Переменные командной оболочки

В данной главе мы познакомимся с методикой работы с **переменными окружения** с использованием командной оболочки. Эти переменные обычно требуются для работы приложений.

Символ доллара ($)

Еще одним важным интерпретируемым командной оболочкой символом является символ доллара
$. Командная оболочка будет ис-кать **переменную окружения** с именем, соответствующим разме-щенной после **символа доллара** строке, и заменять данный символ
и имя переменной на значение этой переменной (или ни на что в том случае, если переменной не существует).

Ниже приведено несколько примеров использования перемен-ных **$HOSTNAME, $USER, $UID, $SHELL** и **$HOME**.

```
[paul@RHELv4u3 ~]$ echo Это командная оболочка $SHELL
Это командная оболочка /bin/bash
[paul@RHELv4u3 ~]$ echo Данная командная оболочка $SHELL
используется на компьютере $HOSTNAME
Данная командная оболочка /bin/bash используется на
компьютере RHELv4u3.localdomain
[paul@RHELv4u3 ~]$ echo Идентификатор пользователя $USER
равен $UID
Идентификатор пользователя paul равен 500
[paul@RHELv4u3 ~]$ echo Моей домашней директорией являет-
ся директория $HOME
Моей домашней директорией является директория /home/paul
```

Зависимость от регистра

В данном примере показано, что имена переменных командной оболочки зависят от регистра!

```
[paul@RHELv4u3 ~]$ echo Привет $USER
Привет paul
[paul@RHELv4u3 ~]$ echo Привет $user
Привет
```

Создание переменных

В данном примере осуществляется создание переменной **$MyVar** с последующей установкой ее значения. После этого в примере используется команда **echo** для проверки значения созданной переменной.

```
[paul@RHELv4u3 gen]$ MyVar=555
[paul@RHELv4u3 gen]$ echo $MyVar
555
[paul@RHELv4u3 gen]$
```

Кавычки

Обратите внимание на то, что двойные кавычки также позволяют осуществлять раскрытие переменных в строке команды, в то время, как одинарные кавычки позволяют предотвратить такое раскрытие.

```
[paul@RHELv4u3 ~]$ MyVar=555
[paul@RHELv4u3 ~]$ echo $MyVar
555
[paul@RHELv4u3 ~]$ echo "$MyVar"
555
[paul@RHELv4u3 ~]$ echo '$MyVar'
$MyVar
```

Командная оболочка bash будет заменять переменные на их значения в строках, помещенных в двойные кавычки, но не будет осуществлять такую замену в строках, помещенных в одинарные кавычки.

```
paul@laika:~$ city=Burtonville
paul@laika:~$ echo "Сейчас мы находимся в городе $city."
Сейчас мы находимся в городе Burtonville.
```

```
paul@laika:~$ echo ' Сейчас мы находимся в городе $city.'
Сейчас мы находимся в городе $city.
```

Команда set

Вы можете использовать команду set для вывода списка переменных окружения. В системах Ubuntu и Debian команда **set** также выведет список функций командной оболочки после списка переменных командной оболочки. Поэтому для ознакомления со всеми элементами списка переменных окружения при работе с данными системами рекомендуется использовать команду **set | more**.

Команда unset

Следует использовать команду **unset** для удаления переменной из вашего окружения командной оболочки.

```
[paul@RHEL4b ~]$ MyVar=8472
[paul@RHEL4b ~]$ echo $MyVar
8472
[paul@RHEL4b ~]$ unset MyVar
[paul@RHEL4b ~]$ echo $MyVar

[paul@RHEL4b ~]$
```

Переменная окружения $PS1

Переменная окружения **$PS1** устанавливает формат приветствия вашей командной оболочки. При вводе строки форматирования вы можете использовать обратный слэш для экранирования таких специальных символов, как символ **\u**, предназначенный для вывода имени пользователя, или **\w**, предназначенный для вывода имени рабочей директории. На странице руководства командной оболочки **bash** представлен полный список специальных символов.

В примере ниже мы несколько раз изменяем значение переменной окружения **$PS1**.

```
paul@deb503:~$ PS1=приглашение
приглашение
приглашениеPS1='приглашение '
приглашение
приглашение PS1='> '
>
```

```
> PS1='\u@\h$ '
paul@deb503$
paul@deb503$ PS1='\u@\h:\W$'
paul@deb503:~$
```

Для того, чтобы избежать неисправимых ошибок, вы можете использовать зеленый цвет для приглашений командной оболочки, выводимых обычным пользователям, и красный цвет для приглашений командной оболочки, выводимых пользователю **root**. Добавьте следующие строки в ваш файл **.bashrc** для использования зеленого цвета в приглашениях, выводимых обычным пользователям.

```
# цветное приглашение командной оболочки, созданное paul
RED='\[\033[01;31m\]'
WHITE='\[\033[01;00m\]'
GREEN='\[\033[01;32m\]'
BLUE='\[\033[01;34m\]'
export PS1="${debian_chroot:+($debian_chroot)}
$GREEN\u$WHITE@$BLUE\h$WHITE\w\$ "
```

Переменная окружения $PATH

Переменная окружения **$PATH** устанавливает директории файловой системы, в которых командная оболочка ищет бинарные файлы, необходимые для исполнения команд (за исключением тех случаев, когда команда является встроенной или представлена псевдонимом команды). Данная переменная содержит список путей к директориям с символами двоеточия в качестве разделителей.

```
[[paul@RHEL4b ~]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:
```

Командная оболочка не будет осуществлять поиск бинарных файлов, которые могут быть исполнены, в текущей директории. (Функция поиска исполняемых файлов в текущей директории являлась простейшим механизмом несанкционированного доступа к данным, хранящимся на компьютерах под управлением PC-DOS). В том случае, если вы хотите, чтобы командная оболочка осуществляла поиск исполняемых файлов в текущей директории, вам следует добавить символ . в конец строки, являющейся значением переменной $PATH вашей командной оболочки.

```
[paul@RHEL4b ~]$ PATH=$PATH:.
[paul@RHEL4b ~]$ echo $PATH
```

```
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:.
[paul@RHEL4b ~]$
```

Значение переменной $PATH вашей командной оболочки может отличаться в случае использования команды **su** вместо команды **su -**, так как последняя команда позволяет дополнительно использовать значения переменных окружения целевого пользователя. К примеру, в представленный значением переменной **$PATH** список директорий пользователя **root** обычно добавляются директории /**sbin**.

```
[paul@RHEL3 ~]$ su
Password:
[root@RHEL3 paul]# echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin
[root@RHEL3 paul]# exit
[paul@RHEL3 ~]$ su -
Password:
[root@RHEL3 ~]# echo $PATH
/
usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/b
in:
[root@RHEL3 ~]#
```

Команда env

Команда **env** в случае использования без параметров выведет список экспортированных **переменных окружения**. Отличие данной команды от команды **set** с параметрами заключается в том, что команда **set** выводит список всех **переменных окружения**, включая те переменные, которые не экспортируются в дочерние командные оболочки.

Кроме того, команда **env** может также использоваться для запуска "чистой" командной оболочки (командной оболочки без наследования какого-либо окружения). Команда **env -i** позволяет очистить окружение дочерней командной оболочки.

При рассмотрении данного примера следует обратить внимание на то, что командная оболочка **bash** установит значение переменной окружения **$SHELL** при запуске.

```
[paul@RHEL4b ~]$ bash -c 'echo $SHELL $HOME $USER'
/bin/bash /home/paul paul
[paul@RHEL4b ~]$ env -i bash -c 'echo $SHELL $HOME $USER'
```

```
/bin/bash
[paul@RHEL4b ~]$
```

Вы можете использовать команду **env** для установки значения переменной **$LANG** или любой другой переменной окружения одного экземпляра командной оболочки **bash** в рамках одной команды. В примере ниже данная возможность используется для демонстрации влияния значения переменной **$LANG** на работу механизма поиска файлов по шаблонам (для получения дополнительной информации о данном механизме следует обратиться к главе, посвященной поиску файлов по шаблонам).

```
[paul@RHEL4b test]$ env LANG=C bash -c 'ls File[a-z]'
Filea  Fileb
[paul@RHEL4b test]$ env LANG=en_US.UTF-8 bash -c 'ls
File[a-z]'
Filea  FileA  Fileb  FileB
[paul@RHEL4b test]$
```

Команда export

Вы можете экспортировать переменные командной оболочки в другие командные оболочки с помощью команды **export**. В примере ниже с помощью данной команды осуществляется экспорт переменной окружения в дочерние командные оболочки.

```
[paul@RHEL4b ~]$ var3=три
[paul@RHEL4b ~]$ var4=четыре
[paul@RHEL4b ~]$ export var4
[paul@RHEL4b ~]$ echo $var3 $var4
три четыре
[paul@RHEL4b ~]$ bash
[paul@RHEL4b ~]$ echo $var3 $var4
четыре
```

При этом с помощью данной команды переменная не экспортируется в родительскую командную оболочку (ниже приведено продолжение предыдущего примера).

```
[paul@RHEL4b ~]$ export var5=пять
[paul@RHEL4b ~]$ echo $var3 $var4 $var5
четыре пять
[paul@RHEL4b ~]$ exit
```

```
exit
[paul@RHEL4b ~]$ echo $var3 $var4 $var5
три четыре
[paul@RHEL4b ~]$
```

Разграничения переменных

До текущего момента мы сталкивались с тем, что командная оболочка bash интерпретирует переменную начиная с символа доллара, продолжая интерпретацию до появления первого не алфавитно-цифрового символа, который не является символом подчеркивания. В некоторых ситуациях такое поведение может оказаться проблемой. Для решения этой проблемы могут использоваться фигурные скобки таким образом, как показано в примере ниже.

```
[paul@RHEL4b ~]$ prefix=Super
[paul@RHEL4b ~]$ echo Привет $prefixman и $prefixgirl
Привет  и
[paul@RHEL4b ~]$ echo Привет ${prefix}man и ${prefix}girl
Привет Superman и Supergirl
[paul@RHEL4b ~]$
```

Несвязанные переменные

В примере ниже представлена попытка вывода значения переменной **$MyVar**, но она не является успешной ввиду того, что переменной не существует. По умолчанию командная оболочка не будет выводить ничего в том случае, если переменная не связана (ее не существует).

```
[paul@RHELv4u3 gen]$ echo $MyVar

[paul@RHELv4u3 gen]$
```

Однако, существует параметр командной оболочки **nounset**, который вы можете использовать для генерации ошибки в том случае, если используемой переменной не существует.

```
paul@laika:~$ set -u
paul@laika:~$ echo $Myvar
bash: Myvar: unbound variable
paul@laika:~$ set +u
paul@laika:~$ echo $Myvar
```

```
paul@laika:~$
```

В командной оболочке bash команда **`set -u`** идентична команде **`set -o nounset`** и, по аналогии, команда **`set +u`** идентична команде **`set +o nounset`**.