SHELL Programming Part 2

Command-Line Arguments

The command-line arguments $1, $2, $3, ...$9 are positional parameters, with $0 pointing to the actual command, program, shell script, or function and $1, $2, $3, ...$9 as the arguments to the command.

Following script uses various special variables related to the command line −

```
#!/bin/sh



echo "File Name: $0"

echo "First Parameter : $1"

echo "Second Parameter : $2"

echo "Quoted Values: $@"

echo "Quoted Values: $*"

echo "Total Number of Parameters : $#"
```

Here is a sample run for the above script −

```
$./test.sh Zara Ali
File Name : ./test.sh
First Parameter : Zara
Second Parameter : Ali
Quoted Values: Zara Ali
Quoted Values: Zara Ali
```

```
Total Number of Parameters : 2
```

Special Parameters $* and $@

There are special parameters that allow accessing all the command-line arguments at once. **$*** and **$@** both will act the same unless they are enclosed in double quotes, **""**.

Both the parameters specify the command-line arguments. However, the "$*" special parameter takes the entire list as one argument with spaces between and the "$@" special parameter takes the entire list and separates it into separate arguments.

We can write the shell script as shown below to process an unknown number of commandline arguments with either the $* or $@ special parameters −

```
#!/bin/sh



for TOKEN in $*

do

   echo $TOKEN

done
```

Here is a sample run for the above script −

```
$./test.sh Zara Ali 10 Years Old
Zara
Ali
10
Years
Old
```

Note − Here **do...done** is a kind of loop that will be covered in a subsequent tutorial.

Exit Status

The **$?** variable represents the exit status of the previous command.

Exit status is a numerical value returned by every command upon its completion. As a rule, most commands return an exit status of 0 if they were successful, and 1 if they were unsuccessful.

Some commands return additional exit statuses for particular reasons. For example, some commands differentiate between kinds of errors and will return various exit values depending on the specific type of failure.

Following is the example of successful command −

```
$./test.sh Zara Ali
File Name : ./test.sh
First Parameter : Zara
Second Parameter : Ali
Quoted Values: Zara Ali
Quoted Values: Zara Ali
Total Number of Parameters : 2
$echo $?
0
$
```

Escape Characters

Certain characters are significant to the shell; we have seen, for example, that the use of double quotes (") characters affect how spaces and TAB characters are treated, for example:

```
$ echo Hello       World
```

```
Hello World
$ echo "Hello       World"
Hello     World
```

So how do we display: **`Hello     "World"`** ?

```

$ echo "Hello   \"World\""
```

The first and last " characters wrap the whole lot into one parameter passed to **`echo`** so that the spacing between the two words is kept as is. But the code:

```
$ echo "Hello   " World ""
```

would be interpreted as three parameters:

1. "Hello   "
2. World
3. ""

So the output would be

```
Hello    World
```

Note that we lose the quotes entirely. This is because the first and second quotes mark off the Hello and following spaces; the second argument is an unquoted "World" and the third argument is the empty string; "".
Thanks to Patrick for pointing out that this:

```

$ echo "Hello   "World""
```

is actually only one parameter (no spaces between the quoted parameters), and that you can test this by replacing the **`echo`** command with (for example) **`ls`**.