Раскрытие команд командной оболочкой

Команды и аргументы

В данной главе вашему вниманию представляется обзор механизма раскрытия команд **командной оболочки** (shell expansion), созданный в ходе подробного рассмотрения методик обработки **команд** и **аргументов**. Понимание принципа работы механизма раскрытия команд командной оболочки является важным ввиду того, что многие команды в вашей системе Linux подвергаются обработке и с высокой вероятностью последующей модификации средствами командной оболочки перед исполнением.

Интерфейс командной строки системы или командная оболочка, используемая в большинстве систем Linux, носит имя **bash**, которое расшифровывается как **Bourne again shell** (название "Born again shell" - "возрожденная командная оболочка" было изменено с целью упоминания автора оригинальной командной оболочки sh Стивена Борна). Командная оболочка **bash** реализует возможности командных оболочек **sh** (оригинальная командная оболочка Стивена Борна), **csh** (командная оболочка Билла Джоя с поддержкой сценариев, синтаксис которых основан на синтаксисе языка программирования C), а также **ksh** (командная оболочка Дэвида Корна).

В данной главе для демонстрации возможностей командной оболочки будет периодически использоваться команда **echo**. Команда **echo** является достаточно простой командой: она всего лишь осуществляет вывод переданных ей данных.

```
paul@laika:~$ echo Burtonville
Burtonville
paul@laika:~$ echo Smurfs are blue
Smurfs are blue
```

Аргументы

Одной из важнейших возможностей командной оболочки является возможность **обработки строк команд**. При вводе команды после приглашения командной оболочки и нажатии клавиши Enter командная оболочка приступает к обработке строки команды, разделяя ее на **аргументы**. При обработке строки команды командная оболочка может внести множество изменений в переданные вами аргументы.

Данный процесс называется раскрытием команд командной оболочки. После того, как командная оболочка заканчивает обработку и модификацию переданной строки команды, будет осуществляться непосредственное исполнение результирующей команды.

Удаление пробелов

Части строки команды, которые разделены с помощью одного или нескольких последовательно расположенных символов **пробелов** (или табуляции), рассматриваются как отдельные **аргументы**, причем все пробелы удаляются. Первым **аргументом** является сама команда, которая должна быть исполнена, остальные **аргументы** передаются этой команде. Фактически командная оболочка производит разделение вашей строки команды на один или несколько аргументов.

Это полностью объясняет эквивалентность следующих четырех команд после их раскрытия средствами **командной оболочки**.

```
[paul@RHELv4u3 ~]$ echo Hello World
Hello World
[paul@RHELv4u3 ~]$ echo Hello   World
Hello World
[paul@RHELv4u3 ~]$ echo   Hello   World
Hello World
[paul@RHELv4u3 ~]$    echo      Hello      World
Hello World
```

Команда **echo** будет выводить каждый из принятых от командной оболочки аргументов. Также команда **echo** осуществляет добавление пробелов между всеми принятыми аргументами.

Одинарные кавычки

Вы можете предотвратить удаление пробелов из строки команды, поместив ее в одинарные кавычки. Содержимое экранированной таким образом строки рассматривается как единый аргумент. В примере ниже команда **echo** принимает только один **аргумент**.

```
[paul@RHEL4b ~]$ echo 'Строка с      одинарными    кавычками'
Строка с      одинарными    кавычками
[paul@RHEL4b ~]$
```

Двойные кавычки

Вы также можете предотвратить удаление пробелов из строки команды, поместив ее в двойные кавычки. Как и в примере выше, команда **echo** примет только один **аргумент**.

```
[paul@RHEL4b ~]$ echo "Строка с      двойными    кавыч-
ками"
Строка с      двойными    кавычками
[paul@RHEL4b ~]$
```

Позднее при обсуждении **переменных** в рамках данной книги мы разберемся с важными различиями между одинарными и двойными кавычками.

Команда echo и кавычки

Строки, помещенные в кавычки, могут содержать специальные обозначения символов, идентифицируемые командной **echo** (в случае использования команды **echo -e**). В примере ниже продемонстрирована методика использования обозначения символа **\n** для вставки символа переноса строки, а также обозначения символа **\t** для вставки символа табуляции (обычно эквивалентного восьми символам пробела).

```
[paul@RHEL4b ~]$ echo -e "Строка с \nсимволом переноса
строки"
Строка с
символом переноса строки
[paul@RHEL4b ~]$ echo -e 'Строка с \nсимволом переноса
строки'
Строка с
символом переноса строки
[paul@RHEL4b ~]$ echo -e "Строка с \tсимволом табуляции"
Строка с     символом табуляции
[paul@RHEL4b ~]$ echo -e ' Строка с \tсимволом табуляции'
Строка с     символом табуляции
[paul@RHEL4b ~]$
```

Команда **echo** может генерировать и другие символы помимо символов пробелов, табуляции и переноса строки. Обратитесь к странице руководства для ознакомления со списком допустимых обозначений символов.

Команды

Внешние или встроенные команды?

Не все исполняемые командной оболочкой команды являются **внешними**; некоторые из них являются **встроенными**. **Внешние команды** реализованы в форме программ, представленных отдельными бинарными файлами, которые размещены в какой-либо директории файловой системы. Многие бинарные фалы, реализующие функции внешних команд, размещаются в директории **`/bin`** или **`/sbin`**. **Встроенные команды** являются неотъемлемой частью самого приложения командной оболочки.

Команда `type`

Для установления того, будет ли переданная командной оболочке команда исполнена как **внешняя команда** или как **встроенная команда**, следует использовать специальную команду **`type`**.

```
paul@laika:~$ type cd
cd is a shell builtin
paul@laika:~$ type cat
cat is /bin/cat
```

Как вы можете заметить, команда **`cd`** является **встроенной**, а команда **`cat`** - **внешней**.

Также вы можете использовать данную команду для установления того, является ли введенная команда **псевдонимом** команды.

```
paul@laika:~$ type ls
ls is aliased to `ls --color=auto'
```

Исполнение внешних команд

Некоторые команды имеют как встроенные, так и внешние реализации. В случае исполнения одной из таких команд приоритет отдается встроенной реализации. Для исполнения внешней реализации вам придется ввести полный путь к бинарному файлу, являющемуся реализацией данной команды.

```
paul@laika:~$ type -a echo
echo is a shell builtin
echo is /bin/echo
paul@laika:~$ /bin/echo Исполнение внешней реализации команды echo...
```

```
Исполнение внешней реализации команды echo...
```

Команда which

Команда **which** осуществляет поиск бинарных файлов в директории, заданной с помощью переменной окружения **$PATH** (переменные будут рассматриваться позднее). В примере ниже устанавливается, что **cd** является **встроенной** командой, а **ls, cp, mkdir, pwd** и **which** - **внешними** командами.

```
[root@RHEL4b ~]# which cp ls cd mkdir pwd
/bin/cp
/bin/ls
/usr/bin/which: no cd in
(/usr/kerberos/sbin:/usr/kerberos/bin:...
/bin/mkdir
/bin/pwd
```

Псевдонимы команд

Создание псевдонима команды

Командная оболочка позволяет вам создавать **псевдонимы** команд (aliases). Псевдонимы команд обычно используются для создания лучше запоминающихся имен для существующих команд или для упрощения передачи параметров команд.

```
[paul@RHELv4u3 ~]$ cat count.txt
один
два
три
[paul@RHELv4u3 ~]$ alias dog=tac
[paul@RHELv4u3 ~]$ dog count.txt
три
два
один
```

Сокращения команд

Команда создания **псевдонима** команды (alias) также может оказаться полезной в случае необходимости сокращения длины имени существующей команды.

```
paul@laika:~$ alias ll='ls -lh --color=auto'
```

```
paul@laika:~$ alias c='clear'
paul@laika:~$
```

Стандартные параметры команд

Псевдонимы команд могут использоваться для передачи командам стандартных параметров. Например, ниже показана методика передачи параметра **-i** по умолчанию при вводе команды **rm**.

```
[paul@RHELv4u3 ~]$ rm -i winter.txt
rm: удалить обычный файл "winter.txt"? no
[paul@RHELv4u3 ~]$ rm winter.txt
[paul@RHELv4u3 ~]$ ls winter.txt
ls: невозможно получить доступ к winter.txt: Нет такого
файла или каталога
[paul@RHELv4u3 ~]$ touch winter.txt
[paul@RHELv4u3 ~]$ alias rm='rm -i'
[paul@RHELv4u3 ~]$ rm winter.txt
rm:удалить пустой обычный файл "winter.txt"? no
[paul@RHELv4u3 ~]$
```

В некоторых дистрибутивах используются стандартные псевдонимы команд для защиты пользователей от случайного удаления файлов ('rm -i', 'mv -i', 'cp -i').

Просмотр объявлений псевдонимов команд

Вы можете передать один или несколько псевдонимов команд в качестве аргументов команды **alias** для вывода их объявлений. Исполнение команды без аргументов приведет к выводу полного списка используемых на данный момент псевдонимов.

```
paul@laika:~$ alias c ll
alias c='clear'
alias ll='ls -lh --color=auto'
```

Команда unalias

Также вы можете прекратить использование псевдонима команды, воспользовавшись командой **unalias**.

```
[paul@RHEL4b ~]$ which rm
/bin/rm
[paul@RHEL4b ~]$ alias rm='rm -i'
[paul@RHEL4b ~]$ which rm
```

```
alias rm='rm -i'
        /bin/rm
[paul@RHEL4b ~]$ unalias rm
[paul@RHEL4b ~]$ which rm
/bin/rm
[paul@RHEL4b ~]$
```

Вывод информации о раскрытии команд командной оболочкой

Вы можете активировать режим вывода информации о раскрытии команд командной оболочкой с помощью команды **set -x** и остановить вывод этой информации с помощью команды **set +x**. У вас может возникнуть потребность в использовании данной возможности как при изучении данного курса, так и в случаях, когда возникают сомнения насчет того, как командная оболочка обрабатывает переданную вами команду.

```
[paul@RHELv4u3 ~]$ set -x
++ echo -ne '\033]0;paul@RHELv4u3:~\007'
[paul@RHELv4u3 ~]$ echo $USER
+ echo paul
paul
++ echo -ne '\033]0;paul@RHELv4u3:~\007'
[paul@RHELv4u3 ~]$ echo \$USER
+ echo '$USER'
$USER
++ echo -ne '\033]0;paul@RHELv4u3:~\007'
[paul@RHELv4u3 ~]$ set +x
+ set +x
[paul@RHELv4u3 ~]$ echo $USER
paul
```