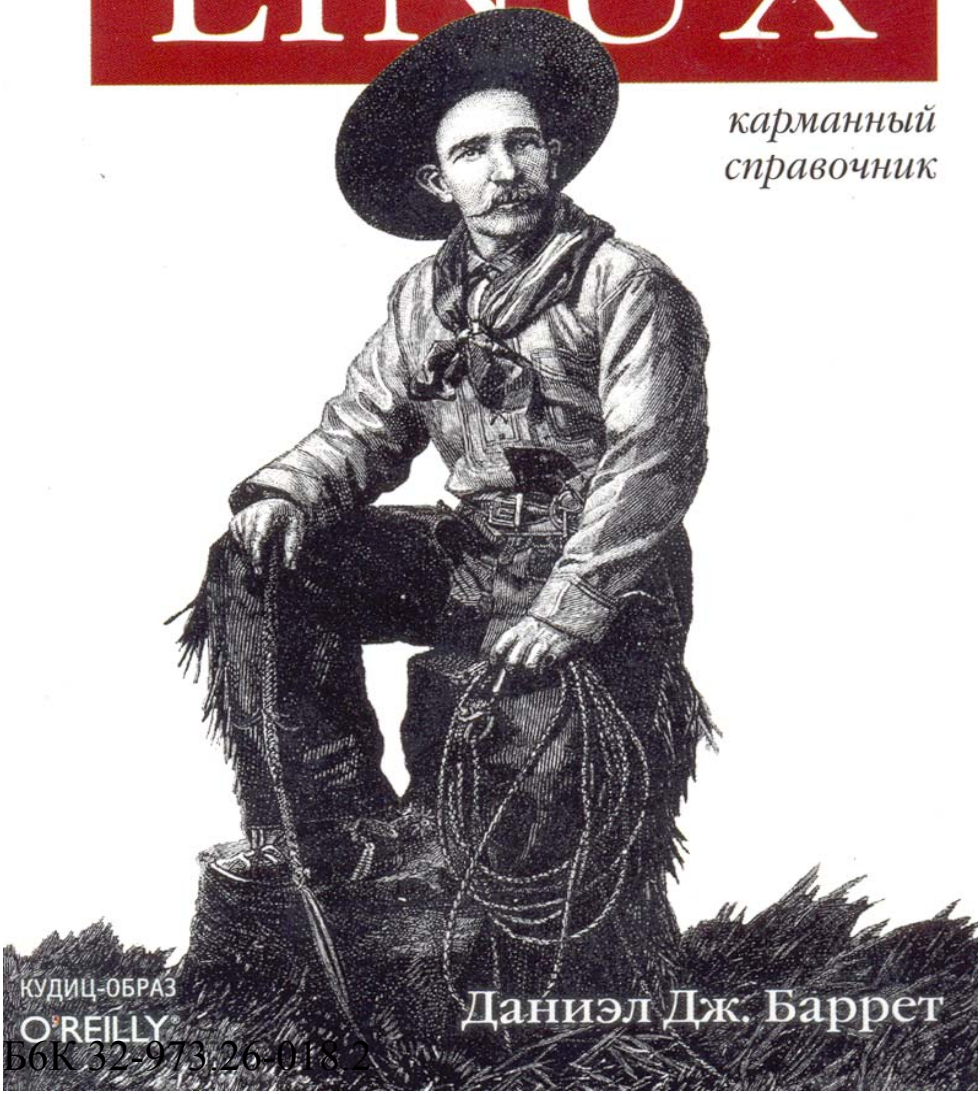
Основные команды

включает Fedora Linux

LINUX

карманный справочник



Даниэл Дж. Баррет

КУДИЦ-ОБРАЗ
O'REILLY
ББК 32.973.26-018.2

Д. Дж. Еаррет

Linux: основные команды. Карманный справочник /

Пер. с англ. - М.: КУДИЦ-ОБРАЗ, 2005. - 288 с.

Карманный справочник Шил — это одновременно и краткий справочник для

опытных пользователей, и руководство для новичков.

Книга предваряется общими сведениями об операционной системе файлы и директории, командный процессор shell, система X Window. Затем приводится подробное описание команд и программ Linux, покрывающих почти весь спектр прикладных задач, возникающих пире.

Данюл Дж. Баррет

Linux: основные команды. Карманный справочник

Учебно-справочное издпиие

Научный редактор П. С. Воликов. инструктор учебный «ИД КУДИЦ-ОБРАЗ»

Тел.: 333-82-11; E-mail: ok@kudiis.ra;]irtp://boob.kudi 119049. Москва, Ленинский пр-т., д. 4, стр. 1А

Формат 70х90/32. Бумага офсетная Пе'1.1т, офсетная Усл. пет. п. 10,53, Тираж 3000. Зака, :$_5$36

Содержание

Linux.
Карманный справочник

Добро пожаловать в Linux! Если вы новичок в мире Linux, то эта книга может послужить вам быстрым введением в операционную систему Linux в целом и в дистрибутив Fedora Linux в частности, а также руководством по часто используемым командам. Если у вас есть опыт использования Linux, вы легко можете пропустить вводный материал.

О чем эта книга?

Эта книга - краткое руководство, а не всесторонний справочник. Мы рассматриваем важные, полезные аспекты Linux для того, чтобы вы могли начать продуктивно работать с ним. Однако мы не приводим здесь все возможные команды и опции (заранее приносим извинения, если ваша любимая команда здесь не встретится), не вдаемся в подробности внутреннего устройства и работы этой операционной системы. Коротко, лаконично и по сути - вот наш девиз.

Мы фокусируем внимание на *командах,* этих надоедливых маленьких словах, которые вы набираете в командной строке для того, чтобы сказать системе Linux, что делать, например, таких, как Is (вывести список файлов), grep (найти заданный текст в файле), xmms (проиграть аудиофайл) и df (показать, сколько свободного места осталось на диске). Мы кратко затрагиваем графические оконные среды GNOME и KDE, каждой из которых можно было бы посвятить отдельный Карманный справочник.

Мы организовали материал по функциям так, чтобы вам было удобнее изучать его. Например, чтобы вы смогли просмотреть содержимое файла, мы вводим все команды просмотра файлов одновременно: cat - для коротких текстовых файлов, less - для более длинных, od - для бинарных, ghostview -для Postscript-файлов, и т. д. Затем мы рассматриваем каждую из этих команд в отдельности, кратко описывая ее предназначение и опции.

Мы предполагаем, что у вас есть учетная запись в Linux-системе и вы знаете, как войти в нее, используя ваше учетное имя и пароль. В противном случае поговорите с вашим системным администратором либо, если это ваша собственная система, используйте учетную запись, созданную при установке Linux.

Что такое Linux?

Linux - это популярная компьютерная программная среда с открытым исходным кодом, которая конкурирует с системами Microsoft Windows и Apple Macintosh. Она имеет четыре основные части.

Ядро

Это операционная система низкого уровня, обрабатывающая файлы, работающая с дисками, сетью и выполняющая другие необходимые операции

Программы

Тысячи программ для работы с файлами, текстовые редакторы, математические программы, программы для работы с аудио- и видеоинформацией, для создания веб-сайтов, шифрования, записи ком-.. пакт-дисков и т. д.

Командный процессор (shell)

Пользовательский интерфейс для набора команд, их исполнения и отображения результатов. Существуют различные командные процессоры: Bourne shell, Kom shell, C shell и др. В этой книге рассматривается bash, Bourne Again Shell, который, как правило, устанавливается для пользователей по умолчанию. Тем не менее, все эти командные процессоры имеют схожие основные функции.

X

Это графическая система, которая обеспечивает поддержку окон, меню, иконок, мыши и других известных элементов GUI - графического интерфейса пользователя. На основе X строятся более сложные графические среды; наиболее популярные из них - KDE и GNOME. На протяжении этой книги мы рассмотрим программы, которые для работы открывают собственные X-окна.

Что такое Fedora Unux?

Fedora Linux - это отдельный Linux-дистрибутив, созданный компанией Red Hat, Inc. и проектом Fedora (для более подробной

информации зайдите на сайт *http://fed.ora. redhat. com),* ранее называвшийся Red Hat Linux*. Наш материал основан на Fedora Core 1, первой официальной версии дистрибутива (ноябрь 2003). Мы рассматриваем поставляемые в дистрибутиве программы и командный процессор, кратко затрагивая систему X и ядро по необходимости.

Что такое команда?

Команда в Linux, как правило, состоит из *названия программы,* за которым следуют *опции и аргументы,* которые набираются в командном процессоре. Название программы ссылается на программу, расположенную где-то на диске (которую командный процессор найдет и выполнит). Опция, которая, как правило, начинается со знака минус, определяет действие программы, а аргументы, как правило, представляют собой входные и выходные данные. Например, следующая команда, которая считает количество строк в файле:

```
$ we -I myfile
```

состоит из имени программы (we - программа "word count"), опции (-1), которая говорит, что про грамма должна подсчитать количество строк, и аргумента *(myfile),* задающего файл, над которым нужно проделать эту операцию (значок доллара -это *приглашение на ввод команды* в командном процессоре, указывающее на то, что он ждет вашей команды). Опции можно задавать по отдельности:

* Сейчас компания Red Hat сконцентрировалась на более профессиональных продуктах. Большая часть этой книги дает информацию, применимую для Enterprise и других дистрибутивов Linux.

```
$ myprogram -a -Ь -c rayfile  Три отдельные опции
```

либо их можно объединять после одного минуса:

```
$ myprogram -abc myfile  То же самое, что -а -Ь -с
```

хотя некоторые программы ведут себя своеобразно и не распознают объединенные опции.

Команды могут быть намного сложнее, чем просто запуск одной программы.

• Они могут запускать несколько программ одновременно, либо последовательно (одну за другой), либо объединять их в "конвейер",

когда выходные данные предыдущей команды становятся входными данными для последующей.

• Опции не стандартизованы. Одна и та же опция (скажем, -1) может иметь разный смысл в разных программах: в программе we опция -1 означает "подсчитать количество строк текста", а в программе Is она означает "подробный вывод в одноколоночном формате". С другой стороны, две программы могут использовать разные опции для обозначения одного и того же действия "выполнить "молча"" (подавляя выдачу стандартной информации): -q ("quietly") и -s ("silently").

•Так же обстоит дело и с аргументами. Чаще всего они задают названия файлов входных и выходных данных, но они могут быть и директориями или регулярными выражениями.

•Тесктовый пользовательский интерфейс Linux - ***командный процессор*** — имеет встроенный язык программирования. Можно, например, вместо команды "запустить эту программу"сказать: "если сегодня четверг, то запустить эту программу, иначе выполнить другую команду шесть раз для каждого файла, оканчивающегося на *.txt"*.

Пользователи и суперпользователи

Linux - многопользовательская операционная система. На конкретном компьютере пользователь определяется *именем пользователя (username),* например "smith" или "funkyguy", и обладает личной (в разумных пределах) частью системы, которую он может использовать. Также существует специально определенный пользователь с именем *root, {суперпользователь),* который имеет право осуществлять в системе любые операции. Обычные пользователи ограничены в своих правах: хотя они и могут выполнять большинство программ, в целом они могут изменять только те файлы, которыми владеют они сами. С другой стороны, суперпользователь может создавать, изменять или удалять любой файл и запускать любую программу.

Некоторые команды, описанные в этой книге, может исполнять только суперпользователь. В таких случаях мы будем использовать значок "решетка" (#) в качестве приглашения командного процессора.

```
# далее вводится команда
```

В противном случае мы будем использовать значок доллара в качестве приглашения выполнить команду обычном пользователю.

```
$   дадее  вводится команда
```

Для того чтобы стать суперпользователем, вам не нужно выходить из системы и снова входит в нее; просто выполните команду su (см. раздел "Получение прав суперпользователя" на странице 189) и введите пароль суперпользователя.

```
$   SU   -1
Password:   ********
#
```

Как читать эту книгу

Когда мы описывае команду, сначала мы даем общую информацию о том, как ее использовать. Например, программа we (word count) имеет такой синтаксис:

```
we   [опции]   [файлы]
```

что означает, что вы должны набрать "we", а затем, если нужно, опции и имена файлов. Вам не нужно набирать квадратные скобки " [" и "]": они лишь указывают на то, что содержимое необязательно; а курсивный шрифт означает, что вместо этого слова вы должны ввести свое значение, например имя файла. Если вы встретите вертикальную черту между опциями или аргументами, возможно, заключенными в скобки:

```
Is   (файл   |   директория)
```

то это указывает на возможность выбора: при выполнении команды Is вы можете в качестве аргумента предоставить либо имя файла, либо имя директории.

Ввод и вывод

Большинство программ в Linux получают входные данные из стандартного потока ввода (standard input), которым, как правило, является клавиатура, а выходные данные выводят в стандартный поток вывода (standard output), как правило, на экран вашего монитора. Кроме того, сообщения об ошибках выводятся в стандартный поток ошибок (standard error), которым обычно тоже является ваш экран, но этот поток отделен от стандартного устройства вывода*. Позже мы увидим, как перенаправлять стандартные потоки ввода, вывода и сообщений об ошибках в файлы, каналы (pipe) и обратно. А пока давайте определимся с терминологией. Когда мы говорим, что команда "читает" (данные), то мы имеем в виду, что она это делает из стандартного потока ввода, если не оговорено другое. А когда команда "печатает", то она это делает в стандартный поток вывода, если только не упоминается принтер.

* Например, вы можете направлять стандартный поток вывода в файл и в то же время выводить стандартные сообщения об ошибках на экран.

Стандартный заголовок

Описание каждой команды начинается с заголовка. Например, для команды Is (list files – вывести список файлов):

Is [опции\ [файль/[coreutils

/bin stdin stdout -file --opt —help —version

Заголовок включает в себя название (Is) команды, ее синтаксис, директорию, в которой она расположена (/bin), название RPM-пакета, который ее установил (coreutils), и шесть свойств команды, напечатанных либо жирным (присутствующие), либо светлым (отсутствующие) шрифтом.

stdin

Команда читает данные из стандартного потока ввода, т. е. клавиатуры, по умолчанию.

stdout

Команда пишет в стандартный поток вывода, т. е. на экран, по умолчанию.

-file

Если вы введете минус (-) в качестве имени файла с входными данными, то команда будет читать данные из стандартного потока ввода; и аналогично, если ввести минус в качестве имени файла для выходных данных, то команда будет писать в стандартный поток вывода. Например, следующая командная строка we (word count - подсчет слов) читает файлы fuel, file2, затем читает данные из стандартного потока ввода, и затем file3.

$ we filel file2 - file3

- - opt

Если вы указываете командную опцию "--", это означает "конец перечисления опции": все, что находится в командной строке после этого, не является опцией. Иногда это необходимо при работе с файлами, названия которых начинаются с минуса и которые в противном случае воспринимались бы (ошибочно) как опции. Например, если название файла -foo, то команда we -foo не сработает, так как -foo будет воспринято как (неправильная) опция, в то время как we foo будет выполнена корректно. Если команда не поддерживает синтаксис "--", то вы можете присоединить спереди к имени файла имя текущей директории, так чтобы минус не был первым символом.

$ we ./-foo

- - *help*

Опция --help инициирует вывод командой справочного сообщения, которое объясняет, как правильно использовать эту команду, и завершает работу команды.

- - version

Опция --version инициирует вывод командой номер ее версии и завершает работу команды.

Стандартные символы

На протяжении книги мы будем использовать определенные символы для обозначения действия с клавишами. Как и в другой Linux-документации, мы используем символ ^, который обозначает действие "нажать и удерживать клавишу Ctrl", например, AD (произносится как "контрол D") обозначает действие "нажать клавишу Ctrl и, удерживая ее, нажать клавишу D". Также мы пишем ESC, подразумевая, что нужно "нажать клавишу Esc". Такие клавиши, как Enter и пробел, говорят сами за себя.

Ваш друг - команда echo

Во многих наших примерах мы будем выводить информацию на экран с помощью команды echo, которую мы формально опишем как "Вывод на экран" на странице 224. echo - одна из простейших команд: она просто печатает свои аргументы в стандартный поток вывода после того, как эти аргументы обработаны командным процессором.

```
S   echo My dog has   fleas
My dog has   fleas
$   echo My name     is   $USER Здесь USER-

переменная командного процессора
My name   is   smith
```

Получение справки

Если вам нужно больше информации, чем предоставляет эта книга, есть несколько способов получить ее.

Используйте команду man

Команда man выводит на экран страницу оперативной справки по заданной команде. Например, для того чтобы посмотреть на документацию по команде вывода списка файлов Is, выполните следующую команду:

man Is

Для того чтобы сделать поиск страниц справки по ключевому слову, используйте опцию -к, после которой укажите ключевое слово.

$ man -k database

Используйте команду info

Команда info - это расширенная гипертекстовая справочная система, охватывающая многие Linux-программы.

$ info Is

Если для интересующей вас программы документации нет, то info выведет man-страницу для нее.

Используйте опцию —help (если она имеется) Многие Linux-команды отвечают на опцию --help выводом короткого справочного сообщения. Попробуйте набрать следующую команду.

$ Is --help

Изучите директорию /usr/share/doc

Эта директория содержит справочную документацию для многих программ, как правило, разбитую по названиям программ и версиям. Например, файлы для текстового редактора Emacs версии 21.3 находятся в директории /usr/share/doc/emacs-21.3

Справка по GNOME и КОЕ

Для получения справочной информации по GNOME и KDE выберите вкладку Help (Справка) в главном меню.

Веб-сайты, посвященные Fedora Linux

Официальный сайт - *http://fedora.redhat.com.* Неофициальная страница часто задаваемых вопросов (FAQ) появилась на странице *http://fedora.artoo.net.* И конечно же, есть веб-страница у этой книги. http://www.oreilly.com/catalog/linuxpg/

Группы новостей USENET

Сеть Usenet имеет десятки новостных групп по Linux-тематике, например *comp.os.liniix.misc* и *comp.os. linux.questions.* Информацию, касающуюся Red Hat, можно найти в группах *alt.os.linux.redhat, comp.os.limix. redhat, linux.redhat* и *lima.redhat.misc.* Поиск среди сообщений новостных групп можно сделать на странице Google Groups, *http://groups.googie.com,* которая является кладезем справочной информации.

Google

Используйте Google для поиска документации и учебных пособий: *http://www.google.com.*

Fedora: первый взгляд

Когда вы входите в систему Fedora (или другую) Linux, вы, скорее всего, видите графический рабочий стол*, как на рис. 1, который содержит следующую информацию.

* Только если вы не входите в систему удаленно по сети, в случае чего вы увидите командную строку, приглашающую вас ввести команду.

- Панель задач, аналогичная панели задач Windows, снизу, на которой располагаются:

-иконка "red hat" (красная шапочка) слева, при нажатии на которую всплывает главное меню программ;

-иконки для запуска различных программ, например веб-браузера Mozilla, почтовой программы Evolution и менеджера

печати (Print Manager) для настройки принтеров;
-переключатель рабочих столов (квадрат, разделенный на четыре части), который позволяет перемещаться между несколькими рабочими столами;
-голубая "галочка", указывающая на то, что ваше системное программное обеспечение обновлено, либо, в противном случае, красный восклицательный знак;
-часы.

- Другие иконки на рабочем столе, например, "мусорная корзина" для удаления файлов, дисковод и ваша домашняя директория (папка) для хранения персональных файлов

В дистрибутив Fedora входит несколько похожих внешне интерфейсов, и вы будете работать либо с GNOME, либо с KDE*. Узнать, в каком интерфейсе вы работаете, можно нажав по иконке "red hat", чтобы открыть главное меню, и выбрав пункт Help. В появившемся окне Help будет явно указано, какой это графический интерфейс - GNOME или KDF-.

* В зависимости от вашей системной конфигурации интерфейс можег выглядеть иначе, GNOME и K.DE очень гибкие в настройке, а дистрибутив Fedora включает в себя еще и третий графический интерфейс, twin, со своими особенностями и внешним видом (а во обще, для Linux существуют и другие графические интерфейсы).

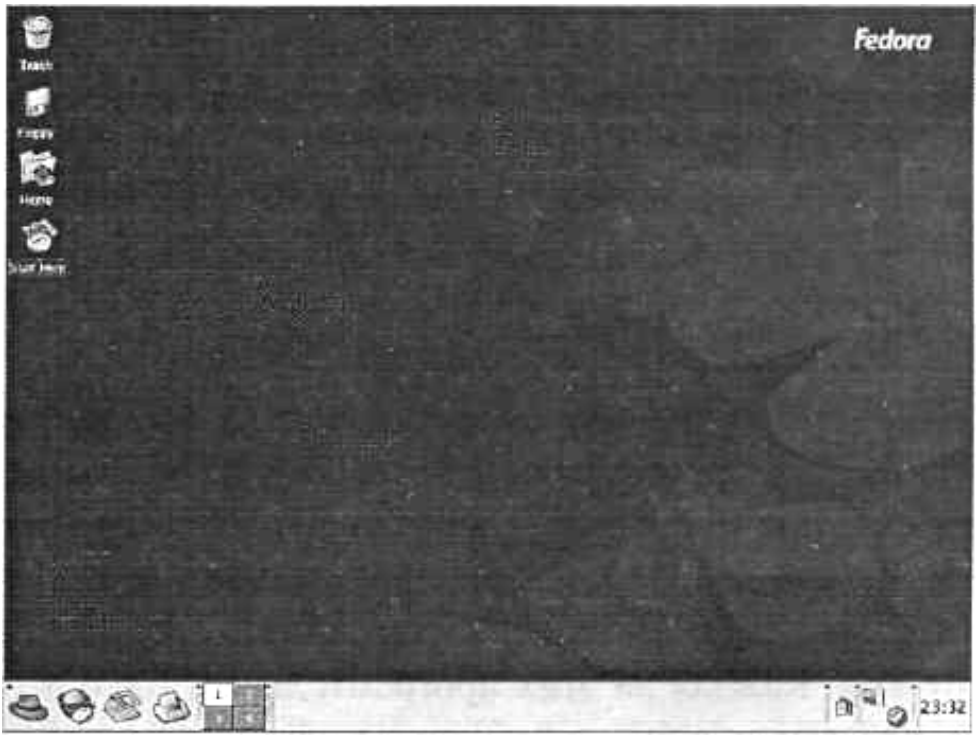


Рис. I. Графический рабочий стол Fedora

Роль командного процессора

Исследуйте иконки и меню в GNOME и KDE. Эти графические интерфейсы являются для некоторых пользователей основным способом работы в Linux. Различные дистрибутивы, включая Fedora, упрощают эти интерфейсы, чтобы пользователи без особых усилий могли редактировать файлы, читать электронную почту и работать с Internet.

Тем не менее, настоящая мощь Linux заключается не в этом. Чтобы извлечь максимум возможностей из Linux, вам нужно научиться пользоваться командным процессором. Сначала это может показаться сложно по сравнению с иконками и меню, но когда вы привыкнете, то поймете, как легко пользоваться командным процессором и насколько это мощный инструмент. На протяжении

этой книги в основном рассматриваются Linux-команды, которые запускаются из командного процессора.

Как запустить командный процессор

Для того чтобы запустить командный процессор в GNOME, KDE или любом другом графическом интерфейсе под Linux, вам нужно открыть *окно командного процессора.* Это делается с помощью таких программ, как xterm, gnome-terminal, konsole и uterm. Каждая их этих программ делает одно и то же: открывает окно, в котором запускается командный процессор, ожидающий ввода команд. Чтобы запустить окно командного процессора, используя три стандартных оконных интерфейса для Fedora, выполните следующие действия.

Интерфейс	Проделайте следующее...	. чтобы запустить командный процессор в окне...
GNOME	Menu -.*System Tools:* Terminal *или на рабочем столе.* Щелкните правой кнопкой мыши: Open Terminal	gnome-terminal
KDE	Menu : System Tools: Terminal или *на рабочем столе.* Щелкните правой кнопкой мыши: Open Terminal	console
twm	*На рабочем столе:*'Щелкните правой кнопкой мыши: XTerm	xterrn

Не путайте окно программы (например, программы konsole) с запущенным внутри нее командным процессором. Окно - это только контейнер, хотя и со своими интересными функциями, однако командным процессором является программа, которая предлагает вам вводить и выполнять команды.

Если вы не работаете с графическим интерфейсом, -скажем, вы заходите в систему удаленно по сети либо непосредственно с подключенного терминала, -командный процессор запустится немедленно, как только вы войдете в систему. В этом случае окно командного процессора не понадобится.

Вход/выход из системы и выключение

Мы предполагаем, что вы знаете, как входить в Linux-систему под вашей учетной записью. Для того чтобы выйти из GNOME или KDE, нажмите на иконку "red hat" (красная шляпа) на панели задач и выберите пункт Logout (завершить сеанс) в глав-ном меню. Для того чтобы выйти из командного процессора при удаленном входе в систему, просто закройте его, набрав команды exit или logout.

Никогда просто так не выключайте питание ком-пьютера с работающей системой Linux: ему требуется более корректное завершение работы. Для того чтобы выключить систему в GNOME, выберите в главном меню Logout (завершить сеанс), а затем Shut Down (выключить компьютер). В KDE сначала закройте свою учетную запись (log out), а затем на экране входа в систему выберете иконку Shutdown. Для того чтобы выключить систему из командного процессора, выполните команду shutdown в режиме суперпользователя. Синтаксис этой команды следующий.

shutdown [*опции*] *время* [сообщения] SysVinit /sbin stdin stdout -file --opt --help --version

Команда shutdown останавливает или перезапускает систему Linux; только суперпользователь может выполнять ее. Ниже приведен пример команды, которая остановит работу системы через 10 минут, послав сообщение "scheduled maintenance" (плановое обслуживание) всем работающим в системе пользователям.

```
# shutdown -h +10  "scheduled maintenance"
```

Параметр *время* может быть числом минут, которому предшествует знак плюса, например +10, абсолютное время в часах и минутах, например 16:25, или слово now (сейчас), означающее немедленное выполнение операции.

При отсутствии опций команда shutdown переводит систему в однопользовательский режим - специальный режим обслуживания, при котором в системе работает только один человек (в системной консоли) и все неосновные службы отключены. Для того чтобы выйти из однопользовательского режима, либо выполните еще раз

команду shutdown, чтобы завершить работу или перегрузиться, либо нажмите ^D, чтобы вернуть систему в нормальный многопользовательский режим.

Полезные опции

- r Перезагрузить систему
-h Остановить систему
-к На самом деле не завершать работу системы, но разослать предупреждающие сообщения всем пользователям, как будто система прекращает работу
-c Отменить текущий shutdown (опустите параметр *время)*
- f При перезагрузке отменить обычную проверку файловой системы, осуществляемую программой *t*sck (описанную в разделе "Диски и файловые системы" на странице 146}
-F При перезагрузке провести обычную проверку файловой системы

Для получения технической информации о завершении работы, однопользовательском режиме и различных состояниях системы обратитесь к man-страницам по init и inittab.

Файловая система

Для того чтобы использовать любую из Linux-систем, вам нужно познакомиться с файлами Linux и их расположением. Каждый файл в Linux содержится в коллекции, называемой *директорией.* Директории - это то же самое, что папки в системах Windows и Macintosh. Директории образуют иерархию или *дерево:* одна директория может содержать другие директории, называемые *поддиректориями,* и так далее до бесконечности. Самая верхняя директория называется *корневой* или *root-директорией* и обозначается символом косой черты (/)*.

* В Linux *все* файлы и директории происходят от корневой директории, в отличие от Windows или DOS, в которых различным дискам соответствуют свои метки.

Мы ссылаемся на файлы и директории, используя синтаксис из имен и косых черт, называемый *путем к файлу (path)*. Например, такой путь:

/on e/two/th ree/four

ссылается на корневую директорию /, которая содержит директорию *one,* которая содержит директорию *two,* которая содержит директорию *three,* которая содержит файл или директорию *four*. Если путь начинается с корневой директории, то он называется *абсолютным,* в противном случае он называется *относительным.* Подробнее об этом читайте ниже.

Когда вы запускаете командный процессор, то он "находится" в некоторой директории(в абстрактном смысле). Говоря более техническим языком, ваш командный процессор имеет *текущую рабочую директорию,* и когда вы выполняете команды в нем, они выполняются относительно этой директории. А конкретно, если вы указываете относительный путь в этом командном процессоре, то он будет вычислен относительно вашей текущей рабочей директории. Например, если ваш командный процессор находится в директории */one/two/three* и вы вьшолняете команду, которая ссылается на файл *myfile,* то абсолютный путь этого файла - */one/two/three/myfile.* Аналогично, относительный путь *a/Ь/c* будет соответствовать абсолютному пути */one/two/three/a/b/c*.

Для двух специальных директорий используются обозначения "." (одна точка) и ".." (две точки). Первое означает вашу текущую директорию, а последнее – родительскую (на один уровень выше). Поэтому если вашей текущей директорией является директория */one/two/three,* то "." будет ссылаться на эту директорию, а ".." - на директорию */one/two.*

"Перемещаться" из одной директории в другую можно с помощью команды cd.

$ cd /one/two/three

Эта команда меняет текущую рабочую директорию вашего командного процессора на директорию */one/ two/three.* Это *абсолютное* изменение (так как директория начинается со знака "/");

естественно, также вы можете осуществлять и относительные перемещения.

$ cd d *Войти в директорию* **d**
$ cd . . /mydir *Перейти в родительскую директорию, а затем из нее -в директорию* **mydir**

Имена файлов и директорий могут содержать различные символы: прописные и строчные* буквы, числа, точки, дефисы, символы подчеркивания и многие другие (за исключением символа "/" - он зарезервирован для разделения директорий). Однако старайтесь избегать использования пробелов, звездочек, круглых скобок и других символов, которые имеют специальное назначение в командном процессоре. Иначе вам придется все время заключать их в кавычки или квотировать их (см. раздел "Использование кавычек" на странице 49).

* Они не являются эквивалентными, так как имена файлов в Linux чувствительны к регистру.

Домашние директории

Личные файлы пользователей часто находятся в директориях */home* (для обычных пользователей) и */root* (для суперпользователей). Вашей домашней директорией, как правило, будет директория */home/your jtsername,* где *yourjtsername* - название вашей учетной записи в системе, например: */home/smith, /home/jones* и т. д. Есть несколько способов переместиться в вашу домашнюю директорию или сослаться на нее.

cd

Команда cd, выполненная без аргументов, переместит вас (т. е. установит рабочую директорию командного процессора) в вашу домашнюю директорию.

Переменная HOME

Переменная среды HOME (см. раздел "Переменные командного процессора" на странице 43) содержит имя вашей домашней директории.

```
$ echo $HOME      Команда echo печатает на
                  экран свои аргументы /home/smith
```

Используемая вместо директории одиночная тильда воспринимается командным процессором как имя вашей домашней директории.

```
$ echo ~/home/smith
```

Если за тильдой будет следовать имя пользователя (например, *~smith),* то командный процессор воспринимает эту строку как домашнюю директорию пользователя.

```
$ cd ~smith
$ pwd    Эта команда печатает рабочую директорию
/home/smith
```

Системные директории

Типичная система Linux имеет десятки тысяч системных директорий. Эти директории содержат файлы операционной системы, приложений, документации и почти всего остального, *за исключением* личных файлов пользователей (которые, как правило, находятся в директории */home).*

Если только вы не являетесь системным администратором, вы редко будете заходить в большинство системных директорий, но с небольшими знаниями вы сможете понять или догадаться о их назначении. Их названия часто содержат три части, которые мы будем называть частями "области действия", "категории" и "приложения" (они не имеют стандартных терминов, но так вам будет проще разобраться). Например, директория */usr/local/share/emacs,* которая содержит локальные данные для текстового редактора Emacs, имеет "область действия" */usr/local* (системные файлы, установленные локально), "категорию" *share* (специфические данные

программ и документация) и "приложение" *emacs* (текстовый редактор), как показано на рис. 2. Мы рассмотрим эти три части, правда, несколько в другом порядке.

/usr/local / share/ emacs
Scope **Category** ***Applicati on***

Рис. 2. Директории *"области действия", "категории"* и *"приложения"*(scope - область действия, category - категория, application - приложение)

Категории

Часть *"категория"* в пути к файлу или директории сообщает о типе файлов, которые находятся в директории. Например, если категория называется *bin,* то можно быть уверенным - эта директория содержит программы. Общие категории перечислены ниже.

Категории для программ

bin **Программы (как правило двоичные файлы)**

sbim **Программы (как правило двоичные файлы), предназначенные для выполнения суперпользователем**

lib **Библиотеки, используемые программами**

libexec **Программы, сбываемые, как правило, другими программами, а не пользователями; читай: "библиотеки исполняемых программ"**

Категории для документации

doc **Документация**

info **Файлы документации для встроенной справочной системы редактора Emacs**

man **Файлы документации {man-страницы), отображаемые программой man; сами файлы трудночитаемы: они содержат форматирующие команды и часто хранятся в сжатом виде**

Share **Специфические для программ файлы. Это могут быть примеры использования или указания по**

установке

Категории для конфигурирования

etc **Конфигурационные файлы для системы (и другие разнообразные аттрибуты)**

init.d **Конфигурационные файлы для загрузки Linux;**

rc.d **также *rc1.d, rc2.d...***

Категории для программирования

include **Заголовочные файлы для программирования**

sre **Исходные тексты программ**

Категории для веб-файлов

cgi-bin **Скрипты/программы, которые выполняются на веб-страницах**

html **Веб-страницы**

public_html **Веб-страницы (как правило в домашних директориях пользователей)**

www **Веб-страницы**

Категории для графической системы

fonts **Шрифты (сюрприз!)**

X11 **Файлы системы X Window**

Категории для оборудования

dev **Файлы устройств для работы с дисками и другим аппаратным обеспечением**

mnt **Точки монтирования: директории, через которые**

m/sc **осуществляется доступ к дискам**

Категории для рабочих файлов

var **Файлы, специфичные для данного компьютера, создаваемые и изменяемые во время работы компьютера**

lock **Файлы блокировки, созданные программами, чтобы сказать: "Я работаю"; существование файла блокировки может не позволить другой программе**

/ли другому экземпляру той же самой программы запуститься или произвести какое-либо действие

log **Файлы журналов, которые отслеживают важные системные события и содержат сообщения об ошибках, предупреждающие и информационные сообщения**

mail **Почтовые ящики для входящей почты**

run **PID-файлы, которые содержат ID (идентификаторы) запущенных процессов**

spool **Файлы, поставленные в очередь на отправку или отправляемые в данный момент, например исходящая электронная почта, задания печати и запланированные задания**

tmp **Временное хранилище, используемое программами и/или людьми**

proc **Состояние операционной системы: см. раздел "Директории операционной системы" на странице 35**

Области действия

Часть *"область действия"* в пути к файлу или директории описывает предназначение всей иерархии директорий. Вот некоторые общие из них.

/ Системные файлы, поставляемые с Linux

/usr **Дополнительные системные файлы, поставляемые с Linux**

/usr/games **Игры (сюрприз!)**

/usr/kerberos **Файлы, относящиеся к системе аутентификации Kerberos**

/usr/local **Системные файлы, разработанные "локально" либо для организации, либо для данного отдельного компьютера**

/usr/X11R6 **Файлы, принадлежащие к системе X Window**

Итак, для категории, например, *lib* (библиотеки) ваша система Linux может иметь директории */lib, /usr/lib, /usr/ local/lib, /usr/games/lib* и */usr/Xl JR6/lib.* В вашей системе могут быть и другие "области действия" по усмотрению вашего системного администратора: */my-company/ lib, /my-division/lib* и т. д.

На практике нет четкого функционального отличия директории / от */usr,* но директория / "более низкоуровневая" и находится "ближе" к операционной системе. •Так, например, */bin* содержит такие фундаментальные программы, как Is и cat, */usr/bin* содержит различные приложения, поставляемые с дистрибутивом Linux, а */usr/local/bin* содержит программы, которые установил ваш системный администратор. Это правило не является обязательным, но это общепринятая практика.

Приложения

Часть *"приложения"* в пути к файлу или директории -это, как правило, название программы. После частей "области действия" и "категории" (скажем */usr/local/doc)* программа может иметь свою собственную поддиректорию (скажем, */usr/local/doc/myprogram),* содержащую нужные ей файлы.

Директории операционной системы

/boot

Файлы для загрузки системы. Именно в этой директории находится ядро; как правило, его файл называется */boot/vmlinuz.*

/lost+found

Поврежденные файлы, которые были восстановлены программой восстановления диска.

/proc

Описывает запущенные в текущий момент процессы; для продвинутых пользователей.

Файлы в директории */proc* предоставляют возможность посмотреть на внутреннюю работу ядра и имеют специальные свойства. Они всегда имеют нулевой размер, текущий момент в качестве даты создания/модификации и доступны только для чтения.

$ Is -I /proc/version
-r--r--r--l root root 0 Oct 3 22:55 /
proc/version

Однако их содержимое содержит информацию о ядре Linux.

```
$ cat /proc/version
Linux version 2 . 4 . 22-1. 2115 .nptl ...
```

В основном эти файлы используются программами. Вот несколько примеров.

/proc/ioports Список используемых портов ввода/вывода вашего компьютера

/proc/version Версия операционной системы. Команда uname выводит эту же информацию

/proc/uptime Время работы системы, т. е. число секунд с того момента, когда система была запущена

/proc/nnn Информация о процессе Linux с номером (ID) *nnn,* где *nnn* -положительное целое число

/proc/self Информация о текущем процессе, который вы выполняете; символьная ссылка на файл */proc/nnn,* автоматически обновляемая. Попробуйте набрать Is -I /proc/self несколько раз в строке: вы увидите, как эта ссылка изменяется

Защита файлов

Система Linux может иметь много пользователей со своими учетными записями. Для обеспечения секретности и безопасности каждый пользователь может осуществлять доступ не ко всем, а только *к некоторым* файлам в системе. Этот контроль доступа заключается в двух вопросах.

Кто имеет права? Каждый файл и директория имеют владельца, который имеет права делать с ними все, что угодно. Как правило, пользователь, который создал файл, и является его владельцем (owner), но отношения могут оказаться более сложными.

В добавок к этому доступ к файлу может иметь предопределенная *группа (group)* пользователей. Группы определяются системным администратором, они рассматриваются в разделе "Работа с группами" на странице 190.

Наконец, файл или директория могут быть доступны *всем пользователям* с учетными записями в данной системе. Такую группу пользователей обозначают просто *"остальные"*.

Какие права предоставлены? Владельцы файла, группы и остальные могут иметь права на *чтение (read), запись (write, modify)* и *выполнение (execute, run)* конкретных файлов. Разрешения также распространяются на директории, которые пользователи могут читать (осуществлять доступ к файлам внутри директорий), писать в них (создавать и удалять файлы внутри директорий) и исполнять (входить в директории).

Для того чтобы узнать, кто является владельцем файла, и посмотреть, какие права выставлены для этого файла, выполните следующую команду:

$ Is -I *filename*

Для того чтобы узнать, кто является владельцем директории, и посмотреть, какие права выставлены для этой директории, выполните следующую команду:

$ Is -Id *directory_name*

Права на файл - это 10 символов слева в выходных данных команды, строка из г (read - чтение), w (write - запись), x (execute - выполнение) и других букв. Например:

drwxr-x

Вот что означают эти буквы и символы.

Позиция	**Значение**
1	**Тип файла: - = файл, d = директория, 1 = символьная ссылка, р = именованный канал, с = устройство посимвольного ввода-вывода, Ь = блочное устройство**
2-4	**Права на чтение, запись и исполнение для владельца файла**
5-7	**Права на чтение, запись и исполнение для группы**

8-10 Права на чтение, запись и исполнение для остальных пользователей

Более детально мы рассмотрим команду Is в разделе "Основные операции с файлами" ika странице 66. Для того чтобы сменить владельца, группу или права на файл, используйте команды chown, chgrp и chmod соответственно, которые будут рассмотрены в разделе "Свойства файла" на странице 95.

Командный процессор

Для запуска команд в системе Linux вам нужно где-то набирать их. Этим "где-то" служит *командный процессор (shell),* являющийся командным пользовательским интерфейсом Linux: вы набираете команду и нажимаете Enter, и командный процессор запускает запрошенную

вами программу (или программы). Для того чтобы начать работу с командным процессором, обратитесь к разделу "Fedora: первый взгляд" на странице 21.

Например, чтобы посмотреть, кто работает в системе в текущий момент, вы можете выполнить в командном процессоре следующую команду:

```
$ who
Barret   :0          Sep 23 20:4
t                                4
Byrnes   pts/        Sep 15 13:5
         0                       1
Silver   pts/        Sep 22 21:1
         1                       5
Silver   pts/2       Sep 22 21:1
                                 8
```

(символ доллара "$" - это приглашение командного процессора, которое означает, что он готов выполнить команду). Также одна

команда может вызвать несколько программ одновременно и даже объединить программы так, чтобы они взаимодействовали. Ниже приведена команда, которая перенаправляет выходные данные программы who на вход программы we, которая подсчитывает число строк текста в файле; результатом является число строк в выходных данных программы who:

```
$ who | we -1
4
```

сообщающее, сколько пользователей работают в системе*. Вертикальная черта, называемая конвейером, осуществляет соединение между программами who и we.

Командный процессор фактически сам является программой, и в Linux есть несколько командных процессоров. Мы рассматриваем Bash ("Bourne-Again Shell"), который находится в файле */bin/bash* и который является стандартным (используется по умолчанию) в дистрибутиве Fedora Linux.

*А фактически, сколько интерактивных командных процессоров открыли эти пользователи. Если пользователь открыл несколько командных процессоров, как, например, пользователь silver в этом примере, то команда who выведет для него столько же строк.

Командный процессор и программы

Когда вы выполняете команду она либо может вызывать Linux-программу (например, who), либо может являться *встроенной командой,* функцией самого командного процессора. Вы можете определить это с помощью команды type.

```
$ type who
who is /usr/bin/who
$ type cd
cd is a shell builtin
```

Полезно знать различия между тем, что предоставляет командный процессор, и тем, что делает Linux. Следующие несколько разделов описывают функции командного процессора.

Избранные функции bash

Командный процессор делает больше, нежели просто запускает команды. Также он предоставляет мощные функции для того, чтобы облегчить эту задачу.

Например, групповые символы для поиска файлов по шаблонам имен, перенаправление выходных данных команд в файлы и входных данных из файлов, конвейеры (pipe) для того, чтобы делать выходные данные одной команды входными данными для другой, псевдонимы (alias) для быстрого вызова наиболее употребительных команд, переменные для хранения значений, которые может использовать командный процессор, и т. д. Мы всего лишь слегка затронем эту тему для того, чтобы продемонстрировать вам набор полезных инструментов. Выполните команду info bash, чтобы получить полную документацию.

Групповые символы

Групповые символы предоставляют возможность задавать наборы файлов со схожими именами. Например, a* означает все файлы, которые начинаются с прописной буквы "a". Групповые символы "заменяются" командным процессором на набор имен файлов, которым они соответствуют. То есть если вы набираете:

```
$ ls a*
```

то командный процессор сначала заменяет a* на имена файлов в вашей рабочей директории, которые начинаются с буквы "a", как будто если бы вы набрали следующую команду.

```
ls aardvark adamantium apple
```

Команда ls никогда не узнает, что вы использовали групповой символ: она видит только окончатель

ный список имен файлов после того, как команд ный процессор обработает групповой символ.

Групповой символ	**Значение**
*	Любой набор символов, в том числе пустая строка
?	Любой одиночный символ
[*набор]*	Любой одиночный символ из заданного *набора,* который, как правило, является последовательностью символов, например [aeiouAEiou] для всех гласных букв, либо диапазоном с дефисом, например, [A-z] для всех заглавных букв
[^*набор]* [!*набор]*	Любой одиночный символ, не заданный в *наборе*

Если вы хотите включить в набор символ минуса, то поместите его первым или последним. Чтобы включить в набор закрывающую квадратную скобку, поместите ее первой. Чтобы включить в набор символы ^ или !, не помещайте их на первое место.

Подстановка фигурных скобок

Аналогично групповым символам выражения с фигурными скобками также заменяются на аргументы команды. Следующее выражение, разделенное скобками:

{a,b,cc,ddd}

заменяются на

a b c dddd

Фигурные скобки работают с любыми строками, в отличие от групповых символов, которые ограничиваются именами файлов. Например, строка **sand{ X, Y, ZZZ}wich** заменяется на

```
$ echo sand{X,Y,ZZZ}wich
sandXwich sandYwich sandZZZwich
```

независимо от того, какие файлы есть в текущей директории.

Подстановка тильды

Командный процессор воспринимает тильды (~) как специальные символы, если они появляются отдельно или в начале слова.

-	**Ваша домашняя директория**
-smith	**Домашняя директория пользователя smith**

Переменные командного процессора

Вы можете определить переменные и их значения следующим образом.

```
$ MYVAR=3
```

Чтобы сослаться на значение, просто поместите символ доллара перед именем переменной.

```
$ echo $MYVAR
3
```

Некоторые переменные являются стандартными и, как правило, определяются вашим командным процессором при входе в систему.

Переменная	**Значение**
DISPLAY	Имя вашего сеанса X-window
HOME	Имя вашей домашней директории
LOGNAME	Название вашей учетной записи
MAIL	Путь к вашей директории с входящей электронной почтой
OLDPWD	Предыдущая рабочая директория вашего командного процессора
PATH	Путь поиска для вашего командного процессора: директории, разделенные двоеточиями
PWD	Текущая директория командного процессора

SHELL	Путь к вашему командному процессору, например /bin/bash
TERM	Тип вашего терминала например xterm или vt100
USER	Название вашей учетной записи

Чтобы увидеть список переменных командного процессора, выполните следующую команду

```
$ printenv
```

Область действия этих переменных (т. е. какие программы знают о них) по умолчанию распространяется на тот командный процессор, в котором они определены. Чтобы сделать эти переменные и их значения доступными другим программам, которые вызывает ваш командный процессор, используйте команду export.

```
$ export MYVAR
```

или

```
$ export MYVAR=3
```

Теперь ваша переменная называется *переменной окружения* (переменной среды), так как она доступна другим программам в "среде" вашего командного процессора. Чтобы сделать так, чтобы определенное значение переменной использовалось для определенной программы только один раз, припишите в начале командной строки строку *"переменная=значение"*.

```
$ echo $HOME
/home/smith
$ HOME=/hame/sally echo "My home is $HOME"
My home is /home/sally
$ echo $HOME
/home/smi th      Исходное значение не изменилось
```

Путь поиска

Очень важной переменной является переменная PATH, которая сообщает командному процессору, где искать программы. Когда вы набираете любую команду:

```
$ who
```

то командный процессор должен найти соответствующую программу (программы). Он обращается к значению переменной PATH, которым является последовательность директорий, разделенных двоеточием:

```
$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/
XllR6/bin:/home/smith/bin
```

и ищет программу who в каждой из этих директорий. Если он находит who (скажем, */usr/bin/who),* то он выполняет команду. Иначе он пишет следующее.

```
bash: who: command not found
```

Чтобы добавить директории к пути поиска вашего командного процессора временно, измените переменную PATH. Например, чтобы добавить директорию */usr/sbin* к пути поиска вашего командного процессора, необходимо ввести следующую команду.

```
$ PATH=$PATH:/usr/sbin
$ esho $PATH
/usr/local/bin:/bin:/usr/bin:/usr/
XllR6/bin:/home/smith/bin:/usr/sbin
```

Чтобы сделать это изменение постоянным, измените значение переменной PATH в вашем файле инициализации *~/.bash_profile,* как показано в разделе "Изменение поведения командного процессора" на странице 59. Затем выйдите и войдите в систему заново.

Псевдонимы

Встроенная команда alias обеспечивает удобный сокращенный ввод длинных команд. Например:

```
$ alias ll="ls -l"
```

определяет новую команду 11, которая эквивалента команде Is -1.

```
$ 11
total 436
-rw-r--r-- I smith 3584 Oct 11 14:59
filel
-rwxr-xr-x 1 smith 72 Aug 6 23:04 file2
```

Чтобы псевдонимы бьши доступны при входе в сис тему, определите их в вашем файле *~/.bash_profile* (см. раздел "Изменение поведения командного процессора" на странице 59). Чтобы вывести список всех определенных псевдонимов, наберите alias. Если функциональности псевдонимов вам недостаточно (у них нет параметров или ветвления), обратитесь к разделу "Программирование в командном процес соре" на странице 256 или выполните команду info bash и прочитайте раздел "Shell functions" (функции командного процессора).

Перенаправление ввода/вывода

Командный процессор перенаправляет стандартный поток ввода, стандартный поток вывода и стандартный поток ошибок в файлы и из файлов. Другими словами, любая команда, которая читает из стандартного потока ввода, может вместо этого получать свои входные данные из файла с помощью оператора "<" (знак "меньше").

```
$ mycommand < infile
```

Аналогично, любая команда, которая пишет в стандартный поток вывода, может вместо этого писать свои выходные данные в файл.

$ mycoitrmand > outfile ***Создать/переписать***

выходной файл outfile

$ mycommand >> outfile ***Добавить в данные***

в выходной файл outfile

Команда, которая пишет в стандартный поток ошибок, может также перенаправлять свои выходные данные в файл.

$ mycommand 2> errorfile

Чтобы перенаправить одновременно стандартный поток вывода и стандартный поток ошибок в файлы, необходимо выполнить следующую команду.

$ mycommand > outfile 2> errorfile ***В разные файлы*** **$ mycommand > outfile 2>&1** ***В один файл***

Конвейеры

Используя командный процессор, вы можете делать так, чтобы стандартный поток вывода одной команды становился стандартным потоком ввода для другой. Для этого используется оператор командного процессора "|" (конвейер). Например, команда:

$ who | sort

посылает выходные данные команды who программе sort, печатающей в алфавитном порядке список пользователей, работающих в данный момент в системе.

Объединение команд

Для вызова последовательности нескольких команд в одной командной строчке разделите команды точкой с запятой.

$ ***commandl ; command2 ; command3***

Чтобы запустить последовательность команд, показанную выше, и иметь возможность остановить выполнение в случае, если одна из команд не сможет выполниться, разделите их символами "&&" (логическое "и").

$ ***commandl && command2 && comma.nd.3***

Чтобы запустить последовательность команд так, чтобы исполнение завершилось, если одна из них выполнится, разделите эти команды символами || (логическое "или").

$ ***commandl \ \ command2 || command3***

Использование кавычек

Как правило, командный процессор рассматривает пробелы просто как символы, разделяющие слова в командной строке. Если вы хотите, чтобы слово *содержало* пробел (например, имя файла, содержащее пробел), окружите его одинарными или двойными кавычками, чтобы командный процессор воспринимал его как отдельный элемент строки. При использовании одинарных кавычек командный процессор будет рассматривать строку как есть, тогда как в случае двойных кавычек он может интерпретировать ее, например, если в ней содержится переменная.

$ echo "The variable HOME has value $HOME' The variable HOME has value $HOME $ echo "The variable HOME has value $HOME" The variable HOME has value /home/smith

При использовании обратных кавычек их содержимое воспринимается как команда; в этом случае содержимое заменяется стандартным потоком вывода этой команды.

$ /usr/bin/whoami
smith
$ echo My name is "/usr/bin/whoami[4]
My name is smith

Маскировка

Если символ имеет специальный смысл для командного процессора, а вы хотите использовать его в строке (например, символ "*" - это знак сноски в обычном тексте, а не групповой символ), то поставьте перед этим символом левую косую черту "\". Это называется маскировкой специального символа.

$ echo a* ***Здесь * является групповым символом для подбора файлов с именами начинающимися с буквы "a"***

aardvark agnostic apple
$ echo a* ***Здесь *является простым символом***
a*
$ echo "I live in $HOME" ***Символ доллара означает значение переменной***
I live in /home/smith
$ echo "I live in \$HOME" ***Обычный символ доллара***
I live in $HOME

Также вы можете маскировать управляющие символы (табуляции, новой строки, ^D и т. д.), чтобы использовать их

буквально в командной строке, если перед ними вводить ^V. В частности, это полезно для символов табуляции (^I), которые в противном случае командный процессор использует для завершения имен файлов (см. раздел "Завершение имен файлов" на странице 53).

```
$ echo "There is a tab between
here^V^I land here"
There is a tab between here      and here.
```

Редактирование командной строки

bash позволяет редактировать командую строку, с которой вы работаете, используя клавишные комбинации, которые перешли от текстовых редакторов emacs и vi (см. раздел "Создание и редактирование файлов" на странице 86). Чтобы иметь возможность редактировать командную строку с помощью комбинаций emacs, выполните следующую команду (и поместите ее в ваш файл *~/.bashj>rofile,* чтобы она выполнялась автоматически при запуске командного процессора).

```
$ set -o emacs
```

Для комбинаций vi выполните следующую команду.

```
$ set -o vi
```

Клавишные комбинации emacs	**Клавишные комбинации vi (сначала нужно нажать ESC)**	**Функция**
^ P или стрелка вверх	**к**	Предыдущая командная строка
^ N или стрелка вниз	i	Следующая командная строка
^ F **или** стрелка вправо	1	На один символ вперед
^ В или стрелка влево	h	На один символ назад
^ A	0	В начало строки
^ E	$	В конец строки
^ D	**x**	Удалить следующий символ
^ U	^ U	Удалить всю строку

История командной строки

Вы можете возвращаться к командам, которые вы выполняли раньше, т. е. обращаться к *истории* командного процессора, и затем изменить и выполнить их. Некоторые полезные для работы с историей команды перечислены ниже.

Команда	**Функция**
history	Вывести вашу историю
history *N*	Вывести последние N команд в вашей истории
history -c	Удалить вашу историю
!!	Предыдущая команда
N	Команда номер N в вашей истории
\-*N*	Команда, которую вы набрали N команд назад
!$	Последний параметр предыдущей команды; очень полезен при проверке наличия файлов перед их удалением: $ Is a* $пл!$
!*	Все параметры предыдущей команды

Завершение имен файлов

Нажмите клавишу TAB, когда вы уже набрали часть имени файла, и командный процессор автоматически завершит его. Если тому, что вы набрали, соответствуют имена нескольких файлов, то командный процессор подаст звуковой сигнал, сообщающий о том, что выбор неоднозначен. Нажмите сразу после этого TAB, и командный процессор покажет альтернативные варианты. Попробуйте проделать следующее.

```
$ cd /usr/bin
$ Is un<TAB><TAB>
```

Управление задачами

jobs	**Перечисляет ваши задачи**
&	**Выполнить задачу в фоновом режиме**
^Z	**Приостановить выполнение текущей (интерактивной) задачи**
suspend	**Приостановить командный процессор**
fg	**Перевести задачу в интерактивный режим выполнения**

bg **Перевести приостановленную задачу в фоновый режим выполнения**

Все командные процессоры Linux имеют *возможность управления задачами:* возможность выполнять программы в фоновом (невидимая многозадачность) и интерактивном (чтобы программа выполнялась как активный процесс в сеансе вашего командного процессора) режимах. Задача (job) ~ это просто рабочая единица командного процессора.

Когда вы запускаете команду, ваш текущий командный процессор определяет ее как задачу и следит за ней. Когда команда выполнена, соответствующая задача исчезает. Задачи находятся на более высоком уровне, чем процессы Linux; операционная система Linux ничего о них не знает. Они являются всего лишь элементами командного процессора. Вот некоторые важные термины из лексикона задач.

интерактивное задание (foreground job) Выполняемое в командном процессоре, занимающее сеанс командного процессора, так что вы не можете выполнить другую команду.

фоновое задание (background job)

Выполняемое в командном процессоре, но не занимающее сеанс командного процессора, так что вы можете выполнить другую команду в этом же командном процессоре.

приостановить (suspend)

Временно приостановить интерактивный процесс.

возобновить (resume)

Вернуться к вьшолнению приостановленной задачи.

jobs

Встроенная команда j obs выводит список задач, запущенных в вашем текущем командном процессоре.

```
$ jobs
[1]- Running  emacs myfile  &
[2]+  Stopped  su
```

Целое число слева - это номер задачи, а знак плюса указывает на ту задачу, к которой по умолчанию будут применяться команды fg (foreground) или Ъд (background).

&

Амперсанд, помещенный в конце строки, приводит к тому, что заданная команда будет выполняться как фоновая задача.

$ emacs myfile & [2] 28090

Ответ командного процессора включает в себя номер задачи (2) и идентификатор процесса(28090). __

^Z

Нажатие ~Z в командном процессоре во время выполнения интерактивной задачи приостановит ее выполнение. Задача просто перестает выполняться, но ее состояние запоминается.

$ mybigprogram
[1]+ Stopped mybigprogram $

Теперь вы можете набрать команду Bg, чтобы перевести эту команду в фоновый режим, либо команду f g, чтобы возобновить ее выполнение в интерактивном режиме.

suspend

Встроенная команда suspend приостанавливает текущий командный процессор, если возможно, как если бы вы набрали ^Z в самом командном процессоре. Например, если вы выполнили команду su и хотите вернуться в ваш исходный командный процессор.

$ whoami
smith
$ su -1
Password: *****

#whoami
root
 # suspend
[1]+ Stopped su
$ whoami
smith

bg *[%jobnumber*

Встроенная команда bg запускает выполнение приостановленной задачи в фоновом режиме. При выполнении без аргументов bg действует на задачу, которая была приостановлена самой последней. Чтобы задать конкретное задание (из списка, выводимого командой jobs), укажите номер задания с предшествующим символом процента.

$ bg %2

Некоторые задачи не могут оставаться в фоновом режиме, например, если они ожидают ввода данных. Если вы попытаетесь выполнить эту команду для такой задачи, то командный процессор приостановит ее и выведет следующее.

[2]+ Stopped *далее идет командная строка*

Теперь вы можете возобновить выполнение задачи (с помощью команды fg) и продолжить работу.

fg *[%jobnumber]*

Встроенная команда fg переводит выполнение приостановленного или фонового задания в интерактивный режим. При выполнение без аргументов эта команда, как правило, выбирает задание, приостановленное или отправленное в фоновый режим последним. Чтобы задать конкретное задание (из списка, выводимого командой jobs), укажите номер задания с предшествующим символом процента.

$ fg %2

Прекращение выполнения команды

Если вы запустили из командного процессора команду в интерактивном режиме и хотите немедленно прекратить ее выполнение, введите ^C. Командный процессор воспримет нажатие ^C как "остановить выполнение текущей задачи немедленно". Поэтому, если вы выводите очень длинный файл (скажем, командой cat) и хотите остановить вывод, нажмите ^C.

$ cat bigfile
This is a very long file with many
lines. Blah blah blah blah blah blah
blah blahblahblah ^C
$

Чтобы прекратить выполнение программы, работающей в фоновом режиме, вы можете перевести ее в интерактивный режим с помощью команды fg и затем набрать ^C, или, на выбор, использовать команду kill (см. раздел "Управление процессами" на странице 174).

В целом, использование комбинации ^C не является корректным способом остановки выполнения программы. Дело в том что ^C останавливает программу немедленно, не оставляя ей ни единого шанса освободить занимаемые ей ресурсы. Остановка программы может привести к тому, что ваш командный процессор не будет реагировать на команды и нажатия клавиш. Если это произошло, выполните следующие действия.

1. Нажмите ^J, чтобы добраться до приглашения командного процессора. Эта комбинация дает такой же эффект, как и клавиша Enter (символ новой строки), но работает даже тогда, когда не работает Enter.

2.Наберите слово reset (даже если буквы не появляются во время набора) и нажмите ^J снова, чтобы выполнить эту команду. Так вы переинициализируете ваш командный процессор. Комбинация ^C работает только в окне командного процессора. Скорее всего, она не возымеет эффекта в каком-либо другом окне, не являющимся окном командного процессора. Кроме того, некоторые программы

написаны так, что они фиксируют нажатие ^C и игнорируют это действие: одним из примеров является текстовый редактор emacs.

Завершение работы командного процессора

Для того чтобы завершить работу командного процессора, либо выполните команду exit, либо нажмите ^D*.

```
$ exit
```

Изменение поведения командного процессора

Чтобы настроить работу каждого вашего командного процессора, редактируйте файлы *.bash_profile* и *.bashrc* в вашей домашней директории. Эти файлы выполняются каждый раз, когда вы входите в систему *(~/.bash_profile)* или открываете командный процессор *(-/.bashrc)*. В этих файлах можно задавать переменные и псевдонимы, запускать из них программы, печатать ваш гороскоп и т. д.

Эти два файла являются примерами скриптов командного процессора: исполняемыми файлами,

* Комбинация Ctrl-D обозначает "конец файла" для любой программы, осуществляющей чтение из стандартного потока ввода. В данном случае такой программой является сам командный процессор, работу которого мы хотим завершить.
которые содержат команды командного процессора. Мы рассмотрим эту тему более подробно в разделе "Программирование в командном процессоре" на странице 256.

Установка программного обеспечения

Возможно, вам время от времени нужно будет устанавливать дополнительное программное обеспечение в вашей Linux-системе. Наиболее распространенная форма программных пакетов для Fedora и многих других Linux-дистрибутивов - это

**.rpm-файлы.*

Файлы Red Hat Package Manager (RPM) устанавливаются и управляются программами rpm (вручную) и up2date (автоматически).

*файлы *.tar.gz, *.tar.Z, *.tar.bz2*

Упакованные tar-файлы - упакованы с помощью программы tar и сжаты с помощью архиваторов gzip *(.gz),* compress (.Z) или bzip2 *(.bz2)*.

Большая часть программного обеспечения должна устанавливаться суперпользователем, поэтому вам нужно выполнить команду su (или ее эквивалент) перед установкой. Например:

```
$ su -1
Password: ********
# rpm -ivh mypackage.rpm
...etc...
```

Для поиска нового программного обеспечения изучите компакт-диски вашего Linux-дистрибутива или зайдите, например, на следующие сайты.

http://freshmeat. net/ http://freshrpms. net/ http://rpmfmd. net/ http.V/sourceforge. net/

```
up2date [опции] [файль!\   up2date
/usr/bin stdin stdout -file --opt --help -version
```

up2date - это самый простой способ обновления вашей системы Fedora. В режиме суперпользователя просто выполните команду:

```
#up2date
```

и следуйте инструкциям. Эта команда вызывает графический пользовательский интерфейс. Также вы можете запустить команду up2date в командном режиме:

#up2date -1

чтобы вывести список всех обновленных RPM-паке-тов (если таковые есть) для вашей системы. Чтобы скачать эти пакеты, выполните следующую команду:

#up2date -d packages

Чтобы установить RPM-пакеты, которые вы уже скачали с помощью команды up2date -d, выполните следующую команду:

#up2date -i packages

Программа up2date скачивает RPM-пакеты с серверов Red Hat или Fedora по сети Internet, поэтому вам нужно будет зарегистрировать вашу систему на этих серверах в первый раз, когда вы выполните команду up2date.

Некоторые пользователи Linux предпочитают использованию up2date другие программы, например

yum (http://linux.duke.edu/projects/yum/) и
apt (http://ayo.freshrpms.net/).

rpm [опции][файлы] rpm
/bin stdin stdout -file -opt —help --version

Если вы предпочитаете устанавливать RPM-пакеты вручную, используйте программу rpm, тот же самый менеджер пакетов, который использует программа up2date. Программа rpm не только устанавливает программное обеспечение, но также и проверяет, что ваша система удовлетворяет всем предварительным условиям для установки. Например, если пакет supers tuff требует того, чтобы в системе был установлен пакет otherstuff, который вы не устанавливали, то программа rpm не установит superstuff. Если ваша система прошла этот тест, то rpm полностью установит новое программное обеспечение.

Имена RPM-файлов, как правило, имеют вид название-версия.архитектура.rpm. Например, файл emacs-20.7-l7.i386.rpm соответствует пакету emacs версии 20.7-17 для компьютеров

архитектуры i386 (Intel 386 и выше). Примите к сведению, что программа rpm иногда требует аргумент в виде имени файла (например, emacs-20.7-17.i386.rpm), а иногда просто название пакета (например, emacs).

Для работы с RPM-пакетами часто используются следующие команды:

rpm -q package_name

Сообщает информацию о том, установлен ли пакет раскаде_пате в вашей системе, и если да, то какой версии. Например: rpm -q textutils. Если вы не знаете названия пакета (как в известной проблеме, что появилось раньше - курица или яйцо?), выведите список всех пакетов и используйте команду grep для поиска пакетов с похожими названиями.

$ rpm -да | grep -i похожее_название
rpm -ql package_name

Выводит список файлов в указанном установленном пакете. Попробуйте выполнить, например, команду rpm -ql emacs.

rpm -qi package_name

Выводит общую информацию о пакете.

rpm -qlp package.rpm
Выводит содержимое RPM-файла, необязательно установленного. Используйте опцию -qip для получения общей информации о RPM-файле.

rpm -qa

Выводит список всех установленных RPM-паке-тов. Этот список можно перенаправить на вход команде grep для поиска названия нужного пакета.

$ rpm -qa | grep -i emacs

rpm -qf filename

Выводит название пакета, который установил указанный файл в вашей системе.

$ rpm -qf /usr/bin/who sh-utils-2.0-11

rpm -ivh packagel. rpm package2. rpm... Устанавливает пакеты, отсутствующие в вашей системе.

rpm -Fvh packagel. rpm package2. rpm... Обновляет пакеты, которые уже установлены в вашей системе.

rpm -e package__names

Удаляет пакеты из вашей системы. В этом случае не нужно указывать номер версии пакета, нужно задать только его название. Например, если вы установили пакет GNU Emacs из файла emacs-20.7-17. i386.rpm, то для его удаления нужно выполнить команду rpm -e emacs, а негрm -e emacs-20. 7-17.1386.rpm.

Файлы tar.gz и tar.bz2

Упакованные программные файлы с именами, оканчивающимися на .tar.gz и Jar.bz2, как правило, содержат исходный код пакета, который перед установкой вам нужно будет скомпилировать ("собрать").

1. Просмотрите содержимое пакета файл за файлом.

Убедитесь в том, что при распаковке ни одним из новых файлов вы не перепишете уже имеющиеся в системе, случайно или намеренно.

$ tar tvzf package, tar.gz I less Длягzip-файлов
$ tar tvjf package.tar.bz2 j less Дляbzipl-файлов

2. Если все нормально, распакуйте файлы в новую директорию.

$ mkdir newdir
$ cd newdir
$ tar xvzf package, tar . gz Для gzip-файлов

$ tar xvjf package, tar . bz2 Для bzip2-файлов

3. Найдите среди распакованных файлов файл INSTALL или README. Прочитайте его, чтобы узнать, как скомпилировать пакет.

$ cd newdir $ less INSTALL

4. Как правило, файлы INSTALL или README говорят о том, что нужно выполнить скрипт с на званием configure из текущей директории, затем выполнить команду make и затем – make install. Изучите, какие опции вы можете передавать скрипту configure.

$ ./configure --help

Затем установите программу.

$. /configure options
$ make
$ su -1
Password: ********
 # make install

Основные операции с файлами

Is	Вывести список файлов в директории
cp	Копировать файл
mv	Переименовать ("переместить") файл
rm	Удалить файл
In	Создать ссылки (альтернативные имена) на файл

Одна из первейших задач, с которыми вам придется столкнуться при работе в Linux-системе, это работа с файлами: копирование, переименование, удаление и т. д.

Is [опции][файлы] coreutils /bin stdin stdout -file —opt —help —version

Команда Is (произносится как "эл эс") выводит список атрибутов файлов и директорий. Вы можете вывести список файлов в текущей директории:

$ Is

в заданных директориях:

$ Is dirl dir2 dir3
или для отдельных файлов:
$ Is filel file2 file3

Наиболее важные опции - это -a и -1. По умолчанию Is скрывает файлы, имена которых начинаются с точки; опция -a отображает все файлы. Опция -1 осуществляет подробный вывод в од-ноколоночном формате:

-rw-r-r-- I smith users 149 Oct 28 2002 my.data

который включает в себя слева направо: права на файл (-rw-r--r —), владельца (smith), группу (users), размер (149 байт) дату последней модификации (Oct 28 2002) и имя. Обратитесь к разделу "Защита файлов" на странице 36 для более подробной информации о правах на файл.

Полезные опции

-a Вывести список всех файлов, включая файлы, имена которых начинаются с точки

-l Подробный вывод, включая атрибуты файлов. Добавьте опцию -h ("human-readable" - удобный для восприятия), чтобы размер файлов выводился в килобайтах, мегабайтах и гигабайтах вместо байтов

-F Пометить имена определенных файлов целевыми символами, указывающими на топ файлов. Приписывает"/" к иректориям, "*" к исполняемым файлам, "@" к символьным ссылкам, "|" к именованных каналам и "=" - к сокетам. Это просто визуальные индикаторы, которые на самом деле не являются частью имени файла

- i Добавить к выводу информационные дескрипторы (inode) файлов

- s Добавить размер файла в блоках, полезно для сортировки файлов по размеру:

$ Is -s | sort -n

-R При выводе содержимого директории вывести рекурсивно дерево подкаталогов с выдачей их содержимого
-d При выводе содержимого директории вывести имена каталогов так, как если бы они были обычными файлами, а не показывать их содержимое

cp [опции] файлы (файл \ директория) coreutils ~/bin stdin stdout -file -opt -help --versloti

Команда cp просто копирует файл:

$ cp filel file2

или копирует несколько файлов в директорию:

$ cp filel file2 file3 file4 dir

Используя опции -a или -R, вы также можете рекурсивно копировать директории.

Полезные опции

-p Копировать не только содержимое файла, но и права на файл, временные метки и, если у вас достаточно прав на это, его владельца и группу (как правило, владельцем копии будете вы, временной меткой будет момент копирования, а права будут установлены на основе оригинальных прав файла с применением к ним маски режима создания файла для пользователя)
- a Копировать иерархию директорий рекурсивно, сохраняя специальные файлы, права, символьные ссылки и жесткие ссылки. Эта опция комбинирует опции -R (рекурсивное копирование, включая специальные файлы), -p (права) и -d (ссылки)
- i Интерактивный режим. Выдать предупреждение прежде чем переписать существующий файл
- f Перезаписывать файлы при копировании (если такие уже есть) без дополнительных предупреждений

mv [опции] источник цель coreutils /bin stdin stdout -file --opt -help —version

Команда mv может либо переименовывать файлы:

$ mv filel file2

либо перемещать файлы и директории в заданную директорию:

$ mv filel file2 file3 file4 destination_directory 68

Полезные опции

-i Интерактивный режим. Выдать предупреждение, прежде чем переписать существующий файл

- f Перезаписывать файлы при копировании (если такие уже есть) без дополнительных предупреждений

rm [опции] файлы \ директории coreutils /bin stdin stdout -file —opt --help -version

Команда rm может либо удалять файлы:

$ rm filel file2 file3

либо рекурсивно удалять директории:

$ rm -r dirl dir2

Полезные опции

-i Интерактивный режим. Выдать предупреждение, прежде чем
удалить какой-либо файл

- f Удалять файлы, игнорируя любые ошибки и предупреждения

-r Рекурсивно удалить директорию и ее содержимое. Используйте с осторожностью, в особенности в комбинации с опцией - f

ln [опции] источник цель **coreutils**
/bin stdin stdout -file -opt -help -version

Ссылка (link) - это созданный командой ln указатель на другой файл. Существует два вида ссылок. Символьная ссылка указывает на файл по его пути точно так же, как "ярлык" в Windows или "псевдоним" в Macintosh.

$ ln -s myfile softlink

Если вы удаляете оригинальный файл, то такая ссылка станет неработоспособной, будет указывать на несуществующий файл. С другой стороны, жесткая ссылка - это просто второе название физического файла на диске (строго говоря, она указывает на тот же самый информационный дескриптор). Удаление оригинального файла не делает такую ссьшку неработоспособной.

$ ln myfile hardlink

Символьные ссылки могут указывать на файл, расположенный на другом разделе диска, так как они являются всего лишь ссылками на файл по пути; жесткие ссылки этого не могут, так как номер информационного дескриптора (inode) на одном разделе не имеет смысла на другом. Также символьные ссылки могут указывать на директории, тогда как жесткие - не могут... только если вы не суперпользователь и не используете опцию -d.

Полезные опции

-s Создать символьную ссылку. По умолчанию - жесткую ссылку
-i Интерактивный режим. Выдать предупреждение, прежде чем переписать существующий файл
-f Перезаписывать файл (если такой уже есть) без дополнительных предупреждений
-d Позволяет суперпользователю создать жесткую ссылку на директорию

Очень просто узнать, куда указывает символьная ссылка (скажем, linkname), с помощью любой из следующих команд.

$ readlink linkname $ Is -1 linkname

Работа с директориями

cd Сменить вашу текущую директорию

pwd Вывести имя вашей текущей директории, т. е. "где вы находитесь сейчас" в файловой системе

basename Вывести последнюю часть пути к файлу (т. е. имя файла без имени директории, в которой он находится)

dirname Отбросить последнюю часть пути к файлу (т. е. вывести только имя директории, в которой находится файл)

mkdir Создать директорию

rmdir Удалить пустую директорию

rm –r Удалить непустую директорию и ее содержимое

Мы рассмотрели структуру директорий в Linux в разделе "Файловая система" на странице 27. Теперь мы рассмотрим команды, с помощью которых можно создавать, изменять, удалять и управлять директориями в рамках этой структуры.

cd [директория] bash
встроенная команда stdin stdout -file --opt --help -version

Команда cd (сменить директорию) устанавливает вашу текущую рабочую директорию. Если вы не укажете директорию, то по умолчанию команда cd сменит текущую директорию на вашу домашнюю директорию.

pwd bash
встроенная команда stdin stdout -file --opt --help --version

Команда pwd выводит абсолютный путь вашей текущей рабочей директории.

$ pwd
/users/smith/mydir

basename путь coreutils
/bin stdin stdout -file -opt —help —version

Команда basename выводит последний компонент пути к файлу; поэтому, например, для примера, показанного выше, результат будет следующим.

$ basename /users/smith/mydir
mydir

dirname путь coreutils
/usr/bin stdin stdout -file --opt —help —version

Команда dirname отбрасывает последний компонент пути к файлу.

$ dirname /users/smith/mydir /users/smith

Команда dirname просто обрабатывает строку, которая является именем директории. Она не изменяет вашу текущую рабочую директорию.

mkdir [опции] директории coreutils
/bin stdin stdout -file —opt —help —version
Команда mkdir создает одну или несколько директорий.

$ mkdir dl d2 d3

Полезные опции

-p Если вы указываете путь к директории (а не просто имя директории), то команда создаст все необходимые родительские директории автоматически. Например, командаткс Иг -p /one/two/three создаст директории /one и /one/two, если они не существуют, а затем и саму директорию /one/two/three

-m права Создать директорию с заданными правами доступа: $ mkdir -m 0755 mydir По умолчанию, переменная umask вашего командного процессора управляет правами доступа. Обратитесь к информации о команде chmod в разделе "Свойства файла" на странице 95, и разделу "Защита файлов" на странице 36.

rmdir[опции] директории coreutils /bin stdin stdout -file --opt -help -version

Команда rmdir (удалить директорию) удаляет одну или более пустых директорий, которые вы указываете. Чтобы удалить непустую директорию и ее содержимое, используйте (только с осторожностью) команду rm -г директория. Используйте команду rm -ri директория для интерактивного удаления или команду rm -rf директория для того, чтобы игнорировать любые сообщения об ошибках и другие уведомления во время удаления.

Полезные опции

-p Ели вы указали путь к директории (а не только имя директории), то команда удалит не только заданную директорию, но и указанные родительские директории автоматически, каждая из этих директорий должна быть пустой. Поэтому команда rmdir -p /one/two/ three удалит не только директорию /one/tmflhree, нО также и директории /one/two и /one, если они существуют

Просмотр файлов

cat Просмотреть файлы целиком
less Просмотреть файлы постранично
head Просмотреть первые строки файла
tail Просмотреть последние строки файла
nl Просмотреть файлы с пронумерованными строками
od Просмотреть данные в восьмеричной системе (или других форматах)
xxd Просмотреть данные в шестнадцатеричной системе

gv Просмотреть Postscript- или PDF-файлы
xdvi Просмотреть TeX DVI-файлы.

В Linux вам нужно будет просматривать различные типы файлов: текст, Postscript, двоичные данные и множество других. Ниже мы рассмотрим, как их просматривать. Обратите внимание на то, что команды для просмотра графических файлов рассмотрены в разделе "Графика и хранители экрана" на странице 246, а для аудио-файлов - в разделе "Аудио и видео" на странице 251.

cat [опции] [файлы] coreutils
/bin stdin stdout -file --opt^help --version

Простившей программой просмотра является cat, которая просто копирует файлы в стандартный поток вывода, объединяя их (объединять по-английски - concatenate, отсюда название). Обратите внимание на то, что большие файлы, скорее всего, не будут умещаться на экране, поэтому подумайте об использовании для просмотра команды less. Вообще говоря, cat в основном полезен для отправки набора файлов в конвейер.

Также cat умеет управлять своими выходными данными, опционально отображая непечатаемые символы, номера строк (хотя команда nl больше подходит для этой цели) и игнорируя пробелы.

Полезные опции

-T Выводить знаки табуляции в виде "I
-E Выводить символы новой строки в виде $
-v Выводить другие непечатаемые символы в удобном для восприятия формате
-n Присоединять в начале каждой строки ее номер
-b Присоединять в начале каждой непустой строки ее номер
-s Заменить каждую последовательность пустых строк одной пустой строкой

less [опции] [файлы] less
/usr/bin stdin stdout* -file —opt -help —version

Используйте команду less для постраничного просмотра текста. Она хорошо подходит для чтения текстовых файлов или в качестве последней команды в конвейере командного процессора с большими выходными данными.

$ command1 | command2 | command3 | command4 | less

При выполнении команды less нажмите h для вывода справочного сообщения, описывающего все ее возможности. Вот некоторые полезные клавиши для постраничного просмотра файлов.

Клавиша или комбинация	**Функция**
h, н	Просмотреть справочную страницу
Пробел, f, ~V, AF	Перейти на один экран вперед
Enter	Перейти вперед на одну строку
Ь, Лв, Esc-b	Вернуться назад на один экран
/	Перейти в режим поиска. Укажите после этого регулярное выражение и нажмите Enter, и less будет искать первую соответствующую этому выражению строку
?	То же самое что и /, но поиск будет производиться в обратном направлении по файлу
n	Повторить последний поиск
N	Повторить последний поиск в обратном направлении
v	Редактировать текущий файл вашим стандартным текстовым редактором (значение переменной окружения VISUAL, ИЛИ, если она не определена, переменной EDITOR, или, если и она не определена, то будет использован vi
<	Перейти в начало файла
>	Перейти в конец файла
: n	Перейти к следующему файлу

: p Перейти к предыдущему файлу

* Хотя технически команду less можно вставить в середину конвейера либо перенаправить ее выходные данные в файл, особого смысла делать это нет.

Команда less имеет невероятное число функций; мы рассмотрим только самые общие из них. Рекомендуется обратиться к man-странице.

Полезные опции

-c Очищать экран перед тем, как отобразить следующую страницу
-m Вывод информации о том, какая часть файла выведена к настоящему моменту (в процентах)
-N Выводить номера строк
-r Выводить управляющие (непечатаемые) символы; как правило, less преобразует их в удобный для восприятия формат
-s Объединять несколько соседних пустых строк в одну
-s Урезать длинные строки до длины экрана вместо переноса

head [опции] [файлы] coreutils
/usr/bin stdin stdout -file —opt -help --version

Команда head печатает первые 10 строк файла: она полезна для предварительного просмотра содержимого файлов.

$ head myfile

$ head * | less Просмотреть первые строки всех файлов в текущей директории

Полезные опции

-N Вывести первые N строк вместо 10
-n N
-c N Вывести первые N байт файла

-q Тихий" режим: при обработке более чем одного файла не
выводить заголовок над каждым файлом. Обычно head выводит заголовок, содержащий имя файла

tail [опции] [файлы] coreutils /usr/bin stdin stdout -file --opt --help -version

Команда tail выводит последние 10 строк файла, а также имеет другие функции.

$ tail myfile

Полезные опции

-N Вывести последние N строк файла вместо 10
-n N
+N Вывести все строки, кроме первых /Устрок
-c N Вывести последние ^байт файла
- f Оставить файл открытым и, если в него будут затем добавляться строки, выводить их. Это очень полезно. Добавьте опцию - - retry, если файл еще не существует, но вы ожидаете его появления
-q Тихий" режим: при обработке более чем одного файла не
выводить заголовок над каждым файлом. Обычно tail выводит заголовок, содержащий имя файла

nl [опции] [файлы] coreutils usr/bin stdin stdout -file -opt --help^-version

Команда nl копирует указанные ей файлы в стандартный поток вывода, добавляя перед каждой строкой ее номер. Это более удобно, чем использовать команду cat с опциями -п и -Ь, и эта команда предоставляет огромный контроль над нумерацией строк. Команду nl можно использовать двумя способами: с простыми текстовыми файлами и со специальными размеченными текстовыми файлами с предопределенными верхними и нижними колонтитулами.

Полезные опции

-b [a|t|n|pR] Добавлять номера ко всем строкам (a), непустым строкам (t), не нумеровать строки (п) или нумеровать только те строки, которые содержат регулярное выражение Я (по умолчанию - а)

-v N Начать нумерацию с целого числа iY(no умолчанию с 1)

-i N Увеличивать номер на Л'для каждой следующей

строки, то есть, например, вы можете использовать только нечетные номера (-12) или только четные (-v2 -i2) (по умолчанию - 1)

-n [ln|rn|rz] Выравнивать номера влево (In), вправо (rn) или

вправо с добавлением нулей в начале (rz) (по умолчанию - In)

-w N Ширина номера должна равняться N колонкам

(по умолчанию - 6)

-s S Вставить строку 5между номером строки и текстом (по умолчанию - TAB)

Кроме этого, команда nl имеет интересную способность делить текстовые файлы на виртуальные страницы, каждую со своим верхним колонтитулом, телом и нижним колонтитулом с разными схемами нумерации. Однако для этого вы должны вставить специфичные для nl разделительные строки в файл: \:\:\: (начало верхнего колонтитула), \:\: (начало тела) и \: (начало нижнего колонтитула). Каждая из них должна вставляться на отдельной строке. Затем вы можете использовать дополнительные опции (обратитесь к man-странице), чтобы задать схему нумерации в колонтитулах ваших оформленных файлов.

od [опции] [файлы] coreutils
/usr/bin stdin stdout -file --opt --help --version

Когда вы хотите просмотреть двоичный файл, обратитесь к команде od (восьмеричный вывод). Она копирует один или более файлов в стандартный поток вывода, отображая данные в ASCII-, восьмеричном, десятичном, шестнадцатеричном форматах и в формате с плавающей точкой, различных размеров (byte, short, long). Например, следующая команда:

```
$ od -w8 /usr/bin/who

0000000 042577 043114 000401 000001
0000010 000000 000000 000000 000000
0000020 000002 000003 000001 000000
0000030 106240 004004 000064 000000
```

отображает байты двоичного файла /usr/bin/who в восьмеричном формате, по восемь байт в строке. Колонка слева содержит смещение от начала файла для каждой строки, опять же, в восьмеричном формате.

Полезные опции

-N B — Вывести только первые 5байт каждого файла, в десятичном, шестнадцатеричиом (с префиксом Ох или ох) форматах, в блоках по 512 байт (с суффиксом Ь), килобайтах (с суффиксом к) или мегабайтах (с суффиксом т). По умолчанию выводится весь файл

-j B — Начать вывод с (5+1)-го байта каждого файла; форматы такие же, как и у опции -N. По умолчанию вывод с 0-го байта

-w [B] — Выводить по В байт в строке; форматы такие же, как и у опции -N. Использование -w без аргументов эквивалентно опции -w32. По умолчанию выводится по 16 байт в строке

-s [B] — Группировать данные в строках в группы по 5байт, разделенные пробелами; форматы такие же, как и у опции -N. Использование -s без аргументов эквивалентно опции - s 3. По умолчанию группировка по 2 байта

-A (d|o|x|n) Отображать смещение от начала файла в самой левой колонке в десятеричном (d), восьмеричном (o), шестнадцатеричном (h) форматах или не отображать совсем (п). По умолчанию - o

-1 (a | c) [z] Вывести данные в символьном формате, причем непечатаемые символы выводить в виде ESC-последовательностей (a), либо в виде названий (c). Информация по параметру z дана ниже

-t (d| o |u | x) [SizElz]] Вывести данные в целочисленном формате, включая восьмеричный (o), десятичный со знаком (d), десятичный без знака (и), шестнадцатеричный (x) (для вывода в двоичном представлении используйте команду xxd). ^^представляет собой размерность целого числа в байтах; она может быть положительным целым числом, либо любым из значений C, S, I и L, которые означают размерность типов данных char, short, int и long соответственно. Информация по параметру z дана ниже.

-1 f [SIZE [z]] Вывести данные в формате с плавающей точкой. 5/7£представляет собой размерность целого числа в байтах; она может быть положительным целым числом, либо любым из значений F, D или Ц которые означают размеры типов данных float, double или long double соответственно. Информация по параметру z дана ниже. Если опция -1 опущена, то по умолчанию используется опция -to2. Если добавить z к параметру опции -t, то будет выведена новая колонка справа, отображающая печатаемые символы для каждой строки, почти как и в стандартных выходных данных команды xxd

xxd [опции] [файлы] vim-common
/usr/bin stdin stdout -file --opt -help -version

Аналогично команде od, xxd осуществляет шест-надцатеричный или двоичный вывод файла в различных форматах. Также она может делать обратную задачу, преобразовывать из

шестнадцатерично-го представления обратно в исходный вид. Например, команда:

```
$ xxd /usr/bin/who

0000000: 7f45 4c46 0101 0100 0000 0000
0000 0000 .ELF
0000010: 0200 0300 0100 0000 a08c 0408
3400 0000 4. . .
0000020: 6824 0000 0000 0000 3400 2000
0600 2800 h$ 4. . . . (.
0000030: 1900 1800 0600 0000 3400 0000
3480 0408 4. . .4. . .
```

осуществляет шестнадцатеричныи вывод двоичного файла /usr/bin/who, по 16 байт в строке. Колонка слева указывает на смещение от начала файлы для строки, следующие восемь колонок содержат данные, а последняя колонка отображает печатаемые символы в строке, если таковые имеются.

Команда xxd осуществляет трехколоночный вывод по умолчанию: смещение от начала файла, данные в шестнадцатеричном представлении и данные в виде текста (только печатаемые символы).

Полезные опции

- i N — Вывести только первые Мбайт (по умолчанию выводится весь файл)
- s N — Выводить не с начала файла. Первый вариант пропускает первые
- s -N — N байт. Второй вариант (-N) выводит /Убайт с конца файла (также существует вариант +#для более сложной обработки стандартного входного потока; обратитесь к man-странице)
-c N — Выводить по/Убайт в строке (по умолчанию выводится по 16 байт в строке)
-g N — Группировать данные в строках в группы по N байт, разделенные пробелами, аналогично команде od -s (no умолчанию группировка по 2 байта)

- b Выводить данные в двоичном, а не в шестнадцатеричном формате

-u Выводить данные в шестнадцатеричном формате в верхнем, а не в нижнем регистре

-p Выводить данные в простом шестнадцатеричном формате, по 60 байт подряд в строке

- i Выводить данные в виде структуры данных языка программирования С. При чтении из файла создается массив элементов типа unsigned chars, содержащий данные, и элемент типа unsigned int, содержащий длину массива. При чтении из стандартного потока ввода создается только разделенный запятыми список байт в шестнадцатеричном формате

-r Обратная операция: преобразовать из шестнадцатеричного вывода xxd обратно в исходный формат файла. Работает со стандартным форматом шестнадцатеричного вывода и, при добавлении опции-p, с простым форматом шестнадцатеричного вывода. Если вам скучно, попробуйте использовать какую-нибудь из этих команд для преобразования и обратного преобразования файла через конвейер, воссоздавая исходный файл в стандартном потоке вывода:

$ xxd myfile | xxd -r
$ xxd -p myfile | xxd -r -p

gv [опции] файл gv
/usr/X11R6/bin stdin stdout -file --opt --help --version

Программа GhostView отображает Adobe Postscript- или PDF-файлы в графическом окне. Вы можете вызывать ее с помощью команд gv или ghostview. Принцип управления этой программой прост: щелкните по номеру нужной страницы, чтобы перейти к ней. После нескольких минут работы вы освоитесь в программе.

GhostView - это приоритетная программа просмотра Postscript-файлов для Linux, но также существуют другие свободно распространяемые программы для просмотра PDF-файлов, например acroread (http://www.adobe.com/) и xpdf

(http:// www.foolabs.com/xpdf/).

Полезные опции

-page P	Начать с P-п страницы (по умолчанию с 1-й)
-monochrome	Задать режим вывода:монохромный, оттенки серого
-grayscale	или цветной соответственно
-color	
-portrai t	Выбрать ориентацию страницы; как правило,
- landscape	программа gv определяет это автоматически
-seascape	
-upsidedown	
-scale N	Задать коэффициент масштабирования для вывода. Целое число Сможет быть положительным (изображение будет больше) или отрицательным (меньше).
-watch	При изменении Postscript-файла автоматически
-nowatch	перезагружать его (watch), либо не делать этого
(nowatch)	

xdvi [опции] файл tetex-xdvi
/usr/bin stdin stdout -file --opt -help -version

Система обработки документов TeX создает двоичные выходные файлы в формате DVI, с расширением .dvi. Программа просмотра xdvi выводит DVI-файлы в графическом окне. При желании вы можете преобразовать DVI-файл в Postscript-файл с помощью команды dvips, а затем использовать программу GhostView (gv) для его просмотра:

$ dvips -o myfile.ps myfile.dvi
$ gv myfile.ps

При выводе файла программа xdvi предоставляет колонку кнопок справа с очевидными функциями, например, Next (следующая страница) для перехода к следующей странице (вы можете скрыть кнопки, вызвав программу xdvi с опцией -expert). Также вы можете перемещаться по файлу с помощью клавиш.

Комбинация клавиш	Функция
q	Выход
n, пробел, Enter, Pagedown	Перейти к следующей странице. Наберите перед нажатием число N, чтобы переместиться на N страниц вперед
p, Backspace, Delete, Pageup	Перейти к предыдущей странице. Наберите перед нажатием число N, чтобы переместиться на N страниц назад
<	Перейти к первой странице
>	Перейти к последней странице
^L	Обновить страницу
R	Перечитать DVI-файл, скажем, после того, как вы его изменили
Нажатие кнопок мыши	Увеличить прямоугольную область под курсором мыши

Программа xdvi имеет множество командных опций для изменения цветов, геометрии, масштаба и поведения программы в целом.

Создание и редактирование файлов

emacs	Текстовый редактор от Free Software Foundation
vim	Текстовый редактор, расширение редактора Unix vi
umask	Задать маску правдоступа для новых файлов и директорий
soffice	Офисный пакет для редактирования документов Microsoft Word, Excel и PowerPoint
abiword	Редактировать документ Microsoft Word gnumeric Редактировать электронную таблицу Excel

Чтобы освоиться в Linux, вы должны научиться работать с одним из его текстовых редакторов. Два основных редактора - это emacs от Free Software Foundation и vim, последователь Unix-редактора vi. Полное изучение этих редакторов выходит за рамки

этой книги, но каждый из них имеет online-cnpa-вочники и учебные пособия, а мы перечислим наиболее общие операции в табл. 1. Для того чтобы отредактировать файл myfile, запустите любую из следующих команд.

```
$ emacs myfile
$ vim myfile
```

Если файл myfile не существует, то он будет автоматически создан. Также вы можете быстро создать пустой файл (чтобы позднее его отредактировать), используя команду touch (обратитесь к разделу "Свойства файла" на странице 95):

```
$ touch newfile
```

либо вы можете сразу записать данные в новый файл, перенаправив в него выходные данные какой-нибудь программы (обратитесь к разделу "Перенаправление ввода/вывода" на странице 47):

```
$ echo Мама мыла раму > newfile
```

На случай, если вы обмениваетесь файлами с Microsoft Windows-системами, мы рассматриваем Linux-программы, в которых можно редактировать документы Microsoft Word, Excel и PowerPoint.

Ваш стандартный редактор

Различные программы в Linux запускают в случае необходимости текстовый редактор, и по умолчанию этим редактором является vim. Например, ваша почтовая программа может вызвать редактор для создания нового сообщения, а программа less вызывает редактор, когда вы набираете "v". Но что, если вы не хотите, чтобы vim был вашим стандартным редактором (т. е. чтобы по умолчанию открывался другой редактор)? Установите переменные окружения VISUAL и EDITOR по своему усмотрению, например, следующим образом.

```
$ EDITOR=emacs
```

$ VISUAL=emacs
$ export EDITOR VISUAL Необязательно

Необходимо задать обе эти переменные, так как различные программы проверяют либо одну, либо другую из них. Установите переменные EDITOR и VISUAL в вашем файле инициализации ~/.bash_profile, если хотите, чтобы ваши настройки стали постоянными. Любую программу можно сделать вашим стандартным редактором, лишь бы она в качестве аргумента принимала имя файла.

Независимо от того, как вы установите эти переменные, любой системный администратор должен знать по крайней мере основные команды редакторов vim и emacs на случай, если какая-нибудь системная программа вдруг запустит редактор для редактирования принципиально важного файла.

emacs [опции] [файлы] emacs
/usr/bin stdin stdout -file --opt -help --version

Программа emacs - это чрезвычайно мощная среда редактирования, в которой команд больше, чем вы можете себе представить, а также она имеет встроенный язык программирования для определения ваших собственных свойств редактирования. Для того чтобы вызвать учебник emacs, выполните команду:

$ emacs

и наберите ^h t.

Большинство функциональных комбинаций клавиш emacs включают в себя клавишу CTRL (например, комбинация "F) или метаклавишу, как правило, ее роль играет Alt. В собственной документации emacs метаклавиша сокращенно обозначается с помощью M- (например, M-F означает "нажать метаклавишу и, удерживая ее, нажать F"), так она будет обозначаться и ниже. Информация об основных комбинациях клавиш приведена в табл. 1.

vim [опции] [файлы] vim-enhanced
/usr/bin stdin stdout -file -opt -help -version

vim - это усовершенствованная версия старого стандартного Unix-редактора vi. Чтобы открыть учебник по vim, выполните команду:

$ vimtutor

vim может работать в двух режимах: в режиме ввода (insert) и в нормальном режиме (normal). Режим ввода предназначен для набора текста обычным способом, тогда как нормальный режим предназначен для выполнения таких команд, как "удалить строку" или копировать/вставить. Основные клавишные комбинации для работы в нормальном режиме приведены в табл. 1.

Таблица 1. Основные клавишные комбинации в редакторах emacs и vim

Таблица 1. Основные клавишные комбинации в редакторах emacs и vim

Функция	**Emacs**	**vim**
Открыть редактор в текущем окне	$ emacs -nw *[file]*	$ vim [file]
Открыть редактор в новом графическом окне	*$* emacs *[file]*	$ gvim [file]
Набрать строку text	*Text*	i text ESC
Сохранить и выйти!	^x^s затем ^x^c	:wq
Выйти без сохранения изменений	^x^c Ответить "нет" при появлении вопроса о сохранении буферов	:q!
Сохранить	^x^s	**:** w
Сохранить как	^x^w	: wимя_файла
Отменить действие	^_	u
Приостановить редактор (не для	^z	^z
Переключиться в режим	*(N/A)*	ESC

Переключиться в командный режим	M-x	:
Отменить выполнение команды	^q	ESC
Перейти вперед	^f или стрелка вправо	1 или стрелка
Перейти назад	^b или стрелка влево	h или стрелка

Таблица 1. Основные клавишные комбинации в редакторах emacs и vim (Продолжение)

Функция	emacs	vim
Перейти вверх	^p или стрелка вверх	r или стрелка вверх
Перейти вниз	^n или стрелка вниз	j или стрелка вниз
Перейти к следующему Слову	м- f	w
Перейти к предыдущему Слову	м-b	b
Перейти в начало строки	^a	0
Перейти в конец строки	^e	$
Перейти вниз на один Экран	^v	^f
Перейти вверх на один Экран	M-v	^b
Перейти в начало	м-< буфера	gg
Перейти в конец буфера	м->	*G*
Удалить следующий Символ	^d	x
Удалить предыдущий Символ	Backspace	X
Удалить следующее Слово	M-d	de

Удалить предыдущее Слово	M-Backspace	db
Удалить текущую строку	^a^k^k	dd
Удалить до конца строки	^к	d$
Определить область (Нажмите эту кобинацию для того, чтобы пометить начало области, а затем переместите курсор в конец нужной обаслти)	^пробел	V

Таблица 1. Основные клавишные комбинации в редакторах emacs и vim (Продолжение)

Функция	**emacs**	vim
Вырезать область	^w	**D**
Копировать область	M-w	У
Вставить область	^У	Р
Вызвать справку	^**h**	:help
Вызвать руководство пользователя	^**h i**	:help

umask *[опции] [маска]* bash
встроенная команда stdin stdout -file --opt --help --version

Команда umask устанавливает или выводит на экран маску прав доступа. При создании вами файлов или директорий она определяет права на чтение, запись и выполнение для вас, вашей группы и всех остальных (для более подробной информации обратитесь к разделу "Защита файлов" на странице 36 и параграфу, посвященному команде chmod, в разделе "Свойства файлов" на странице 95).

$ umask
0002 " ''' ''' '
$ umask -S
u=rwx,g=rwx,o=rx

Сначала несколько технических деталей. Значение umask - это маска, т. е. двоичное значение, которое комбинируется (посредством двоичной операции XOR) со значением Об б б для файлов и 0777 для директорий, в результате чего получаются права доступа по умолчанию . Например, операция 0002 XOR 0666 дает права 0664 для файлов, а 0002 XOR 0777 - режим 0775 для директорий.

Если такое объяснение вам не понятно, то вот простой рецепт. Используйте маску 0022, чтобы предоставить себе полные права, а всем остальным -права только на чтение/исполнение.

$ umask 0022
$ touch newfile && mkdir dir
$ Is -Id newfile dir
-rw-r--r— 1 smith smith 0 Nov 11 12:25 newfile
drwxr-xr-x 2 smith smith 4096 Nov 11 12:25 dir

Чтобы предоставить себе и вашей группе полные права, а всем остальным права только на чтение/исполнение, используйте маску 0 002.

$ umask 0002
$ touch newfile && mkdir dir
$ Is -Id newfile dir
-rw-rw-r-- 1 smith smith 0 Nov 11 12:26 newfile
drwxrwxr-x 2 smith smith 4096 Nov 11 12:26 dir

Чтобы предоставить себе полные права, а всем остальным не предоставлять никаких прав, используйте маску 0077.

$ umask 0077
$ touch newfile && mkdir dir
$ Is -Id newfile dir
-rw 1 smith smith 0 Nov 11 12:27 newfile

drwx 2 smith smith 4096 Nov 11 12:27 dir

soffice [файлы] openoffice.org
/usr/lib/openoffice/programs stdin stdout -file -opt --help -version
OpenOffice.org* - это комплексный интегрированный офисный программный пакет, в котором можно редактировать файлы Microsoft Word, Excel и PowerPoint. Просто выполните следующую команду:

$ soffice

и можно приступать к работе. В одной и той же программе можно редактировать все три типа файлов t. Это большая программа, которая требует много памяти и дискового пространства.

Также пакет OpenOffice.org умеет работать с изображениями (команда sdraw), факсами (sfax), почтовыми марками (slabel), и т. д. На сайте http:/ /www.openoffice.org/ можно найти много информации, либо вы можете использовать меню Help программы soffice.

abiword [опции][файлы\ abiword
/usr/bin stdin stdout -file --opt --help --version

abiword - это еще одна программа для редактирования документов Microsoft Word. Она меньше и быстрее пакета soffice, хотя и не такая мощная, и идеально подходит для многих задач редактирования. Если вы задаете файлы в командной строке, то они должны существовать: программа abiword не создаст файлы автоматически.

* "org" - это часть названия программного пакета.

t Пакет soffice включает в себя отдельные программы Writer (команда swr i ter) для работы с текстом, Calc (scale) для работы с электронными таблицами и impress (simpress) для создания презентаций, которые при желании вы можете запускать отдельно.

gnumeric [опции][файль/\ gnumeric
/usr/bin stdin stdout -file --opt —help —version

gnumeric - это программа для работы с электронными таблицами, в которой можно редактировать документы Microsoft Excel. Она довольно мощная и быстрая, и если вы использовали до этого программу Excel, то в gnumeric вы встретите много знакомого. Если вы задаете файлы в командной строке, то они должны существовать: программа gnumeric не создаст файлы автоматически.

Свойства файла

stat	Вывести атрибуты файлов и директорий
we	Подсчитать количество байт, слов, строк в файле,
du	Показать, сколько места на диске занимают файлы и директории
file	Определить (попытаться угадать) тип файла
touch	Изменить временные метки файлов или директорий
chown	Изменить владельца файлов или директорий ~
chgrp	Изменить принадлежность группедля файлов или директорий
chmod	Изменить права доступа файлов или директорий
chattr	Изменить расширенные атрибуты файлов или директорий
Isattr	Вывести список расширенных атрибутов файлов или директорий

При изучение файлов в Linux их содержимое - это только половина предмета. Каждый файл и директория, помимо прочего, имеют атрибуты, которые описывают владельца, размер, права доступа и другую информацию. Команда Is -1 (см. раздел "Основные операции с файлами" на странице 66) выводит некоторые из этих атрибутов, а для просмотра остальных атрибутов используются описанные ниже команды.

stat [опции] файлы coreutils
/usr/bin stdin stdout -file --opt -help -version

Команда выводит список важных атрибутов файлов (по умолчанию) или файловых систем (опция -f). Информация о файле выглядит так:

$ stat myfile
File: "myfile"

Size: 1264 Blocks: 8 Regular File
Access: (0644/-rw-r--r--) Uid: (600/smith)
Gid: (620/users)
Device: 3 0a Inode: 99492 Links: 1
Access: Fri Aug 29 00:16:12 2003
Modify: Wed Jul 23 23:09:41 2003
Change: Wed Jul 23 23:11:48 2003

и включает в себя имя файла (myfile), его размер в байтах (1264), размер в блоках (8), тип файла (Regular File), права на файл в восьмеричной системе (0644), права в формате команды "Is -I" (-rw-r--r--), идентификатор владельца (600), имя владельца (smith), идентификатор группы (620), имя группы (users), тип устройства (30a), номер информационного дескриптора (99492), количество жестких ссылок (1) и временные метки последних операций, проделанных с файлом (чтения, модификации и изменения статуса). Информация о файловой системе выглядит так:

$ stat -f myfileb
File: "myfile"
ID: bffff358 ffffffff Namelen: 255 Type: EXT2
Blocks: Total: 2016068 Free: 876122
Available: 773709 Size: 4096
Inodes: Total: 1026144 Free: 912372

и включает в себя имя файла (myfile), идентифика-торфайловой системы (bffff358 ffffffff), максимальную длину имени файла для данной файловой системы (255 байт), тип файловой системы (EXT2), полное количество блоков, количество свободных и доступных блоков в этой файловой системе (2016068, 876122, и 773709 соответственно), размер блока для этой файловой системы (4096), полное количество и количество свободных информационных дескрипторов (1026144 и 912372 соответственно).

Опция -t позволяет получить те же самые данные, но в одной строке, без заголовков. Это удобно для обработки информации с помощью shell-скриптов или других программ.

$ stat -t myfile

myfile 1264 8 81a4 500 500 30a 99492 1 44
le 1062130572 1059016181 1059016308
$ stat -tf myfile
myfile bffff358 ffffffff 255 ef53 2016068
875984 773571 4096 1026144 912372

Полезные опции

-l Вычислять символические ссылки и работать с файлами, на которые они указывают
- f Сообщать о файловой системе, в которой находится файл, а не о самом файле
-t Краткий режим: вывести информацию в одной строке

we [опции] [файлы] coreutils
/usr/bin stdin stdout -file -opt -help --version

Программа we (от англ. word count) выводит количество байт, слов и строк в текстовом (хотя и необязательно) файле.

$ we myfile
24 62 428 myfile

В этом файле 24 строки, 62 слова, разделенных пробелами, и 428 байт.

Полезные опции

- l Выводит только количество строк.
-w Выводит только количество слов
-c Выводит только количество байт (символов)
- L Найти самую длинную строку в каждом файле и вывести ее длину в байтах

du [опции] [файлы\ директории] coreutils
/usr/bin stdin stdout -file -opt -help -version

Команда du (от англ. disk usage) вычисляет объем дискового пространства, занимаемого файлами или директориями. По умолчанию, она подсчитывает пространство, занимаемое текущей

директорией и всеми ее поддиректориями, выводя занимаемый объем для каждой из них и общий занимаемый объем в конце.

```
$ du
8./Notes
36      ./Mail
340     ./Files/mine
40      ./Files/bob
416     ./Files
216     . /PC
2404
```

Однако также она умеет измерять размер файлов.

```
$ du  file   file2
4./file
16      ./file2
```

Полезные опци

-b -k -m Вычислить объем занимаемого дискового пространства в байтах (-Ь), килобайтах (-к) или мегабайтах (-т)

-в N Вывести размеры в заданных вами блоках, где 1 блок = N байт (по умолчанию 1 блок = 1024)

-h -H Выводить в удобном для восприятия формате и выбирать наиболее подходящую единицу измерения для каждого размера. Например, если две директории имеют размеры 1 гигабайт и 25 килобайт соответственно, то команда du -h напечатает 1G и 25K. Опция -h использует степени 1024, тогда как опция -н использует степени 1000

-c Вывести общий размер в последней строке. Это стандартное поведение при вычислении размера директории, но при вычислении размера отдельных файлов укажите опцию -c, если хотите увидеть их общий размер

- L Вычислять размер файлов, на которые указывают символьные ссылки

- s Выводить только общий размер

file [опции] файлы file
/usr/bin stdin stdout -file --opt -help —version
Команда file сообщает о типе файла.

$ file /etc/hosts /usr/bin/who letter.doc
/etc/hosts: ASCII text
/usr/bin/who: ELF 32-bit LSB executable, Intel 803 86 ...
letter.doc: Microsoft Office Document

В отличие от некоторых других операционных систем Linux не отслеживает типы файлов, поэтому выходные данные команды - догадка, основанная на содержимом файла и других факторах.

Полезные опции

-b Опустить имена файлов (правая колонка в выходных данных)
-i Вывести MIME-типы файла, такие, как "text/plain" или "audio/mpeg", вместо обычных выходных данных.
-f name_file Считывать имена файлов, построчно, из заданного файла name_file (и сообщать об их типах), и после этого обработать имена файлов, заданные в командной строке, как обычно
-L Сообщать о типе файла, на который ссылается символьная ссылка, а не о самой ссылке
- z Если файл сжат (см. раздел "Сжатие и упаковка файлов" на странице 134), изучить несжатое содержимое, чтобы найти информацию о типе файла, вместо того чтобы просто сооб1щгь, что это "сжатые данные"

touch [опции] файлы coreutils
/bin stdin stdout -file -opt -help -version
Команда touch изменяет две временные метки, связанные с файлом: его время модификации (когда данные файла были последний раз изменены) и время, когда к нему последний раз осуществлялся доступ (когда файл последний раз читался).

$ touch myfile

Вы можете сменить эти временные метки на произвольные, например следующим образом.

$ touch -d "November 18 1975" myfile

Если данный файл не существует, команда touch создаст его, т. е. это удобный способ создания пустых файлов.

Полезные опции

- a	Изменить только время доступа
- m	Изменить только время модификации
- c	Если файл не существует, не создавать его (по умолчанию команда touch создает его)
-d временная_метка	Установить временную_метку(метки) файла. Поддерживается огромное количество форматов временных меток, например, "12/28/2001 3pm", "28-May" (подразумевается текущий год и время 00.00), "next tuesday 13:59", "0" (полночь сегодня). Поэкспериментируйте и проверьте результаты ваших изменений с помощью команды stat. Всю документацию можно посмотреть с помощью команды info touch
-t временная_метка	Менее изящный способ задания временной метки файла, использующий формат [[CC]YY]MMDDhhmm[.ss], где CC - это две цифры, обозначающие век, YY - две цифры, обозначающие год, MM - две цифры, обозначающие месяц, DD – две цифры, обозначающие день, hh две цифры, обозначающие часы, mm - две цифры, обозначающие минуш, и ss - две

цифры, обозначающие. Например, -t 20030812150047 означает August 12,2003, at 15:00:47

chovn [опций\ user spec файлы coreutils /bin stdin stdout -file --opt -help -version

Команда chown (от англ. change ownership) устанавливает владельца файлов или директорий.

$ chown smith myfile myfile2 mydir

Параметр user_spec может принимать одно из следующих значений:

- имя пользователя (или идентификатор пользователя) для установки владельца;
- имя пользователя (или идентификатор пользователя), за которым может следовать двоеточие и имя группы (или идентификатор группы), для установки владельца и группы;
- имя пользователя (или идентификатор пользователя), за которым следует только двоеточие, для установки владельца и установки в качестве группы первичной группы этого пользователя ;
- имя группы (или идентификатор группы), перед которым ставится двоеточие, для установки только группы;
- --ref erence=file для установки владельца и группы такими же, какие они у другого файла/г/е.

Полезные опции

- -dereference	Выполнить операцию над файлом, на который указывает символьная ссылка
-R	Выполнить операцию рекурсивно по всем подиреетториям

chgrp [опции] user spec файлы coreutils /bin stdin stdout -file —opt —help —version

Команда chgrp (от англ. change group) устанавливает группу для файлов или директорий.

$ chgrp smith myfile myfile2 my_dir

Параметр groupspec может принимать одно из следующих значений:

• имя группы или идентификатор группы;

• --reference=/r/e для установки группы такой же, как у файла file.

Обратитесь к разделу "Работа с группами" на странице 190 для более подробной информации о группах.

Полезные опции

--dereference	Выполнить операцию над файлом, на который указывает символьная ссылка
- R	Выполнить операцию рекурсивно по всем поддиректориям

chmod [опции] права файлы coreutils /bin stdin stdout -file --opt -help -version

Команда chmod (от англ. change mode) устанавливает права доступа к файлам и директориям. Не каждый файл должен быть доступен для всех (понимаете это не Windows 95), и chmod - это инструмент, который позволяет решать эту задачу. Обычные права включают в себя права на чтение, запись и выполнение, и они могут относиться к владельцу файла, группе пользователей, и/или ко всем остальным пользователям. Аргумент права может задаваться тремя способами:

- —reference=/i/e для установки таких же прав, как у файла file;
- восьмеричное число, размером до четырех цифр, которое определяет абсолютные права на файл в битах. Самая левая цифра специальная (ее мы опии шем позже), а вторая, третья и четвертая относятся к владельцу файла, группе файла и к остальным пользователям соответственно. На рис. 3 показан пример, поясняющий значение режима 0640;
- одна или более разделенных запятыми строк, задающих абсолютные или относительные права (т. е. относительно текущих прав на файл), которые нужно установить для файла.

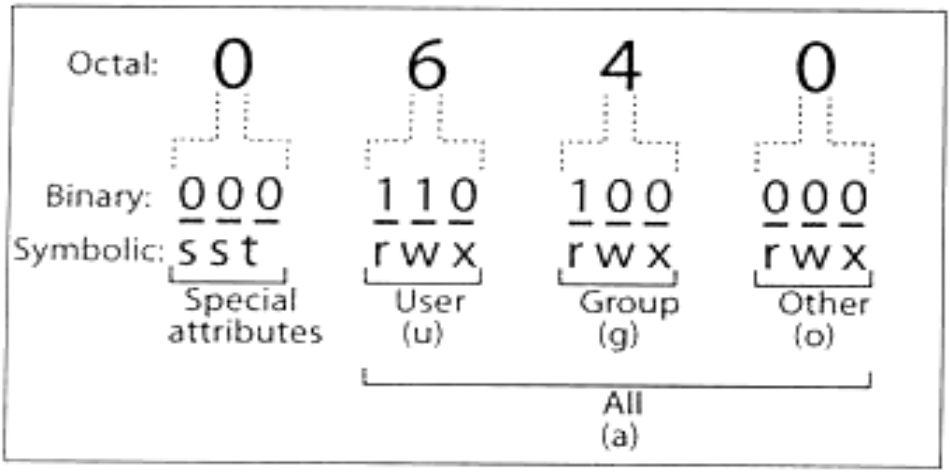


Рис. 3. Значение битов в правах на файл

В третьем варианте каждая строка состоит из трех частей: необязательной области действия, команды и прав.

Область действия (опциональна)

u — владелец, g — группа, o — остальные пользова-тепи(не владелец и не входящие в группу4), a - все пользователи. По умолчанию - a.

Команда

"+" - добавить права, "-" - удалить права, "=" -установить абсолютные права, игнорируя существующие.

Права

r - на чтение, w - на запись/изменение, x - на выполнение (в случае директорий это означает право использовать команду cd с данной директорией), X - для условного выполнения (мы поясним это позже), и - для копирования прав пользователя, g - для копирования прав группы, o - для копирования прав "остальных пользователей", s - для установки атрибутов setuid или setgid, и t - для установки бита принадлежности (sticky bit).

Например, строка ug+rw добавляет права на чтение и запись для владельца и группы, строка a-x (или просто -x) удаляет права на выполнение для всех, а строка u=r сначала удаляет все существующие права а затем устанавливает права только на чтение файла для его владельца. Вы можете объединять эти строки разделяя их запятыми, например ug+rw, a-x.

Атрибуты setuid и setgid устанавливаются для исполняемых файлов (программ и скриптов). Представьте себе, что у вас есть исполняемый файл F с владельцем "smith" и группой "friends". Если для файла F установлен атрибут setuid (от англ. set user ID), то любой, кто исполняет файл F, "станет" пользователем smith, со всеми его правами и привилегиями, на время выполнения программы. Аналогично, если для файла F установлен атрибут setgid (от англ. set group ID), то любой, кто исполняет файл F, станет членом группы friends на время выполнения программы. Нетрудно догадаться, что атрибуты setuid и setgid могут подорвать безопасность системы, поэтому не используйте их, если только вы не знаете точно, что делаете. Одна неуместная команда chmod +s может сделать всю вашу систему уязвимой для атак.

Права на условное исполнение (X) означают то же самое, что и в случае параметра x, за исключением того, что они устанавливаются только в том случае, если объект является исполняемым файлом или директорией. В противном случае команда не возымеет эффекта.

Полезные опции

-R Выполнить операцию рекурсивно для всех поддиректорий

chattr [опции[[+ -=\[атрибугь1[[файлы] e2fsprogs /usr/bin stdin stdout -file --opt -help -version

Если вы выросли на других Unix-системах, вы, возможно, будете удивлены тем, что файлы в Linux могут иметь дополнительные атрибуты помимо прав доступа. Если файл находится в файловой системе ext2 или ext3 (стандартная для Fedora), то вы можете установить эти расширенные атрибуты с помощью команды chattr (от

англ. change attribute) и вывести список этих атрибутов с помощью команды lsattr.

Как и в случае с командой chmod, атрибуты можно устанавливать (+) или добавлять (-) относительно, или устанавливать абсолютно (=).

Атрибут	Значение
a	Файл только для добавления данных, т. е. разрешается только добавлять данные и нельзя редактировать как-либо еще. Только для суперпользователя
A	Доступ к файлу не фиксируется во временных метках: доступ к файлу не обновляет его временную метку доступа (atime)
c	Сжатый: данные сжимаются при записи и распаковываются при чтении
d	Неархивируемый: программа dump должна игнорировать этот файл при резервном копировании (см. раздел "Резервное копирование и удаленное хранение" на странице 154)
i	Неизменяемый: файл не может быть изменен или удален (только для суперпользователя)
j	Журналируемые данные (только для файловой системы ехй)
s	Безопасное удаление: в случае удаления данные этого файла переписываются нулями
s	Синхронное обновление: изменения записываются на диск немедленно, как будто вы набираете команду sync после сохранения (см. раздел "Диски и файловые системы" на странице 146)
u	Неудаляемый: файл не может быть удален

Полезные опции

-R Рекурсивно обрабатывать директории.

lsattr [опции] [файлы] e2fsprogs
/usr/bin sldin stdout -file —opt —help --version

Если вы установили расширенные атрибуты с помощью команды chattr, то вы можете просмотреть их с помощью команды lsattr (от англ. list attributes). В ее выходных данных используются те же буквы, что и в команде chattr; например, в следующем примере файл является неизменяемым и неудаляемым.

$ lsattr myfile
-u--i myfile - .,

Полезные опции

- R Рекурсивно обрабатывать директории
-a Вывести список всех файлов, включая те файлы, имена которых начинаются с точки
-d В случае директории не перечислять ее содержимое, а выводить информацию только о самой директории. Если файлы не указываются, то команда lsattr вьшодит атрибуты всех файлов в текущей директории.

Поиск файлов

find Искать файл в иерархии директорий
slocate Создать индекс файлов и искать строку в этом индексе
which Искать исполняемые файлы в ваших директориях поиска (команда)
type Искать исполняемые файлы в ваших директориях поиска (встроенная функция bash)
whereis Искать исполняемые файлы, документацию и файлы с исходным кодом

Linux-системы запросто могут содержать десятки и сотни тысяч файлов. Как найти конкретный файл, когда он вам понадобился? Первый способ - это упорядочить ваши файлы по директориям с продуманной схемой, но есть несколько других способов поиска файлов, каждый из которых нужно использовать в зависимости от того, что вы ищете.

Для поиска произвольного файла есть простая программа find, которая упорно перебирает файл за файлом во всех директориях в поисках нужного файла. Программа slocate намного быстрее, она

осуществляет поиск в заранее созданном индексе, который вы генерируете в случае необходимости (Fedora генерирует этот индекс ежедневно по умолчанию).

Для поиска программ команды which и type проверяют все директории в пути поиска вашего командного процессора. Команда type встроена в командный процессор bash (и, следовательно, доступна только при работе в bash), тогда как which является программой (как правило /usr-/bin/which); type работает быстрее и может искать среди псевдонимов* командного процессора. В отличие от них, команда whereis исследует заранее известный набор директорий, а не директории вашего пути поиска.

find [директории] [выражение] findutils /usr/bin stdin stdout -file -opt --help -version

Команда find ищет в одной или более директориях (и рекурсивно в их поддиректориях) файлы, соответствующие определенному критерию. Она очень мощная, с более чем 50 опциями и, к сожалению, не совсем обычным синтаксисом. Ниже приведены некоторые простые примеры, которые осуществляют поиск по всей файловой системе из корневой директории.

Найти определенный файл с именем myfile.

$ find / -type f -name myfile -print

Вывести имена всех директорий.

$ find / -type d -print

Командный процессор tcsh проделывает определенный фокус для того, чтобы команда which видела псевдонимы.

Полезные опции

-name шаблон
-path шаблон
-lname шаблон
-iname шаблон
-ipath шаблон
-ilname шаблон

Имя (-name), путь (-path) или символьная ссылка (-lname) искомого файла должна соответствовать заданному шаблону командного процессора, который может включать в себя групповые символы *, ?, и []. Пути задаются относительно дерева директорий, в котором осуществляется поиск. Опции -iname, -ipath and –

	i lname аналогичны опциям -name, -path и -lname соответственно, но нечувствительны к регистру
-regex regexp	Путь (относительно дерева директорий, в котором осуществляется поиск) должен соответствовать заданному регулярному выражению regexp
-type f|d|l|b|c|p|s	Искать только регулярные файлы (f), директории (d), символические ссылки (1), блочные устройства (Ь), символьные устройства (с), именованные каналы (р) или сокеты (s)
atime N ctime N mtime N	Доступ к файлу осуществлялся последний раз (-atime), файл модифицировался последний раз (-mtime) или изменялся его статус (- ctime) точно N*24 часов назад. Использовать +/VB случае "больше N" или -NB случае "меньше N1
amin N cmin N irani n N	Доступ к файлу осуществлялся последний раз (- amin), файл модифицировался последний раз (-mmin) или изменялся его статус (-cmin) точно Жминут назад. Использовать -ьЛ'в случае "больше /V, или -/VB случае "меньше №
-anewer other_f ile -cnewer other_file -newer other_fiie	Доступ к файлу осуществлялся (-anewer), файл модифицировался (-newer) или изменялся его статус (-cnewer) позже, чем у файла other_f He
-maxdepth N -mindepth N	Рассматривать файлы минимум (-mindepth) или максимум (-maxdepth) на Л/уровней вглубь дерева директорий, в котором осуществляется поиск
- f o 11 ow	Вычислять символьные ссылки
-depth	Выполнить используя поиск в глубину: полностью изучить содержимое директории (рекурсивно) перед тем, как искать в самой

	директории
-xdev	Ограничить поиск одной файловой системой, т. е.не выходить за границы устройства хранения
-size N[bckw]	Искать среди файлов размера N, который можно задать в блоках (Ь), однобайтовых символах (с), килобайтах (к) или двухбайтовых словах (w). Используйте +Л/в случае "больше /V', или -N в случае "меньше N'
-empty	Файл имеет нулевой размер и является регулярным файлом или директорией
-user имя	Файл принадлежит владельцу с заданным
-group имя	именем пользователя или группе с заданными именем группы
-perm режим	Файл имеет права, аналогичные режиму.
-perm -режим	Используйте -режим для того, чтобы
-perm +режим	проверить, что установлены все заданные биты, или +режим, чтобы проверить, что установлен хотя бы один из этих битов

Также вы можете группировать и инвертировать части с помощью следующих операций.

выражение I -а выражение!

"И" (эта операция проделывается по умолчанию, если два выражения пишутся рядом, поэтому опцию "-а" можно и не указывать).

выражение1 -о выражение! "ИЛИ".

/ выражение -not выражение Отрицание выражения (выражение)

Маркеры предшествования, почти как в алгебре. Сначала обрабатывается выражение в скобках. Вам может понадобиться квотировать их в командном процессоре с помощью символа "\".

выражение! , выражение2

То же самое, что и оператор запятой в языке программирования C. Обработать оба выражения и вернуть значение второго.

Когда вы указали критерий поиска, вы можете сделать так, чтобы команда find проделывала определенные действия с файлами, соответствующими критерию.

Полезные опции

-print	Просто вывести путь к файлу, относительно директории поиска
-printf	строка Вывести данную строку, в которой можно делать подстановки в стиле библиотечной функции C, printf (). Полный список выходных данных можно найти на man-станице
-printo	Аналогична -print, но вместо разделения выходных строк символом новой строки использует символ нуль (ASCII 0). Используйте эту опцию, когда вы перенаправляете выходные данные команды find на вход другой программе, а ваш список имен файлов содержит символы пробела. Конечно, получающая программа должна уметь читать и анализировать такие разделенные нулями строки, например, xargs -0
-exec cmd ;	Вызвать указанную команду командного процессора, cmd.
-ok cmd ;	Не забудьте замаскировать метасимволы командного процессора, включая обязательную заключительную точку с запятой, чтобы они не обрабатывались сразу в командной строке. Кроме тога, символ"{}" (не забудьте поместить его в кавычки или замаскировать) представляет собой путь к найденному файлу. Действие - ok выдает приглашение пользователю перед вызовом команды процессора; -exec этого не делает
-Is	Выполнить команду Is -dils над файлом

Команда find, которая выдает список файлов в стандартный поток вывода, очень удобна вкупе с командой xargs, которая читает список файлов из стандартного потока ввода и применяет к ним команду (см. man xargs). Например, чтобы искать в вашей текущей

директории файлы, содержащие слово "myxomatosis", нужно выполнить следующую команду.

$ find . -printO | xargs -0 grep myxomatosis

slocate [опции] slocate
/usr/bin stdin stdout -file --opt --help -version

Команда slocate (безопасный поиск) создает индекс (базу данных) расположения файлов и осуществляет быстрый поиск по нему. Если вы планируете помещать много файлов в иерархию директорий, которые не будут сильно изменяться, то slocate для этого случая подходит лучше всего. Для поиска одного файла или осуществления более сложных операций над найденными файлами используйте команду find.

Fedora Linux автоматически индексирует всю файловую систему раз в день, но если вам понадобится создать индекс самостоятельно (скажем, он будет находиться в директории /tmp/myindex), выполните следующую команду.

$ slocate -u -o /tmp/myindex

Чтобы создать индекс конкретной директории и всех ее поддиректорий, используйте следующую команду.

$ slocate -U directory -o /tmp/myindex

Затем, чтобы найти строку string в индексе, выполните следующую команду.

$ slocate -d /tmp/myindex string

Что делает команду slocate "безопасной"? Во время поиска она не выводит названия файлов, которые вы, обычно, не имеете права видеть. Поэтому, если суперпользователь создал индекс защищенной директории, пользователь не являющийся суперпользователем, может осуществлять поиск по нему, но он не увидит защищенных файлов.

Опции индексирования

-u	Создать индекс, начиная с корневой директории
-u директория	Создать индекс, начиная с заданной директории
-1 (0 11)	Отключить (0) или включить (1) безопасность. По умолчанию 1
-e директории	Исключить одну или более директориям индекса. Разделите пути к этим директориям запятыми
-o выходной _файл	Записать индекс в выходной _файл

Опции поиска

-d индекс	Указать, какой индекс использовать (в нашем примере /tmp/myinde^
- I	Поиск, нечувствительный в регистру
-r регулярное_выражение	Искать файлы, соответствующие заданному регулярному_выражению

which файл which
/usr/bin stdin stdout -file —opt -help --version

Команда which осуществляет поиск исполняемых файлов в пути поиска вашего командного процессора. Если вы можете вызвать программу, набирая ее название:

$ who

то программа which сообщит вам, где эта программа расположена:

$ which who
/usr/bin/who

Вы можете найти даже саму программу which.

$ which which a _ti
/usr/bin/which

Если несколько программ в пути поиска имеют одинаковые названия (скажем, /usr/bin/who и /usr/local/ bin/who), то команда which вьщаст первую из них.

type [опции] команды bash
встроенная команда stdin stdout -file —opt --help -version

Команда type, как и which, осуществляет поиск исполняемых файлов в пути поиска вашего командного процессора.

```
$ type grep who
grep is /bin/grep
who is /usr/bin/who
```

Однако команда type является встроенной в командный процессора, тогда как which - это программа на диске.

```
$ type which type rm if
which is /usr/bin/which
type is a shell builtin
rm is aliased to ч/bin/rm'
if is a shell keyword
```

Так как type встроена в командный процессор, она работает быстрее which; однако она доступна только тогда, когда вы работаете в bash.

whereis [опции] файлы util-linux
/usr/bin stdin stdout -file --opt --help --version

Команда whereis осуществляет попытку найти заданные файлы в жестко запрограммированном списке директорий. Она может искать исполняемые файлы, документацию и файлы с исходным кодом. Команда whereis несколько неудобная, поскольку ее список директорий может не включать нужную вам директорию.

Полезные опции

-b Выводить список только исполняемых

-m файлов (-b), тап-страниц(-m) или
-s файлов с исходным кодом (-s)
-B директории.. . -f файлы. .. Искать исполняемые файлы(-ъ), man-
-M директории... -f файлы. . . страницы (-м), или файлыс исходным
-S директории... -f файлы. .. кодом (-S) только в заданных директориях. Список директорий должен оканчиваться опцией - f перед тем, как вы начнете вводить названия файлов, которые вам нужно найти

Работа с текстом в файлах

grep	Найти строки в файле, которые соответствуют регулярному выражению
cut	Выделить колонки из файла
paste	Вставить колонки
tr	Преобразовать символы в другие символы
sort	Сортировать строки текста в соответствии с различными критериями
uniq	Найти идентичные строки в файле
tee	Копировать файл и вывести его в стандартный поток вывода одновременно

Одна из сильных сторон Linux - это работа с текстом: обработка текстовых файлов (или стандартного потока ввода) различными преобразованиями. Любая программа, которая читает данные из стандартного потока ввода и пишет данные в стандартный поток вывода, имеет отношение к текущему разделу, но здесь мы рассмотрим только самые употребительные и мощные из них.

grep [опции] шаблон [файлы] grep /bin stdin stdout -file —opt —help —version

Команда grep - одна из наиболее полезных и мощных в арсенале Linux. Ее предпосылки просты: если на вход подаются несколько файлов, то она печатает все строки в этих файлах, которые соответствуют определенному шаблону регулярного выражения. Например, если файл myfile содержит следующие строки:

The quick brown fox jumped over the lazy dogs!

My very eager mother just served us nine pancakes.
Film at eleven.

а мы ищем все строки, содержащие "pancake", то в результате выполнения этой команды мы получим:

$ grep pancake myfile
My very eager mother just served us nine pancakes.

grep может использовать два различных типа регулярных выражений: основной и расширенный. Они одинаково мощные, но разные, и вы можете выбрать, какой из них использовать, основываясь на вашем опыте использования других реализаций grep. Основной синтаксис отражен в табл. 2 и 3.

Полезные опции

-v	Выводить только те строки, которые /«соответствуютрегулярному выражению
-l	Выводить только именатех файлов, которые содержат соответствующие регулярному выражению строки, а не сами строки
-L	Выводить имена только тех файлов, которые не содержат соответствующие регулярному выражению строки
- c	Выводить только количество соответствующих регулярному выражению строк
-n	Перед каждой строкой в выходных данных, соответствующих регулярному выражению, выводить номер строки в исходном файле
-b	Перед каждой строкой в выходных данных, соответствующих регулярному выражению, выводить смещение строки в байтах от начала входного файла
- i	Поиск, нечувствительный к регистру
-w	Проверять на соответствие только слова (т. е. учитывать только те слова, которые целиком соответствуют всему регулярному выражению)
-x	Проверять на соответствие только строки (т. е.

	учитывать только те строки, которые целиком соответствуют всему регулярному выражению). Отменяет действие опции -w
-A N	После каждой найденной строки выводить следующие N строк из исходного файла
-в N	Перед каждой найденной строкой выводить предыдущие N строк из исходного файла
- C N	То же самое, что -A N -B N: вывести N строк (из исходного файла) до и после каждой строки, соответствующей регулярному выражению
-r	Рекурсивно осуществлять поиск во всех файлах в директории и ее поддиректориях
-E	Использовать расширенные регулярные выражения (см. egrep)
- F	Использовать список фиксированных строк вместо регулярных выражений (см. f grep)

eg rep [опции] шаблон [файлы] grep
/bin stdin stdout -file --opt -help -version

Команда egrep похожа на grep, но использует другой ("расширенный") язык для регулярных выражений. Она эквивалентна команде grep -E.

Таблица 2. Основные и расширенные регулярные выражения для команды grep

Регулярное выражение	**Значение**
.	Любой одиночный символ
[. . .]	Соответствует одиночному символу из заданного списка
[^. ..]	Соответствует любому одиночному символу, НЕ входящему в список
(...)	Группировка
^	Начало строки
$	Конец строки
\<	Начало слова
\>	Конец слова

[:alnum:]	Любой алфавитно-цифровой символ
[:alpha:]	Любой алфавитный символ
[:cntrl:]	Любой управляющий символ
[:digit:]	Любая цифра
[:graph:]	Любой графический символ
[:lower:]	Любая строчная буква
[:print:]	Любой печатаемый символ
[:punct:]	Любой знак пунктуации
[:space:]	Любой пробельный символ
[:upper:]	Любая заглавная буква

Таблица 2. Основные и расширенные регулярные выражения для команды grep (Продолжение)

Регулярное выражение	**Значение**
[:xdigit:]	Любая шестнадцатеричная цифра
*	Нуль или более вхождений регулярного выражения
\c	Символ "c", даже если "c" является специальным символом регулярного выражения. Например, используйте комбинацию * для обозначения звездочки или \\ для обратной косой черты. Либо можно поместить символ в квадратные скобки, например, [*] или [\]

Таблица 3. Различия: основные и расширенные регулярные выражения

Основные	**Расширенные**	**Значение**
|	|	Или
\+	+	Одно или более вхождений регулярного выражения
\ ?	?	Нуль или одно вхождение регулярного выражения
\ {n\}	{n}	Точно n вхождений регулярного выражения
\{n,\}	{n, }	n или более вхождений регулярного выражения

\{n,m\}	{n,m}	От n до m (включительно) вхождений регулярного выражения, n < m.

fgrep [опции] [фиксированные строки] [файлы] grep /bin stdin stdout -file --opt -help -version

Команда fgrep аналогична команде grep, но вместо регулярных выражений она на входе принимает список фиксированных строк, разделенных символами новой строки. Она эквивалентна команде

grep -F. Например, чтобы искать строки one, two и three в файле myfile:

```
$ f grep ' one        Обратите внимание на то, что мы вводим
                      символы новой строки
two
three' myfile
```

fgrep, как правило, используется с опцией -f, которая считывает шаблоны из файла. Например, если у вас есть словарный файл с большим количеством строк:

```
$ cat my_dictionary_file
aardvark
aback
abandon
. . .
```

то вы можете без труда проделать поиск этих строк в наборе входных файлов:

```
$ fgrep -f my_dictionary_file *
```

Также команда fgrep хорошо подходит для поиска символов, не являющихся буквами или цифрами, например * и {, так как они будут восприниматься буквально, а не как метасимволы в регулярных выражениях.

cut -(b | c | f) диапазон [опции] [файлы] coreutils /usr/bin stdin stdout -file --opt -help -version

Команда cut выделяет колонки текста из файлов. Колонка определяется либо по смещению в символах (например, с девятнадцатого символа каждой строки):

$ cut -cl9 myfile

либо по смещению в байтах (которое часто является тем же самым, что и смещение в символах, только если в вашем языке не используются многобайтные символы):

$ cut -b19 myfile

либо по разделителям (например, пятое поле в каждой строке файла с разделенными запятыми строками):

$ cut -d, -f5 myfile

Вы не ограничены выводом одной колонки: вы можете указать диапазон (3-16), последовательность разделенную запятыми (3,4,5,6,8,16) или и то, и другое одновременно (3,4,8-16). В случае диапазона, если вы опустите нижнюю границу (-16), то будет использоваться диапазон (1-16); если вы опустите верхнюю границу (5-), то диапазон будет ограничиваться концом строки.

Полезные опции

-de Использовать символ С в качестве входного разделителя между полями при использовании опции -1 По умолчанию используется символ табуляции

--output-delimiter=C Использовать символ С в качестве выходного разделителя между полями при использовании опции -1 По умолчанию используется символ табуляции

-s Подавлять (не выводить) строки, которые не содержат разделяющие символы

paste [опции] [файлы] **coreutils**
/usr/bin stdin stdout -file —opt —help —version

Команда paste - противоположность команды cut: она рассматривает несколько файлов как вертикальные колонки, объединяет их и выводит в стандартный поток вывода:

```
$  cat  letters
A
B
C
$  cat  numbers
1
2
3
4
5
$ paste numbers  letters
1       A
2       B
3       C
4
5
$ paste letters numbers
A       1
B       2
C       3
4
5
```

Полезные опции

-d разделители — Использовать указанные разделители между колонками; по умолчанию используется символ табуляции. Вы можете указать одиночный символ (- d:), который будет использовать везде, либо список символов (-dxyz), которые будут использоваться последовательно в строке

	(первый разделитель - x, затем - y, затем - z, затем - x, затем - y,...)
-s	Переставить местами строки и колонки при выводе: $ paste -s letters numbers ABC 12 3 4 5

tr [опции[набор_символов1 [набор_символов2] coreutils /usr/bin stdin stdout -file -opt --help --version

Команда tr осуществляет простые полезные преобразования одного набора в другой. Например, чтобы заменить все гласные буквы звездочками, нужно проделать следующее.

```
$ cat myfile
This is a very wonderful file.
$ cat myfile | tr aeiouAEIOU '*'
Th*s *s * v*ry w*nd*rf*l f*l*.
```

Чтобы удалить все гласные буквы, введите следующие команды

```
$ cat myfile | tr -d aeiouAEIOU
Ths s vry wndrfl fl.
```

Чтобы сделать все буквы в файле прописными, введите следующие команды:

```
$ cat myfile | tr 'a-z' 'A-Z'
THIS IS A VERY WONDERFUL FILE.
```

Команда tr преобразует первый символ набора_ символов] в первый символ набора_символов2, второй - во второй, третий - в третий, и т. д. Если длина набора_символов1 N символов, то будут использованы только N первых символов набора^ символов2 (случай, когда на бор символовJ длиннее набора_символов2, рассмотрен ниже в описании опции -t).

Набор символов может иметь следующий вид.

Формат	Значение
ABCD	Последовательность символов A, B, C, D
A - в	Диапазон символов от A до B
[x*y]	y вхождений символа x
[: class:]	Такие же классы символов ([: alnum:], [: digit:], и т. д.), какие принимает команда grep

Также команда tr понимает escape-последовательности: "\a" ("G = звонок), "\Ь" (ЛН = забой), "\Г (ль = подача страницы), "\п" (л J = новая строка), "\r" (ЛМ = возврат каретки), "\t" ("I = табуляция) и "\v" (ЛК = вертикальная табуляция), принимаемые функцией print f (см. раздел "Вывод на экран" на странице 224), а также запись \ппп для обозначения символов с восьмеричным значением ппп.

tr прекрасно подходит для быстрых и простых преобразований; но для более сложных преобразований обратитесь к командам sed, awk или perl.

Полезные опции

- d — Удалить символы набора_ символов 1 из входных данных
- s — Удалить соседние дубликаты (символов из набора^ символов 1) из входных данных. Например, команда t r - s aeiouAElOU удалит все повторяющиеся гласные буквы (слово reeeeeeally преобразуется в слово really)
-c — Использовать символы, «^входящие в наборсимволов!
-l — Если набор символов / длиннее чем набор символов2, то привести их к одной длине обрезав набор_ символов 1. Если опция -1 не указана, то последний символ набора' символов2'(незаметно) дублируется до тех пор, пока набор_символов2не станет такой же длины, что и набор символов2

sort [опции] [файлы] coreutils
/bin stdin stdout -file —opt -help -version

Команда sort выводит строки текста в алфавитном порядке либо сортирует их по какому-либо другому критерию, который вы зададите. Все указываемые в командной строке файлы объединяются, и результат объединения сортируется и выводится.

$ cat myfile
def
xyz
abc
$ sort myfile
abc
def
xyz

Полезные опции

- f	Сортировка, нечувствительная к регистру
-n	Сортировать численно (т. е. число 9 идет перед 10),а не в алфавитном порядке (число 10 идет перед 9, так как оно начинается с цифры "1")
-g	Еще один метод численной сортировки другим алгоритмом, который, помимо прочего, распознает экспоненциальное представление чисел (например, 7.4e3, что означает "7.4 умножить на десять в третьей степени" или 7400). Выполните команду info sort, чтобы изучить все детали
-u	Уникальная сортировка: игнорировать повторяющиеся строки (при использовании опции - с для проверки сортированных файлов прервать выполнение, если встретится несколько одинаковых строк подряд)
-c	Не сортировать, а только проверять, отсортированы ли уже входные данные. Если данные отсортированы, то ничего не печатать, иначе вывести сообщение об ошибке
-b	Игнорировать начальные пробелы
-r	Сортировать в обратном порядке (по убыванию)
-t X	Использовать Хв качестве разделителя полей для опции -к

-к F1 [. C1] [, F2 [. C2]] Выбрать ключи сортировки

Ключ сортировки - это часть строки, которая рассматривается при сортировке, вместо того чтобы рассматривалась вся строка. Например, "пятый символ каждой строки". Обычно команда sort будет считать, что следующие строки отсортированы:

aaaaz
bbbby

но если ваш ключ сортировки - это "пятый символ каждой строки", то эти строки будут расставлены в обратном порядке, так как "y" идет раньше "z". Синтаксис этой опции означает следующее.

Параметр	**Смысл**	**Значение по умолчанию**
F1	Начальное поле	Обязательный
C1	Первая позиция в поле 1	1
F2	Последнее поле	Последнее поле
C2	Первая позиция в последнем поле	1

Поэтому команда sort -kl. 5 осуществляет сортировку по первому полю, начиная с его пятого символа; а команда sort -k2.8,5 означает "с восьмого символа второго поля до первого символа пятого поля".

Вы можете повторять опцию -к для того, чтобы определить несколько ключей, которые будут использованы последовательно в том порядке, в котором вы указали их в командной строке.

uniq [опции] [файлы] coreutils
/usr/bin stdin stdout -file --opt -help —version

Команда uniq работает с повторяющимися подряд строками текста. Например, если у вас есть файл ту file:

$ cat myfile
a
b
b
c

b

то команда uniq определит и обработает (так, как вы определите) две последовательные строки b, но не затронет третью b.

$ uniq myfile
a
b

Команда unig часто используется после сортировки файла.

$ sort myfile | uniq
a
b
c

В этом случае останется только последняя Ь. Также вы можете подсчитать повторяющиеся строки, вместо того чтобы удалять их.

$ sort myfile | uniq
1 a
3 b
1 c

Полезные опции

-c	Подсчитать расположенные рядом повторяющиеся строки
- i	Нечувствительное к регистру выполнение команды
-u	Выводить только уникальные строки
-d	Выводить только повторяющиеся строки
-s N	Игнорировать первые /^символов каждой строки при поиске дубликатов
-f N	Игнорировать первые /Уразделенных пробелами полей каждой строки при поиске дубликатов
-w N	Рассматривать только первые ^символов каждой строки при поиске дубликатов. При использовании с опциями – s или - f команда sort сначала будет игнорировать указанное число символов или полей, а затем

использовать следующие N символов

iee [опции] файлы coreutils
/usr/bin stdin stdout -file --opt --help --version

Как и команда cat, команда tee копирует стандартный поток ввода в стандартный поток вывода неизмененным. Однако, в то же время она копирует этот же самый стандартный поток ввода в один или более файлов, tee наиболее часто используется в качестве промежуточного звена конвейера, записывая промежуточные данные в файл и в то же время передавая их по конвейеру следующей команде.

$ who | tee original_who | sort

Эта команда напечатает отсортированные выходные данные команды who в стандартный поток вывода, но одновременно запишет исходный (несортированный) выходной поток команды who в файл original who.

Полезные опции

-a Дописывать, а не переписывать файлы
-i Игнорировать сигналы прерываний

Более сложная обработка

Мы рассмотрели всего лишь небольшую часть функций для работы с текстом в Linux. Эта система имеет сотни фильтров, которые осуществляют куда более сложную обработку данных. Но чем сложнее функция, тем сложнее научиться ее использовать и описать в такой короткой книге. Вот вам несколько фильтров для начала.

awk

awk - это язык поиска шаблонов. Он умеет искать данные с помощью регулярных выражений и выполнять операции на основе этих данных. Вот несколько простых примеров обработки текстового файла myfile.

Вывести второе и четвертое слова каждой строки.

$ awk '{print $2, $4}' myfile

Вывести все строки короче 60 символов.

$ awk '{length($0) < 60}' myfile

sed

Аналогично awk, sed - это механизм поиска по шаблону, который может выполнять операции над строками текста. Его синтаксис тесно связан с синтаксисом vim и строчного редактора ed. Вот несколько тривиальных примеров.

Вывести файл, в котором все вхождения строки "red" будут заменены строкой "hat".

$ sed 's/red/hat/g' myfile

Вывести файл, у которого будут удалены первые 10 строк.

$ sed 'l,10d' myfile

m4

m4 - это язык обработки макросов. Он ищет ключевые слова в файле и заменяет их заданными значениями. Например, для файла:

```
$ cat myfile
My name is NAME and I am AGE years old
 ifelse(QUOTE,yes,No matter where you
go... there you are)
```

команда m4 заменит слова NAME, AGE и QUOTE.

```
$ m4 -DNAME=Sandy myfile
My name is Sandy and I am AGE years old
$ m4 -DNAME=Sandy -DAGE=25 myfile
My name is Sandy and I am 25 years old
$ m4 -DNAME=Sandy -DAGE=25 -DQUOTE=yes myfile
```

My name is Sandy and I am 25 years old
No matter where you go... there you are

perl, python

Perl и Python - это самостоятельные языки написания сценариев, достаточно мощные для написания завершенных, сложных приложений.

Сжатие и упаковка файлов

gzip	Сжать файлы с помощью программы GNU Zip
gunzip	Разжать файлы GNU Zip !
compress	Сжать файлы традиционным методом Unix
uncompress	Разжать файлы традиционным методом Unix
zcat	Сжать/разжать файл через стандартные потоки ввода/вывода (gzip или compress)
bzip2	Сжать файлы в формате BZip
bunzip2	Разжать файлы BZip
zip	Сжать файлы в формате Windows Zip
unzip	Разжать файлы Windows Zip
uuencode	Преобразовать файл в формат uuencode
uudecode	Восстановить файл из формата uuencode

Linux умеет сжимать файлы в множество форматов и разжимать их. Наиболее популярный формат -GNU Zip (gzip), сжатые файлы которого имеют суффикс .gz. Другие часто встречающиеся форматы -это классическое сжатие Unix (суффикс . z), bzip2-сжатие (суффикс .bz2) и Zip-файлы из Windows-систем (суффикс . zip).

Также существует технология преобразования бинарных файлов в текстовый формат, чтобы их можно было (скажем) передавать внутри электронных писем. Сегодня эта операция проделывается автоматически при прикреплении файла к письму и в MIME-инструментах, но мы рассмотрим более старые программы uuencode и uudecode, которые все же используются до сих пор.

Если вы столкнетесь с форматом, который мы не рассмотрели, например с файлами Macintosh hqx/sit, Arc, Zoo и другими, то вы можете узнать .о них больше на сайтах http://www.faqs.org/faqs/compression-faq/partl7section-2.html и http://www-lO6.ibm.com/de-velopenvorks/library/1-lw-comp.html.

gzip [опции] [файль/\ ' gzip
stdin stdout -file --opt -help --version

Команды gzip и gunzip сжимают и разжимают файлы в формате GNU Zip. Сжатые файлы имеют суффикс .gz.

Примеры команды

gzip file	Сжать файл file в файл file. gz. Исходный файл удаляется
gzip -c file	Вывести сжатые данные в стандартный поток вывода
cat file | gzip	Сжать данные из конвейера
gunzip file.gz	Разжать файл file.gz в файл. Исходный файл file.gz удаляется
gunzip -c file.gz	Разжать данные в стандартный поток вывода
cat file.gz | gunzip	Разжать данные из конвейера
zcat file.z	Разжать данные в стандартный поток вывода

tar-файлы, сжатые с помощью gzip: примеры команд

tar czf myf ile. tar .gz dirname	Упаковать директорию dimame
tar tzf myf ile. tar .gz	Вывести список содержимого
tar xzf myf ile. tar .gz	Распаковать

Добавьте опцию v для команды tar, чтобы выводить имена файлов по мере их обработки.

compress [опции] [файлы] ncompress
/usr/bin stdin stdout -file -opt -help -version

Команды compress и uncompress сжимают и разжимают файлы согласно стандартному формату сжатия Unix (Lempel Ziv). Сжатые файлы имеют суффикс .Z.

Примеры команд

compress file	Сжать файл fi 1 e в файл fi 1 e. Z. Исходный файл удаляется
compress -c file	Вывести сжатые данные в стандартный поток вывода

cat file | compress	Сжать данные из конвейера
uncompress file.Z	Разжать файл fi 1 e. Z в fi 1 e. Исходный файл */e.7удаляется
uncompress -c file.Z	Разжать данные в стандартный поток вывода
cat file.Z | uncompress	Разжать данные из канала
zcat file.Z	Разжать данные в стандартный поток вывода

tar-файлы, сжатые с помощью compress: примеры команд

tar czf myfile.tar.z mydir	Упаковать директорию mydir
tar tzf myfile.tar.z	Вывести список содержимого
tar xzf myfile.tar.z	Распаковать

Добавьте опцию v для команды tar, чтобы выводить имена файлов по мере их обработки.

bzip2 [опции] [файль/\ bzip2 /usr/bin stdin stdout -file -opt -help -version

Команды bzip2 и bunzip2 сжимают и разжимают файлы согласно формату Burrows-Wheeler. Сжатые файлы имеют суффикс .bz2.

bzip2 file bzip2 -c file cat file | bzip2 bunzip2 file.bz2 bunzip2 -c file.bz2

Примеры команд

bunzip2 file	Сжать файл ff/ев файл §e.bz2. Исходный файл удаляется
bunzip2 – c file	Вывести сжатые данные в стандартный поток вывода
cat file.| bzip2	Вывести сжатые данные в стандартный поток вывода
bunzip2 file.bz2	Разжать файл Ше.Ьг2ъ Hie. Исходный файл /й?.Лг?удаляется

bunzip2 –c file.bz2 Разжать данные в стандартный поток вывода
cat file.bz2| bunzip2 Разжать данные в стандартный поток вывода
bzcat file.bz2 Разжать данные в стандартный поток вывода

tar-файлы, сжатые с помощью bzip2: примеры команд

tar cjf myf ile. tar .bz2 dirname Упаковать
tar tjf -myf ile. tar .bz2 Вывести список содержимого
tar xjf myf ile. tar .bz2 Распаковать

Добавьте опцию v для команды tar, чтобы выводить имена файлов по мере их обработки.

zip [опции] [файлы] zip
/usr/bin stdin stdout -file --opt --help --version

Команды zip и unzip сжимают и разжимают файлы согласно формату Windows Zip. Сжатые файлы имеют суффикс .zip. В отличие от программ gzip, compress и bzip2, программа zip не удаляет исходный файл (файлы).

zipmyfile.zip filel file2 file3. .. Упаковать
zip -r myfile.zip dirname Упаковать рекурсивно
unzip -I myfile.zip Вывести список содержимого
unzip myfile.zip Распаковать

uuencode [опции] новый файлвходной файл shamb\s
/usr/bin stdin stdout -file -opt --help --version

До того как появились функции прикрепления файлов к письмам и MIME, приходилось повозиться с бинарными файлами для того, чтобы отправить их по электронной почте. Сначала нужно было преобразовать файл с помощью программы uuencode в ASCII-формат, который выглядит примерно так.

```
begin 644 myfile
M(R4N8F%S:%]P<F]F:6QE"B,@4G5N<R!F:
7)S="!W:&5N(&QO9V=I;F<@:6X6
- M=6YD97(@1TY/344*"G1R874@)
PH@('1E<WO@+6X@(B134TA?04=%3E1?4$E$
end
```

После получения этих данных получатель должен был преобразовать их с помощью программы uudecode, чтобы восстановить исходные дынные.

Для того чтобы преобразовать файл myfile в формат uuencode, т. е. создать файл myfile.uu введите следующую команду.

$ uuencode newfile myfile > myfile.uu

Первый аргумент, newfile, - это имя файла, который будет создан при декодировании. Оно появится в первой строке выходных преобразованных данных.

begin 644 newfile
M(R~N8F%S:%]P<F]F:6QE"B,@4G5N<R!F:7)S="!W:
&5N(&QO9V=I;F<@:6X@

Чтобы декодировать файл myfile.uu, и создать файл newfile, выполните следующую команду.

$ uudecode myfile.uu

Сравнение файлов

diff	Построчное сравнение двух файлов или директорий
comm	Построчное сравнение двух сортированных файлов
cmp	Побайтное сравнение двух файлов
md5sum	Вычислить контрольную сумму заданного файла (MD5)

Есть три способа сравнивать файлы в Linux:

- построчный (dif f, dif f3, sdif f, comm), лучше всего подходит для текстовых файлов;
- побайтный (cmp), часто используется для бинарных файлов;
- сравнивать контрольные суммы (md5sum, sum, cksum).

Все эти программы консольные. Пример графического инструмента сравнения файлов - xxdif f, который можно найти на сайте http://xxdiff.sourceforge.net.

diff [опции] файл1 файл2 diffutils /usr/bin stdin stdout -file —opt --help -version

Команда diff сравнивает два файла построчно, либо сравнивает директории. Сравнивая два текстовых файла, diff может создавать детальный отчет об их различиях. В случае бинарных файлов команда diff сообщает лишь о том, различны они или нет. Если различий нет, diff не создает выходных данных; это верно для любых типов файлов.

Традиционный формат выходных данных выглядит следующим образом. .

Номера строк и тип изменения

< Соответствующий раздел файла file 1, если он есть

> Соответствующий раздел файла fde2, если он есть Например, сначала мы имеем файл/z/e_a.

Hello, this is a wonderful file.
The quick brown fox jumped over
the lazy dogs.
Goodbye for now.

Представьте себе, что мы удалили первую строку, заменили слово "brown" на "blue" во второй строке и добавили строку в конец, создав файл filejb.

The quick blue fox jumped over
the lazy dogs.
Goodbye for now.
Linux rOOlz!

Тогда выходные данные команды diff file_a f ile_b будут следующими.

1, 2cl строки 1-2 файла filea перейти в строку I файла file_b
<Hello, this is a wonderful file. Строки
1-2 файла file_a
<The quick brown fox jumped over
разделитель diff
>The quick blue fox jumped over Строка
1 файла filejb

4a4 Строка 4 была добавлена
>Linux rOOlz ! Добавленная строка

Начальные символы < и > - это стрелки, обозначающие файлы filea и file_b соответственно. Такой формат вывода стандартный: существует много других форматов, некоторые из которых можно перенести в другие программы. Попробуйте использовать их, чтобы посмотреть, как они выглядят.

Опция	**Формат выходных данных**
-n	Формат управления версиями RCS, как у команды re sdif f (man rcsdif f)
- c	Контекстный формат di f f, как у команды patch (man patch)
-D macro	Формат препроцессора языка Си с использованием макросов #ifdef macro... #else ... tendif
-u	Унифицированный формат, который сливает файлы и обозначает знаком "-" удаления и знаком "+" вставки
-y	Двухколоночный формат; используйте опцию -W для изменения ширины выходных данных
-e	Создать ed-скрипт, который при выполнении превращает файл file a ъ файл file b
-q	Не сообщать об изменениях, а только о самом факте различия файлов

Также команда diff может сравнивать директории. Следующая команда:

$ diff dirl dir2

сравнивает все файлы с одинаковыми именами в этих директориях, и выводит список всех файлов, которые есть в одной директории, но отсутствуют в другой. Если вы хотите сравнить иерархии директорий полностью, рекурсивно, используйте опцию -г:

$ diff -г dirl dir2

которая создает (потенциально большой) отчет о всех различиях.

Полезные опции

-b Не учитывать пробельные символы
-в Не учитывать пустые строки
- i Игнорировать регистр
- r При сравнении директорий рекурсивно обрабатывать поддиректории

diff- это только один член семейства программ, которые работают с различиями файлов. Вот некоторые из других программ: dif f 3, которая сравнивает три файла одновременно, и sdif f, которая объединяет различия между двумя файлами и создает из них третий файл согласно вашим указаниям.

comm [опции\ файл1 файл2 coreutils
/usr/bin stdin stdout -file —opt —help —version

Команда comm сравнивает два сортированных файла и создает три колонки выходных данных, разделенных знаками табуляции.

1. Все строки, которые есть в файле file], но отсутст вуют в файле file2.
2. Все строки, которые есть в файле file2, но отсутствуют в файле file 1.
3. Все строки, которые есть в обоих файлов.

Например, если файлы file J и file2 содержат такие строки:

filel: file2:
apple baker
baker charlie
charlie dark

то выходные данные команды comm будут такими:

$ comm filel file2 apple
baker
charlie dark

Полезные опции

-1 Опустить колонку 1
- 2 Опустить колонку 2
-3 Опустить колонку 3

cmp [опции] файл1 файл2[смещение1 [смещение2\] diffutils /usr/bin stdin stdout -file --opt -help -version

Команда cmp сравнивает два файла. Если их содержимое одинаковое, то команда cmp ничего не сообщает, а если разное, то она выводит местоположение первого отличия. .

$ cmp myfile yourfile
myfile yourfile differ: char 494, line 17

По умолчанию, команда cmp не сообщает о том, какого типа изменение, а только о том, где оно. Она идеально подходит для сравнения бинарных файлов, в отличие от команды diff, которая лучше всего подходит для работы с текстовыми файлами.

Как правило, команда cmp начинает сравнение с начала каждого файла, но она может начинать сравнение и с другого места, если вы укажете смещение.

$ cmp myfile yourfile 10 20

Эта команда начнет сравнение с десятого символа файла myfile и с двадцатого символа файла yourfile.

Полезные опции

-1 Детальный вывод: выводить все различия побайтно:
$ cmp -I myfile yourfile
494 164 172
Это означает, что на позиции, смещенной на 494 (в десятичной системе) байта, в файле myfile находится символ "t" (164 в восьмеричной системе), а в файле yourfile - символ "z" (172 в восьмеричной системе)

- s Подавлять выходные данные: не выводить ничего, сразу

же завершать работу и возвращать код ошибки; 0, если файлы одинаковые, 1 - если отличаются (или другое значение, если сравнение не было выполнено по какой-либо причине)

md5sum файлы | --check файл coreutils
/usr/bin stdin stdout -file —opt -help —version

Команда md5sum выводит 32-байтную контрольную сумму для заданных файлов, используя алгоритм MD5 (более подробная информация на http:// www.faqs.org/rfcs/rfc 1321 .html).

```
$ md5sum myfile
dd63 602dflcceb57 96 6d085524c3 980f
myfile
```

Два разных файла весьма маловероятно будут иметь одинаковую контрольную сумму MD5, поэтому сравнение контрольных сумм - это достаточно надежный способ определить различие файлов:

```
$ md5sum myfilel > suml
$ md5sum myfile2 > sum2
$ diff -q suml sum2
Files suml and sum2 differ
Файлы suml и sum2 различаются
или что набор файлов изменился, с помощью опции
--check:
$ md5sum filel file2 file3 > mysum
$ md5sum --check mysum
filel: OK ':
file2: OK
f ile3 : OK ',..-.
$ echo "новые данные" > file2
$ md5sum --check mysum
filel: OK
file2: FAILED
file3 : OK .. . .
```

md5sum: WARNING: 1 of 3 computed . checksums did NOT match Внимание: I из 3 контрольных сумм не совпадает

Еще две программы, похожие на md5sum, - это программы sum и cksum, которые используют различные алгоритмы для подсчета контрольных сумм. Программа sum совместима с другими Unix-системами, в частности BSD Unix (по умолчанию) или System V Unix (с опцией -s), а cksum создает контрольную сумму CRC.

$ sum myfile
12410 3
$ sum -s myfile
47909 6 myfile
$ cksum myfile
1204834076 2863 myfile

Первое целое число - это контрольная сумма, а второе - количество блоков. Но, как вы можете видеть, эти контрольные суммы являются небольшими числами и, таким образом, не надежны, так как файлы могут случайно иметь идентичные контрольные суммы. md5sum намного лучше подходит для идентификации.

Диски и файловые системы

Df	Вывести информацию о свободном месте на смонтированных файловых системах
mount	Смонтировать логический диск (он будет доступен)
umount	Размонтировать логический диск (он будет недоступен)
fsck	Проверить файловую систему на ошибки
sync	Сбросить содержимое всех дисковых кэшей на диск

Linux-системы могут иметь несколько дисков или дисковых разделов. В обычном разговоре они называются по-разному: дисками, разделами, файловыми системами, томами, даже директориями. Мы постараемся быть более точными.

Диск - это физическое устройство, он может быть разбит на разделы, которые будут восприниматься как независимые устройства хранения информации. Разделы представляются в Linux-системах в виде специальных файлов, как правило в директории /dev. Например, файл /dev/hda7 может соответствовать разделу на вашем IDE master-

диске. Вот некоторые часто встречающиеся в директории /dev устройства.

hda	Первая IDE-шина, master-устройство; его разделы обозначаются ши Mai, hda2,...
hdb	Первая IDE-шина, slave-устройство; его разделы обозначаются *жМЬ1,МЬ2,...
hdc	Вторая IDE-шина, master-устройство; его разделы обозначаются \/жМс1,Мс2,...
hdd	Вторая IDE-шина, slave-устройство; его разделы обозначаются ШШ2
sda	Первое SCSI-устройство; его разделы обозначаются как sda1, sda2,...
sdb	Второе SCSI-устройство; его разделы обозначаются как sdbi, sdb2,... Аналогично для sdc, sdd,...
htO	Первый ленточный накопитель IDE(затем htl, ht2,...) с автоматической перемоткой
пМО	Первый ленточный накопитель ЮЕ(затем nht1, nht2,...) без автоматической перемотки
stO	Первый ленточный накопитель ЭСБКзатем sti, st2,...)
scdO	Первый SCSI привод CD-ROM (затем scd1, scd2,...)
fdO	Первый флоппи-дисковод (затем fdl, fd2,...), как правило, монтируется в директорию /mnt/floppy

До того как в раздел можно будет помещать файлы, он "форматируется" посредством записи на него файловой системы. Файловая система определяет, как будут размещаться файлы; примеры файловых систем - ext3 (журналируемая файловая система Linux, стандартная в Fedora) и vfat (файловая система Microsoft Windows). Форматирование, как правило, осуществляется при установке Linux.

Когда файловая система создана, вы можете сделать ее доступной, смонтировав ее в пустую директорию. Например, если вы монтируете файловую систему Windows в директорию /mnt/win, то она становится частью иерархии директорий вашей системы, и вы можете создавать и редактировать такие файлы, как, например, /mnt/win/myfile. Также файловые системы можно размонтировать, делая их недоступными, например, с целью профилактики.

Монтирование жестких дисков, как правило, осуществляется автоматически при загрузке.

df [опции] [дисковыеустройства /файлы/ директории] coreutils /bin stdin stdout -file --opt -help -version

Программа df (от англ. disk free) выдает общий объем, объем использованного и свободного пространства на заданном разделе диска. Если вы указываете файл или директорию, то df описывает дисковое устройство, на котором располагается этот файл или директория. При отсутствии аргументов команда df выводит информацию о всех смонтированных файловых системах.

$ df
Filesystem Ik-blocks Used Available Use% Mounted on
/dev/hda 1011928 225464 735060 24%/ /dev/hda9 521748 249148 246096 51%/var /dev/hda8 8064272 4088636 3565984 54%/usr /dev/hdalO 8064272 4586576 3068044 60%/home

Полезные опции

-к	Выводить все размеры в килобайтах (стандартно) или
-m	мегабайтах соответственно
-в N	Выводить размеры в блоках, которые вы определите, где 1 блок = Л'байт (по умолчанию 1 блок = 1024 байта)
-h	Выводить в удобном для восприятия формате и выбирать -н наиболее подходящие единицы измерения для каждого размера.

Например, если на двух ваших дисках имеется 1 гигабайт и 25 килобайт свободного пространства соответственно, то команда df -h выведет 1G и 25К. Опция -h использует степени 1024, тогда как опция -н - степени 1000

-1	Выводить информацию только о локальных файловых системах, опуская сетевые
-m	Включать в выходные данные тип файловой системы (ex12, vfat и т. д.)
-1тип	Выводить информацию о файловых системах только заданного типа

- x тип	Не выводить информацию о файловых системах заданного типа
- i	lnode-режим. Выводить полное количество, количество использованных и свободных информационных дескрипторов (i-node) для каждой файловой системы вместо блоков диска

mount [опции] устройство директория mount /bin stdin stdout -file -opt --help -version

Команда mount делает доступным физическое устройство хранения данных. Наиболее часто она используется с дисковыми устройствами (скажем, /dev/hda/), делая их доступными через существующую директорию (скажем, /mnt/mydir).

```
#mkdir /mnt/mydir
#mount /dev/hdal /mnt/mydir
#df /mnt/mydir
Filesystem lK-blocks Used Available Use% Mounted on
/dev/hdal 1011928 285744 674780 30% /mnt/mydir
```

Команда mount имеет множество опций и способов применения; мы рассмотрим только самые основные из них.

В большинстве случаев mount читает информацию из файла /etc/fstab (таблица файловых систем), чтобы узнать, как смонтировать нужный диск. Например, если вы наберете mount /usr, то команда mount будет искать строку "/usr" в файле /etc/fstab, которая может выглядеть, например, следующим образом.

```
/dev/hda8 /usr ext3 defaults 1 2
```

Из этой строки команда mount узнает, помимо прочего, что дисковое устройство /dev/hda8 нужно монтировать в директорию /usr как файловую систему типа ext3*.

Как правило, команда mount выполняется суперпользователем, но такие общие устройства, как флоппи-дисковод или CD-ROM, часто могут монтироваться и размонтироваться любым пользователем.

$ mount /mnt/cdrom
$ mount /rant/floppy

* В качестве альтернативы вы можете использовать опцию -t команды mount для того, чтобы задать непосредственно тип файловой системы, например mount -t ext3 /dev/hdal /mnt/mydir. Посмотрите информацию, которую дает команда man mount.

umount [опции] [устройство \ директория] mount
/bin stdin stdout -file --opt -help -version

Команда umount - противоположность mount: она делает недоступным раздел диска. Имейте в виду, если вы смонтировали CD-ROM, то вы не сможете вынуть его из дисковода до тех пор, пока не размонтируете диск с помощью команды umount.

$ umount /mnt/cdrom

Перед извлечением сменных носителей из дисководов всегда размонтируйте их, иначе вы рискуете повредить их файловую систему. Чтобы размонтировать сразу все смонтированные устройства, выполните следующую команду.

 # umount -a

Не размонтируйте файловую систему, которая используется в данный момент; фактически, команда umount откажет в выполнении из соображений безопасности.

fsck [опции] [устройств^ e2fsprogs
/sbin stdin stdout -file --opt --help --version

Команда fsck (от англ. filesystem check) проверяет разделы Linux-диска и, если требуется, исправляет найденные на нем ошибки. Программа fsck запускается автоматически при загрузке системы; однако вы можете запустить ее вручную, если захотите. Вообще говоря, надо размонтировать устройство, прежде чем проверять его, чтобы никакая другая программа не работала с ним во время проверки.

#umount /dev/hdalO
#fsck -f /dev/hdalO

Pass 1: Checking inodes, blocks, and sizes Pass 2: Checking directory structure Pass 3: Checking directory connectivity Pass 4: Checking reference counts Pass 5: Checking group summary information /home: 172/1281696 files (11.6% noncontiguous) , 1405555/2562359 blocks

Программа fsck - это интерфейс для ряда программ проверки файловых систем из директории / sbin с названиями, начинающимися с "fsck". Поддерживаются только определенные типы файловых систем; вы можете вывести их список с помощью следующей команды.

$ Is /sbin/fsck.* | cut -d. -f2
fc."

Полезные опции

- A	Проверить все диски, перечисленные в файле /etc/fstab, no порядку
-N	Вывести описание проверки, которая будет проведена, но завершить работу, не проводя ее
- r	Исправлять ошибки интерактивно, запрашивая подтверждение перед каждым исправлением
- a	Исправлять ошибки автоматически (только если вы действительно знаете, что делаете; в противном случае вы можете серьезно повредить файловую систему)

sync coreutils
/bin stdin stdout -file -opt --help --version

Команда sync сбрасывает на диски содержимое всех дисковых кэшей. В нормальном режиме ядро может буферизовать в памяти операции чтения, записи, изменений атрибутов и другие операции с диском. Команда sync записывает накопленные изменения на диск. Как правило, вам не нужно выполнять эту команду, но если (скажем) вы собираетесь проделать какую-то рискованную операцию, которая

может вывести ваш компьютер из строя, то не помешает выполнить непосредственно перед этим команду sync.

Разбиение и форматирование дисков

Такие дисковые операции, как разбиение и форматирование, могут быть сложными в Linux-системах. Вот, для начала, несколько программ, которые вам могут понадобиться (начните с изучения их man-страниц).

parted, fdisk или s fdisk	Разбить (на разделы) жесткий диск. Это делает каждая из указанных программ: просто они имеют различные пользовательские интерфейсы
mkf s	Форматировать жесткий диск, т. е. создать новую файловую систему
f I oppy	Форматировать гибкий диск

Резервное копирование и удаленное хранение

mt	Управление накопителем на магнитной ленте
dump	Записать раздел диска на ленту
restore	Восстановить архив dump
tar	Запись и чтение архивов на ленте
cdrecord	Записать CD-R-диск
rsync	Сделать копию набора файлов на другом устройстве или хосте

Есть несколько способов сделать резервные копии ваших драгоценных файлов в Linux:

- скопировать их на магнитную ленту;
- записать их на CD-R-диск;
- сделать их копию на удаленной машине.

Как правило, ваше ленточное резервное устройство соответствует файлу /dev/htO, если у него интерфейс IDE или файлу /dev/stO в случае SCSI (или IDE с ide-scsi-эмуляцией). Общепринятая практика -делать ссылку с именем /dev/tape на соответствующее устройство.

$ In -s /dev/htO /dev/tape

Мы не описываем все команды Linux для создания резервных копий. Некоторые пользователи предпочитают cpio команде tar, а для низкоуровневых копий диска неоценима команда dd. Обратитесь к man-страницам этих программ, если они вас заинтересовали.

mt mt-st
/bin stdin stdout -file -opt -help --version

Команда mt (от англ. magnetic tape) осуществляет простые операции с магнитной лентой в лентопротяжном устройстве, например перемотку, прогон вперед и назад и сохранение. Вот некоторые часто используемые операции.

status	Показать состояние устройства
rewind	Перемотать ленту
retension	Перенатянуть ленту (одна полная перемотка вперед и назад)
erase	Стереть ленту
o f f I ine	Отключить устройство
eod	Переместиться вперед по ленте до конца данных

Например:

$ mt -f /dev/tape rewind

Также вы можете перемещаться по ленте, по файлам или по записям, но, как правило, вы будете использовать для этого программу для чтения/записи, например, tar или restore.

dump [опции] разделили файлы dump
/bin stdin stdout -file --opt --help --version

Команда dump записывают целый раздел диска, или отдельные файлы, на резервный носитель, например ленту. Она поддерживает полное и инкрементальное резервное копирование, автоматически определяя, какие файлы нужно скопировать (т. е. какие файлы изменились с момента последнего резервного копирования). Чтобы

восстановить файлы с резервного устройства, используйте команду restore.

Чтобы проделать полное резервное копирование заданной файловой системы (скажем, /usr) на ваше ленточное устройство (скажем, /dev/tape), используйте опции -0 (нуль) и -и.

#dump -0 -u -f /dev/tape /usr

Это называется кпированием нулевого уровня. Опция -и делает запись в файл /etc/dumpdates, сообщающую о том, что было выполнено резервное копирование.

Инкрементальное резервное копирование может иметь от 1 до 9 уровней: резервная копия i-го уровня содержит все новые и измененные файлы с момента создания последней резервной копии (i-l)-го уровня.

#dump -I -u -f /dev/tape /usr

Не выполняйте команду dump на "живой" файловой системе, активно используемой в данный момент: размонтируйте ее по возможности.

restore [опции] [файлы] ■-- dump
/sbin stdin stdout -file --opt --help --version

Команда restore читает резервные копии (дампы), созданные командой dump. Затем она может восстанавливать файлы на диск, сравнивать их с их версиями на диске и выполнять другие операции. Самый удобный способ использования команды restore - с опцией -i , чтобы работать в интерактивном режиме, который позволяет просматривать содержимое ленты почти как файловую систему, выбирая файлы и директории и, наконец, восстанавливая их.

 # restore -i -f /dev/tape

Затем restore предложит вам ввести одну из следующих команд.

help	Вывести справочное сообщение
quit	Выйти из программы без сохранения

	файлов
cd директория	Аналогично команде cd командного процессора, установить вашу текущую рабочую директорию в дампе для работы с файлами
Is	Аналогично Linux-команде Is, просмотреть все файлы в текущей рабочей директории дампа
pwd	Аналогично команде pwd командного процессора, вывести имя вашей текущей рабочей директории в дампе
add	Добавить файлы или директории в"список извлечения" - список файлов, которые вы хотите восстановить. При отсутствии аргументов команда add добавляет текущую директорию и все находящиеся в ней файлы
add имя_файла	Добавить файл имя файла в список извлечения add директория Добавить директорию в список извлечения
delete	Противоположность команды add: удалить файлы из списка извлечения. При отсутствии аргументов команда delete удаляет текущую директорию (и ее содержимое) из списка извлечения
delete имя_файла	Удалить файл имя файла из списка извлечения delete директория Удалить директорию из списка извлечения
extract.	Восстановить все файлы, которые вы добавили в список извлечения (совет: если ваша резервная копия располагается на нескольких лентах, начните с последней ленты)

Также команда restore может выполняться в других, неинтерактивных, режимах.

res tore -x	Восстановить всю информации с ленты в существующую файловую систему

	(сначала перейдите в корневую директорию соответствующей файловой системы)
restore –r	Восстановить всю информации с ленты в отформатированный раздел диска (сначала перейдите в корневую директорию соответствующей файловой системы)
res tore -t	Вывести список содержимого дампа
restore –c	Сравнить дамп с исходной файловой системой

tar [опции] [файль/\ tar
/bin stdin stdout -file --opt -help -version

Программа tar (от англ. tape archive) не только читает файлы с ленточных устройств и записывает их туда:

$ tar -cf /dev/tape myfilel myfile2

но также позволяет создавать tar-файлы, которые являются стандартным средством упаковки файлов в Linux- и Unix-системах, и извлекать из них данные.

$ tar -czvf my_archive.tar. gz mydir Создать
$ tar -tzvf my_archive.tar.gz Вывести список содержимого
$ tar -xzvf my_archive.tar.gz Извлечь

Если вы указываете файлы в командной строке, то только они будут обработаны.

$ tar -xvf /dev/tape filel file2 file3

В противном случае будет обработан весь архив.

Полезные опции

-c	Создать архив. Вам нужно будет перечислить входные файлы и директории в командной строке
-r	Поместить файлы в существующий архив

-u Поместить новые/измененные файлы в существующий архив
-A Поместить один архив (например, tar-файл) в конец другого архива: например, tar -A -f /dev/tape myfile.tar
-1 Вывести список содержимого архива
-x Извлечь файлы из архива
-f файл Читать архив из, или записать архив в заданный файл. Этим файлом может быть устройство (например, /dev/tape) или обычный файл, если вы хотите создать традиционный tar-файл Linux
-d Сравнить архив с файловой системой
-z Сжать (при записи) или восстановить сжатые (при чтении)
... данные с помощью gzip
-j Сжать (при записи) или восстановить сжатые (при чтении) данные с помощью bz ip2
-z Сжать (при записи) или восстановить сжатые (при чтении) данные с помощью compress
-b N Использовать блоки размера Л'* 512 байт
-v Подробный режим: выводить дополнительную информацию
-h Разыменовывать символьные ссылки
-1 Не выходить за границы файловой системы
-p При извлечении файлов восстанавливать их исходные права и владельцев

cdrecord [опции] cdrecord /usr/bin stdin stdout -file --opt -help -version

Команда cdrecord записывает CD-R-диск в SCSI- или IDE-устройстве, использующем ide-scsi эмуляцию. Чтобы записать содержимое директории на

CD-ROM, который можно было бы читать в Linux-, Windows- и Macintosh-системах*.

1. Найдите ваше устройство для записи компакт-дис ков, выполнив следующую команду.

$ cdrecord --scanbus

0,0,0 0) *
0,1,0 1) *

0,2,0 2) *
0,3,0 3) 'YAMAHA "CRW6416S "l.Od1
Removable CD-ROM
••-.*;? - - .

В данном случае это устройство 0,3,0.

2. Узнайте скорость записи CD-R- или CD-RW-дисков (в зависимости от того, какой тип вы используете) вашего записывающего устройства. Предположим, это 6-скоростное устройство записи CD-R-дисков, поэтому скорость равна 6.

3.Поместите файлы, которые вы хотите записать, в директорию, скажем, dir. Разместите их в точности так, как вы хотели бы их видеть на компакт-диске. Сама директория dir не будет скопирована на компакт-диск, будет скопировано только ее содержимое.

4. Запишите компакт-диск:

```
$ DEVICE="0,3,0"
$ SPEED=6
$ mkisofs -R -1 dir > mydisk.iso
$ cdrecord -v dev=${DEVICE}
speed=${SPEED} mydisk.iso
```

* В частности, компакт-диск ISO9660 с расширением Rock Ridge. Команда mkisofs может создавать другие форматы, которые сможет записать команда cdrecord: изучите информацию, которую выдает команда man mkisofs.

или, если ваша система достаточно быстрая, вы можете проделать все это в одном конвейере:

```
$ mkisofs -R -1 dir \
| cdrecord -v dev=${DEVICE}
speed=${SPEED} -
```

Также команда cdrecord может записывать музыкальные компакт-диски, но вместо нее вы, возможно, захотите использовать более удобную гра-фическую программу, например xcdroast (см.раздел "Аудио и видео" на странице 251), которая основана на cdrecord.

rsync [опции] источник место назначения rsync
/usr/bin stdin stdout -file —opt —help —version

Команда rsync копирует набор файлов. Она умеет делать точные копии, включая права на файл и другие атрибуты (это называется зеркальным отображением или зеркалированием), либо она может просто копировать данные. Она может работать по сети или на одиночном компьютере. Команда rsync имеет много способов использования и более 50 опций; мы рассмотрим только некоторые наиболее общие случаи, имеющие отношение к резервному копированию.

Чтобы отобразить директорию dirl и ее содержимое в другую директорию dir2 на этом же компьютере, выполните следующую команду.

$ rsync -a dirl dir2

Для того чтобы отобразить директорию dirl по сети на другой хост, например server.example.com, на котором у вас есть учетная запись с именем пользователя smith по безопасному соединению SSH, чтобы предотвратить "прослушивание", выполните следующую команду.

$ rsync -a -e ssh dirl smith@server.example.com:

Полезные опции

-o Копировать информацию о владельце файла (возможно, на удаленном хосте вам понадобятся права суперпользователя)
-g Копировать информацию о группе файла (возможно, на удаленном хосте вам понадобятся права суперпользователя)
-p Копировать права доступа на файл
-t Копировать временные метки
-r Копировать директории рекурсивно, т. е. включая их содержимое
-1 Разрешить копирование символьных ссылок (а не тех

файлов, на которые они указывают)

-D Разрешить копирование устройств (только для суперпользователя)

-a Зеркалирование: копировать все атрибуты исходных файлов. Сочетает все опции: -Dgloprt

-v Подробный режим: вывести информацию о том, что происходит во время копирования. Добавьте - -progress для того, чтобы отображался числовой индикатор прогресса во время копирования файлов

-e команда Задать альтернативную программу соединения с удаленным хостом, например, ssh для обеспечения большей безопасности

Печать файлов

Ipr Напечатать файл

Ipq Просмотреть очередь печати

Iprm Удалить задание из очереди печати

В Linux есть две популярные системы печати, CUPS и LPRng; дистрибутив Fedora включает в себя систему CUPS. Обе системы использую команды с одинаковыми именами: Ipr, Ipq и Iprm. Однако эти команды имеют различные опции в зависимости от того, какую систему вы используете: CUPS или LPRng. Чтобы убить сразу двух зайцев, мы рассмотрим общие опции, которые используются в обеих системах.

Чтобы установить принтер, который будет использоваться в Fedora, выполните команду:

 # redhat-config-printer

и следуйте указаниям.

Ipr [опции] [файлы] cups /usr/bin stdin stdout -file -opt --help --version

Команда Ipr (от англ. line printer) отправляет файл на принтер.

$ Ipr -P irtyprinter myfile

Полезные опции

-P имя_принтера	Отправить файл на принтер имя принтера, который вы до этого установили посредством утилиты redhat-conf ig-printer
-# N	Напечатать #копий файла
-J название	Установить название задания, которое будет печататься на титульной странице (если ваш принтер установлен на печать титульной страницы)

lpq [опции] cups
/usr/bin stdin stdout -file -opt --help --version

Команда lpq (от англ. line printer queue) выводит список всех заданий печати, ожидающих своей очереди.

Полезные опции

-p имя_принтера	Вывести очередь заданий для принтера имя_ принтера
-a	Вывести очередь заданий для всех принтеров
-1	Подробный вывод: выводить информацию в более детальном формате

lprm [опции] [идентификаторызаданий] cups
/usr/bin stdin stdout -file --opt --help --version

Команда lprm (от англ. line printer remove) отме-няет одно или более заданий печати. Используйте команду lpq для того, чтобы узнать идентификатор нужного задания печати (скажем, 61 или 78), а затем выполните следующую команду:

$ lprm -P printername 61 78

Если вы не укажете идентификатор задания, то бу-дет отменено ваше текущее задание печати (только суперпользователь может отменять задания других пользователей) Опция -P указывает, какая очередь печати содержит задание.

Операции контроля правописания

look	Быстрый поиск правописания слова
aspell	Интерактивный блок проверки правописания
spell	Блок групповой проверки правописания

Linux имеет несколько встроенных программ проверки правописания. Если вы привыкли к графическим средствам проверки правописания, то программы Linux могут показаться вам примитивными, но их можно использовать в конвейерах, и это весьма мощное средство.

look [опции] префикс[словарный_файл\иX\\-\\пт
/usr/bin stdin stdout -file —opt --help --version

Команда look выводит (в стандартный поток вывода) слова, которые начинаются с заданной строки префикс. Слова эти расположены в словарном файле (обычно /usr/share/dict/words). Например, выходные данные команды look bigg будут следующими.

bigger
biggest
Biggs

Если вы укажете ваш собственный словарный файл - любой текстовый файл с отсортированными в алфавитном порядке строками, то команда look выведет все строки, начинающиеся с заданного префикса.

Полезные опции

- f	Игнорировать регистр
-1 x	Искать соответствие префиксу только до символа X включительно. Например, команда look -t i big выведет все слова, начинающиеся с "Ы"

aspell [опции] файл \ команда aspell
/usr/bin stdin stdout -file --opt -help -version

aspell - это мощный пакет проверки правописания с множеством опций. Вот несколько полезных команд.

aspell -c file

Интерактивно проверяет правописание и, опционально, исправляет ошибки во всех словах в файле file.

aspell -I < file

Выводит список слов файла file с неправильным написанием в стандартный поток вывода.

aspell dump master

Выводит основной словарь aspell в стандартный поток вывода.

aspell help

Выводит краткое справочное сообщение. Для более подробной информации посетите сайт http://aspell.net.

spell [файлы] aspell
/usr/bin stdin stdout -file —opt -help -version

Команда spell выводит все слова заданных файлов, которые написаны неправильно, согласно ее словарю. Она эквивалентна следующей команде.

$ cat files \ aspell -I | sort -u

Если не указан ни один файл, то команда spell читает данные из стандартного потока ввода.

Мониторинг процессов

ps	Вывести список процессов
uptime	Посмотреть загрузку системы
w	Вывести список активных процессов для всех

	пользователей
top	Интерактивно наблюдать за процессами, интенсивно использующими ресурсы системы
xload	Графический вывод статистики загрузки системы в отдельном окне
free	Вывести объем свободной памяти

Процесс - это единица работы в системе Linux. Каждая программа, которую вы запускаете, представляет из себя один или более процессов, и Linux предоставляет команды для мониторинга и управления ими. Каждый процесс определяется ID-номером процесса, или PID-номером (идентификатор процесса).

Процессы - это не то же самое, что задачи (см. раздел "Управление задачами" на странице 53): процессы являются частью операционной системы, тогда как о задачах известно только командному процессору, в котором они выполняются. Работающая программа заключает в себе один или более процессов; задача состоит из одной или более программ, выполняемых в виде команд командного процессора.

ps [опции] procps
/bin stdin stdout -file -opt -help --version

Команда ps выводит информацию о ваших работающих процессах и, опционально, о процессах других пользователей.

```
$ ps
PID TTY      TIME CMD
4706 pts/2   00:00:01 bash
15007 pts/2  00:00:00 emacs
16729 pts/2  00:00:00 ps
```

Команда ps имеет по крайней мере 80 опций; мы рассмотрим только несколько полезных комбинаций. Если опция кажется вам произвольной или нелогичной, это потому, что команда ps (GNU ps) включает в себя функции нескольких других Unix-реализаций ps, в попытке быть совместимой с ними.

Для просмотра списка всех процессов выполните команду:

$ ps –ux

всех процессов пользователя smith:

$ ps -U smith

всех работающих экземпляров программы program_ пате:

$ ps -C рrодгат_name

процессов в терминале N:

$ ps -tW

конкретных процессов 1, 2 и 3505:

$ ps -pl,2,3505

всех процессов с командными строками, урезанными шириной экрана:

$ ps -ef всех процессов с полными командными строками:

$ ps -efww

и всех процессов в виде дерева, когда дочерние процессы располагаются ниже родительских с отступом:

$ ps -efH

Помните, что вы можете извлекать более лаконичную информацию из выходных данных команды ps, используя команду grep или другие фильтрующие программы.

uptime procps

/usr/bin stdin stdout -file --opt --help --version

Команда uptime сообщает о том, как долго работает система с момента последней загрузки.

```
$ uptime
10:54pm up 8 days, 3:44, 3 users,
load average: 0.89, 1.00, 2.15
```

Эта информация включает в себя, справа налево: текущее время (10:54pm), время работы системы (8 дней, 3 часа, 44 минуты), количество работающих в системе пользователей (3) и среднюю загрузку системы за три отрезка времени - одну минуту (0.89), пять минут (1.00) и пятнадцать минут (2.15). Средняя загрузка - это среднее число процессов, готовых к выполнению в данный интервал времени.

w [имя пользователя] procps
/usr/bin stdin stdout -file --opt --help --version

Команда w выводит текущий процесс для каждого работающего в системе пользователя, или, более точно, для каждого командного процессора каждого пользователя.

$ W
10:51pm up 8 days, 3:42, 8 users, load average: 0.00, 0.00, 0.00 USER TTY FROM LOGIN0 IDLE JCPU PCPU WHAT barrett pts/0 :0 Sat 2pm 27:13m 0.07s 0.07s emacs jones pts/1 hostl 6SepO3 2:33m 0.74s 0.21s bash smith pts/2 host2 6SepO3 0.00s 13.35s 0.04s w

Верхняя строка аналогична строке, которую выводит команда uptime. Колонки обозначают терминал пользователя, хост или X-дисплей (если это применимо^ которого был осуществлен вход в систему, время входа, время простоя, два измерения процессорного времени (выполните команду man w для более подробной информации) и текущий процесс. Укажите имя пользователя, чтобы увидеть информацию только об этом пользователе.

Чтобы максимально сократить вывод, попробуйте выполнить команду w -hf s.

Полезные опции

- h Не выводить строку заголовка
- f Не выводить колонку FROM
- s Не выводить колонки JCPU и PCPU

top [опции\ procps
/usr/bin stdin stdout -file --opt --help --version

Команда top позволяет следить за наиболее активными процессами, обновляя данные через определенные интервалы времени (скажем, каждую секунду). Это консольная программа, которая обновляет данные интерактивно.

```
$ top
116 processes: 104 sleeping, 1 running,
0 zombie, 11 stopped
CPU states: 1.1% user, 0.5% system,
0.0% nice, 4.5% idle
Mem: 523812K av, 502328K used, 21484K free,
OK shrd, 160436K buff
Swap:  530104K av,  OK used, 530104K free
115300K cached
PID   USER PRI       N1 SIZE RSS SHARE STAT
%CPU %MEM TIME       COMMAND
26265 smith 10          0  1092 1092 840  R
4.7  0.2  0:00    top
1 root   0          0   540  540 472  S
0.0  0.1  0:07    init
2 root   0          0     0    0   0  SW
0.0  0.0  0:00    kflushd
```

Когда выполняется программа top, вы можете нажимать клавиши, чтобы изменять ее поведение, например задавать скорость обновления (s), скрывать бездействующие процессы (i) или убивать процессы (к). Нажмите h, чтобы увидеть полный список, и q, чтобы выйти из программы.

Полезные опции

-nw Выполнить ^обновлений, а затем завершить работу
-dw Обновлять данные каждые Л'секунд
-pN -рм . . . Выводить процессы только с идентификаторами N, M,..., до 20 процессов

- c Выводить командные аргументы процессов
-Ь Выводить данные в стандартный канал вывода
неинтерактивно, без выведения экранных кодов. Команда top-b-nl> out file сохраняет мгновенное состояние в файле outfile

xload XFree86-tools
/usr/bin stdin stdout -file --opt --help --version

Выполните команду xload, чтобы посмотреть на статистику загрузки системы в графическом виде в отдельном окне. Она выводит график загрузки процессора (по оси Y) во времени (ось X).

Полезные опции

-update N	Обновлять данные каждые Л'секунд (по умолчанию 10)
-scale JV	Разделить ось Y на N секций (по умолчанию 1). Команда xload может добавлять секции по мере возрастания нагрузки; N - это минимальное число видимых в данный момент секций
-hi цвет	Использовать заданный цвет для линий, разделяющих секции
-1 abe 1 x	Вывести текст Жнад графиком (по умолчанию будет выводиться имя вашего хоста)
-nolabel	Не выводить текстовых надписей над графиком
-jumpscroll N	Когда график достигнет правой границы, прокрутить на ^пикселей влево и продолжить вывод графика (по умолчанию, прокрутка на половину ширины окна)

free procps
/usr/bin stdin stdout -file --opt --help -version

Команда free выводит статистику использования памяти в килобайтах.

$ free
total used free shared buffers cached

Mem: 523812 491944 31868 0
67856 199276
-/+ buffers/cache: 224812 299000
Swap: 530104 0 530104

Ядро Linux резервирует максимально возможный объем для кэширования, поэтому наилучшая оценка свободного объема оперативной памяти в предыдущих выходных данных — 299000.

Полезные опции

- s N Выполнять непрерывно и обновлять данные каждые /Усекунд
-b Выводить объемы памяти в байтах (-Ь) или мегабайтах (-ш) -ш
-1 Добавить строку с суммарным объемом памяти внизу
-o Не выводить строку "buffers/cache"

Управление процессами

kill Завершить процесс (или послать ему сигнал)
nice Вызвать программу с определенным приоритетом
renice Изменить приоритет процесса во время его выполнения

После того как процесс запущен, его можно остановить, перезапустить и изменить его приоритет. Мы рассмотрели некоторые из этих операций командного процессора в разделе "Управление задачами " на странице 53. Теперь мы рассмотрим операции изменения приоритета и завершения.

kill [опции] [идентификаторы/процессов] bash встроенная команда stdin stdout -file --opt --help --version

Команда kill отправляет сигнал процессу. Он может завершить процесс (действие по умолчанию), прервать его, приостановить, аварийно завершить и т. д. Вы должны быть владельцем процесса или суперпользователем, чтобы выполнять над ним операции.

$ kill 13243

Если эта команда не сработает - некоторые программы перехватывают сигнал и не реагируют на него - добавьте опцию -KILL:

$ kill -KILL 13243

которая практически гарантирует выполнение команды. Однако это некорректный выход для программы, он может оставить в системе неосвобожденные ресурсы (или привести к другим несогласо-ванностям) после ее завершения.

Если вы не знаете идентификатора процесса, попробуйте использовать команду pidof:

$ /sbin/pidof emacs

или выполните команду ps и изучите ее выходные данные.

Помимо программы /bin/kill в файловой системе, большинство командных процессоров имеют встроенные команды kill, но их синтаксис и поведение отличаются. Однако, все они поддерживают следующий синтаксис:

$ kill -N PID
$ kill -NAME PID

где РГО - идентификатор процесса, N - номер сигнала, а NAME - это название сигнала без префикса "SIG" (например, используйте -HUP для того, чтобы отправить сигнал SIGHUP). Для того чтобы посмотреть на полный список сигналов, передаваемых командой kill, выполните команду kill -1, хотя ее вывод может отличаться в зависимости от того, какую команду kill вы выполняете. Для того чтобы получить описание сигналов, выполните команду man 7 signal.

nice [-приоритет] командная строка coreutils
/bin stdin stdout -file - -opt --help –version

Вызывая программу, которая интенсивно использует системные ресурсы, вы, возможно, захотите сделать так, чтобы и другие процессы (и пользователи) имели достаточно ресурсов,

посредством задания ее приоритета. Именно для этого и предназначена команда nice. Ниже приведен пример установки приоритета 7 для сложной задачи.

$ nice -7 sort VeryLargeFile > outfile

Если вы не указываете приоритет, то будет установлен приоритет 10. Чтобы узнать, какой приоритет стандартный (т. е. приоритет, который устанавливается, если вы не выполняете команду nice), выполните команду nice без аргументов.

$ nice 0

Если вы суперпользователь, вы также можете повысить приоритет (уменьшить число):

$ nice —10

(да, здесь два минуса, что это означает: опция "минус 10"). Чтобы посмотреть на nice-значения ваших процессов, используйте команду ps и смотрите в колонку "NI".

$ ps -o pid,user,args,nice

renice приоритет [опция] PID util-linux
/usr/bin stdin stdout -file --opt --help --version

Тогда как команда nice может вызывать программы с заданным приоритетом, команда renice изменяет приоритет уже запущенного процесса.

Ниже мы увеличиваем nice-значение (уменьшаем приоритет) процесса 28734 на пять.

$ renice +5 -p 28734

Обычные пользователи могут уменьшать приоритеты (увеличивать число), а суперпользователь может и повышать приоритеты (уменьшать число). Допустимый диапазон - от -20 до +20, но избегайте использования больших отрицательных чисел, иначе вы можете повлиять на жизненно важные системные процессы.

Полезные опции

-p PID Выполнить операцию для процесса с заданным идентификатором (РЮ). Вы можете опустить опцию -p и указать только PID (renice +5 28734)

-и имя_пользователя Выполнить операцию для всех процессов заданного пользователя

Пользователи и их окружение

Logname	Вывести ваше имя пользователя
Whoami	Вывести ваше текущее, действующее имя пользователя
id	Вывести идентификатор пользователя и группы, к которым принадлежит пользователь
who	Вывести подробный список пользователей, работающих в данный момент в системе
users	Вывести краткий список пользователей, работающих в данный момент в системе
finger	Вывести информацию о пользователях
last	Определить, когда в последний раз входил в систему какой-либо пользователь
printenv	Вывести ваше окружение

Кто вы? Только система знает об этом точно. Следующий набор программ сообщит вам все о пользователях: их имена, время входа в систему и свойства их окружения.

logname coreutils
/usr/bin stdin stdout -file --opt --help -version

Команда logname выводит ваше имя пользователя. Она может показаться тривиальной, но очень полезна для использования в скриптах командного процессора.

```
$ logname
smith
```

whoami coreutils
/usr/bin stdin stdout -file --opt —help —version

Команда whoami выводит имя текущего, действующего пользователя. Оно может отличаться от имени пользователя, под которым вы входили в систему (выходные данные команды logname), если вы затем использовали команду su. Следующий пример показывает отличие команды whoami от команды logname.

```
$ logname • -
smith
$ whoami
smith
$ su Password:
#logname
smith
#whoami
root
```

id [опции] [имя_пользователя] coreutils
/usr/bin stdin stdout -file —opt --help -version

Каждый пользователь имеет уникальный численный идентификатор (UID) и уникальный численный идентификатор группы (GID), в которую он входит по умолчанию. Команда id выводит эти идентификаторы с соответствующими им именами пользователя и группы.

```
$ id
uid=500(smith) gid=500(smith)
groups=500(smith),6(disk),490(src),501(cdwrite)
```

Полезные опции

- u Вывести идентификатор действующего пользователя и завершить работу
- g Вывести идентификатор действующей группы и завершить работу
- G Вывести идентификаторы всех других групп, к которым принадлежит пользователь

- n Вывести имена (для пользователей и групп) вместо численных идентификаторов. Должна использоваться в комбинации с опциями -и, -д или -G. Например, команда id -Gn создает такие же выходные данные, что и команда groups
- г Вывести реальные значения вместо действующих. Должна использоваться в комбинации с опциями -u, -g ИЛИ -G

who [опции] [имя файла] coreutils "/usr/bin stdin stdout -file -opt -help -version

Команда who показывает всех вошедших в систему пользователей, по одному командному процессору пользователя на строку.

```
$ who
smith   :0     Sep  6   17:09
barrett pts/1  Sep  6   17:10
jones   pts/2  Sep  8   20:58
jones   pts/4  Sep  3   05:11
```

Как правило, команда who получает свои входные данные из файла /var/run/utmp. Аргумент имя_ файла может задать другой файл с данными, например, /var/log/wtmp с информацией о завершенных сессиях, или /var/log/btmp с информацией о неудачных попытках входа в систему*.

Полезные опции

-H Вывести строку заголовков
-1 Для пользователей, вошедших в систему удаленно, вывести имена удаленных хостов
-n Также вывести время простоя каждого пользователя в его терминале
-T Также указать, доступен ли для записи терминал пользователя (см. информацию о команде mesg у в разделе "Обмен мгновенными сообщениями" на странице 221). В выходных данных знак "плюс"

означает "да", "минус" - "нет", а вопросительный знак означает "неизвестно"

-m Вывести информацию только о себе, то есть, о пользователе связанном с текущим терминалом

-q Вывод только имен пользователей и количества пользователей. Выводит почти ту же информацию, что и команда users, но добавляет количество пользователей.

* Если ваша система сконфигурирована для журналирования этих завершенных сессий и неудачных попыток входа в систему.

users [имя файла] coreutils
/usr/bin stdin stdout -file -opt --help -version

Команда users выводит сжатый список пользователей, имеющих открытые рабочие сессии. Если пользователь работает в нескольких командных процессорах (терминалах), то его имя в списке появится несколько раз.

```
$ users
barrett jones  smith smith  smith
```

Аналогично команде who, команда users по умолчанию получает входные данные из файла /var/ log/utmp, но может читать их и из другого указанного файла.

finger [опции\ [пользователь[@хост\] finger
/usr/bin stdin stdout -file —opt -help -version

Команда finger выводит информацию о пользователях в сжатом формате:

```
$ finger
Login   Name  Tty      Idle Login Time
smith    Sandy Smith   :0     Sep  6 17:09
barrett Daniel Barrett  :pts/l  24  Sep 6 17:10
jones    Jill  Jones     :pts/2  Sep  8 20:58
```

или подробно:

```
$ finger   smith
```

. Login: smith Name: Sandy Smith
Directory: /home/smith Shell: /bin/bash On since Sat Sep 6 17:09 (EDT) on :0 Last login Mon Sep 8 21:07 (EDT) on pts/6 from localhost

No mail.
Project:
Enhance world peace
Plan:
Mistrust first impulses; they are
always right.

В качестве аргумента пользователя может быть имя локального или удаленного пользователя в формате пользователь@хост. Удаленные хосты отвечают на запросы команды finger, только если они сконфигурированы для этого.

Полезные опции

-1 Вывести в подробном формате
- s Вывести в сжатом формате
-p Не выводить разделы Project и Plan, которые обычно читаются из пользовательских файлов -/.projects -/.plan соответственно

last [опции] [пользователи] {терминалы] SysVinit /usr/bin stdin stdout -file --opt --help -version

Команда last выводит историю входа в систему в обратном хронологическом порядке.

$ last
barrett pts/3 localhost Mon Sep 8 21:
07 - 21:08 (00:01)
smith pts/6 :0 Mon Sep 8 20:
25 - 20:56 (00:31)
barrett pts/4 myhost Sun Sep 7 22:
19 still logged in

Вы можете указать имена пользователей или терминалы, чтобы конкретизировать вывод.

Полезные опции

-N	Вывести только последние Л'-строк выходных данных, где N- положительное целое число
-i	Вывести IP-адреса вместо имен хостов
-R	Не выводить имена хостов
-x	Также выводить информацию о выключении системы и изменениях в конфигурации загрузки (например, о смене однопользовательского режима на многопользовательский)
-f имя_файла	Читать из какого-нибудь другого файла с данными вместо /var/run/utmp, см. описание who для более подробной информации

printenv [переменные окружения} coreutils /usr/bin stdin stdout -file -opt —help -version

Команда printenv выводит все переменные окружения, известные вашему командному процес-рору, и их значения:

$ printenv HOME=/home/smith MAIL=/var/spool/mail/smith NAME=Sandy Smith SHELL=/bin/bash

или значения только заданных переменных:

$ printenv HOME SHELL /home/smith /bin/bash

Работа с учетными записями пользователей

useradd	Создать новую учетную запись
userdel	Удалить учетную запись
usermod	Изменить учетную запись
passwd	Изменить пароль
chfn	Изменить персональную информацию пользователя

chsh Изменить командный процессор пользователя

Программа установки Fedora предлагает вам создать две учетные записи, одну для суперпользователя и одну для обычного пользователя. Но вы, возможно, захотите создать и другие учетные записи.

Создание пользователей - это важная операция, к которой стоит относиться серьезно. Каждая учетная запись - это потенциальная дорога для "злоумышленника", желающего попасть в вашу систему, поэтому каждый пользователь должен иметь защищенный, сложно подбираемый пароль, который стоит регулярно менять.

useradd [опции] имя пользователя shadow-utils /usr/sbin stdin stdout -file --opt -help --version"

Команда useradd позволяет суперпользователю создавать учетную запись нового пользователя.

 # useradd smith

Ее настройки по умолчанию обычно не самые подходящие (их можно просмотреть с помощью

useradd -D), так что не забудьте указать все необходимые вам опции.

 # useradd -d /home/smith -s /bin/bash -g users smith

Полезные опции

-d директория	Сделать домашней директорией пользователя указанную директорию
-s командныйпроцессор	Задать командный процессор, который запускается при входе пользователя в систему
-u uid	Установить идентификатор пользователя в значение uid. Если вы не знаете в точности, что делаете, то опустите эту опцию и оставьте значение по умолчанию

-g группа	Установить в качестве первичной группы пользователя указанную группу, которая может задаваться либо численным идентификатором, либо именем, и которая должна существовать к этому моменту
-G грynnai, группа2,...	Сделать пользователя членом дополнительных существующих групп группа!, группа2н т. д.
-m	Скопировать все файлы из "скелетной" директории /etc/skel, в свежесозданную домашнюю директорию. Скелетная директория обычно содержит минимальные (скелетные) версии файлов инициализации, например, -/.bashprofile, для того чтобы новые пользователи могли начать работу. Если вы хотите скопировать файлы из другой директории, добавьте опцию -к (-к другая директория)

userdel [-r] имяпользователя shadow-utils /usr/sbin stdin stdout -file —opt --help --version

Команда userdel удаляет существующего пользователя.

 # userdel smith

Она не удалит файлы из домашней директории пользователя, только если вы не укажете опцию -г. Хорошо подумайте, прежде чем удалять пользователя; возможно стоить деактивировать учетную запись (с помощью команды usermod -L) вместо того, чтобы удалять ее. И убедитесь в том, что у вас есть резервные копии всех файлов пользователя, прежде чем удалять их: когда-нибудь они вам могут понадобиться.

usermod [опции] имя_пользователя shadow-utils /usr/sbin stdin stdout -file --opt --help –version

Команда usermod изменяет различные атрибуты заданной учетной записи пользователя, например домашнюю директорию.

 # usermod -d /home/another smith

Полезные опции

-d директория	Сделать указанную директорию домашней
-1 имя_пользователя	Изменить имя пользователя на имя_ пользователя. Хорошо подумайте, прежде чем делать это, на случай, если что-то в вашей системе зависит от исходного имени. И, естественно, не делайте этого с системными учетными записями (root, daemon и т. д.)!
-s командныи_процессор	Изменить командный процессор пользователя, который запускается при входе в систему на указанный командныйпроцессор
-g группа	Изменить первичную группу пользователя на указанную группу, которая может задаваться либо численным идентификатором группы, либо именем группы, и которая должна существовать к этому моменту
-G группа 1, группа2, . . .	Сделать пользователя дополнительно членом списка существующих групп: группа1, группа2,... Если пользователь до этого принадлежал к другим группам, но здесь вы их не указали, то пользователь больше не будет к ним принадлежать
-L	Блокировать учетную запись,

	чтобы пользователь не смог войти в систему
-U	Разблокировать учетную запись после того, как была использована опция -L

passwd [опции] [имя_пользователя] passwd /usr/bin stdin stdout -file --opt -help --version

Команда passwd изменяет пароль учетной записи, по умолчанию вашей:

#passwd

или пароль другого пользователя (это может сделать только суперпользователь):

#passwd smith

Команда passwd имеет опции, большинство из которых относятся к истечению срока действия пароля. Используйте их только в контексте хорошо продуманной политики безопасности.

chfn [опции] [имя пользователя] util-linux /usr/bin stdin stdout -file —opt -help -version

Команда chfn (от англ. change finger) обновляет некоторые части персональной информации, находящейся в системе: настоящее имя пользователя, домашний телефон, рабочий телефон и адрес офиса, которые выводятся командой finger. Если имя пользователя не указывается, то команда chfn изменяет информацию вашей учетной записи; если при вызове команды (суперпользователем) указывается имя пользователя, то изменяется информация об этом пользователе. При отсутствии опций команда chfn предложит вам ввести необходимую информацию.

$ chfn
Password: ******
Name [Shawn Smith]: Shawn E. Smith**

Office [100 Barton Hall]:
Office Phone [212-555-1212]: 212-555-1234
Home Phone []:

Полезные опции

- f	имя	Изменить полное имя на имя
-h	номер	Изменить номер домашнего телефона на номер
-p	номер	Изменить номер домашнего телефона на номер
-o	офис	Изменить адрес офиса на офис

chsh [опции] [имя пользователя] util-linux /usr/bin stdin stdout -file --opt -help --version

Команда chsh (от англ. change shell) устанавливает командный процессор, который вызывается, когда вы входите в систему. При вызове без имени пользователя команда chsh изменяет вашу учетную запись; если при вызове команды (суперпользователем) указывается имя пользователя, то изменяется командный процессор этого пользователя. При отсутствии опций команда chsh предложит вам ввести необходимую информацию.

$ chsh
Changing shell for smith.
Password: ******
New shell [/bin/bash]: /bin/tcsh**

Новый командный процессор должен быть в списке /etc/shells.

Полезные опции

-s командный^процессор	Задать новый командный процессор
-1	Вывести список всех допустимых командных процессоров

Получение прав суперпользователя

Обычные пользователи, в основном, могут изменять только те файлы, которыми они владеют. Один особенный пользователь, который называется суперпользователем или root, имеет полный доступ к компьютеру и может делать на нем все, что угодно. Для того чтобы стать суперпользователем, войдите в систему под своей учетной записью и наберите следующее.

$ SU -1
Password: # ********

Вам предложат ввести пароль суперпользователя (который, предполагается, вы знаете, если это ваш компьютер). Знак приглашения вашего командного процессора изменится на знак решетки (#), чтобы показать, что вы имеете права суперпользователя. Когда вы закончите выполнять команды в качестве суперпользо-вател, наберите лв или выполните команду exit, чтобы завершить сеанс суперпользователя и снова вернуться в ваш командный процессор.

Это простейший способ получить права суперпользователя в системе. Для этого существуют и другие программы, которые предлагают расширенный контроль, например программа sudo, но их описание выходит за рамки этой книги.

Если вы укажете команде su имя пользователя:

$ su -I jones Password: *****

то вы можете стать этим пользователем (при условии, что вы знаете его пароль).

Полезные опции

-1	Выполнить процедуру входа в систему. Вы почти всегда будете использовать эту опцию, поскольку в этом случае устанавливаются подходящие пути поиска для root
-m	Сохранить ваши текущие переменные окружения в новой сессии

-c команда Выполнить эту команду (в сессии другого пользователя) и выйти (выполнить команду exit). Если вам нужно сделать это много раз, почитайте man-страницу команды sudo

-s командный_процессор Запустить заданный командный процессор (например, /bin/bash)

Работа с группами

groups	Вывести группы, к которым принадлежит пользователь
groupadd	Создать новую группу
groupdel	Удалить группу
groupmod	Изменить группу

Группа - это набор пользовательских учетных записей, который рассматривается как одна категория. Если вы предоставите группе право на осуществление какого-либо действия (например изменение файла), то все члены этой группы получат это право. Например, вы можете предоставить группе friends полные права на чтение, запись и исполнение файла /tmp/sample.

```
$ groups
users smith friends
$ chgrp friends /tmp/sample
$ chmod 770 /tmp/sample
$ Is -1 /tmp/sample
-rwxrwx 1 smith friends 2874 Oct
20 22:35 /tmp/sample
```

Чтобы добавить пользователей в группу, редактируйте файл /etc/group*, будучи суперпользователем. Чтобы изменить группу, которой принадлежит файл, вызовите команду chgrp, описанную в разделе "Свойства файлов" на странице 95.

```
groups \имена_пользовзтелей\ coreutils
/usr/bin stdin stdout -file --opt --help -version
```

Команда groups выводит Linux-группы, к которым принадлежите вы или к которым принадлежат другие пользователи.

$ whoami
smith
$ groups
smith users
$ groups jones root
jones .- jones users
root : root bin daemon sys adm disk wheel src

* Разные системы могут хранить списки членов групп по-разному.

groupadd [опции] группа . shadow-utils
/usr/sbin stdin stdout -file —opt --help -version

Команда groupadd создает новую группу. В большинстве случаев следует использовать опцию - f для того, чтобы предотвратить дублирование групп.

#groupadd -f friends

Полезные опции

-g gid Задать ваш собственный числовой идентификатор группы вместо того, чтобы команда groupadd выбирала его сама

- f Если группа уже существует, сообщить об этом и завершить работу

groupdel группа shadow-utils
/usr/sbin stdin stdout -file --opt --help --version

Команда groupdel удаляет существующую группу.

#groupdel friends

Перед тем как сделать это, стоит определить все файлы, у которых идентификатор группы установлен на заданную группу, чтобы вы смогли обработать их в дальнейшем:

#find / -group friends -print

так как команда groupdel не меняет идентификаторов группы каких бы то ни было файлов. Она просто удаляет имя группы из системных записей.

groupmod [опции] группа shadow-utils
/usr/sbin stdin stdout -file --opt --help --version

Команда groupmod изменяет заданную группу: ее имя или идентификатор группы.

#groupmod -n newname friends

Команда groupmod не изменяет файлы, которые относятся к этой группе: она просто изменяет идентификатор группы или ее имя в системных записях. Будьте аккуратны при изменении идентификатора, иначе он будет указывать на несуществующую группу.

Полезные опции

- g gid	Изменить идентификатор группы на номер gid
-n имя	Изменить имя группы на имя

Основная информация о хосте

uname	Вывести основную информацию о системе
hostname	Вывести имя хоста
dnsdomainname	То же самое, что и команда hostname -d
domainname	То же самое, что и команда hostname -y
nisdomainname	То же самое, что и команда hostname -y
ypdomainname	То же самое, что и команда hostname -y
ifconfig	Установить и вывести информацию о сетевом интерфейсе

Каждая Linux-машина (или хост) имеет имя, сетевой IP-адрес и другие свойства. Ниже описывается, как можно просмотреть эту информацию.

uname [опции] coreutils

/bin stdin stdout -file --opt -help -version

Команда uname выводит основную информацию о вашем компьютере:

```
$ uname -a
Linux server.example.com 2.4.18-27.8.0 #1 Fri Mar
14 06:45:49 EST 2003 1686 i686 i386 GNU/Linux
```

Выходные данные включают в себя название ядра (Linux), имя хоста (server.example.com), версию ядра (2.4.18-27.8.0 #1 Fri Mar 14 06:45:49 EST 2003), название аппаратной платформы (1686), тип процессора (i686), аппаратную платформу (i386) и название операционной системы (GNU/Linux).

Полезные опции

-a	Вся информация
- s	Только название ядра (по умолчанию)
-п	Только имя хоста
- г	Только версию ядра
-т	Только название аппаратной платформы
-р	Только тип процессора
- i	Только аппаратную платформу
- о	Только название операционной системы

hostname [опции] [имя] net-tools
/bin stdin stdout -file --opt --help --version

Команда hostname выводит имя вашего компьютера. В зависимости от ваших настроек это может быть полностью определенное имя хоста:

```
$ hostname
myhost.example.com
```

или короткое имя вашего хоста:

$ hostname
myhost

Также вы можете задать имя вашего хоста, работая в режиме суперпользователя.

Однако имена хостов и имена серверов - это сложные темы, рассмотрение которых выходит за рамки этой книги. И не нужно сразу бросаться изменять имена хостов!

Полезные опции

- i Вывести IP-адрес вашего хоста
-a Вывести псевдоним вашего хоста
-s Вывести короткое имя вашего хоста
- f Вывести полностью определенное имя вашего хоста
-d Вывести DNS-домен вашего хоста
-y Вывести имя NIS- или YP-домена вашего хоста
-F файл Установить имя вашего хоста, считав имя из файла файла
ifconfig интерфейс net-tools
/sbin stdin stdout -file —opt -help —version

Команда i f conf ig выводит и устанавливает различные атрибуты сетевого интерфейса вашего компьютера. Эта тема выходит за рамки данной книги, но мы покажем вам несколько приемов.

Для того чтобы вышести информацию о стандартном сетевом интерфейсе (который обычно называется ethO), выполните следующую команду.

$ ifconfig ethO
ethO Link encap:Ethernet HWaddr 00:50:BA:48:4F:BA
inet addr:192.168.0.10 Bcast:192.168.0.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:1955231 errors:0 droppediO overruns:0 frame: 0
TX packets:1314765 errors:0 dropped:0 overruns:0

carrier:O
collisions:0 txqueuelen:10 0
RX bytes:2320504831 (2213.0 Mb)
TX bytes: 152785756 (145.7 Mb)
Interrupt:11 Base address:0x6000

Выходные данные включают в себя ваш MAC-адрес (00:50:BA:48:4F:BA), ваш IP-адрес (192.168.0. 10), вашу сетевую маску (255.255.255.0) и различную другую информацию. Для того чтобы вывести список всех существующих интерфейсов, выполните следующую команду.

$ ifconfig -a

Если у вас есть опыт работы с сетевыми службами, изучите man-страницу команды ifconfig для того, чтобы узнать детали.

Поиск хоста

host	Искать имена хостов, IP-адреса и информацию DNS
whois	Искать владельцев Интернет-доменов
ping	Проверить, доступен ли удаленный хост
traceroute	Вывести сетевой путь к удаленному хосту

При работе с удаленными компьютерами вы, возможно, захотите узнать о них больше. Кто является их владельцем? Какие у них IP-адреса? Где они расположены в сети?

host [опции] имя [сервер] bind-utils
/usr/bin stdin stdout -file --opt --help -version

Команда host осуществляет поиск имени хоста
или IP-адреса удаленной машины, запрашивая DNS.

$ host www.redhat.com
www.redhat.com has address 66.187.232.50
$ host 66.187.232.50
50.232.187.66.in-addr.arpa domain name
pointer www.redhat.com.

Она может узнать намного больше.

$ host -a www.redhat.com Trying "www.redhat.com"
->>HEADER«- opcode: QUERY, status: NOERROR, id: 50419
flags: gr rd ra,- QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3
; QUESTION SECTION:
www.redhat.com. IN ANY
; ANSWER SECTION: www.redhat.com. 196 IN A 66.187.232.50
AUTHORITY SECTION: redhat.com. 90535 IN NS ns2.redhat.com. redhat.com. 9053 5 IN NS ns3.redhat.com. redhat.com. 9053 5 IN NS nsl.redhat.com.
;; ADDITIONAL SECTION:
ns2.redhat.com. 143358 IN A 66.187.224.210 ns3.redhat.com. 143358 IN A 66.187.229.10 nsl.redhat.com. 143358 IN A 66.187.233.210

Полное обсуждение серверов имен выходит за рамки данной книги. Последний, необязательный, параметр "сервер" позволяет задавать конкретный

DNS-сервер для запроса. Вот пример запроса сервера comcast.net.
$ host www.redhat.com ns01.jdc01.pa.comcast.net
Using domain server:
Name: ns01.jdc01.pa.comcast.net
Address: 66.45.25.71#53
Aliases:
www.redhat.com has address 66.187.232.50
Чтобы вывести список всех опций, наберите просто команду host.

Полезные опции

- a Вывести всю доступную информацию
-1 Выбрать тип запроса: A, AXFR, CNAME, HINFO, KEY,

MX, NS, PTR, SIG, SOA и тому подобные.

$ host -t MX redhat.com
redhat.com mail is handled by 20 mx2.redhat.com.
redhat.com mail is handled by 10 mxl.redhat.com.

Если команда host не делает того, что вы хотите, то попробуйте использовать команду dig, еще одну мощную утилиту DNS-поиска. Также вы можете встретить команду nslookup, которая устарела в наши дни, но ее все еще можно найти в некоторых Linux- и Unix-системах.

whois [опции] имя_домена jwhois
/usr/bin stdin stdout -file -opt -help --version

Команда whois осуществляет поиск регистрационной информации Интернет-домена:

$ whois redhat.com
Registrant:
Red Hat, Inc. (REDHAT-DOM)

P.O. Box 13588
Research Triangle Park, NC 27709

Возможно, вы увидите несколько страниц текста отказа регистратора от обязательств до или после того, как появится нужная информация.

Полезные опции

-h регистратор Выполнить поиск на сервере заданного регистратора. Например,
whois -h whois.networksolutions. com yahoo.com.

-p порт Запросить заданный TCP-порт вместо стандартного порта43 (служба whois)

ping [опции\ хост iputils
/bin stdin stdout -file -opt --help --version

Команда ping сообщает, доступен ли удаленный хост. Она посылает маленькие пакеты (ICMP-пакеты, если быть точным) удаленному хосту и ждет ответов.

$ ping google.com
PING google.com (216.239.37.100) from
192.168.0.10 : 56(84) bytes of data.
64 bytes fromwww.google.com (216.239.37.100)
: icmp_seq=O ttl=49 time=32.390 msec
64 bytes fromwww.google.com (216.239.37.100)
: icmp_seq=l ttl=49 time=24.208 msec
AC
google.com ping statistics
2 packets transmitted, 2 packets received, 0% packet loss round-trip min/avg/max/mdev = 24.208/28.299/32.390/4.091 ms

Полезные опции

-c N	Отправить пакеты максимум N раз
-± N	Ждать Жсекунд (по умолчанию 1 секунду) между отправками
-n	Выводить IP-адреса, а не имена хостов

traceroute [опции] хост[длина_пакета] traceroute /bin stdin stdout -file --opt -help —version

Команда traceroute выводит сетевой путь от вашего локального хоста до удаленного хоста и время, которое требуется пакетам для того, чтобы пройти этот путь.

$ traceroute yahoo.com
1server.example.com (192.168.0.20)
1.397 ms 1.973 ms
2.817 ms
210.221.16.1 (10.221.16.1)
15.397 ms 15.973 ms
10.817 ms
3gbr2-plO.cblma.ip.att.net
(12.123.40.190) 11.952ms

11.720 ms 11.705 ms
16 p6.www.dcn.yahoo.com
(216.109.118.69) 24.757ms 22.659 ms *

.Каждому хосту в пути посылается три пробных пакета и сообщается время, за которое они возвращаются. Если хост не ответит в течение пяти секунд, traceroute выводит звездочку. Также программа traceroute может блокироваться брандмауэрами и может не работать по другим причинам, в случае чего она выводит символ.

Символ	**Значение**
!F	Необходима фрагментация
!H	Хост недоступен
!N	Сеть недоступна
!P	Протокол недоступен
!S	Маршрутизация от источника невозможна
!X	Связь административно запрещена
!N	ICMP недоступность, код N

Стандартный размер пакета 40 байт, но вы можете изменить его с помощью последнего необязательного параметра длина_пакета (например, traceroute myhost 120).

Полезные опции

-n	Численный режим: выводить IP-адреса вместо имен хостов
-w N	Изменить тайм-аут с пяти секунд на /Усекунд

Сетевые соединения

ssh	Безопасно зайти на удаленный хост или выполнять команды на нем
telnet	Зайти на удаленный хост (небезопасно!)
scp	Безопасно копировать файлы на/с удаленного хоста (пакетный режим)
sftp	Безопасно копировать файлы на/с удаленного хоста (интерактивно)
ftp	Копировать файлы на/с удаленного хоста

(интерактивно, небезопасно!)

Работая в Linux, легко создавать сетевые соединения между двумя машинами для удаленной работы или передачи файлов. Только не забудьте убедиться в том, что вы делаете это безопасно.

ssh [опции] хост[команда] openssh-clients /usr/bin stdin stdout -file --opt --help --version

Программа ssh (от англ. secure shell) безопасно соединяет вас с удаленным компьютером, на котором у вас уже есть учетная запись.

$ ssh remote.example.com

Либо она может запускать программу на этом удаленном компьютере без необходимости входа в систему на нем.

$ ssh remote.example.com who

Программа ssh шифрует все данные, которые передаются по соединению, включая ваше имя пользователя и пароль (которые вам понадобятся для доступа к удаленному компьютеру). Также протокол SSH поддерживает другие способы аутентификации, например, публичные ключи и идентификаторы пользователей. Для более подробной информации обратитесь к man-странице sshd.

Полезные опции

-l имя_пользователя	Задать ваше имя пользователя на удаленном компьютере; иначе ssh использует ваше локальное имя пользователя. Также вы можете использовать синтаксис имя пользователя@хост. **$ ssh smith@server.example.com**
-p порт	Использовать другой доргвместо стандартного (22)
-t	Выделить терминал на удаленной

системе; это полезно, если вы пытаетесь выполнять удаленную команду с интерактивным пользовательским интерфейсом, например, текстовый редактор

-v Выдавать подробную информацию, полезно для отладки

telnet [опции] хосг[порт\ telnet /usr/bin stdin stdout -file -opt --help -version

Программа telnet соединяет вас с удаленным компьютером, когда у вас уже есть на нем учетная запись.

$ telnet remote.example.com

Избегайте использования программы telnet для удаленного входа в систему: большинство реализаций небезопасны и посылают ваш пароль по сети незашифрованным, так что кто-нибудь может его узнать. Вместо этого используйте программу ssh, которая защищает ваш пароль и данные с помощью шифрования. Есть два исключения.

• В среде Kerberos (использование усовершенство ванного программного обеспечения Telnet как на стороне сервера, так и на стороне клиента). Fedora telnet умеет работать с Kerberos. Зайдите на сайт http://web.mit.edu/kerberos/ для получения более подробной информации.

• Подключение к удаленному порту, когда вы не посылаете никакой критической информации. Например, чтобы проверить наличие веб-сервера (порт 80) на удаленной системе.

$ telnet remote.example.com 80

Trying 192.168.55.21... Connected to remote.example.com (192. 168.55.21).

Escape character is 1A] ' . xxx Введите что-нибудь и нажмите клавишу Enter <HTML><HEAD> # Yep, it's a web server Действительно, это веб-сервер <TITLE>400 Bad Request</TITLE> </HEAD><BODY> <Hl>Bad Request</Hl> Your

browser sent a request that this server could not understand.<P> </BODYx/HTML> Connection closed by foreign host.

Чтобы не искушать вас использовать в дальнейшем программу telnet, мы даже не будем рассматривать ее опции.

scp ИСТОЧНИК назначение openssh-clients /usr/bin stdin stdout -file -opt —help --version

Команда scp (от англ. secure copy) безопасно копирует файлы и директории с одного компьютера (из источника) на другой (в назначение)одной операцией (если вам нужен интерактивный пользовательский интерфейс, обратитесь к команде sf tp). Она шифрует все данные, передаваемые по соединению между двумя компьютерами.

$scp myfile remote.example.com:newfile
$scp -r mydir remote.example.com:
$scp remote.example.com:myfile .
$scp -r remote.example.com:mydir .
204 | Linux. Карманный справочник

Чтобы задать альтернативное имя пользователя для удаленного компьютера, используйте синтаксис имя_пользователя@хост.

$ scp myfile smith@remote.example.com:

Полезные опции

-p Дублировать все атрибуты файла (права, временные метки) при копировании
- r Рекурсивно копировать директории и их содержимое
-v Выдавать подробную информацию, полезно для отладки

sftp (XOCT | tfM*_f70#b30BareMfi@xoc7)openssh-cr\ents /usr/bin stdin stdout -file -opt --help -version

Программа sftp интерактивно копирует файлы между двумя компьютерами (в отличие от команды scp, которая копирует файлы группой). Пользовательский интерфейс почти такой же, как и у команды ftp.

$ sftp remote.example.com
Password: ******
sftp> cd MyFiles
sftp> Is
README
filel
file2
file3
sftp> get file2
Fetching /home/smith/MyFiles/file2 to
file2
sftp> quit**

Если ваше имя пользователя в удаленной системе отлично от вашего имени пользователя в локальной системе, используйте аргумент имя_пользователя@хост:

$ sftp smith@remote.example.com

Команда	**Функция**
help	Вывести список всех доступных команд
ls	Вывести список файлов в текущей удаленной (Is) или локальной (11s) директории
pwd lpwd	вывести имя удаленной (pwd) или локальной (lpwd) рабочей директории
cd dir lcd dir	Сменить вашу удаленную (cd) или локальную (lcd) рабочую директорию на директорию dir
get файл1 [файл2]	Копировать удаленный файл/на локальной компьютер, опционально переименовав его в файл2
put файл1 [файл2]	Копировать локальный файл1 на удаленный компьютер опционально переименовав его в файл2

mget file*	Копировать несколько удаленных файлов на локальный компьютер, используя групповые символы * и ?
mput file*	Копировать несколько локальных файлов на удаленный компьютер, используя групповые символы * и ?
quit	Выйти из программы sftp

ftp [опции] хост

/usr/bin stdin stdout -file - -opt - - help - -version

Программа ftp (от англ. file transfer protocol) копирует файлы между компьютерами, но не безопасно: ваши имя пользователя и пароль передаются по сети незашифрованными. Вместо ftp, по возможности, используйте программу sftp.

Команды, которые мы перечислили для программы s f tp, также работают и для ftp (но кроме этого, эти две программы поддерживают и различные команды).

Электронная почта

evolution Графический почтовый клиент

mutt Консольный почтовый клиент

mail Консольный почтовый клиент с минимальным набором функций

Дистрибутив Fedora включает в себя несколько программ для просмотра электронной почты. Мы рассмотрим три их них с различными функциями и возможностями. Среди других почтовых программ - приложения pine, RMAIL и vm, встроенные в emacs, и mozilla Mail & News.

Чтобы посмотреть статистику сообщений электронной почты, которые вы отправляете и получаете, просмотрите журнальный файл /var/log/maillog. В режиме суперпользователя вы можете использовать команду mailq для того, чтобы просмотреть все

исходящие сообщения, поставленные в очередь на отправку на вашем компьютере.

evolution evolution

/usr/bin stdin stdout - file -- opt —help —version

Ximian Evolution - это графическая почтовая программа, которая очень похожа на программу Microsoft Outlook. В зависимости от настроек вашей системы вы можете вызвать Evolution из Главного меню в пункте Internet: Evolution Email, или выполнить команду evolution в командном процессоре.

Электронная почта | 207

Чтобы создать почтовую учетную запись, выполните следующие действия.

1. Выберите Tools > Settings... (Сервис > Изменить настройки)
2. В окне Evolution Settings (Параметры Evolution), если у вас еще нет учетной записи, выберите Add (Добавить). В противном случае выберите учет ную запись и выберите Edit (Правка).

3.В окне Evolution Account Editor (Редактор учетных записей Evolution), во вкладке Identity (Идентификация), введите ваше полное имя и адрес электронной почты.

4. Выберите вкладку Receiving Mail (Получение почты) и укажите тип вашего почтового сервера (IMAP, POP, локальная доставка и т. д.) и введите необходимую информацию. Для POP- или IMAP-серверов введите имя хоста и имя пользователя, ко торые вам должен сообщить ваш Интернет-провай дер; в случае локальной доставки введите путь к ва шему локальному почтовому ящику.
5. Выберите вкладку Sending Mail (Отправка почты) и укажите тип сервера для исходящей почты: SMTP в случае, если сервер удаленный (вам будет предло жено ввести имя хоста), или sendmail, если сервер - это локальный компьютер.
6. Редактирование остальных вкладок и опций - на ваше усмотрение. Выберите ОК, чтобы выйти из окна Evolution Account Editor. Теперь вы можете

выполнять основные операции с электронной почтой.

Inbox Просмотреть вашу электронную почту

New Написать новое почтовое сообщение

Send/Receive Проверить почту

Reply Ответить на сообщение

Reply To All Ответить на сообщение всем адресатам в строках To и CC

Forward Переслать сообщение третьей стороне

И есть еще много функций - экспериментируйте!

mutt [опции] mutt

/usr/bin stdin stdout - file -- opt --help --version

mutt - это консольная почтовая программа, которая работает в обычном терминале (или терминальном окне), поэтому его можно использовать как локально (например, в окне xterm), так и удаленно по SSH-соединению. Эта программа очень мощная, с множеством команд и опций. Чтобы вызвать ее, наберите следующую команду.

$ mutt

Когда появится главное окно, будет выведен краткий список сообщений в вашем почтовом ящике, по одному на строку. Доступные команды перечислены в табл. 4.

Таблица 4. Команды mutt, доступные в главном окне

Стрелка вниз Перейти к следующему сообщению

PageUp Переместиться вверх на один экран по сообщениям

PageDown Переместиться вниз на один экран по сообщениям

Home Перейти к первому сообщению

End Перейти к последнему сообщению

Таблица 4. Команды mutt, доступные в главном окне (Продолжение)

Стрелка вниз Перейти к следующему сообщению

m Написать новое почтовое сообщение. Будет открыт ваш

стандартный текстовый редактор. После редактирования сообщения и выхода из редактора нажмите ""y", чтобы отправить сообщение, или "q", чтобы отложить его отправку

г Ответить на текущее сообщение. Работает так же, как

и клавиша m

f Переслать текущее сообщение третьей стороне.

Работает так же, как и клавиша m

i Просмотреть содержимое вашего почтового ящика

C Скопировать текущее сообщение в другой почтовый ящик

d Удалить текущее сообщение

В процессе создания сообщения, после выхода из текстового редактора, доступны команды из табл. 5.

Таблица 5. Команды mutt, доступные при создании сообщения

Клавишная комбинация Функция

a Прикрепить файл (вложение) к сообщению

c Задать список CC

b Задать список BCC

e Снова редактировать сообщение

г Редактировать поле Reply-To (адресат ответа)

S Редактировать поле Subject (Тема)

У Отправить сообщение

C Копировать сообщение в файл

q Отложить сообщение, не отправляя его

Команды, перечисленные в табл. 6, доступны всегда. Таблица 6. Остальные команды mutt

Клавишная комбинация Функция

? Просмотреть список всех команд (нажмите пробел для перемещения вниз, "q" - чтобы выйти)

ЛС Отменить выполняемую команду

q Выйти из программы

Официальный сайт программы mutt - http://www. mutt.org, существует краткое руководство по mutt по адресу http://www. cs. utk. edu/~help/mail/mutt_starting.php.

mail [опции] получатель MailX
/bin stdin stdout - file -- opt --help --version
Программа mail (эквивалентно Mail)* - это быстрый, простой почтовый клиент. Большинство людей предпочитают для регулярного использования более мощные программы, но для быстрого обмена сообщениями через командную строку или скрипт действительно удобной оказьшается программа mail. Чтобы отправить короткое сообщение, введите следующие команды.

$ mail smith@example.com
Subject: my subject
I'm typing a message.
To end it, I type a period by itself on a line.
Cc: jones@example.com
$

* В более старых Unix-системах mail и Mail были разными программами, но в Linux это одно и то же: /usr/bin/Mail- это символьная ссылка на /bin/mail.

Чтобы отправить короткое сообщение одной командой, наберите следующее.

$ echo "Hello world" | mail -s "subject" smith@example.com

Чтобы отправить файл одной командой, используйте любую из следующих команд.

$ mail -s "my subject" smith@example. com < filename
$ cat filename | mail -s "my subject" smith@example.com

Обратите внимание на то, как легко вы можете отправлять выходные данные конвейера в виде почтовых сообщений.

Полезные опции

-s subject Установить поле SubpctfteMa) исходящего сообщения

-v Подробный режим: выводить сообщения о доставке почты

-c addresses Отправить копии сообщения заданным адресатам (CC), адреса в списке addresses разделяются запятыми

-b addresses Отправить слепые копии сообщения заданным адресатам (BCC), адреса в списке addresses разделяются запятыми

Просмотр веб-страниц

mozilla Полнофункциональный веб-браузер

lynx Консольный веб-браузер

wget Скачать веб-страницы на диск .

curl Скачать веб-страницы на диск

Linux предлагает несколько способов работы с www: традиционные браузеры, консольные браузеры и утилиты для скачивания веб-страниц.

mozilla [опции][1/ЯЦ mozilla

/usr/bin stdin stdout -file --opt --help -version

Mozilla - это один из наиболее популярных веб-браузеров для Linux; он работает и на множестве других операционных систем. Запустите его в фоновом режиме с помощью следующей команды.

$ mozilla &

Mozilla предоставляет стандартные функции (просмотр страниц, кнопки перемещения назад и вперед, закладки, историю и так далее), а также просмотр во вкладках, подавление всплывающих окон и многое-многое другое. Также она имеет полнофункциональную почтовую программу и программу для чтения новостей Usenet. Для начала в меню Help (Справка) выберите Help Contents (Содержание), а для полной информации зайдите на сайт http://www.mozilla.org.

Среди других веб-браузеров для Linux - Firebird (упрощенная Mozilla, http://www.mozilla.org/products/fiire-bird), Netscape (основан на том же коде, что и Mozilla, http://www.netscape.com), Opera (http://www.opera.com), Konqueror для KDE (http://www.konqueror.org), Epiphany для GNOME (http://www.gnome.org) и Galeon (также основан на Mozilla, http://galeon.sourceforge.net).

lynx [опции] [URL] lynx

/usr/bin stdin stdout - file - opt --help --version

lynx - это консольный веб-браузер: раритетный в наши дни, но все еще полезный, когда графика не имеет значения, или при работе с медленными сетевыми соединениями.

$ lynx http://www.yahoo.com

Все управление осуществляется с помощью клавиатуры, а не мыши. Многие страницы не будут отображаться правильно, особенно если они содержат много таблиц или рамок, но обычно можно найти способ работы с такими страницами.

Клавишная комбинация	**Функция**
?	Вывести справку
k	Вывести список всех клавишных комбинаций и их функций
^G	Отменить выполнение команды
q	Выйти из lynx
Enter	"Кликнуть" текущую ссылку или завершить ввод в поле
Стрелка влево	Обратно к предыдущей странице
Стрелка вправо	Вперед к следующей странице, или "кликнуть" текущую ссылку
g	Зайти на URL-адрес (вам будет предложено ввести его)
p	Сохранить, напечатать или отправить по почте текущую страницу
Пробел	Прокрутить вниз
b	Прокрутить вверх
Стрелка вниз	Перейти к следующей ссылке или полю для заполнения
Стрелка вверх	Перейти к предыдущей ссылке или полю для заполнения
^A	Перейти в начало страницы
^E	Перейти в конец страницы
m	Вернуться к главной/домашней (main/home) странице
/	Искать текст на странице
a	Сделать текущую страницу закладкой
v	Просмотреть список ваших закладок
г	Удалить закладку
=	Вывести свойства текущей страницы и ссылки

\ Просмотреть исходный HTML-текст (нажмите еще раз, чтобы вернуться к нормальному виду)

lynx имеет более 100 командных опций, так что имеет смысл изучить его man-страницу.

Полезные опции

- dump Вывести отображенную страницу в стандартный поток вывода и выйти из программы (сравните с опцией -source)

-source Вывести исходный HTML-текст в стандартный поток вывода и выйти из программы (сравните с командами wget и curl)

-emacskeys Сделать так, чтобы lynx понимал клавишные комбинации редактора emacs

-vikeys Сделать так, чтобы lynx понимал клавишные комбинации редактора vim (или vi)

-homepage=URL Сделать вашей домашней страницей адрес URL

-color Включить и отключить цветной режим

-nocolor

wget [опции] URL wget

/usr/bin stdin stdout -file -opt --help --version

Команда wget идет по URL-адресу и скачивает информацию в файл или стандартный поток вывода. Она великолепно подходит для скачивания отдельных страниц или всей иерархии веб-страниц до заданной вложенности. Например, давайте скачаем главную страницу Yahoo:

$ wget http://www.yahoo.com --23:19:51-- http://www.yahoo.com/

=> чindex.html'

Resolving www.yahoo.com... done. Connecting to www.yahoo.com[216.109.118.66]: 80... connected.

HTTP request sent, awaiting response... 200 OK Length: unspecified [text/html]

[<=>

] 31,434 220.84K/S

23:19:51 (220.84 KB/s) - чindex.html' saved [31434]

которая будет сохранена в файл index.html в текущей директории, wget имеет дополнительную возможность возобновлять скачивание, если оно прервалось, скажем, из-за сетевой ошибки: просто выполните wget -cc тем же URL-адресом, и она возобновит скачивание с того момента, на котором остановилась.

Еще одна похожая команда - это curl, которая пишет в стандартный поток вывода по умолчанию -

в отличие от wget, которая по умолчанию дублирует исходные имена файлов страниц.

$ curl http://www.yahoo.com > mypage.html

Команда wget имеет более 70 опций, поэтому мы рассмотрим только самые важные из них (команда curl имеет другой набор опций; обратитесь к ее man-странице).

Полезные опции

-i filename Читает URL-адреса из заданного файла filename и скачивает их

-O filename Записать все скачанные HTML-страницы в заданный файл filename, объединив страницы

-c Режим продолжения: если предыдущий сеанс скачивания прервался, и была скачана только часть файла, возобновить скачивание с места разрыва. То есть, если wge t скачал 10ОК файла размером 150К, то при указании опции -с будут скачаны и добавлены к существующему файлу только оставшиеся 50К. Однако wget может быть обманут, если удаленный файл изменился с момента первого (частичного) скачивания, поэтому используйте эту опцию, только если вы знаете, что удаленный файл не изменился

-t N Попробовать /Nраз, прежде чем отменить операцию. N=0 означает пытаться бесконечно долго

--progress=dot Выводить точки [dot) или
--progress=bar прямоугольники {bar) для отображения прогресса скачивания.

--spider Не скачивать, только проверить существование удаленных страниц

-nd Скачать все файлы в текущую директорию, даже если удаленно они находятся в более сложном дереве директорий (по умолчанию wget дублирует удаленную иерархию директорий)

- r Скачать иерархию страниц рекурсивно: все директории и поддиректории

-1 N Скачать файлы максимум до /N-го уровня вложенности (по умолчанию 5)

-к Во всех скачанных файлах изменить ссылки так, чтобы их можно было просматривать локально

-P Скачать все необходимые файлы для полного отображения страницы, например,таблицы стилей и изображения

-L Следовать по относительным ссылкам (в рамках страницы) и не следовать по абсолютным

-A Включение файлов: скачать только те файлы, имена patternl,pattern2,
pattern3, .. .
которых соответствуют заданным шаблонам patternN. Шаблоны могут содержать те же групповые символы, которые используются в командном процессоре

-R Исключение файлов: скачать только те файлы, имена которых не соответствуют заданным шаблонам patternN

patternl,patternZ, pattern3,...

-I Включение директорий: скачать файлы только из тех директорий, которые соответствуют заданным шаблонам patternN

patternl,pattern2,
patterns, .. .

-X Исключение директорий: скачать файлы только из тех директорий, которые не соответствуют заданным шаблонам

patternl,pattern2,
patterni, . . .

Новости Usenet

Сеть Usenet News - это одно из старейших сетевых сообществ на сегодняшний день. Она состоит из десятков тысяч групп новостей, дискуссионных форумов, в которых люди помещают сообщения и отвечают на них. Дистрибутив Fedora включает в себя программу для чтения новостей slrn, но существует еще много программ, доступных в сети

(rn, trn, tin, и так далее). Mozilla также умеет читать новости Usenet News: в меню Window выберите пункт Mail & Newsgroups. Также можно осуществлять поиск по Usenet News на странице Google Groups, http.V/groups.google. com.

Чтобы иметь доступ к Usenet, вам нужно подключиться к серверу новостей, Интернет-хосту, который позволяет читать и размещать новостные статьи. Когда вы подключаетесь к новостному серверу (скажем, news.example.com), автоматически в файл в вашей домашней директории сохраняется запись о группах новостей, на которые вы подписались, и о том, какие статьи вы прочитали. В зависимости от настройки вашей программы этот файл будет называться либо -/.newsrc, либо -/.jnewsrc.

slrn [опции] slrn

/usr/bin stdin stdout -file —opt —help —version

slrn - это программа для чтения новостей Usenet. Прежде чем использовать ее, вы должны задать новостной сервер, установив переменную окружения NNTPSERVER.

$ export MNTPSERVER=news.example.com Затем создайте файл новостных групп (только если ранее вы не использовали на этом компьютере программу S'lrn):

$ slrn --create и можете начать читать новости:

$ slrn

Новости Usenet I 219

При запуске программа slrn выводит страницу Групп Новостей (News Groups) со списком новостных групп, на которые вы подписались. Табл. 7 показывает некоторые полезные команды.

Таблица 7. Клавишные комбинации программы slrn для работы с новостными группами Usenet

Клавишная комбинация	Функция
Стрелка вверх	Выбрать предыдущую группу новостей
Enter	Читать выбранную группу новостей
P	Разместить новую статью в выбранной группе новостей
a	Добавить новую группу новостей (вы должны знать ее название)
u	Отписаться от выбранной группы новостей (она будет удалена после того, как вы выйдете из программы). Нажмите s, чтобы подписаться на нее снова

Когда вы нажмете Enter, чтобы прочитать группу новостей, программа slrn отобразит страницу Group (Группа), содержащую имеющиеся дискуссии (или "темы") в этой группе новостей. Табл. 8 показывает полезные команды, которые можно использовать на этой странице.

Таблица 8. Команды slrn полезные для работы с дискуссионными темами

Клавишная комбинация	Функция
Стрелка вверх	Выбрать предыдущую тему
Enter	Начать чтение выбранной темы
c	Пометить все темы как прочитанные: нажмите комбинацию ESC-u, чтобы отменить эту операцию

Таблица 9 содержит список некоторых команд, которые вы можете использовать во время чтения статьи. Таблица 9. Команды slrn, полезные при чтении статей

Клавишная комбинация	Функция
q	Завершить чтение и вернуться к странице Group
Spacebar	Перейти к следующей странице статьи
b	Вернуться к предыдущей странице статьи
r	Ответить автору по электронной почте
f	Поместить ответную статью
p	Поместить новую статью
o	Сохранить статью в файл
n	Перейти к следующей непрочитанной статье
p	Перейти к предыдущей непрочитанной статье

В любое время вы можете нажать "?", чтобы вызвать страницу со справкой. Программа slrn имеет огромное число команд и опций, и ее можно настраивать в файле -/.slmrc. Мы рассмотрели только основные ее функции; обратитесь к файлам /usr/share/doc/slrn* и сайту http://www.slrn.org для более подробной информации.

Обмен мгновенными сообщениями

gaim Клиент для обмена сообщениями и IRC-клиент
talk Двусторонний чат в Linux/Unix
write Отправить сообщение на терминал
mesg Запретить write и talk

tty Вывести имя вашего терминала

Linux предоставляет несколько способов для отправки сообщений другим пользователям, работающим на том же компьютере или где-либо в Internet - от древних программ talk и write, которые работают в Linux-терминалах (tty), до современных клиентов для обмена мгновенными сообщениями наподобие программы gaim.

gaim [опции] gaim

/usr/bin stdin stdout -file --opt —help --version

gaim - это клиент для обмена мгновенными сообщениями, который работает со многими протоколами, включая AOL, MSN, Yahoo и многие другие. Также он является IRC-клиентом. Он работает в графическом режиме.

$ gaim &

Если у вас еще нет учетной записи в одной их этих IM-служб, вам сначала нужно будет создать ее; например, зайдите на сайт http://www.aim.com, чтобы создать учетную запись AOL Instant Messenger. Когда вы это сделаете, просто нажмите кнопку Accounts (Учетные записи), чтобы сообщить о вашей записи программе gaim, введите ваше учетное имя и пароль в окне регистрации, и после этого вы можете подключиться.

Полезные опции

-u screenname Сделать вашей стандартной учетной записью имя screenname

-1 Автоматически подключаться при запуске gaim (предполагается, что ваш пароль сохранен в конфигурации)

-w [message] Перейти в режим "Отсутствую", с передачей необязательного сообщения message

talk \пользователь@хост\ [tty\ talk

/usr/bin stdin stdout -file —opt —help —version

Программа talk появилась раньше других программ для мгновенного обмена сообщениями: она связывает двух пользователей, работающих на одном и том же или разных хостах, в сеансе "один к одному". Она используется на Linux- и Unix-компьютерах (а также портирована на другие платформы) и работает

в консольном окне, например, в xterm. Она делит окно по горизонтали так, чтобы вы могли видеть то, что пишете сами, и то, что пишет ваш партнер.

$ talk friend@example.com Если ваш партнер работает в нескольких терминалах, вы можете указать один из этих терминалов для соединения.

write пользователь [tty\ Util-linux

/usr/bin stdin stdout -file --opt --help --version

Программа write более примитивна, чем talk: она отправляет строки текста от одного пользователя другому в той же Linux-системе.

$ write smith Hi, how are you? See you later.

Нажатие комбинации ЛБ завершит связь. Также программа write полезна в конвейерах для коротких одноразовых сообщений.

$ echo 'Howdy!' | write smith

mesg [y I n] SysVinit

/usr/bin stdin stdout - file - - opt - -help - -version

Программа mesg управляет тем, смогут ли talk-или write-соединения достигнуть вашего терминала. Команда mesg y разрешает их, mesg n - запрещает, а mesg - выводит текущий статус ("у" -да, или "п" - нет)*, mesg не влияет на современные программы для обмена мгновенными сообщениями, например, gaim.

$ mesg

is n

$ mesg y

tty coreutils

/usr/bin stdin stdout - file - opt --help -version

Программа tty выводит название терминального устройства (терминала), связанного с текущим командным процессором.

$ tty /dev/pts/4

Вывод на экран

echo Вывести простой текст в стандартный поток вывода

printf Вывести форматированный текст в стандартный поток вывода

yes Выводить повторяющийся текст в стандартный поток вывода

seq Вывести последовательность чисел в стандартный поток вывода

clear Очистить экран или окно

* На момент выхода книги команда mesg y в системе Fedora аварийно завершала работу с сообщением "tty device is not owned by group 'try'". Мы надеемся, что эта проблема будет решена.

Linux предоставляет несколько команд для вывода сообщений в стандартный поток вывода, на случай если вы любите разговаривать сами с собой.

$ echo hello world hello world

Каждая команда имеет разные возможности и функции. Эти команды бесценны для получения информации о Linux, отладки и написания shell-скриптов (см. раздел "Программирование скриптов командного процессора" на странице 256).

echo [опции] строки bash

встроенная команда stdin stdout - file - - opt - - help - -version

Команда echo просто выводит свои аргументы.

$ echo We are having fun We are having fun

К сожалению, есть несколько различных команд echo с немного различным поведением. Существует команда /bin/echo, но командные процессоры Linux обычно подменяют ее встроенной командой с названием echo. Чтобы узнать, какую из этих команд вы используете, наберите type echo.

Полезные опции

-n Не выводить символ новой строки в конце

-e Распознавать и интерпретировать управляющие последовательности. Например, попробуйте выполнить команды echo ' hel lo\a' и echo -e ' hello\a '. Первая выведет текст буквально, а вторая сгенерирует сигнал

-E Не интерпретировать управляющие последовательности: противоположность опции-e

Доступные управляющие последовательности перечислены ниже.

\a Предупреждение (звуковой сигнал)

\Ь Забой (Backspace)

\ с Не выводить символ новой строки (действует так же, как и опция - п)

\f Новая страница

\п Новая строка

\г Возврат каретки

\t Горизонтальная табуляция

\ v Вертикальная табуляция

\ \ Обратная косая черта

\'Одинарная кавычка

\" Двойная кавычка

\ппп Символ с ASCII-значением ппп в восьмеричной системе

printf формат[аргументь/\ bash

встроенная команда stdin stdout -file --opt --help --version
Команда printf - это команда echo с расширенной функциональностью: она выводит форматированные строки в стандартный поток вывода. Она работает почти как функция printf () языка программирования Си, которая применяет заданный формат к последовательности аргументов для того, чтобы создать определенные выходные данные.

$ printf "User %s is %d years old.\n" sandy 29 User sandy is 29 years old.

Первый аргумент - это форматирующая строка, которая в нашем примере содержит два спецификатора, %s и %d. Следующие аргументы, sandy и 29,

вставляются командой printf в строку формата, и затем она выводится. Спецификаторы позволяют работать с числами с плавающей точкой.

$ printf "ThatVll be $%0.2f, sir.Xn" 3 That'11 be $3.00, sir.

Существует две команды printf в Linux: одна встроена в командный процессор bash, другая - команда /usr/bin/printf. Они идентичны за исключением одного спецификатора, %q, который поддерживается только командой, встроенной в bash: она выводит езс-символы"\", так что ее выходные данные можно свободно

использовать в качестве входных данных командного процессора. Заметьте разницу.

$ printf "This is a single quote: %s\n" "'" This is a single quote: '

$ printf "This is a quote: %q\n'

This is a single quote: \'

Вы должны следить за тем, чтобы число форматных условий было равно числу аргументов printf. Если аргументов слишком много, то лишние игнорируются, а если мало, то printf вставит стандартные значения (0 - для чисел, "" - для строк). Тем не менее, вы должны считать такие случаи ошибками, несмотря на то что printf их прощает. Если они будут иметь место в shell-скриптах, то рано или поздно они дадут о себе знать.

Спецификаторы подробно описаны на man-странице функции printf (выполните команду man 3 printf). Вот некоторые полезные примеры.

%d Десятичное целое число
%ld Десятичное длинное целое число
%o Восьмеричное целое число
%x Шестнадцатеричное целое число
% f Число с плаваюицей точкой
%lf Число с плавающей точкой двойной точности
%c Одиночный символ
%s Строка
% q Строка с маскировкой любых метасимволов командного процессора
%% Знак процента

Сразу после первого знака процента вы можете вставить числовое выражение, задающее минимальную длину выходных данных. Например, запись "%5d" означает вывод десятичного числа в поле шириной пять символов, а .запись "%6 . 2 f" - вывод числа с плавающей точкой в поле шириной шесть символов с двумя цифрами послеразделителя. Вот некоторые полезные числовые выражения.

n Минимальная ширина п
On Минимальная ширина л, с заполнением ведущими нулями

n. m Минимальная ширина n, с т цифрами после разделителя

Также print f интерпретирует управляющие последовательности, например, "\n" (вывести символ новой строки) и "\a" (звонок). Обратитесь к разделу о команде echo с полным списком знаков перехода.

yes [строка] coreutils

/usr/bin stdin stdout - file -■ opt —help --version

Команда yes выводит заданную строку (или "y" по умолчанию) бесконечно дожо, каждый раз с новой строки.

$ yes again again again again

Хотя с первого взгляда она может показаться бесполезной, команда yes может пригодиться для превращения интерактивных команд в групповые. Хотите избавиться от надоедливого сообщения "Вы УВЕРЕНЫ, что хотите сделать это"? Передайте выходные данные команды yes на вход такой команде, чтобы автоматически ответить на все вопросы.

$ yes | my_interactive_command Когда команда ту^interactive'_command завершит работу, завершит работу и yes.

seq [опции] спецификация coreutils

/usr/bin stdin stdout - file - opt --help -version

Команда seq выводит последовательность целых или вещественных чисел, которую можно перенаправить другим командам. Есть три вида аргумента спецификация.

Одиночное число: верхний предел Вывод команды seq начинается с 1 и продолжается до заданного числа.

$ seq 3
1
2
3

Два числа: нижний и верхний пределы

Вывод команды seq начинается с первого числа и заканчивается по прохождении второго.

$ seq 5 2
5
4

3
2
Три числа: нижний предел, приращение и верхний предел

Вывод команды seq начинается с первого числа, увеличивается на второе число каждый раз и заканчивается третьим числом (или предшествующим ему).

```
$ seq 1.3 2
1
1.3
1. 6
1.9
```

Полезные опции

-w Выводить в начале нули по необходимости, чтобы у всех строк была одинаковая ширина:

```
$ seq -w 8 10
08
09
10
```

-£ формат Задать формат вывода как в printf . Строка формата должна включать %д (используется по умолчанию), %е или % f

```
$ seq -f '**%д**' 3
* * 1 * *
**2**
**3**
```

-s разделитель Использовать заданную строку в качестве разделителя чисел. По умолчанию: символ новой строки

```
$ seq -s ':' 10 1:2:3:4:5:б:7:8:9:10
```


clear ncurses

/usr/bin stdin stdout - file -- opt --help --version

Эта команда просто очищает ваш экран или окно командного процессора.

Математические операции и вычисления

xcalc Открыть графический калькулятор

expr Вычислить простое математическое выражение в командной строке

dc Консольный калькулятор

Вам нужен калькулятор? Linux предоставляет не только знакомый вам графический калькулятор, но также некоторые консольные программы для проведения математических операций.

xcalc [опции] XFree86-tools

/usr/X16R6/bin stdin stdout - file -- opt --help --version

Команда xcalc вызывает простой графический калькулятор. По умолчанию открывается традиционный калькулятор; если вы предпочитаете калькулятор с обратной польской записью (RPN), добавьте опцию -грп.

expr выражение coreutils

/usr/bin stdin stdout - file -- opt -help --version

Команда expr выполняет простые математические операции (и прочую обработку выражений) в командной строке.


```
$ expr 7+3
10
$ expr '(' 7 + 3 ')' '*' 14 Специальные символы командного процессора помещены в кавычки
140
$ expr length ABCDEFG
7
$ expr 15 '>' 16
0
```

Все аргументы должны быть разделены пробелами. Обратите внимание на то, что мы должны помещать в кавычки или маскировать любые символы, если они имеют специальное значение для командного процессора. Круглые скобки (замаскированные) могут быть использованы для группировки. Табл. 10 показывает операторы для команды expr.

Таблица 10. Операторы для команды expr

Оператор Численная операция Строчная операция

+Сложение

- Вычитание
*** Умножение**
/ Целочисленное деление
% Остаток от деления (по модулю)
<Меньше чем Ранее в словаре
<= Меньше или равно Ранее в словаре, до этого места включительно
>Больше чем Далее в словаре
Больше или равно Далее в словаре, с этого места включительно

Таблица 10. Операторы для команды expr (Продолжение)

Оператор Численная операция Строчная операция
=Равенство Равенство
! = Неравенство Неравенство
Логическое "или" Логическое "или"
& Логическое "и" Логическое "и"
s : гедехр Соответствует ли регулярное выражение гедехр строке 5?
substr s p n Вывести n символов строки s, начиная с позиции p (p=1 - это первый символ)

index s chars

Вернуть номер первой позиции в строке 5, содержащий символ из строки chars. Вернуть 0, если он не будет найден. Работает так же, как Си-функция index ()

В логических выражениях число 0 и пустая строка считаются "ложью"; все остальное - "истина". Результат логического выражения: 0 считается "ложью"; 1 - "истиной".

Команда expr не очень эффективная. Для более сложных задач подумайте об использовании такого языка как Perl.

dc [опции] [файлы]

be

/usr/bin stdin stdout - file -- opt -help --version

Команда dc (от англ. desk calculator) - это стековый калькулятор с обратной польской записью (RPN), который читает выражения из стандартного входного потока и-записывает результат в стандартный поток вывода. Если вы знаете, как использовать RPN-калькулятор компании Hewlett-Packard, то вам будет не сложно использовать команду dc, когда вы изучите ее синтаксис. Но если вы использовали традиционные калькуляторы, то dc может показаться непонятной. Мы рассмотрим только некоторые основные команды. Операции для работы со стеком и калькулятором.

qВыйти из программы dc

f Вывести весь стек

с Удалить (очистить) весь стек

рВывести самое верхнее значение в стеке

рВытолкнуть (удалить) самое верхнее значение из стека

лк Задать точность будущих операций до п цифр после запятой

(стандартно 0: целочисленные операции)

Команды, которые забирают с вершины стека два значения, проделывают на ними операцию и возвращают результат в стек.

+Сложение

Вычитание

*** Умножение**

/ Деление

% Остаток

Возведение в степень (основание - второе значение сверху, показатель степени - верхнее значение)

Команда, которая забирает с вершины стека одно значение, проделывает на ним операцию и возвращает результат в стек.

vКвадратный корень

Примеры.

$ dc

4 5 + p Вывести результат сложения 4 и 5

9

2 3 * p Возвести 2 в степень 3 и вывести результат

8
10 * p Умножить верхнее значение стека на 10
и вывести результат 80
f Вывести стек
80 9 +p Вытолкнуть из вершины стека два значения
и вывести их сумму
89 " '
Дата и время
xclock Вывести графические часы
cal Вывести календарь
date Вывести или установить дату и время
ntpdate Установить системное время, используя удаленный сервер времени

Хотите узнать дату? А точное время? Попробуйте использовать эти программы для вывода и установки даты и времени в вашей системе.

xclock [ОПЦИИ] XFree86-tools
/usrfid 1R6/bin stdin stdout - file ■- opt --help --version

Команда xclock выводит простые графические часы. Если вы предпочитаете другой стиль, есть другие программы-часы: например, oclock (круглые), t3d (из двигающихся трехмерных шариков,располагается вне ваших директорий поиска в директории /usr/Xl IR6/lib/xscreensaver/) и часы для панели задач, отображаемые в GNOME и KDE.

Полезные опции

-analog Аналоговые часы со стрелками
-digital [-brief] Цифровые часы с полной датой и временем; добавьте опцию -brief, чтобы вывести только время
-update N Обновлять время каждые /Усекунд
cal [опции] [месяц [год]] util-linux
/usr/bin stdin stdout - file -- opt --help --version
Команда cal выводит календарь, по умолчанию -текущий месяц.

```
$cal
  September 2003
Su      Mo      Tu      We      Th      Fr      Sa
 1       2       3       4       5       6
```

78 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30

Чтобы вывести другой календарь, укажите месяц и год из четырех цифр: cal 8 2002. Если вы не укажете месяц (cal 2002), то будет выведен календарь для всего года.

Полезные опции

-y Вывести календарь текущего года

-ш Календарь рабочей недели: сделать левой колонкой понедельник

- j Нумеровать каждый день по его положению в году; в нашем

примере 1 сентября был бы 244-м, второе 2 - 245-м, и т. д.

236 | Linux. Карманный справочник

I

date [опции] [формат] coreutils

/bin stdin stdout - file - opt --help -version

Команда date выводит дату и время. По умолчанию она выводит системные дату и время локального часового пояса.

$ date

Sun Sep 28 21:01:31 EDT 2003

Вы можете форматировать выходные данные, указывая форматирующие строки, начинающиеся со знака плюс.

$ date '+%D'

09/28/03

$ date '+The time is %1:%M %p on a

beautiful %A in %B'

The time is 9:01 PM on a beautiful

Sunday in September

Формат Функция Пример

Дата и время

%c Полные датам время, 12-часовой формат Sun 28 Sep 2003,09:01:25 PM EDT

%D Численная дата, 2 цифры для гада 09/28/03

%x Численная дата, 4 цифры для гада 09/28/2003

%T Время, 24-часовой формат 21:01:25
%x Время, 12-часовой формат 09:01:25 PM
Слова
%a День недели (аббревиатура) Sun
%A День недели (полностью) Sunday
%Ь Месяц (аббревиатура) Sep
%B Месяц (полностью) September
%Z Часовой пояс EDT
%P AM (до полудня) или PM (после PM полудня)

Формат Функция Пример
Числа
%w День недели (0-6, О=воскресенье) 0
%u День недели (1-7, 1=понедельник) 7
%d День месяца, дополнять нулем 02
%e День месяца, дополнять пробелом 2
%j День года, дополнять нулями 005
%m Номер месяца, дополнять нулем 09
%y Год, 2 цифры 03
%Y Год, 4 цифры 2003
%M Минута, дополнять нулем 09
%s Секунды, дополнять нулем 05
%1 Часы, 12-часовой формат, дополнять пробелом 9
%I Часы, 12-часовой формат, дополнять нулем 09
%k Часы, 24-часовой формат, дополнять пробелом 9
%H Часы, 24-часовой формат, дополнять нулем 09
%N Наносекунды 737418000
%s Секунды от точки отсчета Linux: полночь 1 января, 1970 1068583983
Другое
%n Новая строка
%t Табуляция
%% Знак процента %

С помощью своих опций также выводить даты и время команда date может в других форматах.

Полезные опции

-d время Вывести заданное время в формате по вашему

усмотрению

-г имя_файла Вывести временную метку последнего изменения заданного файла, форматированную по вашему усмотрению

- s время Установить системные дату и/или время; это может

делать только суперпользователь

a ntpdate сервер_ времени ntp

/usr/bin stdin stdout - file -- opt --help --version

Команда ntpdate устанавливает текущее системное время, обращаясь к серверу времени в сети. Вы должны работать в режиме суперпользователя, чтобы установить системное время.

 # /usr/sbin/ntpdate timeserver.someplace.edu

7 Sep 21:01:25 ntpdate[2399]: step time server 178.99.1.8 offset 0.51 sec

Чтобы постоянно поддерживать ваше системное время в синхронизации с сервером времени, используйте демон ntpd; обратитесь к сайту http://www.ntp.org. Если вы не знаете адреса ближайшего сервера времени, выполните поиск в Google по строке "public ntp time server" ("публичный ntp сервер времени").

Планирование заданий

sleep Ждать заданное число секунд, ничего не делая

watch Выполнять программу через заданные интервалы

at Назначить задание на одноразовое выполнение в будущем

crontab Назначить задание на многоразовое выполнение в будущем

Если вам нужно выполнять программы в определенные моменты или через регулярные интервалы, Linux предоставляет несколько инструментов планирования различной степени сложности.

sleep время coreutils

/bin stdin stdout - file -- opt --help --version

Команда sleep просто ожидает заданное число секунд. Время можно задавать в виде целого числа (секунд) или целого числа, за которым следует буква s (также секунды), m (минуты), h (часы) или d (дни).

$ sleep 5m Не делать ничего 5минут

Команда sleep полезна для задержки выполнения команды на заданное время.

$ sleep 10 && echo 'Прошло десять секунд.1 (прошло 10 секунд) Прошло десять секунд.

watch [ОПЦИИ] команда procps

/usr/bin stdin stdout - file -- opt —help —version

Программа watch выполняет указанную программу через регулярные интервалы времени; стандартно - каждые две секунды. Команда передается командному процессору (поэтому не забудьте поместить в кавычки или замаскировать все специальные символы), и результат отображается в полноэкранном режиме, так что вы без труда сможете следить за выходными данными и видеть, что изменяется. Например, команда watch -n 6 0 date выполняет команду date раз в минуту, что-то вроде дешевых часов. Нажмите ЛС, чтобы выйти.

Полезные опции

-n секунды Установить время между запусками, в секундах

-d Выделять различия в выходных данных, чтобы

обратить внимание на то, что изменяется между последовательными запусками

at [опции] указание времени at

/usr/bin stdin stdout - file — opt --help --version

Команда at запускает команду командного процессора в заданное время.

$ at 7am next Sunday

at> echo Не забудь сходить за покупками

| mail smith

at> lpr $HOME/shopping-list

at> ~D

<EOT>

job 559 at 2003-09-14 21:30

Форматы времени, понимаемые командой at, чрезвычайно гибки. Вообще, вы можете указать:

• время и после него дату (а не наоборот);

• только дату (берется текущее время);

• только время (считается ближайшее наступление, сегодня и завтра);

• специальное слово, например, now (сейчас), midnight (полночь) или teatime (16:00);

• любое из вышеперечисленного, после чего указыва ется смещение, например,"+ 3 days" (плюс три дня).

Даты принимаются во многих форматах: decem-

ber (декабрь) 25 2003,25 december 2003, de-cember 25, 25 december, 12/25/2003, 25.12. 2003, 20031225, today (сегодня), thursday

(четверг), next thursday (следующий чет-врег), next month (следующий месяц), next year (следующий год) и многих других. Названия месяцев могут быть сокращены до трех букв (j an -январь, f eb - февраль, mar - март,...). Форматы времени тоже различные: записи 8pm (8 часов вечера), 8pm, 8:00pm, 8:00 pm, 20:00 и 2000 эквивалентны. Смещения задаются знаками плюса или минуса, после которых идет пробел и промежуток времени: + 3 seconds (плюс три секунды), + 2 weeks (плюс две недели), - 1 hour (минус один час) и т. д*.

Если вы не указываете дату или время, команда at заменяет отсутствующие данные системными датой или временем. Поэтому "next year" (следующий год) означает один год, считая от настоящего момента, "thursday" (четверг) означает следующий ближайший четверг, "december 25" (25 декабря) - ближайший день 25 декабря, а "4:30pm" (16:30) - ближайший момент времени 16:30.

Ваша команда не будет обрабатываться командным процессором до момента выполнения, поэтому групповые символы, переменные и другие конструкции командного процессора не интерпретируются до того момента. Также ваше текущее окружение (см. print env) сохраняется в рамках каждого задания, так что это задание будет выполняться так, как будто бы вы работали в это время в системе. Однако псевдонимы недоступны команде at, поэтому не включайте их в задание.

* Программисты могут посмотреть точный синтаксис в файле **/usr/ share/doc/at- */timespec.**

Чтобы вывести список заданий at, используйте команду atq ("at queue" - очередь at).

$ atq

559 2003-09-14 07:00 a smith

Чтобы удалить задание at, выполните команду atrm ("at remove" - удалить задание at) с номером задания.

$ atrm 559

Полезные опции

- f файл Читать команды из заданного файла вместо стандартного потока ввода

-c номер Вывести команды задания с указанным номером в стандартный поток вывода

crontab [опции] [файл] vixie-cron

/usr/bin stdin stdout - file -- opt --help --version

Команда crontab, как и at, планирует выполнение задания на определенное время. Однако crontab предназначена для повторного выполнения заданий, например "Выполнять эту команду в полночь во второй четверг каждого месяца". Чтобы сделать это, надо будет отредактировать и сохранить файл (ваш файл crontab):

$ crontab -e

который затем автоматически инсталлируется в системную директорию {/var/spool/cron). Раз в минуту запускается Linux-процесс с названием сгоп, проверяет ваш файл crontab и выполняет все задания, которые нужно выполнить.

$ crontab -e

Редактировать ваш файл crontab в вашем стандартном редакторе ($EDITOR).

$ crontab -I

Вывести ваш файл crontab в стандартный поток вывода.

$ crontab -r

Удалить ваш файл crontab $ crontab myfile

Сделать файл myfile вашим файлом crontab.

Суперпользователь может добавить опцию -и username для работы с crontab-файлами других пользователей.

Файлы crontab содержат по одному заданию на строку (пустые строки и строки комментариев, начинающиеся с "#", игнорируются). Каждая строка имеет шесть полей, разделенных пробелами. Первые пять полей задают время, в которое нужно запускать задание, а последнее - саму команду задания.

Минуты часа

Целые числа от 0 до 59. Это может быть одно число (30), последовательность чисел, разделенных запятыми (0,15,30,45), диапазон (20-30), последовательность диапазонов (0-15,50-59) или звездочку, т. е. "все". Также вы можете указать "каждый n-й раз" с суффиксом /и; например, записи */12 и 0-59/12 означают выполнять на 0-й,12-й,24-й,36-й,48-й минутах (то есть, каждые 12 минут).

Часы дня

Тот же синтаксис, что и для минут.

Дни месяца

Целые числа от 1 до 31; здесь вы также можете использовать последовательности, диапазоны, последовательности диапазонов или звездочку.

Месяцы года

Целые числа от 1 до 12; здесь вы также можете использовать последовательности, диапазоны, последовательности диапазонов или звездочку. Кроме того, вы можете использовать аббревиатуры из трех букв (jan - январь, f eb - февраль, mar - март и т. д.), только не в диапазонах и не в последовательностях.

Дни недели

Целые числа от 0 (воскресенье) до 6 (суббота); здесь вы также можете использовать последовательности, диапазоны, последовательности диапазонов или звездочку. Кроме того, вы можете использовать аббревиатуры из трех букв (sun - воскресенье, топ -понедельник, tue - вторник и т. д.), только не в диапазонах или последовательностях.

Команда для выполнения

Любые команды командного процессора, которые будут выполняться в вашем рабочем окружении, так что вы можете ссылаться на переменные окружения $HOME и быть уверенными что это сработает. Общее правило - используйте только абсолютные пути к вашим командам (например, /usr/bin/who вместо who).

Таплица II показывает некоторые примеры задания времени.
Таблица 11. Примеры задания времени в файле crontab

					Каждую минуту
45			*	*	Через 45 минут после наступления каждого часа (1:45, 2:45 и гак далее)
45	9				Каждый день в 9:45 утра
45	9	8			Восьмой день каждого месяца в 9:45 утра
45	9	8	12	•	8 декабря каждого года в 9:45 утра
45	9	8	dec	*	8 декабря каждого года в 9:45 утра
45	9	*	•	6	Каждую суббогу в 9:45 утра
45	9	*	*	sat	Каждую субботу в 9:45 утра
45	9	•	12	6	Каждую субботу декабря, в 9:45 утра
45	9	8	12	6	8 декабря каждого года И в субботу, в 9:45 утра

Если команда создает какие-либо выходные данные при выполнении, то сгоп отправит их вам по почте.

Графика и хранители экрана

еод Отображать графические файлы

gqview Отображать графические файлы и слайд-шоу

ksnapshot Сделать снимок экрана

gimp Редактировать графические файлы

gnupfot Создавать графики и диаграммы

xscreensaver Запустить хранитель экрана

Для просмотра или редактирования графических файлов Linux имеет удобные инструменты с множеством опций. Мы не будем рассматривать эти программы слишком подробно, только так, чтобы вызвать

у вас интерес. Наша цель - сделать так, чтобы вы узнали об этих программах, а далее изучали и использовали их по мере необходимости.

еод [опции] [файл] еод

/usr/bin " stdin stdout - file ■- opt —help --version

Программа для просмотра изображений еод (Eye of Gnome) отображает графические файлы различных форматов. Если вы

вызываете ее для одного файла, то она выведет этот файл. Если для двух или более файлов:

$ eog filel.jpg file2.gif file3.pbm то каждый файл она отображает в отдельном окне.

Большинство опций еод довольно специализированные, поэтому мы их не рассматриваем; мы упоминаем о них для того, чтобы вы знали, что еод имеет опции, на случай если вы захотите изучить их (еод --help).

gqview [опции] [файл] gqview

/usr/bin " stdin stdout - file -- opt —help --version

Программа для просмотра изображений gqview отображает графические файлы различных форматов и может автоматически переключаться с одного изображения на следующее, как в слайд-шоу. По умолчанию, она отображает имена всех графических файлов в текущей директории, и вы можете выбирать имена файлов, которые вы хотите просмотреть. Экранное меню очевидное, так что изучите его самостоятельно. Нажмите Aq, чтобы выйти из программы.

Полезные опции

- f Отображать изображения в полноэкранном режиме (переключиться между полноэкранным и оконным режимами можно с помощью клавиши v)

- s Показывать изображения в режиме слайд-шоу (включить и отключить слайд-шоу можно с помощью клавиши s)

ksnapshot [опции] kdegraphics

/usr/bin stdin stdout - file -- opt —help —version

Команда ksnapshot - это универсальная утилита для получения снимков экрана. Просто выполните команду:

$ ksnapshot

и она сделает снимок и отобразит его уменьшенную копию. Теперь вы можете сохранить его как графический файл или сделать еще один снимок. Единственная тонкость здесь - это выбор формата файла для выходных данных; это делается во время сохранения посредством выбора имени файла с соответствующим стандартным расширением: jpg для создания JPEG-файла, .Ътр - для растрового файла Windows, .рЪт -для

переносимой растровой графики,. eps - для инкапсулированного Postscript-формата, .ico - для иконок Windows и так далее. Чтобы вывести список всех поддерживаемых форматов файлов, нажмите кнопку Save Snapshot ("Сохранить как") и просмотрите варианты выбора в Filter (Фильтр).

Для более подробной информации нажмите кнопу Help (Справка) в окне ksnapshot или выполните команду ksnapshot --help-all в командном процессоре.

gimp [опции] [файлы] gimp
/usr/bin stdin stdout - file -- opt --help -version

GIMP (GNU Image Manipulation Program) - это полнофункциональный пакет для редактирования изображений, который конкурирует с программой Adobe Photoshop по мощи и функциональности. Он довольно сложен в использовании, но результаты его использования могут быть потрясающими. Для более подробной информации посетите сайт http://www.gimp.org. Чтобы запустить программу, выполните следующую команду.

$ gimp

Чтобы редактировать конкретный файл, выполните следующую команду.

$ gimp filename

Если функциональность GIMP для вас избыточна, скачайте для более простых задач редактирования программу xv с сайта http://www.trilon.com/xv. Просто откройте графический файл:

$ xv myfile.jpg

и щелкните правой кнопкой мыши на изображении. Появится меню с инструментами редактирования.

gnuplot [опции] [файл] gnuplot
/usr/bin stdin stdout - file - - opt - - help - -version

Программа gnuplot создает графики, рисуя точки и соединяя их прямыми и кривыми, и сохраняет их в различные принтерные и плоттерные форматы, например, Postscript. Чтобы использовать gnuplot, вам нужно изучить маленький, но мощный язык программирования. Ниже показан пример построения кривой y = x2 на отрезке от x=1 до x=10, который появится в графическом окне на вашем экране.

$ gnuplot

gnuplot> plot [1:10] x2**
gnuplot> quit

Чтобы сделать то же самое и при этом сохранить результаты в Postscript-файл, выполните следующую команду.

$ echo 'set terminal postscript; plot [1:10] x2' | gnuplot > output.ps**

Посетите сайт http://www.gnuplot.info для получения более подробной информации.

xscreensaver xscreensaver
/usr/X11R6/bin stdin stdout - file -- opt --help --version

Система xscreensaver - это универсальный хранитель экрана с сотнями возможных анимационных роликов. Она работает в фоновом режиме, и вы можете управлять ей различными способами.

После периода бездействия. По умолчанию, графический пользовательский интерфейс Fedora (KDE или GNOME) запускает программу xscreensaver автоматически после пяти минут бездействия. Вы можете настроить работу этой службы в главном меню. В среде KDE запустите приложение Control Cen1er(Центр управления), затем выберите Appearance & Themes (Внешний вид и темы), затем Screen Saver (Хранитель экрана). Либо щелкните правой кнопкой мыши на рабочем столе, выберите пункт Configure Desktop (Настроить рабочий стол), затем Screen Saver (Хранитель экрана). В среде GNOME выберите в главном меню Preferences/ Screensaver (Настройки/Хранитель экрана).

В качестве блокировщика экрана. В любой момент (в GNOME или KDE) вы можете открыть главное меню и выбрать пункт Lock Screen (Заблокировать экран). Ваш экран будет заблокирован до тех пор, пока вы не введет ваш пароль.

Из командной строки. Выполните команду xscreensaver-demo, чтобы просмотреть анимационные ролики и сделать удобные для вас настройки. Затем выполните команду xscreensaver-command, чтобы задать поведение программы.

$ xscreensaver-command -activate	Активизироваться $ xscreensaver-
command -next	Выбрать следующий ролик
$ xscreensaver-command -prev	Выбрать предыдущий ролик
$ xscreensaver-command -cycle	Выбрать произвольный ролик

$ xscreensaver-command -lock	Заблокировать экран сейчас
$ xscreensaver-command -exit	Выйти

Аудио и видео

grip	Проигрыватель компакт-дисков, риппер и MP3-кодировщик
xmms	Проигрыватель аудио-файлов (MP3, WAV)
cdparanoia	Извлечь (rip) аудио-треки с музыкальных компакт-дисков в WAV-файлы
audacity	Редактировать аудио-файлы
xcdroast	Программа для записи компакт-дисков с графическим интерфейсом

Linux-системы умеют, и весьма неплохо, работать с аудио-данными. Большинство из программ, которые мы рассмотрим, имеют очевидный интерфейс пользователя, множество функций и приемлемую документацию, поэтому мы не будет рассматривать их детально. Мы, главным образом, хотим, чтобы вы узнали, какие возможности доступны. Существуют различные вебсайты посвященные аудио и MIDI в Linux, например http://linux-sound.org/ и http://www.xdt.com/ar/linux-snd.

В дистрибутив Fedora не включен ни один видеоплеер, но вы можете скачать и установить их. Некоторые популярные плееры – это mpgl23 (http:// www.mpgl23.de/), smpeg (http://www.lokigames.com/ development/), и mplayer (http://www.mplayerhq.hu).

grip grip
/usr/bin stdin stdout - file -- opt --help --version

grip - это проигрыватель музыкальных компакт-дисков и аудио-риппер: он умеет проигрывать компакт-диски, извлекать аудио-данные с компакт-дисков, сохранять их в WAV-файлы и преобразовывать эти файлы в MP3-формат. Он имеет подробную встроенную справку и довольно очевидные элементы управления.

cdparanoia[опций\ иапазон[выходной_фаЩсА^агапо\а
/usr/bin stdin stdout - file -- opt -help --version

Команда cdparanoia считывает аудио-данные с музыкального компакт-диска и сохраняет их в WAV-файлы (или другие форматы: см. man-страницу). Вот основные способы использования.

$ cdparanoia N Записать трек TV в файл.
$ cdparanoia -B
Записать все треки музыкального компакт-диска в отдельные файлы.
$ cdparanoia -B 2-4
Записать 2-й, 3-й и 4-й треки в отдельные файлы.
$ cdparanoia 2-4
Записать 2-й, 3-й и 4-й треки в один файл.

Если у вас сложности с доступом к вашему диску, попробуйте выполнить команду cdparanoia -Qvs ("искать CD-ROM-приводы с подробным отчетом") и найти проблему. Чтобы преобразовать ваши WAV-файлы в MP3-формат, обратитесь к программам LAME (http://lame.sourceforge.net/) или Not-Lame (http://www. idiap.ch/~sanders/notjame/).

xmms [опции] [файль/\ xmms
/usr/bin stdin stdout - file -- opt --help -version

xmms (X MultiMedia System) - это прекрасный графический проигрыватель аудио-файлов, который поддерживает MP3, WAV, Ogg Vorbis и другие аудио-форматы*. Вы можете проигрывать файлы с помощью кнопок управления, аналогичных кнопкам управления CD-проигрывателя, создавать списки файлов для воспроизведения и многое другое. Самый простой способ начать знакомство с ним -это запустить его, либо без аргументов:

* Дистрибутив Fedora включает в себя xmms без поддержки тр3-формата; посетите сайт http://vnvw.xmms.org/, чтобы установить этот формат или самую последнюю, неурезанную версию.

$ xmms либо указав аудио-файлы в командной строке:
$ xmms filel.mp3 file2.wav file3.ogg ... Вот некоторые полезные действия.
Действие Функция

Щелчок правой кнопкой мыши	Вывести главное меню по строке заголовка
Нажать кнопку PL	Отобразить список воспроизведения (нажмите Add, чтобы добавить файлы в него)
Нажать кнопку EQ	Открыть графический эквалайзер
Двойной щелчок левой кнопкой	Проиграть трек мыши по треку в Списке воспроизведения
Щелчок правой кнопкой мыши	Открыть меню списка воспроизведения в списке воспроизведения

audacity [файлы] audacity
/usr/bin stdin stdout - file - - opt - - help - -version

audacity - это графический редактор аудио-файлов для внесения изменений в WAV, MP3 и Ogg файлы. Когда файл загружен, вы можете просматривать форму его сигнала, вырезать и вставлять аудио-данные, накладывать фильтры и специальные эффекты на звук (эхо, усиление басов, обращение и т. д.) и многое другое. Программа audacity не включена в дистрибутив Fedora, но очень рекомендуется скачать ее с сайта http://audacity.sourceforge.net.

xcdroast [опции] xcdroast
/usr/bin stdin stdout - file -- opt --help --version

xcdroast - это программа для записи компакт-дисков с графическим пользовательским интерфейсом. Она поддерживает только SCSI-приводы. Если ваше устройство для записи компакт-дисков является IDE-приводом, вам нужно будет настроить его так, чтобы он использовал модуль эмуляции ide-scsi. Эта задача выходит далеко за рамки нашей маленькой книги, но, как правило, если ваши CD-приводы есть в выходных данных команды:

$ cdrecord -scanbus

то все должно быть нормально. В противном случае обратитесь к руководству по записи компакт-дисков "CD-Writing Howto" (http://www.tldp.org/HOW-TO/CD-Writing-HOWTO.html). Кроме того, перед тем как использовать xcdroast, зайдите на сайт http://

www.xcdroast.org и прочитайте документацию, особенно FAQ (частозадаваемые вопросы), так как настройка может оказаться нетривиальной. Затем выполните следующую команду.

$ xcdroast

Нажмите кнопку Setup (Настройка) и убедитесь в том, что все настройки вас устраивают. Нажмите кнопку Save Configuration (Сохранить настройки), а затем OK, чтобы вернуться в главное окно. Теперь выберите Duplicate CD (Копировать компакт-диск) или Create CD (Создать компакт-диск), на ваше усмотрение, и т. д. В зависимости от того, как настроена ваша система, вам, возможно, нужно будет работать в режиме суперпользователя, чтобы иметь возможность записывать компакт-диски.

Программирование в командном процессоре

Ранее, когда мы рассматривали командный процессор (bash), мы сказали, что он имеет встроенный язык программирования. Фактически, вы можете писать программы, или shell-скрипты, для выполнения таких задач, которые не может выполнить одна команда. Как и любой другой хороший язык программирования, язык командного процессора имеет переменные, конструкции ветвления (if-then-else), циклы, ввод и вывод данных, и многое другое. Целые книги были посвящены написанию shell-скриптов, поэтому мы будем рассматривать самый минимум, необходимый для того, чтобы начать работать. Чтобы получить полную документацию, выполните команду info bash.

Пробел и обрыв строки

Shell-скрипты bash очень критичны к пробелам и обрывам строк. Так как "ключевые слова" этого языка программирования фактически являются командами, обрабатываемыми командным процессором, вам нужно разделять их аргументы пробелами. Аналогично, обрыв строки в середине команды введет командный процессор в заблуждение: он подумает, что команда не полная. Следуйте правилам, которые мы вам показываем, и все будет нормально.

Переменные

Переменные мы рассмотрели ранее.

```
$ MYVAR=6
$ echo $MYVAR
6
```

Все значения, содержащиеся в переменных, являются строками, но если они численные, то командный процессор будет соответственно воспринимать их как числа.

```
$ NUMBER="10"
$ expr $NUMBER + 5
15
```

Когда вы ссылаетесь на значение переменной в shell-скрипте, следует помещать ее в двойные кавычки, чтобы избежать определенных ошибок во время его работы. Неопределенная переменная или переменная, у которой в значении есть пробелы, может быть воспринята непредсказуемым образом, если она не будет заключена в кавычки, что приведет к неправильной работе скрипта.

```
$ FILENAME="My Document"          Пробел в имени
$ Is $FILENAME                    Попробуем вывести ее
Is: My: No such file or directory Ой! Команда Is
                                  увидела 2 аргумента
Is: Document: No such file or directory
$ Is -1 "$FILENAME"               Выведем ее правильно
My Document Команда Is            увидела только l аргумент
```

Если имя переменной обрабатьшается вместе с другой соседней строкой, поместите ее в фигурные скобки, чтобы избежать непредвиденного результата.

```
$ HAT="fedora"
```

```
$ echo "The plural of $HAT is $HATs"
The plural of fedora is Ой!Переменная
"HA Ts " отсутствует
$ echo  "The plural of $HAT is "${HAT}s"
```

Ввод и вывод данных

Вывод данных в скриптах осуществляется с помощью команд echo и printf, которые мы описали в разделе "Вывод на экран" на странице 224.

```
$  echo  "Hello  world"
Hello  world
$ printf "Мне %d лет\n" 'expr 20 + 20"
Мне 40 лет
```

Ввод данных в скриптах осуществляется с помощью команды read, которая считывает одну строку из стандартного потока ввода и сохраняет ее в переменной.

```
$ read name
Sandy Smith <ENTER>
$ echo "I read the name $name"
 I read the name Sandy Smith
```

Логические переменные и возвращаемые значения

Для того чтобы описать операторы ветвления и циклы, нам нужно понятие логической (истина/ ложь) проверки. Для командного процессора значение 0 означает "истину" или "успех", а все остальные значения - "ложь" или "неудачу".

Кроме того, каждая Linux-команда при завершении работы возвращает командному процессору целочисленное значение, называемое возвращаемым значением (return code) или статусом выхода (exit status). Вы можете увидеть это значение в специальной переменной $?.

$ cat myfile ■
My name is Sandy Smith and
I really like Fedora Linux
$ grep Smith myfile
My name is Sandy Smith and Совпадение было
найдено... $ echo $?
0...поэтому возвращаемым значением будет
"успех"
$ grep aardvark myfile
$ echo $? Совпадений не было найдено...
1... поэтому возвращаемым значением
будет "неудача"

Возвращаемые значения команды, как правило, описаны на ее man-странице.

test и"["

Команда test (встроенная в командный процессор) обрабатывает простые логические выражения, включающие в себя числа и строки, и, в зависимости от результата, возвращающая значение 0 (истина) или 1 (ложь).

$ test 10 -It 5 10меньше, чем 5?
$ echo $?
1Нет
$ test -n "hello" Строка "hello"имеет
ненулевую длину?
$ echo $?
О - --Да

Список наиболее общих аргументов команды test для работы с целыми числами строками и файлами дан в табл. 12.

Команда test имеет необычный псевдоним, символ "[" (левая квадратная скобка), сокращенный вариант для работы с операторами ветвления и циклами. Если вы его используете, то вы должны в качестве последнего аргумента поставить символ "]" (правая квадратная скобка), чтобы обозначить конец команды test.

Следующие команды проверки идентичны тем, которые мы рассмотрели выше.

```
$[  10  -It  5  ]
$echo  $?
1
$[  -n  "hello"  ]
$echo  $?
```

Запомните, что "[" - это такая же команда, как и остальные, поэтому за ней следуют отдельные аргументы, разделенные пробелами. И если вы случайно забудете пробел:

```
$  [  5  -It  4]      Отсутствует пробел между
аргументами 4 и ] bash:   [:  missing  ']'
```

то команда test подумает, что последним аргументом является строка "4]", и посчитает, что финальная скобка отсутствует.

Таблица 12. Некоторые общие аргументы команды test

Проверка файлов

-d имя	Файл *«и»является директорией
-f имя	Файл /«появляется регулярным файлом
-L имя	Файл /лияявляется символьной ссылкой
-г имя	Файл имя существует и доступен для чтения
-W имя	Файл ни» существует и доступен для записи
-х имя	Файл ии» существует и доступен для выполнения
-S ИМЯ	Файл имя существует и имеет ненулевую длину
fl -nt f2	Файл //более новый, чем файл f2
fl -ot f2	Файл //более старый, чем файл 12

Проверка строк

si = s2	Строка 5/равна строке s2

si != s2	Строка 5/не равна строке s2
-z si	Строка 5/имеет нулевую длину
-n si	Строка 5/имеет ненулевую длину

Проверка чисел

a -eq b	Целые числа а и b равны
a -ne b	Целые числа аи Л не равны
a -gt b	Целое число а больше целого числа b
a -ge b	Целое число а больше или равно целому числу b
a -It b	Целое число а меньше целого числа b
a -le b	Целое число а меньше или равно целому числу Ь

Объединение и отрицание при проверке

tl -a tl	И: проверки // и £?возвращают результат "истина"
tl -o t2	ИЛИ: или проверка //, или проверка t2 возвращают результат "истина"
! тест	Отрицание результата проверки, то есть, когда гесгвозвращает результат "ложь"
\(тест \)	Круглые скобки используются для группировки, как в алгебре

true и false

Командный процессор bash имеет встроенные команды true и false, которые просто возвращают значение 0 и 1 соответственно.

```
$true
$echo $?
O
$false    .    .    -
$echo $?
1       ' : '
```

Они будут полезны, когда мы будем рассматривать операторы ветвления и циклы.

Операторы ветвления

Этот оператор осуществляет выбор между альтернативами, каждая из которых может содержать сложную проверку. Простейшая форма - это оператор if-then.

```
i f  команда        Если статус выхода команды О then
тело
fi          -.■■,-,...
Например:
if  [  'whoami4  =  "root"  ] then
echo  "Вы являетесь суперпользователем" fi
```

Затем идет оператор if-then-else.

```
if  команда then
тело! else
тело2 fi
Например:
if  [  4whoami"  =  "root"  ] then
echo  "  Вы являетесь суперпользователем  " else
echo  "Вы  являетесь  обычным пользователем" fi
```

Наконец, есть формат if-then-el if-else, в котором можно реализовать столько проверок, сколько вам будет нужно.

```
if  команда! then
тело!
elif  команда2 then
тело2 elif  ...
else
TenoN fi
Например:
if  [  'whoami1  =  "root"  ] then
echo  "  Вы являетесь  суперпользователем  " elif  [  "$USER"
=  "root"  ] then
echo  "  Вы,  должно  быть,  являетесь суперпользователем  "
elif  [  "$bribe"  -gt  10000  ] then
echo  "  Вы можете  заплатить  за  то, чтобы  стать
суперпользователем  " else
echo  "Вы  все  еще  обычный  пользователь" fi
```

Оператор case оценивает одно значение и выполняет соответствующий ему кусок кода.

echo 'Что вы хотите делать?' read answer case "$answer" in есть)
echo "OK, съешьте гамбургер"
спать)
echo "Тогда спокойной ночи"
echo "Я не понимаю, что вы хотите сделать"
echo "Надеюсь увидимся завтра"
esac Общий вид этого оператор таков.
case строка in выражение 1) тело!
выражение2) тело2
выражениеЮ
тело_ИНАЧЕ

Строка - это любое значение, как правило, переменная, например $myvar, а выражения 1-N - это шаблоны (выполните команду info bash reserved case для более подробной информации), а последний шаблон, символ *, выполняет функцию оператора "else" (иначе). Каждый набор команд должен заканчиваться символами;; (как показано).

```
case $letter in X)
echo "$letter - это буква X"
[aeiou])
echo "$letter - это гласная буква"
[0-9])
echo "$letter - это цифра"
*) ' '
echo "Я не могу обработать это"
esac
```

Циклы

Цикл while повторяет набор команд до тех пор, пока выполняется условие (т. е. пока оно возвращает значение "истина").

while команда Пока команда возвращает
значение О do
тело done
Например, рассмотрим скрипт myscript.
i=0

```
while [ $i -It 3 ]'  .•
do
echo "$i"
i=~expr $i +1' done
$ ./myscript
0
1
2
```

Цикл until повторяется до тех порка, пока условие не начнет выполняться (т. е. пока оно не вернет значение "истина"):

```
until  команда  Пока команда возвращает
ненулевое значение do
тело done
Например:
i = 0
until   t   $i   -gt   3   ]
do
echo   "$i"
i=4expr   $i   +  Г done
$   ./myscript
0
1
2
```

Цикл for итерирует по значениям из списка.

```
for  переменная in  список do
тело done
Например:
for  name   in  Tom Jack  Harry do
echo   "$name  -  мой друг" done
```

$ ./myscript Tom - мой друг Jack - мой друг Harry - мой друг

Цикл for особенно удобен для обработки списков файлов, например всех файлов определенного типа в текущей директории.

```
for  file  in   *.doc do
echo   "$file - это отвратительный файл Microsoft  Word" done
```

Для бесконечного цикла используйте цикл while с условием true ("истина") или цикл until с условием false ("ложь").

```
while   true do
echo   "навсегда" done
until   false do
echo   "и  снова  навсегда" done
```

Вероятно, вы будете использовать операторы break или exit для того, чтобы завершить эти циклы при выполнении какого-нибудь условия.

Break и Continue

Команда break осуществляет выход из текущего цикла. Рассмотрим простой скрипт myscript.

```
for  name   in  Tom Jack  Harry do
echo  $name
echo "снова" done echo "все сделано"
$ ./myscript
Tom
снова
Jack      . : ■
снова
Harry
снова
все сделано
```

А теперь с оператором break.

```
for  name   in  Tom Jack  Harry do
echo   $name
if   [   "$name"   =   "Jack"    ]
then break
fi
echo   "снова" done echo   "все  сделано"
Tom
снова
Jack      Выполнен оператор break
все   сделано
```

Команда continue форсирует переход цикла к следующей итерации.

```
for name   in  Tom Jack Harry do
echo  $name
```

```
if [ "$name" = "Jack" ]
then
continue
fi
echo "снова" done echo "все сделано"
$ ./myscript
Tom
снова
Jack      Выполнен оператор continue
Harry
снова
все сделано
```

Также операторы break и continue принимают численные аргументы (break N, continue N) для того, чтобы управлять несколькими уровнями циклов (например, выйти из N вложенных друг в друга циклов), но такой стиль написания скриптов приводит к созданию запутанного кода, и мы не рекомендуем его придерживаться.

Создание и запуск shell-скриптов

Для того чтобы создать shell-скрипт, просто поместите команды bash в файл, как будто если бы вы набирали их в командном процессоре. Чтобы запустить скрипт, у вас есть три способа.

Добавить в начале файла строку #!/bin/bash и сделать файл исполняемым

Это самый распространенный способ запуска скриптов. Добавьте строку:

#!/bin/bash

в самом начале файла скрипта. Это должна быть самая первая строка файла, выровненная влево. После этого сделайте файл исполняемым.

$ chmod +x myscript

Опционально, поместите его в вашу директорию поиска. Затем запустите его как любую другую команду.

$ myscript

Если скрипт находится в вашей текущей директории, но текущая директория "." не является директорией поиска, то вам нужно будет приписать вначале "./", чтобы командный процессор мог найти скрипт. -

$ ./myscriptъ

Текущая директория, как правило, не является вашей директорией поиска по соображениям безопасности.

Передать в качестве аргумента команде bash Команда bash интерпретирует свой аргумент как имя скрипта и запускает его.

$ bash myscript

Запустить в текущем командном процессоре с командой "."

Предыдущие методы запускают ваш скрипт как независимый процесс, который не влияет на ваш текущий командный процессор.* Если вы хотите, чтобы ваш скрипт вносил изменения в ваш командный процессор (изменял переменные, рабочую директорию и так далее), его можно запустить в текущем командном профессоре с командой ".".

$. myscript

Аргументы командной строки

Shell-скрипты могут принимать командные аргументы и опции точно так же, как и команды Linux (фактически, некоторые общеупотребительные команды Linux являются скриптами). В пределах своего shell-скрипта вы можете обращаться к этим аргументам с помощью переменных $1, $2, $3 и т. д.

$ cat myscript
#!/bin/bash

echo "Меня зовут $1 и я живу в $2"

* Технически скрипт вьшолняется в отдельном (дочернем) командном процессоре, который наследует атрибуты оригинального командного процессора но не может изменить их у своего "родителя".

$ ./myscript Johnson Wisconsin

Меня зовут Johnson и я живу в Wisconsin

$ ./myscript Bob

Меня зовут Bob и я живу в

Ваш скрипт может проверять количество полученных аргументов с помощью переменной $ #.

if [$# -It 2] then

echo "$0 ошибка: вы должны задать два аргумента" else

echo " Меня зовут $1 и я живу в $2" fi

Специальное значение $0 содержит имя скрипта, и ее удобно использовать для справочных сообщений и сообщений об ошибках.

$ ./myscript Bob

./myscript ошибка: вы должны задать два

аргумента

Чтобы осуществить итерацию по всем командным аргументам, используйте цикл for со специальной переменной $@, которая содержит все аргументы.

for arg in $@ ■ ■■- .

do

echo "Я нашел аргумент $arg" done

Завершение работы с возвратом значения

Команда exit завершает работу вашего скрипта и передает заданное возвращаемое значение командному процессору. По традиции скрипты должны возвращать 0 в случае успеха и 1 (или другое ненулевое значение) в случае неудачи или ошибки. Если ваш скрипт не вызывает команду exit, то возвращаемым значением автоматически будет 0.

if [$# -It 2] then

echo "ошибка: вы должны задать два аргумента"

exit 1 else

echo "Меня зовут $1 и я живу в $2" fi exit 0

```
$ ./myscript Bob
./myscript ошибка: вы должны задать два
аргумента
$ echo $?
1
```

За пределами shell-скриптов

Shell-скрипты хорошо подходят для многих целей, но Linux имеет более мощные скриптовые языки, а также компилируемые языки программирования. Вот некоторые из них.

Язык Программа Чтобы начать...
Perl perl man perl
http://www.perl. com/
Python python man python
http://www.python.org/
C, C++ gcc man gcc
http://www.gnu.org/software/gcc/
Java javaca http://java.sun.com/
FORTRAN g77 man g77
http://www.gnu. org/software/fortran/fortran. html
Ada gnat info gnat
http://www.gnu.org/software/gnat/gnat.html

а Не включен в Fedora, как и во многие другие дистрибутивы Linux

Заключительные слова

Хотя мы рассмотрели многие команды и возможности системы Linux, но это лишь малая их часть. Fedora и другие дистрибутивы имеют тысячи различных программ. Мы поддерживаем ваш интерес к изучению Linux-систем. Удачи!

Слова благодарности

Искренне признателен моему редактору Майку Лукидису (Mike Loukides), персоналу O'Reilly, техническим редакторам (Ron Bellomo, Wesley Crossman, David Debonnaire, Tim Greer, Jacob Heider, и Eric van Oorschot), Алексу Сковтка (Alex Schowtka) и

Роберту Дюлани (Robert Dulaney) из компании VistaPrint, и моей замечательной семье, Лизе и Софии.

Предметный указатель