Глава 11

Работа в режиме командной строки

В этой главе...

- Сыграем в оболочки
- Синтаксис и структура команд оболочки bash
- Запуск программ из командной оболочки
- Использование символов подстановки
- Работа с длинными командами
- Работа с переменными
- Перенаправление и каналы
- Настройка окружения командной оболочки
- Набор первой помощи

Чтобы уничтожить человека, компьютер сводит его с ума.
Аноним

Большинство программистов старой закалки прямо-таки влюблены в режим командной строки. Те же, кто привык водить мышью по красивым картинкам, считают командную строку чем-то "ископаемым" и пригодным только для занудных дедушек. Истина, как всегда, лежит посередине. Свои преимущества есть и у интерфейса командной строки (command line interface, CLI), и у графического интерфейса пользователя (graphical user interface, GUI). Искусство настоящего профессионала как раз и состоит в том, чтобы уметь воспользоваться нужным преимуществом в нужном месте. В любом случае для успешной работы в Linux необходимо понимать, что режим командной строки — это чрезвычайно мощное средство управления компьютером. Если вы когда-нибудь наблюдали за действиями опытного любителя Linux, то, должно быть, заметили, что сразу же после входа в систему он начинает выстукивать на клавиатуре десятки совершенно жутких команд и при этом страшно доволен полученными результатами.

В настоящей главе мы рассмотрим программу, обеспечивающую интерфейс командной строки для операционной системы Linux. Данная программа называется оболочкой `bash`. Существуют и другие оболочки для Linux, однако самой популярной является именно `bash`, и это далеко не случайно. Создателям `bash` удалось наделить свое творение всеми замечательными свойствами, которые есть у других оболочек.

Каждая оболочка использует собственные методы обработки команд и содержит собственный набор дополнительных средств. Вначале я объясню, что же такое командная оболочка, после чего вы будете готовы окунуться в изучение `bash`. Основное внимание будет уделено тому, что можно сделать с помощью некоторых наиболее популярных средств оболочки `bash`. Затем мы продолжим работу в командной строке и внедрение во внутреннее устройство `bash`.

Каждая оболочка укомплектована целым набором средств, предназначенных для выполнения определенных функций. Большинство этих средств добавлялись к оболочкам по мере того, как перед пользователями возникали все новые и новые задачи. Рассмотрение всех возможностей оболочки `bash` выходит за рамки данной главы, поэтому мы настоятельно рекомендуем вам прочитать страницу справочного

руководства man (электронной справочной системы Linux), посвященную оболочке bash. Кстати, это, пожалуй, наиболее полная и понятная страница справочного руководства man из всех существующих. О том, как работать со страницами справочного руководства man, будет рассказано далее в этой главе, в разделе “Помогите!”.

Сыграем в оболочки

Чтобы компьютер что-нибудь сделал, ему нужно сообщить, что от него требуется. В Linux один из способов общения с компьютером называется *командной оболочкой,* или просто *оболочкой (shell)*. Оболочка — это не графический интерфейс, а совокупность команд и синтаксических правил, которые позволяют выполнять необходимые задачи.

С точки зрения графического интерфейса командная оболочка до невозможности однообразна и скучна. Все, что в ней есть, — это краткое приглашение на ввод команды (например, знак $) и мигающий курсор. Далее я расскажу, как запустить командную оболочку, однако вначале несколько слов о bash.

По умолчанию в Linux используется оболочка bash. В ее основе лежит командная оболочка Unix, которая называлась *Bourne shell* или просто sh. Аббревиатура bash расшифровывается как *Bourne Again shell* (снова оболочка Bourne). Оболочка bash входит в состав большинства дистрибутивов Linux.

Чтобы запустить оболочку bash, щелкните на пиктограмме с изображением монитора на панели рабочего стола GNOME или выберите в главном меню команду System Tools⇨ Terminal (Управление системой⇨Терминал). В случае успеха на экране появится окно, изображенное на рис. 11.1.

Рис. 11.1. Рабочий стол GNOME с открытым экземпляром оболочки bash

Помимо символа $, приглашение командной строки может содержать кое-какую полезную информацию. К примеру, если вы работаете в Fedora Core 1 на компьютере deepthink и вошли в систему под именем пользователя evan, приглашение командной строки будет выглядеть следующим образом:

```
[evan@deepthink evan]$
```

Прежде чем переходить к изучению возможностей bash, следует сказать еще об одном методе запуска сеанса оболочки. Как показано на рис. 11.1, при вызове окна оболочки из графического режима оно открывается в пределах графического рабочего стола. Ну а что если вы хотите запустить сеанс оболочки, находясь в текстовом режиме?

Чтобы перейти в текстовый режим, нажмите комбинацию клавиш <Ctrl+Alt+F2>. Привычный графический интерфейс тут же исчезнет. Не волнуйтесь — он останется работать в фоновом режиме, и вы сможете вернуться к нему, как только захотите. Однако перед этим мы еще немного поговорим о том скучном экране, на который (как я смею надеяться) вы сейчас смотрите.

При переходе в текстовый режим перед вами появляется виртуальный терминал — один из нескольких виртуальных терминалов, имеющихся в системе по умолчанию. В большинстве случаев в нем будет содержаться надпись следующего характера:

```
deepthink login:
```

Введите имя пользователя и пароль. В случае успеха на экране появится сообщение о том, когда вы в последний раз входили в оболочку bash, а затем приглашение на ввод командной строки, которое в нашем случае будет выглядеть примерно следующим образом:

```
[evan@deepthink evan]$
```

Обратите внимание на сходство между вышеприведенным приглашением и тем, которое было в окне оболочки, оставленном нами на рабочем столе графического интерфейса. Ничего удивительного — ведь оба приглашения указывают на то, что вы запустили сеанс оболочки bash. Несмотря на это, данные приглашения относятся к разным *экземплярам* одной и той же программы. Другими словами, только что открытое окружение никак не связано с тем, которое было открыто на рабочем столе графического интерфейса.

Где я нахожусь?

Больше всего недоразумений при работе с оболочками вызывают постоянные входы и выходы. Каждый раз при входе в оболочку вы запускаете *экземпляр оболочки* — отдельную оболочку со своими параметрами. Это значит, что, войдя в оболочку из двух разных виртуальных терминалов, вы на самом деле запустите две оболочки. Если вы измените параметры одной оболочки, они не будут изменены в другой. Впрочем, если внесенные изменения постоянны, они будут применяться ко всем новым экземплярам оболочки, открытым после внесения изменений.

Хотите знать, куда девался графический интерфейс? Вначале успокойтесь (можете немного попрыгать или поприседать — хорошо помогает). Как мы уже говорили, существует несколько виртуальных терминалов. По умолчанию графический интерфейс расположен в виртуальном терминале номер 7. В данный момент вы находитесь в виртуальном терминале номер 2. Расставьте пальцы так, как будто сидите за пианино, и сыграйте аккорд <Ctrl+Alt+F7>. Пара секунд — и перед вами снова появится любимый графический интерфейс со все еще открытым окном экземпляра оболочки bash. Понятно? Впрочем, и это еще не все! Сеанс оболочки, который был открыт во втором окне терминала, тоже никуда не исчез; вы из него не выходили! Попробуйте нажать комбинацию клавиш <Ctrl+Alt+F2>. Ура — и этот экземпляр оболочки на месте! Вы можете перемещаться из второго терминала в седьмой и обратно, пока не надоест (а когда все-таки надоест, вспомните, что у Linux есть еще парочка терминалов, для перехода в которые используются комбинации клавиш <Ctrl+Alt+F1>, <Ctrl+Alt+F2> и т.д. до <Ctrl+Alt+F6>). Ну что, нравится?

Когда вам наскучит забавляться с виртуальными терминалами, выйдите из всех открытых терминалов и вернитесь в графический режим с открытым окном экземпляра оболочки. Сейчас вы узнаете, в чем же кроются причины всенародной любви к оболочке `bash`.

Синтаксис и структура команд оболочки bash

Многие пользователи живут и работают в Linux, даже не подозревая о том, какие возможности таит в себе оболочка `bash`. Тем самым они лишают себя множества интересных вещей. Чем больше вы знаете о том, как работает оболочка, тем больше вы сможете сделать с ее помощью.

Основная операция, которая выполняется при работе с оболочкой `bash`, — это ввод команд, их ключей и аргументов. К примеру, чтобы просмотреть список содержимого домашнего каталога в расширенном формате, включая и скрытые файлы, имена которых начинаются с точки (обычно это файлы настроек и каталоги файлов настроек), необходимо ввести команду `ls -la ~`. Кстати, обратите внимание на последний символ данной команды. Этот загадочный знак называется *тильдой*. В оболочке `bash` тильда используется для быстрого доступа к домашнему каталогу текущего пользователя. В нашем примере мы применили ее для просмотра содержимого домашнего каталога.

Каждую команду можно разбить на три компонента:

- ✓ имя команды;
- ✓ параметры, или флаги;
- ✓ аргументы.

Для большей наглядности рассмотрим пример.

Возьмем какую-нибудь простую команду — допустим, команду `du`, которая выводит на экран список файлов и подкаталогов текущего каталога, объем дискового пространства, который занимает каждый из этих элементов, а также общий объем дискового пространства, занятого каталогом. Вначале наберите саму команду `du` так, как показано ниже:

```
du
```

Команда, конечно же, хорошая, однако в результате ее выполнения у вас появится гораздо больше вопросов, чем ответов. Да, вы получили длиннющий список некоего содержимого, но *что* это за содержимое? Что значат эти цифры — байты, килобайты, или, может быть, количество посланий из другой галактики? Чтобы ответить на данный вопрос, к команде `du` достаточно добавить всего лишь один параметр:

```
du -h
```

Вы выполнили ту же самую команду, что и в прошлый раз, однако теперь вы добавили к ней дополнительные указания относительно того, каким образом следует вывести список содержимого каталога. Параметр `-h` указывает команде `du` на необходимость отобразить размер содержимого каталога в более читабельном формате. Теперь возле чисел стоят обозначения килобайт (`K`), мегабайт (`M`) и гигабайт (`G`), поэтому мы можем легко понять, насколько велики размеры элементов каталога. Но и это еще не все. Что если нас просто интересует, какой объем диска занимает текущий каталог со своими подкаталогами? Для этого к команде `du` следует добавить флаг `-s`:

```
du -s
```

А как узнать объем дискового пространства, занимаемый другим каталогом? Или только одним из подкаталогов текущего каталога? У меня в системе, к примеру, есть подкаталог `Music`. В нем находятся композиции, скопированные с компакт-дисков и преобразованные

в формат Ogg Vorbis (см. главу 15). Чтобы узнать, какой объем диска занимает подкаталог `Music`, и при этом отобразить данную информацию в читабельном формате (а не сидеть с калькулятором, переводя байты в гигабайты), можно воспользоваться следующей командой:

```
du -sh ~/Music
```

В данном примере `du` — это имя команды, `-sh` — параметры (флаги), а `~/Music` — аргумент. К флагу `-l` можно добавить еще несколько флагов, и каждый из них будет указывать на необходимость выполнения командой определенных действий.

А как же узнать, какие параметры и аргументы есть у конкретной команды? Можно воспользоваться двумя источниками. Во-первых, это страницы справочного руководства man, которые будут рассмотрены в разделе "Помогите!". Во-вторых, практически у каждой команды есть параметр `--help`, который выводит на экран список всех ее параметров. Впрочем, последний способ хорош тогда, когда вы уже знаете, какие параметры есть у команды, только не можете вспомнить, как они выглядят. Чтобы просмотреть список параметров команды `ls`, выполните следующее:

```
du --help
```

Ну как, впечатляет?

Запуск программ из командной оболочки

Наиболее очевидное (хотя, возможно, и не столь явное) предназначение командной оболочки — запуск других программ. Большинство служебных средств, используемых в Linux, в действительности являются отдельными и самостоятельными программами. Пользователю нужен метод, чтобы запускать эти программы. В графическом окружении все делается очень просто — вы щелкаете на нужном ярлыке или выбираете команду меню, а система сама выполняет все шаги, необходимые для запуска соответствующего приложения. Обратите внимание, что для запуска программ система использует информацию из переменных окружения, являющихся частью командной оболочки. (О переменных окружения мы поговорим несколько позднее.) По этой причине графический интерфейс зачастую вызывает программы через оболочку `bash`. Таким образом, как вы видите, даже графическому интерфейсу нужна командная оболочка, хоть он это и скрывает.

Попробуйте, к примеру, перейти в окно терминала и выполнить в нем следующую команду:

```
mahjongg
```

Через несколько секунд на экране появится игра Mahjongg. Точно так же из окна оболочки можно запускать и другие программы рабочего стола GNOME, если вы знаете имена этих программ. Обратите внимание, что попытка запуска некоторых программ из виртуального терминала (<Alt+F1>) может привести к ошибке. Отдельные приложения могут работать только в графическом окружении, которого у символьного терминала нет и быть не может.

Использование символов подстановки

Компьютерная жизнь была бы невероятно скучна, если бы одну и ту же команду приходилось повторять по тысяче раз. В конце концов, разве компьютеры не были созданы для того, чтобы выполнять повторяющиеся, рутинные задачи? На помощь приходят *символы подстановки (wildcards)*, благодаря которым одна команда может быть применена сразу к нескольким файлам. Наиболее употребляемые символы подстановки — это "звездочка" (`*`) и знак вопроса (`?`), которые используются для замены в тексте команды имени или части имени файла. К примеру, чтобы просмотреть список всех файлов текущего каталога, имеющих расширение `.doc`, можно воспользоваться следующей командой:

```
ls -l *.doc
```

Полученный список может включать в себя файлы `resume.doc`, `cover_letter.doc`, `to_editor.doc` и т.д. (Более подробно о символах подстановки рассказывалось в главе 10.)

В отличие от Microsoft Windows, Linux не обращает внимания на расширения файлов, такие как `.doc`, `.exe` или `.txt`. К примеру, точка в расширении `.doc` интерпретируется как обычный символ имени файла. Тем не менее общепринятое соглашение о расширениях очень удобно, потому что позволяет с ходу определить тип файла. Просто запомните, что, в отличие от файловой системы Windows, соблюдение этого соглашения в Linux не является обязательным.

Работа с длинными командами

Как только вы немного поработаете с интерфейсом командной строки, вам, конечно же, захочется сделать его более удобным. В данном разделе я расскажу о некоторых средствах оболочки `bash`, призванных максимально облегчить вашу работу в режиме командной строки. Это такие средства, как автоматическое завершение командной строки, возможность редактирования, а также сохранение списка предыдущих выполненных команд.

Автоматическое завершение команд и имен файлов

Как вы догадываетесь, в режиме командной строки придется набирать гораздо больше текста, чем при работе с графическим интерфейсом. Поэтому просто незаменимым является такое средство, как автоматическое завершение. Это функция оболочки, которая дает возможность автоматически завершать имена файлов и (или) системные команды после введения нескольких первых символов.

Файловая система Linux позволяет работать с именами какой угодно длины. Это значит, что имена некоторых файлов могут достигать гигантских размеров. Набирать такие длинные имена не слишком-то приятно. К счастью, использование средств автозавершения и редактирования делает набор длинных команд куда менее обременительным.

Средство автозавершения можно использовать в двух ситуациях — для ввода команды или для завершения имени файла.

Автозавершение команды

Представьте себе, что вы хотите выполнить команду, но помните только то, что она начинается с букв `up` и показывает, сколько времени прошло с момента последней перезагрузки системы. Наберите в командной строке слово `up` и нажмите клавишу <Tab>:

```
[evan@deepthink evan]$ up[TAB]
```

После этого возможен один из двух вариантов действий.

- ✓ Если в *области поиска* (каталоги, в которых находятся программы) с букв `up` начинается только одна команда, система автоматически завершит командную строку и будет ожидать, когда вы нажмете клавишу <Enter> для выполнения команды.
- ✓ Если вы услышите звуковой сигнал, это значит, что с `up` начинаются несколько команд. Тогда следует нажать клавишу <Tab> во второй раз — и на экране появится список всех команд, которые начинаются с `up`. Найдите в списке нужную команду и наберите еще несколько символов ее имени, чтобы набранная последовательность символов однозначно идентифицировала данную команду. После этого еще раз нажмите клавишу <Tab>, и система автоматически завершит команду.

Автозавершение имени файла

Средство автозавершения можно использовать не только для команд, но и для имен файлов. Чтобы набрать в командной строке имя файла, введите несколько первых символов этого имени и нажмите клавишу <Tab>. Система просмотрит текущий каталог в поисках подходящих имен файлов и автоматически завершит имя файла в командной строке. Данное средство работает точно так же, как и автозавершение команд — если под набранную последовательность символов подходит несколько имен файлов, вы услышите звуковой сигнал. В этом случае вам будет необходимо нажать клавишу <Tab>, чтобы увидеть список всех подходящих имен.

Разумеется, к использованию средства автозавершения нужно привыкнуть. Впрочем, уже через несколько дней вы будете с удивлением думать, как вы вообще могли без него обходиться.

Список предыдущих выполненных команд

Нравится вам это или нет, но командная оболочка запоминает все ваши действия. Впрочем, большинству пользователей это все-таки нравится, потому что избавляет от необходимости еще раз вводить эти кошмарные, длиннющие команды, которые с таким трудом были набраны несколько часов, а то и несколько дней, назад. Для большей наглядности рассмотрим пример. Представьте себе, что вчера вы выполнили команду, которая находит и удаляет все файлы *дампа оперативной памяти* (это огромные файлы, которые содержат отладочные данные, понятные только компьютеру или, в крайнем случае, эксперту в области программирования). Это выглядело примерно следующим образом:

```
find / -name core -exec rm {} \;
```

Чтобы снова выполнить данную команду, ее необязательно набирать заново — достаточно найти ее в списке предыдущих выполненных команд и запустить еще раз. Для этого нажимайте клавишу со стрелкой, указывающей вверх, пока в командной строке не появится нужная команда. Затем нажмите клавишу <Enter>, и команда будет выполнена. Вот и все!

Для доступа к списку предыдущих выполненных команд можно воспользоваться двумя способами.

- ✓ **Нажмите клавишу со стрелкой, указывающей вверх.** Это самый простой способ просмотреть список предыдущих выполненных команд. Каждое нажатие данной клавиши прокручивает список недавно набранных команд в обратном порядке. Как только вы дойдете до нужной команды, нажмите клавишу <Enter>, и команда будет выполнена.
- ✓ **Выполните команду** `history`**.** Эта команда выводит на экран список 20 последних выполненных команд.

Создание псевдонимов для команд

Представьте себе программиста старой закалки, который всю жизнь проработал в MS-DOS и не мыслит своего существования без команды `dir`. (Напомним, что в операционной системе MS-DOS команда `dir` применяется для просмотра списка содержимого текущего каталога.) Такому человеку очень сложно переучиться для работы с командами оболочки `bash`. Благодаря использованию псевдонимов вы можете создать новую команду, которая будет носить привычное имя и выполнять похожие действия. К примеру, чтобы создать псевдоним для команды, выводящей список содержимого каталога в расширенном формате (примерно так же, как это делала команда `dir`), необходимо выполнить следующую команду:

```
[evan@deepthink evan]$ alias dir='ls -l'
```

Теперь, выполнив команду `dir`, вы получите список содержимого текущего каталога в расширенном формате.

Наличие псевдонимов позволяет назначить короткие имена длинным часто используемым командам. Чтобы узнать, какие псевдонимы были созданы в Fedora Core по умолчанию, выполните команду `alias` без аргументов. Полученный список будет включать в себя несколько стандартных псевдонимов, а также только что созданный псевдоним `dir`.

Обратите внимание, что при выходе из оболочки или запуске нового экземпляра оболочки созданные псевдонимы уничтожаются. Позднее мы расскажем, как добавить псевдонимы и другие команды оболочки в загрузочный файл `bash`, который запускается при открытии каждого нового экземпляра `bash`. (См. раздел "Настройка окружения командной оболочки" далее в этой главе.)

Работа с переменными

Переменные (variables) оболочки `bash` — это слова или текстовые строки, используемые компьютерами для представления данных. В качестве примера использования переменной можно представить себе переменную под именем `fruit` (фрукт), которой присвоено значение `apple` (яблоко). Существует целый ряд переменных, в которых хранится информация о параметрах учетной записи и рабочего окружения.

Обычные переменные и переменные окружения

В оболочке `bash` используются переменные двух типов.

- ✓ **Переменные (variables).** Обычная переменная используется программой или сеансом командной оболочки, однако она и ее значение доступны только данной программе или данному сеансу оболочки.
- ✓ **Переменные окружения (environment variables).** Переменная окружения также используется программой или сеансом командной оболочки, однако значение этой переменной копируется во все другие программы или оболочки, запущенные в данном окружении.

Переменную окружения оболочки `bash` легко отличить от обычной переменной по ее имени. Имена переменных окружения пишутся прописными буквами, тогда как для обычных переменных используются как прописные, так и строчные буквы.

Основные переменные окружения

Оболочка `bash` использует большое количество переменных окружения. Вы будете поражены тем количеством информации, которая в них содержится. Что самое приятное, значение переменной окружения можно изменить так, как вам этого хочется! Список основных переменных окружения оболочки `bash` приведен в табл. 11.1.

В большинстве случаев значения переменных окружения задаются системным администратором или самой оболочкой. Как правило, переменные окружения используются программами для сбора системной информации и обычные пользователи с ними не сталкиваются. Впрочем, при необходимости значение переменной окружения можно изменить. К примеру, переменная окружения `HISTSIZE` (см. табл. 11.1) определяет размер списка предыдущих выполненных команд. Помните, мы говорили о том, как еще раз выполнить команду, которую вы набрали вчера? (Если нет, вернитесь к разделу этой главы "Список предыдущих выполненных команд".) Если вы назначите переменной окружения `HISTSIZE` большее значение, вы сможете увеличить список выполненных команд, который будет храниться в памяти оболочки `bash`.

Таблица 11.1. Основные переменные окружения оболочки *bash*

Переменная окружения	Описание	Значение
HISTSIZE	Задает размер списка последних набранных команд	Количество команд
HOME	Задает местонахождение домашнего каталога текущего пользователя	Путь к домашнему каталогу
MAILCHECK	Задает периодичность проверки почтового ящика на предмет наличия новых сообщений. Если будут обнаружены новые сообщения, как только вы начнете работать в командной строке, на экране появится сообщение наподобие `You have new mail` (Вам пришло новое сообщение)	Количество секунд между проверками
PATH	Задает список и порядок просмотра каталогов, в которых оболочка `bash` будет искать программу, указанную пользователем в командной строке	Список каталогов, разделенных двоеточиями
PS1	Определяет вид приглашения командной строки	Последовательность команд и атрибутов форматирования, используемых для формирования приглашения командной строки

Присваивание значений переменным окружения и извлечение их значений

Чтобы присвоить переменной значение, необходимо ввести имя переменной, затем поставить знак "равно" (=) и указать значение, которое следует присвоить. Полученная запись должна выглядеть следующим образом:

```
Переменная=Значение
```

Чтобы извлечь значение переменной окружения, необходимо поставить перед ее именем знак доллара ($). Давайте посмотрим, какое значение имеет переменная окружения, определяющая внешний вид приглашения командной строки. Как вы знаете, эта переменная носит имя PS1. Чтобы просмотреть ее содержимое, выполните такую команду:

```
echo $PS1
```

На экране появится нечто наподобие следующего:

```
[\u@\h \W]\$
```

Каждый из символов, стоящих после обратной косой черты, представляет собой атрибут форматирования — специальную инструкцию, указывающую, какую информацию нужно включить в приглашение командной строки. Список атрибутов, используемых для форматирования приглашения командной строки, приведен в табл. 11.2.

Таблица 11.2. Атрибуты форматирования, используемые в значении переменной окружения *PS1*

Атрибут	Описание
\!	Выводит номер команды в списке предыдущих выполненных команд
\#	Выводит количество команд, которые были выполнены в текущем сеансе оболочки
\$	Выводит знак $ для обычного пользователя или знак # для суперпользователя
\d	Выводит текущую дату в формате *день_недели месяц число*

Окончание табл. 11.2

Атрибут	Описание
\h	Выводит имя машины, на которой работает текущий пользователь
\n	Осуществляет переход на следующую строку
\s	Выводит слово bash при работе в командной оболочке bash
\t	Выводит время в 24-часовом формате
\u	Выводит имя пользователя
\w	Выводит текущий каталог (только подкаталог последнего уровня)
\W	Выводит полный адрес текущего каталога

Однако вернемся к нашему примеру. Чтобы заменить стандартное приглашение оболочки необычным, введите следующую команду:

```
PS1='Hello \u, what can I do for you? => '
```

Обратите внимание на одинарные кавычки (о них мы поговорим немного позднее). Теперь каждый раз при нажатии клавиши <Enter> на экране будет появляться гораздо более дружелюбное приглашение, чем раньше. Не волнуйтесь, если стандартное приглашение вам нравилось больше, вы можете присвоить переменной PS1 ее исходное значение или же просто выйти из оболочки и снова войти в нее — исходное приглашение будет восстановлено.

Может быть, вы хотите узнать, какие еще переменные есть в запасе у вашей системы? Чтобы получить список всех переменных текущего окружения, выполните команду env. Вообще-то большинству из вас никогда не придется изменять значения переменных окружения в командной строке. Тем не менее по мере приобретения опыта работы в Linux вам может захотеться познакомиться с возможностями программирования для оболочки bash. В этом случае вы обязательно столкнетесь с изменением значений переменных окружения, как и в любом другом языке программирования.

Так что там с одинарными кавычками? Изменяя значения переменных окружения, необходимо учитывать некоторые тонкости синтаксиса. Присвоить переменной окружения простое числовое значение не составляет никакого труда: например, HISTSIZE=250. Тем не менее если значение переменной должно содержать пробелы, его нужно взять в одинарные или двойные кавычки. Тип кавычек зависит от способа обработки значения переменной.

Чтобы значение переменной окружения отображалось *в точности так же*, как вы его задали, возьмите его в одинарные кавычки. Такое значение называется *литеральной строкой*. Попробуйте, к примеру, выполнить следующую команду и посмотрите, какая фраза будет возвращена системой:

```
echo 'Hello, my name is $USER'
```

Довольно глупо, не так ли? Впрочем, не стоит спешить с выводами. Существует еще один тип строк, который обрабатывается оболочкой совершенно по-другому, — интерполированные строки. *Интерполированная* строка содержит специальные атрибуты. Прежде чем обрабатывать интерполированную строку, командная оболочка интерпретирует содержащиеся в ней атрибуты, заменяя их необходимыми значениями. Попробуйте выполнить ту же команду, что и в предыдущем примере, заменив одинарные кавычки двойными:

```
echo "Hello, my name is $USER"
```

Обратите внимание на полученный результат. Вместо того чтобы вывести на экран строку Hello, my name is $USER, система заменила имя переменной окружения, отмеченное знаком доллара, реальным значением этой переменной.

Если вы собираетесь серьезно заняться настройкой переменных окружения, рекомендуем начать с вышеперечисленных методов. Если вам понравятся сделанные изменения, вы

сможете внести их в файл ~/.bash_profile. После этого выполненные вами настройки станут постоянными. Более подробно о том, как сделать изменения переменных окружения постоянными, будет рассказано немного позднее.

Почему в примере с переменной PS1 были использованы одинарные кавычки? Элементы, которым предшествует обратная косая черта (\), *интерпретируются* оболочкой и в том, и в другом случае. Если бы, однако, мы использовали двойные кавычки, элемент \u был бы интерпретирован только единожды, поэтому приглашение командной строки изменилось бы только в первый раз. В отличие от этого, значения переменных, взятые в одинарные кавычки, интерпретируются каждый раз.

Даже если вы еще не совсем разобрались в вышеизложенном материале, не стоит отчаиваться. По мере приобретения опыта работы в Linux вы познакомитесь со *сценариями оболочки*. Написание сценариев оболочки — это искусство создания настоящих компьютерных программ с помощью средств оболочки. Большинство системных администраторов, работающих с Unix и Linux, говорят на языке сценариев оболочки так же свободно, как мы с вами говорим на своих родных языках.

Перенаправление и каналы

Для управления потоком информации в командной оболочке служат перенаправление и каналы. *Канал (pipe)* перенаправляет результаты выполнения одной команды на вход другой команды. Наличие каналов позволяет соединить несколько служебных команд в своеобразную "цепочку". В конце этой цепочки может осуществляться перенаправление результатов выполнения последней команды на другое устройство или в файл.

Практически все служебные программы Linux, имеющие дело с операциями ввода и вывода, оснащены стандартными интерфейсами stdin (стандартный ввод), stdout (стандартный вывод) и stderr (стандартная обработка ошибок). Наличие общего метода считывания и передачи данных позволяет "связывать" служебные программы в более солидные решения.

Перенаправление результатов выполнения команды

Перенаправление результатов выполнения команды (output redirection) — это наиболее популярный метод управления потоком информации. Одним из примеров перенаправления является отправка результатов выполнения команды не на экран (как это обычно происходит), а в файл. Для начала наберите нашу любимую команду ls -la ~ и нажмите клавишу <Enter>. На экране появится нечто наподобие следующего:

```
total 20
drwx------ 2 sue  users 4096 Oct 30 07:48 .
drwxr-xr-x 5 root root  4096 Oct 30 11:57 ..
-rw-r----- 1 sue  users  24  Oct 30 06:50 .bash_logout
-rw-r----- 1 sue  users 230  Oct 30 06:50 .bash_profile
-rw-r----- 1 sue  users 124  Oct 30 06:50 .bashrc
-rw-rw-r-- 1 sue  users   0  Jan  2 07:48  wishlist
```

Хотите, чтобы эта информация была записана в файл? Тогда воспользуйтесь оператором перенаправления >. Этот оператор указывает, что результат выполнения команды нужно не выводить на экран, а отправить в указанный файл. К примеру, чтобы перенаправить список содержимого домашнего каталога в файл с именем listing, необходимо выполнить следующую команду:

```
ls -la ~ > listing
```

Обратите внимание, что при выполнении этой команды на экране ничего не появилось. Так и должно быть — ведь вся информация была перенаправлена в файл `listing`. Чтобы убедиться в правильности перенаправления, выполните следующую команду:

```
cat listing
```

Команда `cat` отобразит на экране содержимое файла `listing`.

Если вы еще раз выполните команду `ls -la ~ > listing`, содержимое файла `listing` будет перезаписано, т.е. заменено новой информацией. Этого можно избежать, используя оператор перенаправления >>. В данном случае результат выполнения команды будет дописан в конец указанного файла. Таким образом, если сразу же после выполнения предыдущей команды вы выполните команду `ls -la ~ >> listing`, содержимое файла `listing` будет выглядеть примерно так:

```
total 20
drwx------ 2 sue  users 4096 Oct 30 07:48 .
drwxr-xr-x 5 root root  4096 Oct 30 11:57 ..
-rw-r----- 1 sue  users  24  Oct 30 06:50 .bash_logout
-rw-r----- 1 sue  users 230  Oct 30 06:50 .bash_profile
-rw-r----- 1 sue  users 124  Oct 30 06:50 .bashrc
-rw-rw-r-- 1 sue  users   0  Jan 2 07:48   wishlist
total 20
drwx------ 2 sue  users 4096 Oct 30 07:48 .
drwxr-xr-x 5 root root  4096 Oct 30 11:57 ..
-rw-r----- 1 sue  users  24  Oct 30 06:50 .bash_logout
-rw-r----- 1 sue  users 230  Oct 30 06:50 .bash_profile
-rw-r----- 1 sue  users 124  Oct 30 06:50 .bashrc
-rw-rw-r-- 1 sue  users   0  Jan 2 07:48   wishlist
```

Прокладываем каналы

Еще один оператор оболочки `bash` позволяет связывать команды так, что результат выполнения одной команды становится аргументом следующей команды. Этот оператор называется *каналом (pipe)*. Рассмотрим небольшой пример. Предположим, вы хотите просмотреть список всех файлов каталога `/etc` в расширенном формате. Если вы прибегнете к помощи команды `ls -la /etc`, на экране пробежит длиннющий список файлов, б*о*льшая часть которого моментально скроется из виду. Разумеется, вы сможете вернуться назад с помощью комбинации клавиш <Shift+Page Up>, однако даже в этом случае вы, скорее всего, не увидите весь список.

Чтобы сделать это, воспользуйтесь одним из двух способов.

- ✓ Список содержимого каталога `/etc` можно перенаправить в файл посредством команды `ls -la /etc > ~/etclisting` (или что-то вроде того). После этого вы сможете просмотреть содержимое файла `~/etclisting` с помощью любимого текстового редактора.
- ✓ Результат выполнения команды `ls -la /etc` можно перенаправить на вход команды `more` посредством канала. Команда `more` предназначена для постраничного вывода информации на экран. Чтобы перейти к просмотру следующей страницы, необходимо нажать клавишу пробела. Многие пользователи Linux находят эту команду чрезвычайно полезной.

Чтобы перенаправить результат выполнения команды `ls -la` на вход команды `more` с помощью канала, выполните команду `ls -la` *`путь_к_каталогу`* `|` `more`, где *`путь_к_каталогу`* — интересующий вас каталог. Символ `|` (на клавиатуре он обозначен не одной, а двумя расположенными друг над другом вертикальными черточками) — это и есть канал, который указывает на необходимость перенаправить результат выполнения команды `ls -la` на вход команды `more`.

Как уже говорилось, команда `more` предназначена для постраничного вывода информации. Существует еще одно средство, которое действует аналогично команде `more`, однако обладает гораздо большими возможностями. Как ни странно, оно называется `less`[1]. Команда `less` позволяет прокручивать информацию как вперед, так и назад, а также выполняет поиск по ключевым словам. Таким образом, в вышеприведенном примере вместо команды `more` можно использовать `less`. Как видите, Linux подтверждает поговорку "лучше меньше, да лучше".

Настройка окружения командной оболочки

Параметры окружения `bash` хранятся в нескольких файлах настроек. Каждый из них является простым текстовым файлом и может быть открыт в любом текстовом редакторе. Для настройки окружения командной оболочки достаточно изменить или добавить параметры в следующих файлах.

- ✓ `/etc/profile`. Глобальные параметры окружения, действительные для всех пользователей. При входе пользователя в систему этот файл считывается первым.
- ✓ `.bash_profile`. Параметры окружения, действительные для учетной записи конкретного пользователя. Этот файл считывается после файла `/etc/profile`.
- ✓ `.bashrc`. Файл настроек, который считывается при создании каждого экземпляра *вложенной оболочки (subshell)*. Вложенная оболочка вызывается каждый раз при выполнении новой программы. Данный файл всего лишь обеспечивает еще один уровень настроек оболочки для этого события.
- ✓ `.bash_logout`. Этот файл считывается при выходе пользователя из сеанса оболочки `bash`. Он очень удобен для автоматического удаления временных файлов или выполнения другой рутинной работы, необходимой для аккуратного завершения сеанса оболочки.

Следует отметить, что последние три файла хранятся в домашнем каталоге соответствующего пользователя и являются скрытыми (на это указывает точка в начале имени каждого из них).

Если вас то и дело тянет поэкспериментировать с вышеперечисленными файлами, создайте отдельную учетную запись, чтобы спокойно изменять параметры оболочки, не затрагивая собственное окружение. Этот совет в особенности касается файла `/etc/profile`. С его помощью можно повредить все учетные записи, имеющиеся в системе! Во избежание неприятностей создайте резервную копию этого файла с помощью команды `cp /etc/profile /etc/profile.original`. Затем отредактируйте исходный файл `/etc/profile`. В случае чего вы всегда сможете удалить его с помощью команды `rm`, а затем воспользоваться командой `mv`, чтобы переименовать `/etc/profile.original` в `/etc/profile`.

Набор первой помощи

Теперь, когда вы знаете, что такое командная строка и как в ней работать, пришло время познакомиться с несколькими полезными средствами, которые помогут вам разобраться в запутанных ситуациях.

[1] *"More" означает "больше", "less" — "меньше".* — Прим. ред.

Я потерялся

Хотите знать, под каким именем вы вошли в систему в последний раз? Или, предположим, вы подошли к пустующему терминалу и хотели бы знать, что это за машина и кто на ней работает? Для получения информации о том, кто вы и где вы находитесь, можно воспользоваться следующими командами.

- ✓ `whoami`. Эта команда определяет, какой пользователь работает в текущем сеансе оболочки.
- ✓ `who`. Данная команда аналогична команде `whoami`, однако в отличие от последней она определяет не только пользователя, работающего в текущем сеансе оболочки, но и текущий терминал и дату открытия сеанса, а также всех других пользователей, вошедших в систему.
- ✓ `uname`. Эта команда предоставляет разнообразные сведения о системе. К примеру, выполнив команду `uname` с параметром `-a`, вы получите информацию о версии ядра и времени его последней компиляции.
- ✓ `pwd`. Знание того, кто вы, — это лишь один ключ к разгадке тайны. Для полноты ощущений необходимо также знать, где вы находитесь. Как было рассказано в главе 10, навигация в Linux в большинстве случаев связана с перемещением по файлам и каталогам. Если только ваша душа не путешествует в параллельных мирах, в каждый конкретный момент вы можете находиться лишь в одном месте файловой системы. В Linux это место называется текущим рабочим каталогом. Чтобы определить текущий рабочий каталог, выполните команду `pwd`.

Помогите!

Справочное руководство man (man pages) — это электронная справочная система (*man* означает *manual*, т.е. "руководство"), представляющая собой подробное руководство по применению команд и других элементов операционной системы Linux. Помимо всего прочего, справочное руководство man включает в себя параметры команд, форматы файлов и сведения об использовании функций различных программ.

Чтобы просмотреть страницу справочного руководства man, посвященную конкретной команде, выполните команду `man <имя_команды>`.

Ну а что делать, если вы не знаете имя команды, которую ищете, или же просто хотите побольше узнать о сам*о*м справочном руководстве man? Ответ прост — наберите команду `man man`!

Контроль за использованием памяти и дискового пространства

Особо любознательные умы всегда волновала проблема использования памяти и дискового пространства. Команда `df` возвращает список всех подключенных разделов, а также сведения об используемом и свободном дисковом пространстве в каждом из этих разделов. Как и во многих других командах, чтобы получить полный список параметров команды `df`, достаточно набрать `df --help`. К примеру, вы можете узнать, какой параметр команды `df` позволяет отображать сведения о размере дискового пространства в байтах, а не в блоках (раскроем тайну — это параметр `-h`).

Это прекрасно, но что если у вас заканчивается дисковое пространство, и вы хотели бы узнать, какие файлы можно удалить или хотя бы сколько места на диске они занимают? В таком случае вам пригодится команда `du` (сокращение от *disk usage* — использование диска). Если команда `du` выполняется без параметров, она возвращает количество блоков, занятых текущим рабочим

каталогом. Если же в качестве аргумента команды `du` указать путь к каталогу, команда отобразит сведения о количестве блоков, занятых этим каталогом, а также всеми его подкаталогами.

Не думайте, что я хочу ошеломить вас количеством информации, но если вы наберете команду `top`, то получите сведения о распределении системных ресурсов (в частности, памяти) между запущенными процессами. Обратите внимание, что эта информация ежесекундно обновляется. Когда вам надоест следить за выполнением команды `top`, просто нажмите клавишу <Q>, и работа команды будет завершена.

Очистка экрана

Работая в оболочке `bash`, очень удобно использовать команды `clear` и `reset`. Команда `clear` очищает экран от ненужной информации. Не волнуйтесь — она не удаляет файлы и не изменяет параметры окружения; она просто "наводит порядок", чтобы вы снова могли захламлять экран результатами своих исследований.

Команда `reset` более интересна. Попробуйте просмотреть содержимое двоичного файла с помощью команды `cat`. После того как компьютер закончит выводить результат применения команды `cat` к двоичному файлу, вам даже, может быть, удастся разобрать, что написано в приглашении командной строки. К сожалению, вероятнее всего, после выполнения команды `cat` приглашение превратится в набор непонятных символов, а то и вовсе в пустые квадратики. Более того, что бы вы ни попытались набрать в командной строке, это также будет выглядеть как непонятные символы или пустые квадратики. Чтобы вернуться к нормальному состоянию, наберите команду `reset` и нажмите клавишу <Enter>. Конечно же, в силу сложившихся обстоятельств набранное вами слово `reset` будет выглядеть вовсе не как `reset`, однако компьютер все равно поймет, чего вы хотите, и через несколько секунд восстановит исходный язык вашего окружения.

Что здесь происходит?

Существует еще несколько команд, позволяющих отслеживать состояние системы.

- ✓ `ps`. В отличие от упоминавшейся команды `top`, которая позволяет *наблюдать* за происходящим в системе, существует еще одна команда, которая позволяет получить картину происходящего *на момент времени*. Команда `ps` возвращает статистику по процессам, запущенным в окружении текущего пользователя, а при желании — и по всем процессам, запущенным в системе.
- ✓ `kill`. Одно из преимуществ использования виртуальных терминалов и нескольких сеансов оболочки — это возможность незаметно подкрасться и "зарубить" процесс, который начал делать что-то не то. (Только не подумайте, что мы такие кровожадные!) Команда `kill` обычно выполняется со следующими параметрами:

  `kill -`*`сигнал идентификатор_процесса`*

  Параметр -*сигнал* — это число, соответствующее типу сообщения, которое нужно отправить уничтожаемому процессу. Далеко не все сообщения направлены на непосредственное уничтожение процесса — некоторые просто требуют, чтобы процесс завершил свою работу и освободил используемые ресурсы ядра. Самый "жестокий" сигнал — `-9`. В этом случае процесс уничтожается независимо от того, что он выполнял.
- ✓ `killall`. Данная команда аналогична команде `kill`. Она позволяет уничтожить процесс, запущенный заданной командой. К примеру, однажды автору этой книги пришлось несколько раз выполнить команду `killall netscape`, потому что ошибки, имеющиеся в программе Netscape, привели к неконтролируемому размножению ее запущенных экземпляров. К счастью, с появлением обозревателя Mozilla стало возможным устранение этих досадных оплошностей.