Unix / Linux - Fonctions Shell

Dans ce chapitre, nous discuterons en détail des fonctions shell. Les fonctions vous permettent de décomposer la fonctionnalité globale d'un script en sous-sections logiques plus petites, qui peuvent ensuite être appelées à effectuer leurs tâches individuelles en cas de besoin.

L'utilisation de fonctions pour effectuer des tâches répétitives est un excellent moyen de créer **réutilisation du code**. Il s'agit d'une partie importante des principes de programmation modernes orientés objet.

Les fonctions Shell sont similaires aux sous-programmes, procédures et fonctions dans d'autres langages de programmation.

Création de fonctions

Pour déclarer une fonction, utilisez simplement la syntaxe suivante −

```
function_name () {
   list of commands
}
```

Le nom de votre fonction est **nom_fonction**, et c'est ce que vous utiliserez pour l'appeler d'ailleurs dans vos scripts. Le nom de la fonction doit être suivi de parenthèses, suivi d'une liste de commandes jointes à des accolades.

Exemple

L'exemple suivant montre l'utilisation de la fonction −

Démo en direct

```
#!/bin/sh

# Define your function here
Hello () {
   echo "Hello World"
}

# Invoke your function
Hello
```

Lors de l'exécution, vous recevrez la sortie suivante −

```
$./test.sh
Hello World
```

Passer des paramètres à une fonction

Vous pouvez définir une fonction qui acceptera les paramètres lors de l'appel de la fonction. Ces paramètres seraient représentés par **$ 1**, **$ 2** et ainsi de suite.

Voici un exemple où nous passons deux paramètres *Zara* et *Ali* puis nous capturons et imprimons ces paramètres dans la fonction.

Démo en direct

```
#!/bin/sh

# Define your function here
Hello () {
   echo "Hello World $1 $2"
}

# Invoke your function
Hello Zara Ali
```

Lors de l'exécution, vous recevrez le résultat suivant −

```
$./test.sh
Hello World Zara Ali
```

Retour des valeurs des fonctions

Si vous exécutez un **sortie** commande de l'intérieur d'une fonction, son effet n'est pas seulement de mettre fin à l'exécution de la fonction mais également du programme shell qui a appelé la fonction.

Si vous souhaitez simplement terminer l'exécution de la fonction, il existe un moyen de sortir d'une fonction définie.

En fonction de la situation, vous pouvez renvoyer n'importe quelle valeur de votre fonction à l'aide de **retour** commande dont la syntaxe est la suivante −

```
return code
```

Ici **code** peut être tout ce que vous choisissez ici, mais vous devez évidemment choisir quelque chose de significatif ou utile dans le contexte de votre script dans son ensemble.

Exemple

La fonction suivante renvoie une valeur 10 −

Démo en direct

```
#!/bin/sh

# Define your function here
Hello () {
   echo "Hello World $1 $2"
   return 10
}

# Invoke your function
Hello Zara Ali

# Capture value returnd by last command
ret=$?

echo "Return value is $ret"
```

Lors de l'exécution, vous recevrez le résultat suivant −

```
$./test.sh
Hello World Zara Ali
Return value is 10
```

Fonctions imbriquées

L'une des caractéristiques les plus intéressantes des fonctions est qu'elles peuvent s'appeler ainsi que d'autres fonctions. Une fonction qui s'appelle est connue comme ***fonction récursive***.

L'exemple suivant montre la nidification de deux fonctions −

Démo en direct

```
#!/bin/sh

# Calling one function from another
number_one () {
   echo "This is the first function speaking..."
   number_two
}

number_two () {
   echo "This is now the second function speaking..."
}
```

```
# Calling function one.
number_one
```

Lors de l'exécution, vous recevrez le résultat suivant −

```
This is the first function speaking...
This is now the second function speaking...
```

Appel de fonction depuis l'invite

Vous pouvez mettre des définitions des fonctions couramment utilisées dans votre ***.profile***. Ces définitions seront disponibles chaque fois que vous vous connectez et vous pouvez les utiliser à l'invite de commandes.

Alternativement, vous pouvez regrouper les définitions dans un fichier, par exemple ***test.sh***, puis exécutez le fichier dans le shell actuel en tapant −

```
$. test.sh
```

Cela a pour effet de provoquer des fonctions définies à l'intérieur ***test.sh*** à lire et à définir sur le shell actuel comme suit −

```
$ number_one
This is the first function speaking...
This is now the second function speaking...
$
```

Pour supprimer la définition d'une fonction du shell, utilisez la commande unset avec le **.f** option. Cette commande est également utilisée pour supprimer la définition d'une variable dans le shell.

```
$ unset -f function_name
```