Unix / Linux - Variables spéciales

Dans ce chapitre, nous discuterons en détail de la variable spéciale dans Unix. Dans l'un de nos chapitres précédents, nous avons compris comment faire attention lorsque nous utilisons certains caractères non alphanumériques dans des noms de variables. En effet, ces caractères sont utilisés dans les noms des variables Unix spéciales. Ces variables sont réservées à des fonctions spécifiques.

Par exemple, le **$** caractère représente le numéro d'identification du processus, ou PID, du shell actuel −

```
$echo $$
```

La commande ci-dessus écrit le PID du shell actuel −

```
29949
```

Le tableau suivant montre un certain nombre de variables spéciales que vous pouvez utiliser dans vos scripts shell −

Sr.No.	Variable et description
1	**$ 0** Le nom de fichier du script actuel.
2	**$ n** Ces variables correspondent aux arguments avec lesquels un script a été invoqué. Ici **n** est un nombre décimal positif correspondant à la position d'un argument (le premier argument est $ 1, le deuxième argument est $ 2, et ainsi de suite).
3	**$#** Le nombre d'arguments fournis à un script.
4	**$*** Tous les arguments sont cités deux fois. Si un script reçoit deux arguments, $ * équivaut à $ 1 $ 2.
5	**$@** Tous les arguments sont cités individuellement deux fois. Si un script reçoit deux arguments, $ @ équivaut à $ 1 $ 2.
6	**$?** L'état de sortie de la dernière commande exécutée.
7	**$ $** Le numéro de processus du shell actuel. Pour les scripts shell, il s'agit de l'ID de processus sous lequel ils s'exécutent.
8	**$!** Numéro de processus de la dernière commande d'arrière-plan.

Arguments en ligne de commande

Les arguments de ligne de commande $ 1, $ 2, $ 3, ... $ 9 sont des paramètres de position, $ 0 pointant vers la commande, le programme, le script shell ou la fonction réels et $ 1, $ 2, $ 3, .

9 comme arguments de la commande.

Le script suivant utilise diverses variables spéciales liées à la ligne de commande −

```
#!/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 : $#"
```

Voici un exemple d'exécution pour le script ci-dessus −

```
$./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
```

Paramètres spéciaux $ * et $@

Il existe des paramètres spéciaux qui permettent d'accéder à tous les arguments de la ligne de commande à la fois. **$*** et **$@** les deux agiront de la même manière à moins qu'ils ne soient enfermés entre guillemets, **""**.

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.

L'état de sortie est une valeur numérique renvoyée par chaque commande à sa fin. En règle générale, la plupart des commandes renvoient un état de sortie de 0 si elles ont réussi, et 1 si elles ont échoué.

Certaines commandes renvoient des statuts de sortie supplémentaires pour des raisons particulières. Par exemple, certaines commandes différencient les types d'erreurs et renverront diverses valeurs de sortie en fonction du type spécifique de défaillance.

Voici l'exemple de la commande réussie −

```
$./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
$
```