**11. Окружение**

Как обсуждалось выше, командная оболочка на протяжении всего сеанса работы использует массу информации, которая называется *окружением*. Данные, хранящиеся в окружении, используются программами для выяснения деталей конфигурации.

Даже при том, что для хранения своих настроек большинство программ использует конфигурационные файлы, некоторые программы также учитывают значения, хранящиеся в окружении. Зная это, можно использовать окружение для настройки некоторых параметров командной оболочки.

В этой главе мы будем работать со следующими командами:

• printenv — выводит часть или все окружение.

• set — устанавливает параметры командной оболочки.

• export — экспортирует окружение для программ, которые будут выполняться позднее.

• alias — создает псевдоним команды.

**Что хранится в окружении?**

Командная оболочка хранит в окружении данные двух основных типов, хотя bash практически не делает различий между типами. Эти данные хранятся в *переменных окружения* и в *переменных командной оболочки*. Переменные командной оболочки — это фрагменты данных, инициализируемые командой bash, а переменные окружения — практически все остальное. Помимо переменных командная оболочка хранит также программируемые данные, а именно *псевдонимы* и *функции командной оболочки*. Мы уже познакомились с псевдонимами в главе 5, а о функциях (которые имеют отношение к сценариям командной оболочки) поговорим в части IV книги.

**Исследование окружения**

Увидеть, что хранится в окружении, можно при помощи встроенной в bash коман­ды set или программы printenv. Команда set выводит переменные обоих видов — командной оболочки и окружения, — тогда как printenv выводит только последние. Так как список содержимого окружения очень велик, его лучше просматривать, передавая вывод любой из команд по конвейеру в less:

[me@linuxbox ~]$ **printenv | less**

Запустив эту команду, вы должны увидеть нечто похожее:

KDE\_MULTIHEAD=false

SSH\_AGENT\_PID=6666

HOSTNAME=linuxbox

GPG\_AGENT\_INFO=/tmp/gpg-PdOt7g/S.gpg-agent:6689:1

SHELL=/bin/bash

TERM=xterm

XDG\_MENU\_PREFIX=kde-

HISTSIZE=1000

XDG\_SESSION\_COOKIE=6d7b05c65846c3eaf3101b0046bd2b00-1208521990.996705-1177056199

GTK2\_RC\_FILES=/etc/gtk-2.0/gtkrc:/home/me/.gtkrc-2.0:/home/me/.kde/share/  
              config/gtkrc-2.0

GTK\_RC\_FILES=/etc/gtk/gtkrc:/home/me/.gtkrc:/home/me/.kde/share/config/gtkrc

GS\_LIB=/home/me/.fonts

WINDOWID=29360136

QTDIR=/usr/lib/qt-3.3

QTINC=/usr/lib/qt-3.3/include

KDE\_FULL\_SESSION=true

USER=me

LS\_COLORS=no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35:bd=40;33;01:  
         cd=40;33;01:or=01;05;37;41:mi=01;05;37;41:ex=00;32:\*.cmd=00;32:\*.exe:

Это список переменных окружения с их значениями. Например, в списке можно увидеть переменную с именем USER, содержащую значение me. Команда printenv может также вывести значение конкретной переменной:

[me@linuxbox ~]$ **printenv USER**

me

Команда set при вызове без параметров и аргументов выводит переменные обоих типов — командной оболочки и окружения, — а также все объявленные функции командной оболочки.

[me@linuxbox ~]$ **set | less**

В отличие от printenv она сортирует вывод в алфавитном порядке.

Получить значение единственной переменной можно также с помощью команды echo, например:

[me@linuxbox ~]$ **echo $HOME**

/home/me

Единственный элемент окружения, который не выводится командами set и printenv, это псевдонимы. Чтобы вывести список псевдонимов, используйте коман­ду alias без аргументов:

[me@linuxbox ~]$ **alias**

alias l.='ls -d .\* --color=tty'

alias ll='ls -l --color=tty'

alias ls='ls --color=tty'

alias vi='vim'

alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --showtilde'

**Некоторые интересные переменные**

Окружение содержит довольно много переменных, и хотя ваше окружение может отличаться от представленного здесь, вы почти наверняка увидите у себя переменные, перечисленные в табл. 11.1.

**Таблица 11.1. Переменные окружения**

| **Переменная** | **Содержит** |
| --- | --- |
| DISPLAY | Имя вашего дисплея, если вы работаете в графическом окружении. Обычно это :0, что означает первый дисплей, сгенерированный X сервером |
| EDITOR | Имя программы, используемой в качестве текстового редактора |
| SHELL | Имя программы командной оболочки |
| HOME | Путь к домашнему каталогу |
| LANG | Определяет набор символов и порядок сортировки для вашего языка |
| OLD\_PWD | Предыдущий рабочий каталог |
| PAGER | Имя программы для постраничного просмотра. Часто имеет значение */usr/bin/less* |
| PATH | Список каталогов, разделенных двоеточием, в которых производится поиск выполняемых программ по их именам |
| PS1 | Строка приглашения к вводу № 1. Определяет содержимое строки приглашения к вводу в командной оболочке. Как будет показано позднее, эту строку можно менять весьма существенно |
| PWD | Текущий рабочий каталог |
| TERM | Тип терминала. Unix-подобные системы поддерживают множество протоколов для работы с терминалами; эта переменная определяет протокол, который будет использоваться при обмене данными с эмулятором терминала |
| TZ | Определяет часовой пояс. В большинстве Unix-подобных систем внутренние часы компьютера устанавливаются в *координированное универсальное время* (Coordinated Universal Time, UTC), а при выводе значения времени к нему добавляется смещение, определяемое этой переменной |
| USER | Имя пользователя |

Не волнуйтесь, если какие-то переменные у вас отсутствуют. Они могут отличаться в разных дистрибутивах.

**Как устанавливается окружение?**

Когда мы входим в систему, запускается программа bash и читает содержимое серии конфигурационных сценариев, называемых *файлами запуска* (startup files), где определяется окружение по умолчанию, общее для всех пользователей. Затем она читает дополнительные файлы запуска в вашем домашнем каталоге, где определяется личное окружение. Точная последовательность обработки файлов зависит от типа запускаемого сеанса командной оболочки.

**Оболочка входа и простая оболочка**

Сеансы работы с командной оболочкой входа могут быть двух типов: сеанс командной оболочки входа (login shell session) и сеанс простой командной оболочки (non-login shell session).

*Сеанс командной оболочки входа* (login shell session) — это сеанс, который на входе запрашивает имя пользователя и пароль, например, когда вход выполняется в виртуальной консоли. *Сеанс простой командной оболочки* (non-login shell session) обычно начинается, когда запускается терминал в графическом окружении.

Командные оболочки входа читают один или несколько файлов запуска, перечисленных в табл. 11.2.

Обычные сеансы командной оболочки читают файлы, перечисленные в табл. 11.3.

Помимо чтения настроек из файлов запуска, перечисленных выше, обычные командные оболочки наследуют окружение от родительского процесса, каковым обычно является командная оболочка входа.

**Таблица 11.2. Файлы запуска для сеансов командной оболочки входа**

|  |  |
| --- | --- |
| **Файл** | **Содержит** |
| */etc/profile* | Общесистемный конфигурационный сценарий, настройки из которого применяются для всех пользователей |
| *~/.bash\_profile* | Личный пользовательский файл запуска. Может использоваться для расширения и/или переопределения общесистемных настроек |
| *~/.bash\_login* | Если файл *~/.bash\_profile* присутствует в домашнем каталоге, bash пытается прочитать его |
| *~/.profile* | Если в домашнем каталоге нет ни *~/.bash\_profile*, ни *~/.bash\_login*, bash пытается прочитать этот файл. Используется по умолчанию в дистрибутивах на основе Debian, таких как Ubuntu |

**Таблица 11.3. Файлы запуска для обычных сеансов командной оболочки**

|  |  |
| --- | --- |
| **Файл** | **Содержит** |
| */etc/bash.bashrc* | Общесистемный конфигурационный сценарий, настройки из которого применяются для всех пользователей |
| *~/.bashrc* | Личный пользовательский файл запуска. Может использоваться для расширения и/или переопределения общесистемных настроек |

Загляните в свою систему и посмотрите, какие файлы запуска у вас имеются. Помните: поскольку большинство имен файлов из перечисленных выше начинается с точки (такие файлы считаются скрытыми), при использовании команды ls ей необходимо передавать параметр -a.

С точки зрения обычного пользователя, файл *~/.bashrc* является, пожалуй, самым важным файлом запуска, потому что его содержимое читается практически всегда. Обычные командные оболочки читают его по умолчанию, а большинство файлов запуска для командных оболочек входа написаны так, что оболочка также прочитает файл *~/.bashrc*.

**Что находится в файлах запуска?**

Если заглянуть внутрь типичного файла *.bash\_profile* (взятого из системы CentOS-4), можно увидеть следующее:

# .bash\_profile

# Загрузить псевдонимы и функции

if [ -f ~/.bashrc ]; then

        . ~/.bashrc

fi

# Определение пользовательского окружения и запуск программ

PATH=$PATH:$HOME/bin

export PATH

Строки, начинающиеся с #, — это *комментарии,* они не читаются командной оболочкой, а предназначены для человека. Первый интересный фрагмент начинается в четвертой строке:

if [ -f ~/.bashrc ]; then

        . ~/.bashrc

fi

Этот код называется *составной условной командой*, полное описание которой будет дано в части IV книги, где обсуждается программирование на языке командной оболочки, а пока приведем ее перевод на человеческий язык:

Если файл "~/.bashrc" существует, тогда

     прочитать файл "~/.bashrc" file.

Как видите, этот фрагмент вынуждает командную оболочку входа прочитать содержимое файла *.bashrc*. Следующая операция, выполняемая в файле запуска, имеет отношение к переменной PATH.

Приходилось ли вам задумываться над тем, как командная оболочка находит коман­ды, которые вводятся в командной строке? Например, когда мы вводим ls, командная оболочка не обыскивает весь компьютер целиком, чтобы найти */bin/ls* (полный путь к команде ls), а просматривает только каталоги, перечисленные в переменной PATH.

Переменная PATH часто (но не всегда, в зависимости от дистрибутива) устанавливается в файле запуска */etc/profile*, как показано ниже:

PATH=$PATH:$HOME/bin

Здесь в конец списка в переменной PATH добавляется каталог *$HOME/bin*. Этот код может служить примером использования механизма подстановки параметров, с которым мы познакомились в главе 7. Для демонстрации попробуйте выполнить следующий пример:

[me@linuxbox ~]$ **foo="This is some"**

[me@linuxbox ~]$ **echo $foo**

This is some

[me@linuxbox ~]$ **foo=$foo" text."**

[me@linuxbox ~]$ **echo $foo**

This is some text.

Используя этот прием, можно добавлять текст в конец содержимого переменной.

При добавлении строки $HOME/bin в конец содержимого переменной PATH происходит добавление каталога *$HOME/bin* в список каталогов, где будет выполняться поиск вводимых команд. Это означает, что если мы решим создать каталог в своем домашнем каталоге для хранения личных программ, командная оболочка уже будет готова к этому. Нам останется только дать имя *bin* этому каталогу.

**ПРИМЕЧАНИЕ**

Многие дистрибутивы предоставляют настройки PATH по умолчанию. Некоторые дистрибутивы на основе Debian, такие как Ubuntu, проверяют наличие каталога *~/bin* во время входа, и если он имеется, динамически добавляют его в переменную PATH.

Наконец, у нас осталась еще одна строка:

export PATH

Команда export указывает командной оболочке сделать содержимое переменной PATH доступным дочерним процессам этой оболочки.

**Изменение окружения**

Теперь, зная, где находятся файлы запуска и что они содержат, мы можем изменить их, чтобы настроить окружение.

**Какие файлы следует изменять?**

Как правило, изменение содержимого переменой PATH или определение дополнительных переменных окружения следует производить в файле *.bash\_profile* (или эквивалентном ему, в зависимости от дистрибутива, — например, в Ubuntu используется файл *.profile*). Во всех остальных случаях изменения должны производиться в *.bashrc*. Если вы не системный администратор и вам не требуется вносить изменения, касающиеся всех пользователей системы, изменяйте только файлы в своем домашнем каталоге. Конечно, можно изменять файлы в */etc*, такие как *profile*, и во многих случаях в этом есть определенный смысл, но давайте пока избегать рискованных действий.

**Текстовые редакторы**

Для редактирования (то есть изменения) файлов запуска командной оболочки, а также большинства других конфигурационных файлов в системе используется программа, которая называется *текстовым редактором*. Текстовый редактор, подобно текстовому процессору, позволяет редактировать слова на экране, перемещая курсор. От текстового процессора эта программа отличается только поддержкой простого текста и нередко наличием особенностей, необходимых при разработке программ. Текстовые редакторы — основной инструмент, используемый программистами для создания программного кода и системными администраторами для управления конфигурационными файлами, определяющими настройки системы.

Для Linux существует огромное число текстовых редакторов; в вашей системе почти наверняка установлено несколько из них. Почему было создано так много редакторов? Вероятно, потому, что программистам нравится писать их, а так как программисты очень активно пользуются редакторами, они стремятся воплотить в них свои взгляды на то, как должны работать эти редакторы.

Текстовые редакторы делятся на две основные категории: с графическим и с текстовым интерфейсом. Оба окружения рабочего стола, GNOME и KDE, включают несколько популярных редакторов с графическим интерфейсом. В состав GNOME входит редактор с названием gedit, который в меню GNOME обычно называется **Text Editor** (Текстовый редактор). Вместе с KDE обычно распространяется три редактора (в порядке увеличения сложности): kedit, kwrite и kate.

Существует множество редакторов с текстовым интерфейсом. Наиболее популяр­ные из них, с которыми, возможно, вы столкнетесь: nano, vi и emacs. Редактор nano — простой в использовании редактор, созданный как замена редактору pico, поставляемому в составе пакета программ для работы с электронной почтой PINE. Редактор vi (в большинстве систем Linux его замещает программа vim, название которой является сокращением от *Vi IMproved* (Vi улучшенный)) — традиционный редактор для Unix-подобных систем. Подробнее о нем рассказывается в главе 12. Редактор emacs был написан Ричардом Столлманом (Richard Stallman). Это невероятная, универсальная среда программирования, построенная по принципу «все в одном». Но, несмотря на свою доступность, он редко устанавливается по умолчанию в большинстве систем Linux.

**Использование текстового редактора**

Любой текстовый редактор можно запустить из командной строки, введя имя редактора и имя файла, который требуется отредактировать. Если указанный файл не существует, редактор решит, что вы хотите создать новый файл. Ниже приводится пример использования gedit:

[me@linuxbox ~]$ **gedit some\_file**

Эта команда запустит текстовый редактор gedit и загрузит в него файл с именем *some\_file*, если таковой существует.

Все текстовые редакторы с графическим интерфейсом имеют интуитивно понятный интерфейс, поэтому мы не будем описывать их здесь. Вместо этого сосредоточимся на редакторе с текстовым интерфейсом nano. Давайте запустим nano и внесем изменения в файл *.bashrc*. Но перед этим поговорим немного о мерах предосторожности. Всякий раз, собираясь редактировать важный конфигурационный файл, создайте сначала его резервную копию. Это обезопасит вас, если в процессе редактирования вы безнадежно испортите содержимое файла. Чтобы создать резервную копию файла *.bashrc*, выполните следующую команду:

[me@linuxbox ~]$ **cp .bashrc .bashrc.bak**

Неважно, как вы назовете файл с резервной копией; просто дайте ему такое имя, чтобы было понятно, что это за файл. Наиболее часто для имен файлов с резервными копиями используются расширения *.bak*, *.sav*, *.old* и *.orig*. Да, и не забудьте, что команда cp *без лишних вопросов затирает существующие файлы*.

Теперь, когда резервная копия создана, можно запускать редактор:

[me@linuxbox ~]$ **nano .bashrc**

После запуска nano вы увидите на экране примерно такую картину:

  GNU nano 2.0.3         File: .bashrc

# .bashrc

# Загрузить глобальные определения

if [ -f /etc/bashrc ]; then

    . /etc/bashrc

fi

# Пользовательские псевдонимы и функции

                          [ Read 8 lines ]

^G Get Help^O WriteOut^R Read Fil^Y Prev Pag^K Cut Text^C Cur Pos

^X Exit    ^J Justify ^W Where Is^V Next Pag^U UnCut Te^T To Spell

**ПРИМЕЧАНИЕ**

Если в вашей системе не установлен редактор nano, можете вместо него использовать редактор с графическим интерфейсом.

Экран редактора делится на три части: заголовок в верхней части, область редактирования текста в середине и меню команд внизу. Так как nano проектировался как замена текстового редактора, входящего в состав почтового клиента, он не обладает развитыми функциями редактирования.

Первая команда, которую нужно узнать при использовании любого редактора, — это команда выхода из программы. Чтобы покинуть nano, нажмите **CTRL+X**. Эта ­команда присутствует в меню, в нижней части экрана. Нотация **^X** означает **CTRL+X**. Это распространенная форма записи управляющих комбинаций, используемая во многих программах.

Вторая команда, которую следует знать, — как сохранить изменения. В nano сохранение выполняется нажатием **CTRL+O**. Теперь, обладая новыми знаниями, приступим к правке текста. Используя клавишу со стрелкой вниз и/или **Page Down**, переместите курсор в конец файла и добавьте в *.bashrc* следующие строки:

umask 0002

export HISTCONTROL=ignoredups

export HISTSIZE=1000

alias l.='ls -d .\* --color=auto'

alias ll='ls -l --color=auto'

**ПРИМЕЧАНИЕ**

Ваш дистрибутив уже может включать некоторые из этих строк, но если повторить их, это ничему не навредит.

Эти изменения описаны в табл. 11.4.

**Таблица 11.4. Дополнения в файле .bashrc**

| **Строка** | **Значение** |
| --- | --- |
| umask 0002 | Определяет маску umask для устранения проблемы с общими каталогами, обсуждавшейся в главе 9 |
| export HISTCONTROL=ignoredups | Предписывает механизму истории командной оболочки игнорировать команду, если непосредственно перед ней точно такая же команда была записана в историю |
| export HISTSIZE=1000 | Увеличивает историю команд с 500 строк по умолчанию до 1000 |
| alias l.='ls -d .\* --color=auto' | Создает новую команду с именем l., которая выводит все элементы каталога с именами, начинающимися с точки |
| alias ll='ls -l –color=auto' | Создает новую команду с именем ll, которая выводит содержимое каталога в подробном формате |

Как видите, догадаться о назначении многих новых строк непросто, поэтому нелишним будет снабдить их комментариями, чтобы прояснить смысл для тех, кто будет читать файл *.bashrc*. Используя редактор, добавьте пояснения, как показано ниже:

# Изменить маску umask, чтобы упростить использование общих каталогов

umask 0002

# Игнорировать дубликаты в истории команд и увеличить

# объем истории до 1000 строк

export HISTCONTROL=ignoredups

export HISTSIZE=1000

# Добавить несколько удобных псевдонимов

alias l.='ls -d .\* --color=auto'

alias ll='ls -l --color=auto'

Так намного лучше! Закончив правку, нажмите **CTRL+O**, чтобы сохранить измененный файл *.bashrc*, и **CTRL+X**, чтобы выйти из nano.

**важность комментариев**

Всякий раз, изменяя конфигурационные файлы, добавляйте краткие комментарии, описывающие эти изменения. Вне всяких сомнений, вы будете помнить назначение своих изменений завтра, но вспомните ли вы об этом через шесть месяцев? Сделайте себе подарок, добавьте несколько комментариев. Кроме того, хорошо бы завести файл журнала и в нем фиксировать произведенные изменения.

Комментарии в сценариях на языке командной оболочки и в файлах запуска начинаются с символа #. В других конфигурационных файлах для этой цели могут использоваться другие символы. Комментарии можно найти в большинстве конфигурационных файлов. Используйте их как руководство.

В конфигурационных файлах вам часто будут встречаться *закомментированные* строки, чтобы предотвратить их влияние на соответствующую программу. Это делается с целью показать читателю возможные варианты настройки или примеры правильного синтаксиса оформления настроек. Например, файл *.bashrc* в Ubuntu 8.04 содержит следующие строки:

# несколько других псевдонимов команды ls

#alias ll='ls -l'

#alias la='ls -A'

#alias l='ls -CF'

Последние три строки — это допустимые определения псевдонимов, только закомментированные. Если удалить начальные символы # из этих трех строк (это называется *раскомментировать*), псевдонимы будут активированы. Напротив, если добавить символ # в начало строки, можно деактивировать конфигурационную строку, сохранив информацию, хранящуюся в ней.

**Активация изменений**

Изменения, произведенные в файле *.bashrc*, не вступят в силу, пока вы не закроете терминал и не запустите новый, потому что оболочка читает содержимое файла *.bashrc* только в начале сеанса. Однако существует возможность принудить bash повторно прочитать измененный файл *.bashrc* следующей командой:

[me@linuxbox ~]$ **source .bashrc**

После этого изменения должны вступить в силу. Попробуйте, например, один из новых псевдонимов:

[me@linuxbox ~]$ **ll**

**Заключительное замечание**

В этой главе вы приобрели основные навыки правки конфигурационных файлов в текстовом редакторе. По мере чтения страниц справочного руководства (man) для команд обращайте внимание на переменные окружения, поддерживаемые коман­дами. Эта информация может оказаться весьма ценной. В следующих главах вы познакомитесь с еще одним мощным инструментом — функциями командной оболочки, которые также можно включать в файлы запуска bash, чтобы расширить арсенал собственных команд.