжения. В последнем случае уже слишком поздно, чтобы предотвратить любое повреждение, но по крайней мере мы осознали проблему на ранней стадии. Существуют два основных типа IDS: защищающие сети и защищающие индивидуальные хосты.

IDS хостов представляют собой утилиты, которые следят за файловой системой на предмет изменений. Системные файлы, которые так или иначе изменились, но не должны были меняться, привлекают к себе внимание. Это говорит о том, что кто-то вошел в систему с правами администратора, и вносит системные изменения.

Обнаруженное вторжений обрабатывается системой, которая видит весь трафик, который пропускает брандмауэр.

Дополнительные советы

Некоторые общеизвестные факты, которые следует иметь в виду:

- Не входите в систему как администратор. Разработчики UNIX более двух десятилетий назад придумали команду **su** для дополнительной безопасности.
- Прямой доступ root всегда опасен и допускает человеческие ошибки, и не важно входите ли вы в систему как root или используете команду su. Поэтому вместо использования **su**, лучше использовать **sudo** для выполнения тех команд, которые требуют привилегий, и возвращаться потом к своей среде.
- Относитесь к паролям серьезно. Используйте теневые пароли. Регулярно изменяйте пароли.
- Старайтесь всегда использовать SSH или SSL. Избегайте **telnet**, FTP, E-mail клиентов и других клиентских программ, которые отправляют незашифрованные пароли по сети. Важно не только обеспечить безопасность вашего компьютера, но и безопасность ваших паролей.
- Ограничьте ресурсы с помощью **quota** и/или **ulimit**.
- Проверяйте подлинность нового ПО, берите его из надежного места/сайта. Проверьте новые пакеты перед установкой.
- При использовании непостоянного подключения к Интернет, отключайте его, как только вам оно больше не нужно.
- Запускайте личные службы на незанятых портах, а не тех, которые ожидают возможных атак хакеров.
- Знайте вашу систему. Спустя какое-то время вы сможете почти чувствовать, когда что-то происходит.

Меня хакнули?

Как вы можете об этом догадаться? Это контрольный список подозрительных событий:

- Таинственные открытые порты, странные процессы.
- Системные утилиты (часто используемые команды) ведут себя странно.
- Войти в систему проблемно.

- Необъяснимое использование пропускной способности сети.
- Поврежденные или отсутствующие файлы журналов, странное поведение демона syslog.
- Интерфейсы в необычных режимах.
- Неожиданно изменение конфигурационных файлов.
- Странные записи в файлы истории оболочки.
- Неизвестные временные файлы.

Восстановление после вторжения

Итак, сохраняйте спокойствие. Затем выполните следующие действия в указанном порядке:

- Отключите компьютер от сети.
- Постарайтесь выяснить, насколько это возможно то, каким образом безопасность была нарушена.
- Создайте резервную копию важных несистемных данных. Если это возможно, сравните эти данные с существующими резервными копиями, сделанными до того как системы была взломана, для обеспечения целостности данных.
- Переустановите систему.
- Используйте новые пароли.
- Восстановление резервные копии данных.
- Примените все доступные обновления.
- Пересмотреть систему: заблокируйте ненужные службы, проверьте правила брандмауэра и другие политики доступа.
- Подключитесь к сети.

Резюме

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Linux и сети развиваются совместно. Ядро Linux включает поддержку всех распространенных и самых необычных сетевых протоколов. Стандартные сетевые инструменты UNIX предоставляются с каждым дистрибутивом. Наряду с этим, большинство дистрибутивов предлагают инструменты для легкой установки и управления сетью.

Linux известен как стабильная платформа для запуска различных сервисов Интернет, количество его интернет-программы бесконечно. Как и UNIX, Linux может быть также использоваться и управляться в удаленном режиме, используя одно из нескольких решений для удаленного выполнения программ.

Мы кратко затронули тему безопасности. Linux является идеальной системой с межсетевым экраном, легким и дешевым, а также может использоваться для других сетевых функций, таких как маршрутизаторы и прокси-серверы.

Увеличение безопасности сети в основном осуществляется путем применения регулярных обновлений и здравого смысла.

Вот краткий обзор команд, связанных с работой в сети:

Таблица 10.2. Новые команды из главы 10: Сеть

Команда	Значение
ftp	Передача файлов на другой компьютер (небезопасная).
host	Получение информации о компьютерах в сети.
ifconfig	Вывод на экран IP-адресов.
ip	Вывод на экран IP-адресов.
netstat	Отображает информацию о маршрутизации и сетевую статистику.
ping	Отправка запросов на другие узлы сети и получение ответов.
rdesktop	Отображение информации о компьютерах с MS Windows на Linux-машине.
route	Показать информацию о маршрутизации.
scp	Безопасное копирование файлов на/с другого компьютера.
sftp	Безопасное ftp-соединение.
ssh	Создание зашифрованного соединения с другим хостом.
ssh-keygen	Генерация ключей аутентификации для Secure SHell.
telnet	Создание небезопасного соединения с другим хостом.
tracepath/traceroute	Вывод маршрута, которым пакеты следуют на другой узел.
whois	Получение информации об имени домена.
xclock	Приложение часов X Window, удобно для тестирования удаленного дисплея.
xhost	X Window инструмент контроля доступом.

Упражнения

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Основы сетей

- Отобразите сетевую информацию для вашей рабочей станции: IP-адрес, маршруты, имена серверов.
- Предположим, что отсутствует DNS. Что бы вы сделали, чтобы получать доступ к машине соседа без постоянного ввода IP-адреса?

- Как бы вы надолго сохранили информацию о прокси для браузера текстового режима, такого как **links**?
- Какие серверы управляются доменом redhat.com?
- Отправить E-mail в вашу локальную учетную запись. Попробуйте два различных способа отправлять и читать ее. Как вы можете убедиться, что письмо действительно пришло?
- Ваша машина принимает анонимные FTP-соединения? Как вы используете программу **ncftp** для проверки вашего имени пользователя и пароля?
- На компьютере работает веб-сервер? Если нет, запустите его. Проверьте файлы журналов.

Удаленные подключения

- С вашего компьютера отобразите графическое приложение, например, **xclock** на экране вашего соседа. Должны быть созданы необходимые учетные записи. Используйте безопасное соединение!
- Настройте ключи SSH так, чтобы вы смогли подключиться к компьютеру соседа без необходимости ввода пароля.
- Сделайте резервную копию домашнего каталога в /var/tmp на вашем ближайшем "сервере бэкапов", используя **scp**. Выполните архивирование и сжатие перед началом передачи данных! Подключение к удаленному хосту с помощью **ssh**, распакуйте архив и переместите один файл обратно на свой компьютер, используя **sftp**.

Безопасность

- Составьте список открытых (прослушиваемых) портов на вашем компьютере.
- Предположим, вы хотите запустить веб-сервер. Какие службы вы бы отключили? Как бы вы это сделали?
- Установить доступные обновления.
- Как вы можете узнать, кто подключен к вашей системе?
- Создайте задание, которое каждый месяц будет напоминать вам изменить пароль и пароль администратора также.

Глава 11. Звук и видео

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Аннотация

В этой главе рассматривается решение следующих задачь (кратко, т.к. область звука и видео очень широка):

- Настройка звуковой карты
- Воспроизведение компакт-дисков и их копирование
- Воспроизведение музыкальных файлов
- Регулирование громкости
- Видео и телевидение
- Запись звука

Базовая информация по звуку

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Установка

Скорее всего, ваша система уже установлена с аудио-драйверами, и конфигурация была выполнена во время установки. Также, если вам когда-нибудь понадобится заменить аудио-оборудование, то большинство систем предоставляют инструменты, позволяющие легко выполнить установку и настройку устройства. Большинство имеющихся в настоящее время plug-and-play звуковых карт должны обнаруживаться автоматически. Если во время выполнения специальной утилиты настройки вы услышите звук, нажмите OK, и все будет выполнено за вас.

Если ваша карта не определяется автоматически, может быть предложен список звуковых карт и/или свойств звуковых карт, из которых можно выбирать. После этого вы должны предоставить правильный порт ввода/вывода, установки IRQ и DMA. Информацию об этих параметрах можно найти в документации к звуковой карте. Если на вашем компьютере также установлена MS Windows, то эту информацию можно найти в ее Панели управления.

Драйверы и архитектуры

Существуют два основных типа звуковой архитектуры: более древняя Open Sound System или OSS, которая работает в каждой UNIX-подобной системе, а также новая Advanced Sound Linux Architecture или ALSA, которая имеет лучшую поддержку для Linux, как следует из названия. ALSA также имеет больше функций и позволяет ускорить разработку драйверов. Мы сосредоточимся здесь на системе ALSA.

Сегодня поддерживаются почти все основные аудио-чипсеты. Лишь некоторые производители высоко профессиональных решений и обычных карт отказываются документировать свои спецификации чипсета, и они не поддерживаются.

Настройка системы, установленной с ALSA, осуществляется с помощью инструмента **alsaconf**. Кроме того, дистрибутивы обычно предоставляют свои собственные инструменты для настройки звуковой карты; эти инструменты могут даже включать старый и новый способ настройки звуковых устройств.

Воспроизведение звука и видео

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Проигрывание и копирование CD

Пакет *cdp* поставляется с большинством дистрибутивов и предоставляет программу **cdp** или **cdplay**, текстовый проигрыватель компакт-дисков. Десктопные менеджеры обычно включают в себя графический инструмент, например, проигрыватель **gnome-cd** в Gnome, который может быть запущен из меню.

Надо понимать, что есть разница между аудио-дисками и CD с данными. Вам не надо монтировать аудио-CD к файловой системе, чтобы послушать его. Это связано с тем, что данные на таких CD не являются данными файловой системы Linux; они сразу доступны и отправляются в канал аудио выхода проигрывателем компакт-дисков. Если же ваш CD является диском с данными, который содержит файлы .mp3, вам придется сначала примонтировать его к файловой системе и только потом слушать музыку с

помощью одной из программ, которые мы перечислим ниже.

Утилита **cdparanoia** из одноименного пакета читает аудио непосредственно в виде данных с компакт-диска без аналогового преобразования и записывает их в файл или канал в различных форматах, в том числе .wav, как, вероятно, самый популярный. Различные инструменты для конвертирования в другие форматы, например, .mp3, поставляются с большинством дистрибутивов или загружаются как отдельные пакеты. Проект GNU предоставляет несколько проигрывателей CD, программ для копирования, кодирования и управления звуком.

Очень легко создавать аудио-CD с помощью **kaudiocreator**, входящего в состав пакета KDE. Информация о нем есть в справочном центре KDE.

Проигрывание музыкальных файлов

Файлы mp3

Популярный формат .mp3 поддерживается Linux. Большинство дистрибутивов включают несколько программ, которые могут воспроизводить такие файлы. Например, XMMS, который представлен на скриншоте ниже, является одним из наиболее распространенных, отчасти потому, что он похож на проигрыватель Windows.

Рисунок 11.1. mp3-проигрыватель XMMS

Также очень популярны для воспроизведения музыки AmaroK — приложение KDE, которое постепенно набирает популярность, и MPlayer, который также может воспроизводить фильмы.

Ограничения

Некоторые дистрибутивы не позволяют воспроизводить MP3 без изменения конфигурации, это из-за лицензионных ограничений на MP3-инструменты. Возможно, потребуется установить дополнительное программное обеспечение для проигрывания музыки.

В текстовом режиме вы можете использовать команду **mplayer**:

```
[tille@octarine ~]$ mplayer /opt/mp3/oriental/*.mp3
MPlayer 1.0pre7-RPM-3.4.2 (C) 2000-2005 MPlayer Team
CPU: Advanced Micro Devices Duron Spitfire (Family: 6, Stepping: 1)
Detected cache-line size is 64 bytes
CPUflags:  MMX: 1 MMX2: 1 3DNow: 1 3DNow2: 1 SSE: 0 SSE2: 0
Playing /opt/oldopt/mp3/oriental/Mazika_Diana-Krozon_Super-Star_Ensani-Ma-
Bansak.mp3.
Cache fill:  1.17% (98304 bytes)    Audio file detected.
Clip info:
Title: Ensani-Ma-Bansak.mp3
Artist: Diana-Krozon
Album: Super-Star
Year:
Comment:
Genre: Unknown
==========================================================================
```

```
Opening audio decoder: [mp3lib] MPEG layer-2, layer-3
mpg123: Can't rewind stream by 450 bits!
AUDIO: 44100 Hz, 2 ch, s16le, 160.0 kbit/11.34% (ratio: 20000->176400)
Selected audio codec: [mp3] afm:mp3lib (mp3lib MPEG layer-2, layer-3)
==========================================================================
Checking audio filter chain for 44100Hz/2ch/s16le -> 44100Hz/2ch/s16le...
AF_pre: 44100Hz/2ch/s16le
AO: [oss] 44100Hz 2ch s16le (2 bps)
Building audio filter chain for 44100Hz/2ch/s16le -> 44100Hz/2ch/s16le...
Video: no video
Starting playback...
A: 227.8 (03:23:.1) 1.8% 12%
```

Другие форматы

Это увело бы нас слишком далеко, если обсуждать все возможные форматы аудио и способы их воспроизведения. Неполный обзор распространенного ПО для воспроизведения звука и работы с ним:

- Ogg Vorbis: свободный формат аудио. Он был разработан из-за того, что MP3 оказался запатентован.
- **realplay**
- SoX или Sound eXchange: конвертер звука, поставляется с программой **play**. Проигрывает .wav, .ogg и других форматы, в том числе бинарные форматы.
- Playmidi: MIDI-плеер.
- AlsaPlayer: из проекта Advanced Linux Sound Architecture.
- **mplayer**: играет все, что угодно, в том числе mp3-файлы.
- **hxplay**: поддерживает RealAudio и RealVideo, mp3, mp4-аудио, Flash, wav и другие.
- **rhythmbox**: основанный на GStreamer фреймворк, может играть то, что поддерживается в GStreamer, который, как утверждается, способен играть все.

Посмотрите вашу системную документацию и man-страницы для конкретных программ, там найдутся подробные разъяснения о том, как все это использовать.

 **У меня в системе нет этих приложений!**
Большинство программ и приложений, упомянутых выше, являются дополнительным программным обеспечением. Вполне возможно, что они не установлены в системе по умолчанию, но вы можете их найти в вашем дистрибутиве в качестве дополнительных пакетов. Также может оказаться, что приложения, которое вы ищете, нет в вашем дистрибутиве вообще. В этом случае, вам необходимо скачать его с сайта программы.

Регулятор громкости

aumix и **alsamixer** — это два распространенных текстовых инструмента для управления аудиосистемой. Для переключения настроек используются клавиши стрелок. **alsamixer** имеет графический интерфейс при запуске из меню Gnome или доступен как **gnome-alsamixer** из командной строки. Программа **kmix** делает то же самое в KDE.

Независимо оттого, что вы собираетесь слушать, помните, что могут быть и другие люди, которые не

заинтересованы в том, чтобы слушать вас или ваш компьютер. Постарайтесь быть вежливыми, особенно в офисе. Используйте большие наушники. Так будет лучше для ваших ушей и вызывать меньше раздражения у ваших коллег.

Запись

Здесь также доступны различные инструменты, позволяющие записывать голос и музыку. Для записи голоса можно использовать **arecord** в командной строке:

```
alexey@russia:~> arecord /var/tmp/myvoice.wav
Recording WAVE '/var/tmp/myvoice.wav' : Unsigned 8 bit, Rate 8000 Hz, Mono
Aborted by signal Interrups...
```

"Interrupt" (прерывание) означает, что приложение поймало Ctrl + C. Проиграть образец можно с помощью простой команды **play**.

Это хороший тест, который можно выполнить перед тестированием приложений, которым требуется голосовой ввод, таким как передача голоса по IP (VoIP). Имейте в виду, что микрофонный вход должен быть активирован. Если вы не слышите собственный голос, проверьте настройки звука. Часто случается, что микрофон отключен или находится на очень низкой громкости. Это может легко отрегулировать с помощью **alsamixer** или графического интерфейса для звуковой системы вашего дистрибутива.

В KDE можно запустить утилиту **krec**, Gnome предоставляет **gnome-sound-recorder**.

Просмотр видео, каналов и телевидения

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Для этих целей доступны различные проигрыватели:

- **xine**: свободно-распространяемый проигрыватель видео
- **ogle**: DVD-плеер
- **okle**: KDE версия **ogle**
- **mplayer**: Movie Player для Linux
- **totem**: воспроизведение аудио и видео файлов, аудио CD, VCD и DVD
- **realplay**: от RealNetworks.
- **hxplay**: альтернативный Real
- **kaffeine**: мультимедийный проигрыватель для KDE3.

Скорее всего, вы найдете один из них в вашем графическими меню.
Имейте в виду, что всех необходимых кодеков для просмотра различных типов видео может не быть в вашей системе по умолчанию. Вы можете загрузить w32codecs и libdvdcss.

Для просмотра телевидения можно выбрать среди следующих инструментов, которые, как и многие другие, подходят для просмотра и захвата ТВ-передач, видео и другого:

- **tvtime**: великолепная программа с широкими возможностями управления, взаимодействия с телетекстом, режимами для фильмов и многим другим.

- **zapping**: программа для просмотра телепередач для Gnome.
- **xawtv**: X11 ТВ-вьювер.

Интернет-телефония

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Что это?

Интернет-телефония — это чаще всего передача голоса по IP (VoIP) или цифровая телефония, которая позволяет участникам обмениваться голосовыми данными по сети. Разница заключается в том, что потоки данных через сеть общего пользования Интернет отличаются от традиционной телефонии, которая использует выделенную сеть линий для передачи голоса. Хотя при особых обстоятельствах две сети могут быть между собой соединены, сейчас это не практикуется как стандарт. Другими словами, возможность связаться с людьми, которые используют обычный телефон, существует. Если это возможно, то скорее всего вам придется платить за подписку.

В настоящее время существуют различные приложения, доступные для бесплатного скачивания, как свободные, так и проприетарные. Но есть ряд недостатков телефонии через Интернет. Такая система ненадежна, она может работать медленно или в ней может проходить много шума по связи, и поэтому не может использоваться для замены традиционной телефонии. Хотя некоторые провайдеры принимают меры предосторожности, нет никакой гарантии, что вы дозвонитесь, куда вам надо.

Большинство приложений в настоящее время не используют шифрование, так что надо понимать, что кому-то достаточно легко подслушивать ваши разговоры. Если безопасность важна для вас, читайте документацию, которая поставляется с вашим клиентом IP-телефонии. Кроме того, если вы используете брандмауэр, он должен быть настроен на разрешение входящих соединений из любого места, так что использование VoIP также включает в себя принятие рисков на уровне безопасности сайта.

Что вам нужно?

На стороне сервера

Прежде всего, необходим провайдер предоставляющий такую услугу. Данный сервис может быть включен в традиционную телефонию, может быть платным или нет. Среди других есть SIPphone, Vonage, Lingo, AOL TotalTalk, также доступны технологии провайдеров, которые предлагают так называемый "полный телефонный сервис". Услуги интернет-телефонии предлагают Skype, SIP Broker, Google и многие другие.

Клиентская сторона

На стороне клиента, приложения, которые можно использовать, зависят от конфигурации сети. Если у вас есть прямое подключение к Интернету, то не будет никаких проблем, при условии, что вы знаете, к какому серверу можете подключиться, и, как правило, должны быть имя пользователя и пароль для аутентификации в сервисе.

Если вы находитесь за межсетевым экраном, который делает Network Address Translation (NAT), некоторые службы могут не работать, так как они будут видеть только IP-адрес брандмауэра, а не адрес вашего компьютера, который вполне может быть немаршрутизируемым через Интернет , например, когда вы находитесь в корпоративной сети, и ваш IP-адрес начинается с 10., 192.168. или иной немаршрутизируемого

префикса подсети. Это зависит от протокола, который используется приложением.

Также блокирующим фактором может быть доступная пропускная способность: некоторые приложения, оптимизированы для низкого потребления пропускной способности, в то время как другие могут требовать высокую пропускную способность соединения. Это зависит от способа кодирования-декодирования, который используется приложением.

Самое известное приложение — это Skype, который имеет интерфейс, напоминающий обмен мгновенными сообщениями и X-Lite, бесплатную версию программного телефона XTen, который выглядит как мобильный телефон. Хотя эти программы доступны для бесплатной загрузки и популярны, они не свободны: они используют закрытые протоколы и/или доступны только в виде исполняемых файлов.

 **Клиентское оборудование**
Даже если на вашем компьютере, особенно ноутбуке, есть встроенный микрофон, результат будет намного лучше, если вы подключите гарнитуру. Если у вас есть выбор, выборайте в пользу USB-гарнитуры, т.к. она функционирует независимо от имеющегося аудио-оборудования. Используйте alsamixer для настройки входного и выходного уровня звука.

Резюме

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

На платформе GNU/Linux доступна полная мультимедийность. Поддерживается широкой спектр устройств, таких как звуковые карты, ТВ-карты, наушники, микрофоны, проигрыватели CD и DVD. Список доступных приложений является бесконечным, поэтому ниже мы ограничились аудио-командами.

Таблица 11.1. Новые команды в главе 11, связанные с аудио

Команда	Значение
alsaconf	Конфигурация звуковой системы ALSA
alsamixer	Настройка выходных уровней драйвера ALSA
arecord	Запись образцов звука
aumix	Инструменты аудио-миксера
cdp	Проигрывание аудио-CD
cdparanoia	Копирование компакт-дисков
cdplay	Проигрывание аудио-CD
gnome-alsamixer	Интерфейс ALSA для Gnome
gnome-cd	Интерфейс для проигрывания аудио-CD в Gnome
gnome-sound-recorder	Интерфейс для записи образцов звука в Gnome

kaudiocreator	Интерфейс KDE для создания аудио-CD
kmix	Интерфейс KDE для звуковых настроек
krec	Интерфейс для записи образцов звука в KDE
mplayer	Проигрыватель мультимедиа
play	Инструмент командной строки для проигрывания образцов звука

Упражнения

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

1. В меню Gnome или KDE откройте панель настройки звука. Убедитесь, что колонки или наушники подключены к системе, и найдите уровень выходного сигнала, который вас устраивает. Убедитесь в том, что ваша система ALSA совместима, что вы используете соответствующую панель.
2. Если у вас есть микрофон, попробуйте записать образец вашего собственного голоса. Убедитесь, что входная громкость не слишком высокая, так как это приведет к высоким тонам, когда вы общаетесь с другими, или к передаче фонового шума. В командной строке, вы можете даже попытаться использовать **arecord** и **aplay** для записи и воспроизведения звука.
3. Найдите звуковые файлы в вашей системе и попробуйте послушать их.
4. Вставьте компакт-диск и попытаться воспроизвести его.
5. Найдите партнера в чате и настройте программу IP-телефонии. (Возможно, вам потребуется сначала ее инсталлировать.)
6. Вы можете слушать интернет-радио?
7. Если у вас есть DVD-плеер и фильм на диске DVD, попробуйте посмотреть его.

Приложение B. Сравнение команд DOS и Linux

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

В этом приложении команды DOS ставятся в соответствие с некоторыми командами Linux.

Это может помочь сориентироваться новым пользователям, имеющим опыт работы в Windows. Следует помнить, что команды Linux, как правило, имеют ряд опций. Следует прочитать info- или man-страницы команды.

Таблица B.1. Обзор команд DOS/Linux

Команды DOS	Команды Linux
command /?	man command или command --help
cd	cd

chdir	pwd
cls	clear
copy	cp
date	date
del	rm
dir	ls
echo	echo
edit	vim (или другой редактор)
exit	exit
fc	diff
find	grep
format	mke2fs или mformat
mem	free
mkdir	mkdir
more	more или less
move	mv
ren	mv
time	date

Приложение C. Особенности Shell

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Здесь дается обзор базовых функций оболочки (одинаковых для всех оболочек) и некоторых специфичных особенностей.

C1. Базовые функции

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

Следующие функции являются стандартными для каждой оболочки. Обратите внимание, что команды **stop,**

suspend, jobs, bg и **fg** доступны только в системах, которые поддерживают соответствующее управление.

Таблица C.1. Базовые функции shell

Команда	Значение
>	Перенаправление вывода
>>	Добавление в файл
<	Перенаправление ввода
<<	"Здесь" документ (перенаправление ввода)
|	Канальный вывод
&	Запустить процесс в фоновом режиме
;	Разделение команд
*	Соответствует любому(ым) символу(ам) в имени файла
?	Соответствует одному символу в имени файла
[ ]	Соответствует любому символу, заключенному в скобки
( )	Выполнить в дочерней оболочке
` `	Заменить вывод заключенной в кавычки командой
" "	Частичное цитирование (допустимы переменные и командные расширения)
' '	Полное цитирование (нет расширений)
\	Цитировать следующий символ
$var	Использовать значение переменной
$$	id процесса
$0	Имя команды
$n	n-ый аргумент (n от 0 до 9)
$*	Все аргументы как простое слово
#	Начать комментарий
bg	Фоновое исполнение

break	Прервать цикл
cd	Изменить директорию
continue	Прервать очередной цикл программы
echo	Отобразить вывод
eval	Оценить аргументы
exec	Выполнить новую оболочку
fg	Выполнить на переднем плане
jobs	Показать активные задания
kill	Прервать запущенные задания
newgrp	Изменения в новой группе
shift	Сдвиг позиционных параметров
stop	Приостановить фоновое задание
suspend	Приостановить задание
time	Время команды
umask	Установить или посмотреть разрешения на файлы
unset	Удалить переменную или определения функций
wait	Ожидать пока выполняется фоновое задание

C2. Отличительные особенности оболочек

Introduction to Linux. A Hands on Guide —
Введение в Linux. Руководство по работе

В таблице ниже показаны основные различия между обычной shell (**sh**), Bourne Again SHell (**bash**), Korn shell (**ksh**) и C shell (**csh**).

 **Совместимость оболочек**
Так как Bourne Again SHell является расширением sh, все команды sh будут также работать в bash - но не наоборот. Сам по себе bash включает много возможностей, и, как показывает таблица ниже, многие функции включены из других оболочек.
Поскольку оболочка Turbo C является надстройкой csh, все команды csh будут работать в tcsh, но не наоборот.

Таблица C.2. Различные свойства оболочек

sh	bash	ksh	csh	Значение/Действие
$	$	$	%	Приглашение к вводу по умолчанию
	>|	>|	>!	Принудительное перенаправление
> file 2>&1	&> file или > file 2>&1	> file 2>&1	>& file	Перенаправление стандартных потоков вывода и ошибок в file
	{ }		{ }	Расширить элементы в список
`command`	`command` или $(command)	$(command)	`command`	Заменить вывод заключенной в обратные кавычки командой
$HOME	$HOME	$HOME	$home	Домашний каталог
	~	~	~	Символ домашнего каталога
	~+, ~-, dirs	~+, ~-	=-, =N	Стек доступа к каталогам
var=value	var=value	var=value	set var=value	Назначение переменной
export var	export VAR=value	export VAR=value	setenv var val	Установка переменной окружения
	${nnnn}	${nn}		Возможность ссылаться на более, чем 9 аргументов
"$@"	"$@"	"$@"		Все аргументы в виде отдельных слов
$#	$#	$#	$#argv	Количество аргументов
$?	$?	$?	$status	Код возврата от последней выполненной команды
$!	$!	$!		PID последнего фонового процесса
$-	$-	$-		Текущие опции
. file	source file or . file	. file	source file	Читает команды в файле

	alias x='y'	alias x=y	alias x y	Имя x устанавливается для команды y
case	case	case	switch or case	Выбор альтернативы
done	done	done	end	Конец заявленного цикла
esac	esac	esac	endsw	Окончание case или switch
exit n	exit n	exit n	exit (expr)	Выход со статусом
for/do	for/do	for/do	foreach	Цикл по переменным
	set -f , set -o nullglob|dotglob|nocaseglob|noglob		noglob	Игнорировать замену символов при генерации имени файла
hash	hash	alias -t	hashstat	Отображение хэшированных команд (для которых установлены псевдонимы)
hash cmds	hash cmds	alias -t cmds	rehash	Вспомнить команды
hash -r	hash -r		unhash	Забыть команды
	history	history	history	Список предыдущих команд
	ArrowUp+Enter or !!	r	!!	Повторить предыдущую команду
	!str	r str	!str	Повторить предыдущую команду, которая начинается со “str”.
	!cmd:s/x/y/	r x=y cmd	!cmd:s/x/y/	Заменить “x” на “y” в самой недавней команде, которая начинается с “cmd”.
if [$i -eq 5]	if [$i -eq 5]	if ((i==5))	if ((i==5))	Образец проверки условия
fi	fi	fi	endif	Конец инструкции if
ulimit	ulimit	ulimit	limit	Установка ограничений на ресурсы
pwd	pwd	pwd	dirs	Вывод рабочей директории

read	read	read	$<	Чтение с терминала
trap 2	trap 2	trap 2	onintr	Игнорировать прерывания
	unalias	unalias	unalias	Удалить псевдонимы
until	until	until		Начало цикла until
while/do	while/do	while/do	while	Начало цикла while

У Bourne Again SHell есть намного больше свойств, не перечисленных здесь. Но эта таблица может дать вам представление о том, как эта оболочка включает в себя все полезные идеи из других оболочек: в колонке **bash** нет пустот. Больше информации о специфических свойствах Bash можно получить из info-страниц Bash в разделе "Bash Features".

Как минимум вам надо прочитать руководство к вашей командной оболочке. Предпочтительно было бы выбрать info bash. Bash, будучи оболочкой GNU, также прост для начинающих./TD