Позиционные параметры

Командная оболочка присваивает параметры, введенные в командной строке, специальным переменным, которые называются позиционными параметрами. Эти переменные имеют имена от 0 до 9.

```
user@linux-pc:~/bin$ cat posit_params_1
#!/bin/bash
# Просмотр первых трех позиционных параметров
echo "
\$0 = $0
\$1 = $1
\$2 = $2
"
user@linux-pc:~/bin$ posit_params_1

$0 = /home/user/bin/posit_params_1
$1 =
$2 =

user@linux-pc:~/bin$ posit_params_1 one

$0 = /home/user/bin/posit_params_1
$1 = one
$2 =

user@linux-pc:~/bin$ posit_params_1 one two

$0 = /home/user/bin/posit_params_1
$1 = one
$2 = two

user@linux-pc:~/bin$ posit_params_1 one two tree

$0 = /home/user/bin/posit_params_1
$1 = one
$2 = two

user@linux-pc:~/bin$
```

Переменная $0 всегда содержит путь к файлу выполняемой программы. Если требуется получить только имя сценария, необходимо воспользоваться командой basename, например, вот так

```
name=`basename $0`
```

Если у сценарии больше девяти параметров, именование переменных позиционных меняется: после девятой переменной необходимо указывать фигурные скобки вокруг номера переменной (например, для обращения к переменной с номером 10 необходимо написать ${10}).

Количество параметров, которые были переданы в командной строке, содержит переменная $#.

```
#!/bin/bash
name=`basename $0`
if [ $# != 2 ]; then
  echo Использование: $name num_1 num_2
  exit 1
fi
sum=$(($1 + $2))
echo Сумма равна $sum
```

В операторе if проверяется количество параметров, указанных в командной строке. Если параметров недостаточно, выводится сообщение и сценарий завершается.

Переменные $* и $@ включают в себя все параметры командной строки, начиная с $1. Переменная $* содержит все параметры как одно слов, в то время как переменная $@ содержит все параметры как отдельные слова.

```
user@linux-pc:~/bin$ cat posit_params_3
#!/bin/bash
# Сравнение переменных $* и $@
count=1
for param in "$*"; do
  echo "\$* параметр #$count = $param"
  count=$((count + 1))
done
echo
count=1
for param in "$@"; do
 echo "\$@ параметр #$count = $param"
 count=$((count + 1))
done
echo
user@linux-pc:~/bin$ posit_params_3 My name is Igor
$* параметр #1 = My name is Igor

$@ параметр #1 = My
$@ параметр #2 = name
$@ параметр #3 = is
$@ параметр #4 = Igor

user@linux-pc:~/bin$
```

Еще один способ перебрать в сценарии значения всех параметров командной строки — использовать команду shift.

После каждого выполнения команды shift происходит перемещение позиционных параметров на однону позицию: $1 = $2, $2 = $3, … . Значение переменной $0 остается неизменным.

```
user@linux-pc:~/bin$ cat posit_params_4
#!/bin/bash
# Пример использования shift
count=1
while [ -n "$1" ]; do
  echo "Параметр #$count = $1"
  count=$((count + 1))
  shift
done
user@linux-pc:~/bin$ ./posit_params_4 My name is Igor
Параметр #1 = My
Параметр #2 = name
Параметр #3 = is
Параметр #4 = Igor
user@linux-pc:~/bin$
```

Сценарий ниже показывает пример анализа параметров командной строки. У сценария три параметра:

- Выходной файл. При указании этого параметра результаты работы программы сохраняются в файле. Указать необходимость использовать файл можно так `-f имя_файла` или так `--file имя_файла`.
- Подробный режим. При указании этого параметра программа выдает на экран дополнительные диагностические сообщение. Указать необходимость этого можно так `-v` или так `--verbose`.
- Справка. Передав параметр `-h` или `--help`, можно потребовать от программы вывести сообщение о правилах ее использования.

```
#!/bin/bash
# Пример обработка параметров командной строки

usage () {
 echo "Использование: $PROGNAME [-f file | -v | -h]"
 return
}

PROGNAME=$(basename $0)

verbose=0
file_name=""
while [[ -n $1 ]]; do
  if [ $1 == "-f" -o $1 == "--file" ]; then
    shift
        file_name=$1
  elif [ $1 == "-v" -o $1 == "--verbose"  ]; then
    verbose=1
  elif [ $1 == "-h" -o $1 == "--help"  ]; then
    usage
    exit
  else
    usage >&2
    exit 1
  fi
  shift
done
```

```
28
29 echo Значение переменной verbose равно $verbose
30 echo Значение переменной file_name равно \"$file_name\"
```

Использование позиционный параметров для передачи аргументов в функции ничем не отличается от передачи параметров в сценарий.

```
1  #!/bin/bash
2  # Пример обработка параметров в функции
3
4  file_info () {
5
6    if [ -e "$1" ]; then
7      echo -e "\nТип файла:"
8      file $1
9      echo -e "\nИнформация о файле:"
10     stat $1
11   else
12     echo "Использование $FUNCNAME file" >&2
13     return 1
14   fi
15 }
16
17 file_info $0
```

Литература

1. Шотсс У. “Командная строка Linux.”, глава 32
2. Блум Р., Бреснахэн К. “Командная строка Linux и сценарии оболочки.”, глава 13