**8. Продвинутые приемы работы с клавиатурой**

Я часто шутливо описываю Unix как «операционную систему для тех, кто любит печатать». Казалось бы, сам факт наличия командной строки доказывает это. Но в действительности пользователи командной строки не любят печатать слишком много. Зачем, если есть так много команд с короткими именами, таких как cp, ls, mv и rm?

Фактически одной из самых заветных целей командной строки является уменьшение объема ввода — возможность выполнить большую часть работы всего несколькими нажатиями клавиш. Другая цель — не позволить рукам оторваться от клавиатуры и коснуться мыши. В этой главе мы рассмотрим возможности bash, увеличивающие скорость и эффективность использования клавиатуры.

Здесь будут представлены следующие команды:

• clear — очищает экран.

• history — выводит содержимое истории команд.

**Редактирование командной строки**

Для поддержки операций редактирования командной строки bash использует библиотеку (коллекцию подпрограмм, которую могут использовать разные программы) с именем *Readline*. Мы уже видели некоторые из них. Например, нам знакомы клавиши со стрелками влево и вправо, перемещающие курсор, но существует еще целое множество других операций. Рассматривайте их как дополнительные инструменты, которые можно использовать в работе. Необязательно стремиться изучить их все, но многие из них весьма практичны. Выбирайте те, что вам понравятся.

**ПРИМЕЧАНИЕ**

Некоторые комбинации клавиш, описываемые далее (особенно те, что включают клавишу **ALT**), могут перехватываться графическим интерфейсом и использоваться для выполнения других функций. Однако все комбинации без исключения должны правильно работать в виртуальной консоли.

**Перемещение курсора**

В табл. 8.1 перечислены комбинации клавиш, используемые для перемещения курсора.

**Таблица 8.1. Команды перемещения курсора**

|  |  |
| --- | --- |
| **Клавиша** | **Действие** |
| **CTRL+A** | Перемещает курсор в начало строки |
| **CTRL+E** | Перемещает курсор в конец строки |
| **CTRL+F** | Перемещает курсор на один символ вперед; действует так же, как клавиша со стрелкой вправо |
| **CTRL+B** | Перемещает курсор на один символ назад; действует так же, как клавиша со стрелкой влево |
| **ALT+F** | Перемещает курсор на одно слово вперед |
| **ALT+B** | Перемещает курсор на одно слово назад |
| **CTRL+L** | Очищает экран и устанавливает курсор в левый верхний угол. То же самое делает команда clear |

**Изменение текста**

В табл. 8.2 перечислены комбинации клавиш для редактирования символов в командной строке.

**Вырезание и вставка (удаление и возврат) текста**

В документации к Readline используется термин *killing and yanking* (удаление и возврат), обозначающий операцию, которую обычно называют вырезанием и вставкой (cutting and pasting). В табл. 8.3 перечислены комбинации клавиш, выполняющие вырезание и вставку. Вырезанные элементы сохраняются в кольцевом буфере, который называется *kill-ring* (кольцо удалений).

**Таблица 8.2. Команды редактирования текста**

| **Клавиша** | **Действие** |
| --- | --- |
| **CTRL+D** | Удаляет символ в позиции курсора |
| **CTRL+T** | Меняет местами два символа — в позиции курсора и предшествующий ему |
| **ALT+T** | Меняет местами два слова — в позиции курсора и предшествующий ему |
| **ALT+L** | Переводит в нижний регистр символы, начиная с символа в позиции курсора и до конца слова |
| **ALT+U** | Переводит в верхний регистр символы, начиная с символа в позиции курсора и до конца слова |

**Таблица 8.3. Команды вырезания и вставки**

|  |  |
| --- | --- |
| **Клавиша** | **Действие** |
| **CTRL+K** | Удаляет символы от позиции курсора до конца строки |
| **CTRL+U** | Удаляет символы от позиции курсора до начала строки |
| **ALT+D** | Удаляет символы от позиции курсора до конца текущего слова |
| **ALT+BACKSPACE** | Удаляет символы от позиции курсора до начала текущего слова. Если курсор находится в начале слова, удаляется предшествующее слово |
| **CTRL+Y** | Извлекает текст из кольцевого буфера удалений и вставляет его в позицию курсора |

**клавиша meta**

Отважившиеся заглянуть в документацию к Readline, которая находится в разделе «READLINE», на странице справочного руководства (man) для bash, столкнутся с термином *клавиша meta* (meta key). На современных клавиатурах ей соответствует клавиша **ALT**, но так было не всегда.

В стародавние времена (до появления IBM-совместимых персональных компьютеров, но после появления Unix) персональные компьютеры не были так широко распространены. Иногда их заменяли устройства, называемые *терминалами*. Терминал — это коммуникационное устройство с текстовым дисплеем и клавиатурой, имеющее внутри столько электроники, сколько необходимо для отображения символов и перемещения курсора. Терминалы подключались (обычно посредством последовательного кабеля) к большому компьютеру или коммуникационной сети большого компьютера. В то время существовало очень много различных терминалов, имевших разные клавиатуры и дисплеи с разными функциональными возможностями. Так как все они поддерживали как минимум набор символов ASCII, разработчикам программного обеспечения, пишущим переносимые приложения, необходимо было прийти к общему знаменателю. В системах Unix применяется очень сложный способ использования терминалов и их разнообразных возможностей. Поскольку разработчики Readline не были уверены в наличии специализированной управляющей клавиши, они изобрели ее и назвали *meta*. На современных клавиатурах роль клавиши meta играет **ALT**, однако если вы все еще используете терминал (до сих пор поддерживаются в Linux!), можно просто нажать и отпустить клавишу **ESC**, и вы получите эффект нажатия и удержания клавиши **ALT**.

**Дополнение**

Другой вариант помощи пользователям реализован в командной оболочке в виде механизма *дополнения* (completion). Дополнение происходит, когда в процессе ввода команды нажимается клавиша **TAB**. Давайте посмотрим, как это работает. Допустим, что ваш домашний каталог содержит следующее:

[me@linuxbox ~]$ **ls**

Desktop    ls-output.txt  Pictures  Templates   Videos

Documents  Music          Public

Попробуйте ввести следующую строку, но не нажимайте клавишу **ENTER**:

[me@linuxbox ~]$ **ls l**

Теперь нажмите клавишу **TAB**:

[me@linuxbox ~]$ **ls ls-output.txt**

Обратили ли вы внимание, как командная оболочка дополнила командную строку за вас? Попробуйте теперь набрать следующую строку — и снова не нажимайте **ENTER**):

[me@linuxbox ~]$ **ls D**

Нажмите **TAB**:

[me@linuxbox ~]$ **ls D**

Дополнения не произошло — просто прозвучал звуковой сигнал. Так получилось потому, что символу D соответствует более одного элемента в каталоге. Чтобы командная оболочка дополнила вашу строку, предложенная вами «подсказка» должна иметь однозначное продолжение. Попробуйте продолжить ввод:

[me@linuxbox ~]$ **ls Do**

Затем нажмите **TAB**:

[me@linuxbox ~]$ **ls Documents**

Дополнение произошло.

Этот пример демонстрирует дополнение путей как наиболее частый случай использования дополнения. Однако дополнение также работает с именами переменных (когда слово начинается с символа $), именами пользователей (когда слово начинается с символа ~), командами (когда дополняемое слово является первым в командной строке) и сетевыми именами компьютеров (когда слово начинается с символа @). Дополнение сетевых имен компьютеров действует только в отношении имен, перечисленных в */etc/hosts*.

С механизмом дополнения связано несколько управляющих комбинаций клавиш (табл. 8.4).

Существует еще несколько команд, смысл которых для меня не совсем ясен. Полный список вы сможете найти на странице справочного руководства (man) для bash, в разделе «READLINE».

**Таблица 8.4. Команды дополнения**

|  |  |
| --- | --- |
| **Клавиша** | **Действие** |
| **ALT+?** | Выводит список возможных дополнений. В большинстве систем аналогичный эффект можно получить, нажав клавишу **TAB** второй раз, что намного проще |
| **ALT+\*** | Вставит все возможные дополнения. Это пригодится в том случае, если требуется использовать больше одного возможного варианта дополнения |

**ПРограммируемое дополнение**

Последние версии bash реализуют механизм *программируемого дополнения*. Программируемое дополнение дает возможность добавлять дополнительные правила. Обычно это делается с целью добавить поддержку определенных приложений. Например, можно добавить дополнение списка параметров команды или файлов определенного типа, поддерживаемых приложением. В Ubuntu определено огромное множество таких правил. Программируемое дополнение реализуется посредством функций командной оболочки — небольших мини-сценариев, о которых будет рассказываться в следующих главах. Если вам любопытно, попробуйте выполнить команду

set | less

и вы увидите их. Однако не все дистрибутивы включают эти функции по умолчанию.

**Использование истории**

Как рассказывалось в главе 1, bash поддерживает историю вводившихся команд. Этот список команд хранится в домашнем каталоге, в файле с именем *.bash\_history*. Механизм истории помогает уменьшить объем ручного ввода, особенно в сочетании с командами редактирования командной строки.

**Поиск в истории**

Просмотреть содержимое истории можно в любой момент с помощью команды:

[me@linuxbox ~]$ **history | less**

По умолчанию bash хранит последние 500 введенных команд. Как изменить это значение, мы узнаем в главе 11. А теперь представим, что вам понадобилось найти команды, использовавшиеся для получения списка содержимого */usr/bin*. Вот один из возможных способов:

[me@linuxbox ~]$ **history | grep /usr/bin**

А теперь представим, что среди результатов нужно выбрать запись с интересующей вас командой:

88 ls -l /usr/bin > ls-output.txt

Здесь число 88 — это порядковый номер записи команды в списке истории. Зная это число, можно воспользоваться еще одной разновидностью подстановки, которая называется *подстановкой записей истории* (history expansion). Для этого введите:

[me@linuxbox ~]$ **!88**

и bash заменит !88 содержимым 88-й записи в списке истории. Подробнее об этой форме подстановки записей истории мы поговорим чуть ниже.

bash также дает возможность выполнять поступательный поиск в списке истории. Это означает, что bash может выполнять поиск в списке истории по мере ввода символов, уточняя результаты с вводом каждого нового символа. Чтобы запустить поступательный поиск, нажмите комбинацию **CTRL+R** и введите искомый текст. Закончив поиск, нажмите **ENTER**, чтобы выполнить команду, или **CTRL+J**, чтобы скопировать запись из списка истории в текущую командную строку. Чтобы найти следующее вхождение текста (переместиться «вверх» по списку истории), нажмите **CTRL+R** еще раз. Чтобы завершить поиск, нажмите **CTRL+G** или **CTRL+C**. Следующий пример демонстрирует, как действует поиск:

[me@linuxbox ~]$

Первое нажатие комбинации **CTRL+R**:

(reverse-i-search)`':

Приглашение к вводу изменится, показывая, что выполняется поступательный поиск в обратном порядке. Под словами «в обратном порядке» подразумевается, что поиск выполняется от «текущего момента» до некоторого момента в прошлом. Далее мы начинаем ввод искомого текста, в данном примере /usr/bin:

(reverse-i-search)`**/usr/bin**': ls -l /usr/bin > ls-output.txt

Механизм поиска сразу же возвращает результат. Теперь, чтобы выполнить найденную команду, необходимо нажать **ENTER**, или вы можете скопировать команду в командную строку для дальнейшего редактирования, нажав **CTRL+J**. Давайте скопируем ее. Нажмите **CTRL+J**:

[me@linuxbox ~]$ ls -l /usr/bin > ls-output.txt

Механизм поиска вернет управление, командная строка заполнится и будет готова для выполнения!

В табл. 8.5 перечислены некоторые комбинации клавиш, используемые для манипуляций со списком истории команд.

**Таблица 8.5. Команды для работы с историей**

|  |  |
| --- | --- |
| **Клавиша** | **Действие** |
| **CTRL+P** | Переход к предыдущей записи в истории. Действует так же, как клавиша со стрелкой вверх |
| **CTRL+N** | Переход к следующей записи в истории. Действует так же, как клавиша со стрелкой вниз |
| **ALT+<** | Переход в начало (к первой записи) списка истории |
| **ALT+>** | Переход в конец (к последней записи) списка истории |
| **CTRL+R** | Инкрементальный поиск в обратном порядке. Поиск выполняется поступательно, от текущей записи вверх по списку истории |
| **ALT+P** | Поиск в обратном порядке, не инкрементальный. При использовании этого вида поиска введите искомую строку и нажмите **ENTER**, и только после этого будет выполнен фактический поиск |
| **ALT+N** | Поиск в прямом порядке, не поступательный |
| **CTRL+O** | Выполнить текущую команду в списке истории и перейти к следующей. Эту комбинацию удобно использовать, если требуется повторно выполнить последовательность команд из списка истории |

**Подстановка записей истории**

Командная оболочка поддерживает специализированный вид подстановки — подстановку записей из списка истории при использовании символа !. Мы уже видели, как восклицательный знак, сопровождаемый числом, замещается записью из списка истории. Этот вид подстановки имеет несколько разновидностей (табл. 8.6).

Не используйте формы !*строка* и !?*строка*, если только вы абсолютно точно не знаете содержимого записей в списке истории.

Механизм подстановки записей истории поддерживает также другие комбинации, но эта тема становится слишком запутанной, и мы не станем перегружать себя лишней информацией. Желающие смогут обратиться к странице справочного руководства (man) для bash, в разделе «HISTORY EXPANSION». Загляните туда!

**Таблица 8.6. Команды механизма подстановки записей истории**

|  |  |
| --- | --- |
| **Последовательность** | **Действие** |
| !! | Повторяет последнюю команду. Проще, пожалуй, нажать клавишу со стрелкой вверх и **ENTER** |
| !*число* | Повторяет команду из записи с указанным номером |
| !*строка* | Повторяет последнюю команду в списке истории, начинающуюся с указанной строки |
| !?*строка* | Повторяет последнюю команду в списке истории, содержащую указанную строку |

**script**

В дополнение к истории команд в bash большинство дистрибутивов Linux включают программу script, которую можно использовать для записи в файлы целых сеансов работы с командной оболочкой. Базовый синтаксис команды:

script [*файл*]

где *файл* — это имя файла для записи. Если *файл* не будет указан, сохранение сеанса будет произведено в файл *typescript*. Полное описание параметров и возможностей программы можно найти на странице справочного руководства (man) для script.

**Заключительное замечание**

В этой главе мы рассмотрели *несколько* приемов работы с клавиатурой, поддерживаемых командной оболочкой, с целью помочь истинным фанатам клавиатуры уменьшить объем работы. Я думаю, что потом, когда вы сроднитесь с командной строкой, вы сможете обратиться к этой главе, чтобы вспомнить описанные здесь приемы. А пока будем считать их необязательными, но потенциально полезными.

**3. Исследование системы**

Теперь, когда мы знаем, как перемещаться по файловой системе, совершим обзорное путешествие по системе Linux. Но прежде чем отправиться, познакомимся еще с несколькими командами, которые пригодятся в пути:

• ls — выводит список содержимого каталога.

• file — определяет тип файла.

• less — выводит содержимое файла.

**Любопытные возможности ls**

Команда ls является, пожалуй, одной из самых часто используемых команд, и не без оснований. С ее помощью можно увидеть, что находится в каталоге, и узнать некоторые важные атрибуты файлов и каталогов. Как мы уже видели, чтобы получить список файлов и подкаталогов в текущем рабочем каталоге, достаточно ввести ­команду ls:

[me@linuxbox ~]$ **ls**

Desktop  Documents  Music  Pictures  Public  Templates  Videos

Команде можно явно указать каталог, содержимое которого требуется вывести:

me@linuxbox ~]$ **ls /usr**

bin  games    kerberos  libexec  sbin   src

etc  include  lib       local    share  tmp

и даже несколько каталогов. Следующий пример выведет содержимое домашнего каталога пользователя (обозначен символом *~*) и каталога */usr*:

[me@linuxbox ~]$ **ls ~ /usr**

/home/me:

Desktop  Documents  Music  Pictures  Public  Templates  Videos

/usr:

bin  games    kerberos  libexec  sbin   src

etc  include  lib       local    share  tmp

Можно также изменить формат вывода, чтобы получить больше информации:

[me@linuxbox ~]$ **ls -l**

total 56

drwxrwxr-x 2 me me 4096 2012-10-26 17:20 Desktop

drwxrwxr-x 2 me me 4096 2012-10-26 17:20 Documents

drwxrwxr-x 2 me me 4096 2012-10-26 17:20 Music

drwxrwxr-x 2 me me 4096 2012-10-26 17:20 Pictures

drwxrwxr-x 2 me me 4096 2012-10-26 17:20 Public

drwxrwxr-x 2 me me 4096 2012-10-26 17:20 Templates

drwxrwxr-x 2 me me 4096 2012-10-26 17:20 Videos

Параметр -l, добавленный в команду, требует использования «длинного» (long) формата вывода.

**Параметры и аргументы**

Мы подошли к очень важному моменту, касающемуся особенностей работы большинства команд. Команды часто сопровождаются одним или несколькими *параметрами*, изменяющими их поведение, и дополнительными, одним или несколькими, *аргументами*, на которые воздействует команда. Поэтому большинство команд выглядят примерно так:

команда -параметры аргументы

Большинство команд используют параметры, состоящие из одного символа, которому предшествует дефис, например: -l. Но многие команды, в том числе команды из проекта GNU, поддерживают *параметры с длинными именами*, состоящие из слова, которому предшествуют два дефиса. Кроме того, многие команды позволяют объединять вместе параметры с короткими именами. В следующем примере команде ls передаются два параметра: параметр l, требующий использовать длинный (long) формат вывода, и параметр t, требующий сортировать результаты по времени (time) изменения:

[me@linuxbox ~]$ **ls -lt**

Добавим параметр с длинным именем --reverse, чтобы изменить порядок сортировки на обратный:

[me@linuxbox ~]$ **ls -lt --reverse**

Команда ls имеет огромное число допустимых параметров. Наиболее популярные из них перечислены в табл. 3.1.

**Таблица 3.1. Наиболее популярные параметры команды ls**

|  |  |  |
| --- | --- | --- |
| **Параметр** | **Длинный параметр** | **Описание** |
| -a | --all | Список всех (all) файлов, даже с именами, начинающимися с точки, которые обычно не выводятся (то есть скрытых) |
| -d | --directory | Обычно в присутствии этого параметра команда ls выводит информацию о самом каталоге, а не его содержимое. Используйте этот параметр в сочетании с параметром -l, чтобы получить дополнительную информацию о каталоге, а не о его содержимом |
| -F | --classify | Добавляет в конец каждого имени символ-индикатор (например, прямой слеш, если это имя каталога) |
| -h | --human-readable | При использовании длинного формата вывода отображает размеры файлов не в байтах, а в величинах с единицами измерения |
| -l |  | Выводит результаты с использованием длинного формата |
| -r | --reverse | Выводит результаты в обратном порядке. Обычно коман­да ls выводит результаты в алфавитном порядке |
| -S |  | Сортировать результаты по размеру (size) |
| -t |  | Сортировать результаты по времени (time) последнего изменения |

**Пристальный взгляд на длинный формат**

Как было показано выше, параметр -l заставляет команду ls выводить результаты с использованием длинного формата. Этот формат предусматривает вывод большого количества полезной информации. Ниже приводится пример вывода содержимого каталога *Examples* в системе Ubuntu:

-rw-r--r-- 1 root root 3576296 2012-04-03 11:05 Experience ubuntu.ogg

-rw-r--r-- 1 root root 1186219 2012-04-03 11:05 kubuntu-leaflet.png

-rw-r--r-- 1 root root   47584 2012-04-03 11:05 logo-Edubuntu.png

-rw-r--r-- 1 root root   44355 2012-04-03 11:05 logo-Kubuntu.png

-rw-r--r-- 1 root root   34391 2012-04-03 11:05 logo-Ubuntu.png

-rw-r--r-- 1 root root   32059 2012-04-03 11:05 oo-cd-cover.odf

-rw-r--r-- 1 root root  159744 2012-04-03 11:05 oo-derivatives.doc

-rw-r--r-- 1 root root   27837 2012-04-03 11:05 oo-maxwell.odt

-rw-r--r-- 1 root root   98816 2012-04-03 11:05 oo-trig.xls

-rw-r--r-- 1 root root  453764 2012-04-03 11:05 oo-welcome.odt

-rw-r--r-- 1 root root  358374 2012-04-03 11:05 ubuntu Sax.ogg

Рассмотрим различные поля для одного из файлов и их назначение (табл. 3.2).

**Таблица 3.2. Поля длинного формата вывода команды ls**

| **Поле** | **Назначение** |
| --- | --- |
| -rw-r-r-- | Права доступа к файлу. Первый символ указывает тип файла. Например, символом дефиса обозначаются обычные файлы, а символом d — каталоги. Следующие три символа сообщают о правах доступа для владельца файла, следующие три — для членов группы, которой принадлежит файл, и последние три — для всех остальных. Более полное обсуждение прав доступа приводится в главе 9 |
| 1 | Число жестких ссылок на файл. Подробнее о ссылках рассказывается в конце этой главы |
| root | Имя пользователя, владеющего файлом |
| root | Имя группы, владеющей файлом |
| 32059 | Размер файла в байтах |
| 2012-04-03 11:05 | Дата и время последнего изменения файла |
| oo-cd-cover.odf | Имя файла |

**Определение типов файлов командой file**

Занимаясь исследованием системы, полезно иметь возможность определять тип содержимого файлов. В этом нам поможет команда file. Как отмечалось выше, имена файлов в Linux не обязаны отражать тип содержимого файлов. Например, увидев имя файла *picture.jpg*, можно предположить, что он содержит изображение в формате JPEG, но в Linux такие предположения могут не оправдываться. Вызвать команду file можно так:

file *имя\_файла*

Команда file выводит краткое описание содержимого файла. Например:

[me@linuxbox ~]$ **file picture.jpg**

picture.jpg: JPEG image data, JFIF standard 1.01

Существует множество разных типов файлов. Одна из известных идей в Unix-подобных системах, таких как Linux, гласит: «Все сущее есть файл». По мере чтения книги вы убедитесь в истинности этого утверждения.

Типы многих файлов в вашей системе будут вам знакомы, например файлы MP3 и JPEG, но иногда будут попадаться файлы с малоизвестными и даже странными типами.

**Просмотр содержимого файлов командой less**

Команда less — это программа для просмотра текстовых файлов. В системе Linux присутствует множество файлов, содержащих обычный, удобочитаемый текст. Программа less предоставляет удобный способ исследовать их содержимое.

Зачем может понадобиться исследовать текстовые файлы? Дело в том, что многие файлы с системными настройками (их называют *конфигурационными файлами*) хранят информацию в этом формате, что дает возможность прочитать их и вникнуть в особенности работы системы. Кроме того, в этом формате хранятся многие программы в системе (их называют *сценариями*). В последних главах мы узнаем, как редактировать файлы с настройками системы и как писать свои сценарии, а пока просто просматривайте их содержимое.

**что есть «текст»**

Существует множество способов представления информации в компьютере. Все способы связаны с определением отношения между смысловой информацией и числами, которые применяются для ее представления. В конце концов, компьютеры могут работать только с числами, и все данные в компьютере преобразуются в числовое представление.

Некоторые из этих систем представления очень сложны (например, сжатые видеофайлы), другие, напротив, очень просты. Одна из самых ранних и простых систем называется *ASCII-текст*. ASCII (произносится «ас-ки») — это аббревиатура названия «American Standard Code for Information Interchange» (американский стандартный код для обмена информацией). Эта простая система кодирования впервые была использована в телетайпах.

Текст — это простое отображение «один в один» символов в числа. Это очень компактный формат. Пятьдесят символов текста преобразуются в пятьдесят байт данных. Но это не то же самое, что текст в документе, созданном текстовым процессором, таким как Microsoft Word или OpenOffice.org Writer. Файлы документов, в отличие от простых файлов с ASCII-текстом, содержат множество нетекстовых элементов, используемых для описания их структуры и форматирования. Файлы с простым ASCII-текстом содержат только сами символы и очень небольшое количество простейших кодов управления, таких как символы табуляции, возврата каретки и перевода строки.

В системе Linux многие файлы хранятся в текстовом формате, и многие инструменты работают с текстовыми файлами. Даже Windows признает важность этого формата. Хорошо известная программа Notepad (Блокнот) — это редактор для простых файлов с ASCII-текстом.

Команда less используется так:

less *имя\_файла*

После запуска программа less позволяет прокручивать текстовый файл взад и вперед. Например, просмотреть содержимое файла со всеми известными системе учетными записями пользователей можно с помощью следующей команды:

[me@linuxbox ~]$ **less /etc/passwd**

После запуска программа less выведет содержимое файла. Если файл занимает больше одной страницы, его можно прокручивать вверх и вниз. Чтобы выйти из программы less, нажмите клавишу **Q**.

В табл. 3.3 перечислены клавиатурные команды, наиболее часто используемые в программе less.

**Таблица 3.3. Команды программы less**

|  |  |
| --- | --- |
| **Команда** | **Действие** |
| **Page Up** или **b** | Прокрутка к началу на одну страницу |
| **Page Down** или **ПРОБЕЛ** | Прокрутка к концу на одну страницу |
| **СТРЕЛКА ВВЕРХ** | Прокрутка к началу на одну строку |
| **СТРЕЛКА ВНИЗ** | Прокрутка к концу на одну строку |
| **G** | Переход в конец текстового файла |
| **1G** или **g** | Переход в начало текстового файла |
| **/***символы* | Поиск вниз по тексту до ближайшего вхождения указанной последовательности символов |
| **n** | Поиск следующего вхождения искомой последовательности символов |
| **h** | Вывод экрана со справкой |
| **q** | Завершить less |

**меньше значит больше**

Программа less создавалась как улучшенная замена более ранней Unix-программы с именем more. Ее имя — это игра слов «less is more» (меньше значит больше) — девиз архитекторов-модернистов и проектировщиков.

less относится к категории программ постраничного просмотра текстовых документов, которые называют пейджерами (pagers). В отличие от программы more, которая может листать страницы только вперед, программа less способна листать текст в обоих направлениях, вперед и назад, и имеет множество других особенностей.

**Обзорное путешествие**

Файловая система в Linux имеет практически ту же компоновку, что и в других Unix-подобных системах. Фактически ее структура определяется опубликованным стандартом с названием «Linux Filesystem Hierarchy Standard». Не все дистрибутивы Linux следуют этому стандарту, но большинство придерживаются его достаточно близко.

А теперь немного попутешествуем по файловой системе и познакомимся с основными достопримечательностями системы Linux. Это даст нам шанс попрактиковаться в навигации. Первое, что мы обнаружим: многие интересные файлы имеют простой текстовый формат. В ходе путешествия пробуйте выполнить следующие действия:

1. С помощью команды cd перейдите в указанный каталог.

2. Выведите содержимое каталога командой ls -l.

3. Если увидите заинтересовавший вас файл, определите его тип командой file.

4. Если файл выглядит как текстовый, попробуйте просмотреть его командой less.

**ПРИМЕЧАНИЕ**

Вспомните трюк с копированием и вставкой! Если вы пользуетесь мышью, выполните двойной щелчок на имени файла, чтобы скопировать его, и щелчок средней кнопкой, чтобы вставить в команду.

В ходе путешествия не бойтесь заглядывать внутрь системы. Обычные пользователи практически ничего не смогут испортить. Это работа системного администратора! Если команда пожалуется на что-то, просто перейдите к чему-нибудь другому. Потратьте некоторое время на знакомство с окрестностями. Это наша система, и мы вправе заниматься ее исследованием. Помните, что в Linux нет секретов!

В табл. 3.4 перечислены несколько каталогов для исследования. Но вы можете заняться исследованием любых других каталогов!

**Таблица 3.4. Каталоги в системе Linux**

| **Каталог** | **Описание** |
| --- | --- |
| */* | Корневой каталог, откуда все начинается |
| */bin* | Содержит двоичные (binaries) файлы (программы), необходимые для загрузки и функционирования системы |
| */boot* | Содержит ядро Linux, образ начального RAM-диска (с драйверами, необходимыми на этапе загрузки) и сам загрузчик.  Интересные файлы:  */boot/grub/grub.conf* или *menu.lst*, используются для настройки загрузчика  */boot/vmlinuz*, ядро Linux |
| */dev* | Специальный каталог, содержащий узлы устройств. «Все сущее есть файл» применяется также к устройствам. Здесь ядро хранит список всех известных ему устройств |
| */etc* | Каталог */etc* содержит все системные конфигурационные файлы. Здесь же хранится коллекция сценариев командной оболочки, запускающих системные службы во время загрузки. Практически все файлы в этом каталоге содержат обычный читаемый текст.  Интересные файлы: в */etc* все интересно, но, на мой взгляд, особенный интерес представляют:  */etc/crontab*, файл, определяющий время запуска автоматизированных заданий;  */etc/fstab*, таблица устройств хранения и соответствующих им точек монтирования;  */etc/passwd*, список всех учетных записей пользователей |
| */home* | В обычных конфигурациях каждому пользователю выделяется свой домашний каталог в каталоге */home*. Простые пользователи могут записывать что-нибудь только в файлы, находящиеся в их домашних каталогах. Это ограничение защищает систему от необдуманных действий пользователей |
| */lib* | Содержит файлы разделяемых библиотек, используемых основными системными программами. Они напоминают библиотеки DLL в Windows |
| */lost+found* | Каждый раздел или устройство, отформатированные с использованием файловой системы Linux, такой как ext3, будут иметь этот каталог. Он используется на случай частичного восстановления повреждений в файловой системе. Если с системой ничего страшного не происходило, этот каталог будет оставаться пустым |
| */media* | В современных системах Linux каталог */media* будет содержать точки монтирования съемных носителей, таких как USB-диски, CD-ROM и т.д., которые монтируются в момент подключения |
| */mnt* | В старых системах Linux каталог */mnt* содержал точки монтирования съемных носителей, монтируемых вручную |
| */opt* | Каталог */opt* используется для установки «необязательного» (optional) программного обеспечения. В основном используется для установки коммерческого программного обеспечения |
| */proc* | Специальный каталог. Не является фактической файловой системой, в том смысле, что файлы в этом каталоге не хранятся на жестком диске. Это виртуальная файловая система, поддерживаемая ядром Linux. Файлы в ней являются «глазк*а*ми», через которые можно заглянуть в ядро. Эти файлы доступны для чтения и помогают «увидеть» компьютер глазами ядра |
| */root* | Домашний каталог пользователя root |
| */sbin* | Каталог содержит системные двоичные файлы (system binaries). Эти программы выполняют жизненно важные задачи и обычно запускаются только суперпользователем |
| */tmp* | Каталог */tmp* играет роль временного хранилища для временных файлов, создаваемых разными программами. В некоторых конфигурациях этот каталог принудительно очищается при каждой перезагрузке системы |
| */usr* | Дерево каталогов */usr* является, пожалуй, самым объемным в системе Linux. В нем хранятся все программы и файлы поддержки, используемые обычными пользователями |
| */usr/bin* | Каталог */usr/bin* содержит выполняемые программы, установленные дистрибутивом Linux. Очень часто в этом каталоге хранятся тысячи программ |
| */usr/lib* | Содержит разделяемые библиотеки для программ в */usr/bin* |
| */usr/local* | Дерево каталогов */usr/local* используется для установки тех программ, которые не входят в состав дистрибутива, но должны быть доступны всем пользователям в системе. Программы, собираемые из исходных текстов, обычно устанавливаются в */usr/local/bin*. В новейших версиях системы Linux это дерево каталогов присутствует, но остается пустым, пока системный администратор не добавит туда что-нибудь |
| */usr/sbin* | Содержит дополнительные программы для администрирования |
| */usr/share* | Каталог */usr/share* содержит все разделяемые данные, используемые программами в */usr/bin*, в том числе конфигурационные файлы с настройками по умолчанию, ярлыки, фоновые изображения для рабочего стола, звуковые файлы и т.д. |
| */usr/share/doc* | Большинство пакетов, установленных в системе, содержат документацию. Вся эта документация, организованная по пакетам, хранится в каталоге */usr/share/doc* |
| */var* | За исключением */tmp* и */home*, все упоминавшиеся выше каталоги остаются относительно статичными; то есть их содержимое почти не меняется. Дерево каталогов */var* — как раз то место, где хранятся часто изменяемые данные: различные базы данных, буферные файлы, почта пользователей и пр. |
| */var/log* | Каталог */var/log* содержит файлы журналов с записями о различных действиях, выполнявшихся в системе. Они очень важны и должны проверяться время от времени. Наиболее полезным является файл */var/log/messages*. Обратите внимание, что из соображений безопасности некоторые системы требуют привилегий суперпользователя для просмотра файлов журналов |

**Символические ссылки**

Просматривая содержимое каталогов, нередко можно увидеть такие записи:

lrwxrwxrwx 1 root root 11 2012-08-11 07:34 libc.so.6 -> libc-2.6.so

Обратили внимание на первую букву l и на присутствие двух имен файлов в конце? Это специальный файл, который называется *символической ссылкой* (иногда их называют *мягкими ссылками* или, на жаргоне, *симлинками*). В большинстве Unix-подобных систем можно дать одному и тому же файлу несколько имен. Даже при том, что на данный момент ценность такого приема может быть не очевидна, в действительности это очень удобная возможность.

Вообразите следующий сценарий: программе требуется некий разделяемый ресурс (например, библиотека), хранящийся в файле с именем *foo*, но номер версии *foo* меняется очень часто. Было бы хорошо включить номер версии в имя файла, чтобы администратор или другое заинтересованное лицо могли видеть, какая версия *foo* установлена. И здесь возникает проблема. Если изменить имя разделяемого ресурса, нам придется проверять каждую программу, использующую этот ресурс, и изменять в ней имя ресурса на новое, после установки новой версии ресурса. Если честно, такая перспектива не выглядит привлекательной.

Символические ссылки помогут спасти положение. Допустим, мы установили *foo* версии 2.6 с именем файла *foo-2.6* и затем создали символическую ссылку с простым именем *foo*, указывающую на ресурс *foo-2.6*. То есть когда программа откроет файл *foo*, в действительности она откроет файл *foo-2.6*. И все будут счастливы. Программы, полагающиеся на имя *foo*, найдут нужный файл, а мы сможем увидеть фактическую версию ресурса. Когда придет время обновить ресурс до версии *foo-2.7*, мы просто добавим файл в систему, удалим символическую ссылку *foo* и создадим новую символическую ссылку, указывающую на новую версию. Такой подход не только решает проблему обновления версий, но также позволяет сохранить на компьютере обе версии ресурса. Представьте, что в версии *foo-2.7* обнаружилась ошибка (ох уж эти разработчики!) и нужно вернуть старую версию. В этом случае достаточно просто вновь удалить символическую ссылку, указывающую на новую версию, и создать новую символическую ссылку, указывающую на старую версию.

Запись выше (получена в каталоге */lib* в системе Fedora) соответствует символической ссылке с именем *libc.so.6*, указывающей на файл разделяемой библиотеки с именем *libc-2.6.so*. Это означает, что программа, ищущая *libc.so.6*, фактически получит файл *libc-2.6.so*. Как создавать символические ссылки, мы узнаем в следующей главе.

**жесткие ссылки**

Пока мы не ушли далеко от темы ссылок, нужно упомянуть, что существует второй тип ссылок, которые называют *жесткими ссылками* (hard link). Жесткие ссылки так же позволяют присвоить одному файлу несколько имен, но они действуют иначе. Подробнее о различиях между жесткими и символическими ссылками рассказывается в следующей главе.