UNIX - SPECIAL VARIABLES

Previous tutorial warned about using certain nonalphanumeric characters in your variable names. This is because those characters are used in the names of special Unix variables. These variables are reserved for specific functions.

For example, the $ character represents the process ID number, or PID, of the current shell:

```
$echo $$
```

Above command would write PID of the current shell −

```
29949
```

The following table shows a number of special variables that you can use in your shell scripts −

Variable	Description
$0	The filename of the current script.
$n	These variables correspond to the arguments with which a script was invoked. Here n is a positive decimal number corresponding to the position of an argument *thefirstargumentis*$1, *thesecondargumentis*$2, *andsoon*.
$#	The number of arguments supplied to a script.
$*	All the arguments are double quoted. If a script receives two arguments, ∗ *isequivalentto*1 $2.
$@	All the arguments are individually double quoted. If a script receives two arguments, *@isequivalentto*1 $2.
$?	The exit status of the last command executed.
$$	The process number of the current shell. For shell scripts, this is the process ID under which they are executing.
$!	The process number of the last background command.

Command-Line Arguments

The command-line arguments 1, 2, 3, . . . 9 are positional parameters, with 0*pointingtotheactualcommand*, *program*, *shellscript*, *orfunctionand*1, 2, 3, ...$9 as the arguments to the command.

Following script uses various special variables related to 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 of the command-line arguments at once. $* and $@ both will act the same unless they are enclosed in double quotes, "".

Both the parameter specifies all command-line arguments but the "$*" special parameter takes the entire list and separates it into separate arguments.

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

```
#!/bin/sh

for TOKEN in $*
do
   echo $TOKEN
done
```

There is one 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 which we would cover in 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
$
```