Командная оболочка и окружение

Командная оболочка и окружение
- Переменные окружения
- Как устанавливается окружение
- Подстановка
- Литература

Переменные окружения

В командной оболочке предусмотрено средство для сохранения информации, которая используется различными приложениями (в том числе и самой оболочкой). Это средство — переменные среды (или окружения). У каждой такой переменной есть имя и значение. Например, переменная с именем PATH содержит список каталогов, в которых командная оболочка ищет исполняемый файл указанной команды.

Переменные среды делятся на
- глобальные переменные;
- локальные переменные.

Доступ к глобальным переменным может быть получен из командной оболочки и из любого дочернего процесса, который запущен командной оболочкой. Локальные переменные доступны только в том сеансе работы командной оболочки, в котором они созданы.

Замечание
Кроме переменных командная оболочка хранит в окружении функции командной оболочки и псевдонимы.

Для просмотра значений глобальных переменных используется команда printenv или просто вызов env без аргументов:

```
user@linux-pc:~$ printenv
SHELL=/bin/bash
EDITOR=vim
NAME=linux-pc
PWD=/home/user
LOGNAME=user
HOME=/home/user
LANG=C.UTF-8

...

USER=user
```

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games

...

user@linux-pc:~$
```

Посмотреть значение отдельной переменной можно с помощью команды echo. При ссылке на переменную среды необходимо помещать знак $ перед именем переменной среды.

```
user@linux-pc:~$ echo $USER
user
```

Как уже говорилось, глобальные переменные доступны в дочерних процесса, работающих под управлением текущей командной оболочки:

```
user@linux-pc:~$ echo $HOME
/home/user
user@linux-pc:~$ bash
user@linux-pc:~$ echo $HOME
/home/user
```

В этом примере после запуска нового экземпляра командной оболочки (команда bash) отображается текущее значение переменной среды HOME.

В Linux нет отдельной команды, которая отображала бы только локальные переменные среды. Но есть команда set, которая отображает весь набор переменных (включая глобальные).

```
user@linux-pc:~$ set
BASH=/bin/bash
BASH_ALIASES=()
BASH_ARGC=([0]="0")
BASH_ARGV=()
BASH_CMDS=()
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSION='5.0.17(1)-release'
COLUMNS=120
DIRSTACK=()
EDITOR=vim
EUID=1000
GROUPS=()
HISTCONTROL=ignoreboth
HISTFILE=/home/user/.bash_history
HISTFILESIZE=2000
HISTSIZE=1000
HOME=/home/user
...
user@linux-pc:~$
```

Замечание
Команда set выводит не только переменных обоих типов, но и функции командной оболочки. Единственный элемент окружения, который не выводится командами set и printenv, это псевдонимы. Для этого используется команда alias.

В командной оболочке предусмотрена возможность создавать собственные локальные переменные:

```
user@linux-pc:~$ favorite_number=17
user@linux-pc:~$ echo $favorite_number
17
user@linux-pc:~$ message='Hello, world!'
user@linux-pc:~$ echo $message
Hello, world!
```

Для имен локальных переменных обычно используются строчные буквы.

Локальные переменные доступны во всех процессах командной оболочки, но не доступны в дочерних процессах:

```
user@linux-pc:~$ bash
user@linux-pc:~$ echo $message

user@linux-pc:~$ exit
exit
user@linux-pc:~$ echo $message
Hello, world!
```

Для создания глобальной переменной необходимо создать локальную переменную и выполнить ее экспорт в глобальную среду с помощью команды export:

```
user@linux-pc:~$ unset message
user@linux-pc:~$ echo $message

user@linux-pc:~$
```

После экспорта переменной среды message была запущена дочерняя командная оболочка, в которой выведено значение переменной message.

Для удаления переменной среды используется команда unset:

```
user@linux-pc:~$ unset message
user@linux-pc:~$ echo $message

user@linux-pc:~$
```

Переменная	Значение
EDITOR	Имя программы, используемой в качестве текстового редактора.
SHELL	Имя программы командной оболочки.

Переменная	Значение
HOME	Путь к домашнему каталогу.
LANG	Определяет набор символов и порядок сортировки.
OLD_PWD	Предыдущий рабосий каталог
PATH	Список каталогов, разделенный двоеточием, в которых производится поиск выполняемых программ по именам.
PWD	Текущий рабочий каталог.
USER	Имя пользователя.

Как устанавливается окружение

При запуске командная оболочка читает несколько так называемых файлов запуска, в которых определяется окружение, общее для всех пользователей. Затем она читает дополнительные файлы запуска в домашнем каталоге пользователя, которые определяют личное окружение. Точная последовательность действий зависит от типа запускаемого сеанса командной оболочки.

Сеансы работы с командной оболочкой делятся на два типа:
- сеанс командной оболочки входа;
- сеанс простой командной оболочки.

Сеанс командной оболочки входа — это сеанс, который при ходе запрашивает имя пользователя и пароль. Сеанс простой командной оболочки начинается, например, при запуске эмулятора терминала в графическом окружении.

Файлы запуска командной оболочки входа

Файл	Содержит
/etc/profile	Общесистемный конфигурационный сценарий, настройки из которого применяются для всех пользователей.
домашний_каталог/.bash_profile	Личный пользовательский файл запуска. Может использоваться для расширения и/или переопределения общесистемных настроек.
домашний_каталог/.bash_login	Если файл .bash_profile присутствует в домашнем каталоге, bash пытается прочитать его.
домашний_каталог/.profile	Если в домашнем каталоге нет ни .bash_profile, ни .bash_login, bash пытается прочитать этот файл.
OLD_PWD	Предыдущий рабосий каталог

Файл	Содержит
PATH	Список каталогов, разделенный двоеточием, в которых производится поиск выполняемых программ по именам.
PWD	Текущий рабочий каталог.
USER	Имя пользователя.

Файлы запуска простой командной оболочки

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

С точки зрения обычного пользователя, файл .bashrc является самым важным, потому что читается практически всегда.

Как правило, изменение содержимого переменной PATH или определение дополнительных переменных окружения следует производить в файле .bash_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.

Изменения, произведенные в файле .bashrc, не вступят в силу, пока вы не закроете эмулятор терминала и не запустите новый, потому что оболочка читает содержимое файла .bashrc только в начале сеанса. В качестве альтернативы можно просто перезапустить оболочку вызовом команды `bash` или насильно прочитать файл с помощью команды `source ~/.bashrc`.

Подстановка

Прежде, чем выполнить команду, командная оболочка обрабатывает текст введенной команды, выполняя так называемую подстановку. Рассмотрим следующий пример:

```
user@linux-pc:~$ echo *
bin ls_errors.txt ls_output.txt my_name.txt one.txt one_two.txt repos test
user@linux-pc:~$
```

Символ “*” означает последовательность любых символов в имени файла. Перед тем как выполнить команду echo командная оболочка заменяет символ “*” именами файлов в текущем рабочем каталоге.

Символ ~ в начале слова заменяется именем домашнего каталога текущего пользователя:

```
user@linux-pc:~$ echo ~
/home/user
user@linux-pc:~$ echo ~/test
/home/user/test
```

Подстановка позволяет использовать командную оболочку как калькулятор:

```
user@linux-pc:~$ echo $((2 + 2))
4
```

Для вычисления арифметических выражений с помощью подстановки используется следующий формат

```
$((выражение))
```

Операция	Описание
+	Сложение
-	Вычитание
*	Умножение
/	Целочисленное деление
%	Остаток от деления
**	Возведение в степень

В выражение можно использовать только целые числа.

Благодаря подстановке, можно использовать вывод одной команды в качестве аргумента другой:

```
user@linux-pc:~$ echo $(ls)
bin ls_errors.txt ls_output.txt my_name.txt one.txt one_two.txt repos test
user@linux-pc:~$
```

Устаревшая форма записи выглядит следующим образом (вместо знака доллара и круглых скобок используются обратные апострофы):

```
user@linux-pc:~$ echo `ls`
bin ls_errors.txt ls_output.txt my_name.txt one.txt one_two.txt repos test
user@linux-pc:~$
```

Иногда требуется, чтобы подстановка не выполнялась. Если заключить текст в двойные кавычки, то все специальные символы теряют свой смысл:

```
user@linux-pc:~$ echo "*"
*
```

Исключение составляют знак доллара, обратный слеш и обратный апостроф. Не будет выполняться подстановка путей, подстановка тильды, разбиение на слова

```
user@linux-pc:~$ echo "*p"
*p
user@linux-pc:~$ echo "~/test"
~/test
user@linux-pc:~$ echo "one two  three"
one two  three
```

Будет выполняться подстановка значений переменных, подстановка арифметических выражение и подстановка команд. Если Вам требуется подавить все подстановки, используйте одиночные кавычки

```
user@linux-pc:~$ echo "$USER have $((2 + 5))\$"
user have 7$
user@linux-pc:~$ echo '$USER have $((2 + 5))\$'
$USER have $((2 + 5))\$
```

Литература

1. Шотсс У. “Командная строка Linux.”, глава 11
2. Блум Р., Бреснахэн К. “Командная строка Linux и сценарии оболочки.”, глава 5