Unix / Linux - Qu'est-ce que les coquillages?

A **Coquille** vous fournit une interface avec le système Unix. Il rassemble les commentaires de vous et exécute des programmes basés sur cette entrée. Lorsqu'un programme est terminé d'exécuter, il affiche la sortie de ce programme.

Shell est un environnement dans lequel nous pouvons exécuter nos commandes, programmes et scripts shell. Il existe différentes saveurs d'une coque, tout comme il existe différentes saveurs de systèmes d'exploitation. Chaque saveur de coque a son propre ensemble de commandes et de fonctions reconnues.

Invite de coquille

L'invite, **$**, qui s'appelle le **invite de commande**, est émis par le shell. Pendant que l'invite s'affiche, vous pouvez taper une commande.

Shell lit votre entrée après avoir appuyé **Entrer**. Il détermine la commande que vous souhaitez exécuter en regardant le premier mot de votre entrée. Un mot est un ensemble de caractères ininterrompu. Les espaces et les onglets séparent les mots.

Voici un exemple simple de **date** commande, qui affiche la date et l'heure actuelles −

```
$date
Thu Jun 25 08:30:19 MST 2009
```

Vous pouvez personnaliser votre invite de commande à l'aide de la variable d'environnement PS1 expliquée dans le didacticiel Environnement.

Types de coquillages

Dans Unix, il existe deux principaux types de coques −

- **Coquille de Bourne** − Si vous utilisez un shell de type Bourne, le **$** caractère est l'invite par défaut.
- **Coquille C** − Si vous utilisez un shell de type C, le caractère% est l'invite par défaut.

Le Bourne Shell a les sous-catégories suivantes −

- Coquille de Bourne (sh)
- Coquille de Korn (ksh)
- Bourne Again shell (bash)

- Coque POSIX (sh)

Les différentes coques de type C suivent −

- Coquille C (csh)
- Coquille TENEX / TOPS C (tcsh)

Le shell Unix original a été écrit au milieu des années 1970 par Stephen R. Bourne alors qu'il était aux AT&T Bell Labs dans le New Jersey.

La coque Bourne a été la première coque à apparaître sur les systèmes Unix, elle est donc appelée "la coque".

La coque Bourne est généralement installée comme **/bin / sh** sur la plupart des versions d'Unix. Pour cette raison, c'est le shell de choix pour l'écriture de scripts qui peut être utilisé sur différentes versions d'Unix.

Dans ce chapitre, nous allons couvrir la plupart des concepts Shell basés sur le Borne Shell.

Scripts shell

Le concept de base d'un script shell est une liste de commandes, qui sont répertoriées dans l'ordre d'exécution. Un bon script shell aura des commentaires, précédés de # signe, décrivant les étapes.

Il existe des tests conditionnels, tels que la valeur A est supérieure à la valeur B, des boucles nous permettant de parcourir des quantités massives de données, des fichiers pour lire et stocker des données, et des variables pour lire et stocker des données, et le script peut inclure des fonctions.

Nous allons écrire de nombreux scripts dans les sections suivantes. Ce serait un simple fichier texte dans lequel nous mettrions toutes nos commandes et plusieurs autres constructions requises qui indiquent à l'environnement shell quoi faire et quand le faire.

Les scripts et fonctions shell sont tous deux interprétés. Cela signifie qu'ils ne sont pas compilés.

Exemple de script

Supposons que nous créons un **test.sh** script. Notez que tous les scripts auraient **.sh** extension. Avant d'ajouter autre chose à votre script, vous devez alerter le système qu'un script shell est en cours de démarrage. Ceci est fait en utilisant le **Shebang** construire. Par exemple −

```
#!/bin/sh
```

Cela indique au système que les commandes qui suivent doivent être exécutées par le shell Bourne. *Ça s'appelle un shebang parce que le # le symbole est appelé un hachage, et le ! le symbole est appelé un coup*.

Pour créer un script contenant ces commandes, vous mettez d'abord la ligne shebang puis ajoutez les commandes −

```
#!/bin/bash
pwd
ls
```

Commentaires de Shell

Vous pouvez mettre vos commentaires dans votre script comme suit −

```
#!/bin/bash

# Author : Zara Ali
# Copyright (c) Tutorialspoint.com
# Script follows here:
pwd
ls
```

Enregistrez le contenu ci-dessus et rendez le script exécutable −

```
$chmod +x test.sh
```

Le script shell est maintenant prêt à être exécuté −

```
$./test.sh
```

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

```
/home/amrood
index.htm  unix-basic_utilities.htm  unix-directories.htm
test.sh    unix-communication.htm    unix-environment.htm
```

Remarque − Pour exécuter un programme disponible dans le répertoire courant, utilisez **./program_name**

Scripts Shell étendus

Les scripts Shell ont plusieurs constructions requises qui indiquent à l'environnement shell quoi faire et quand le faire. Bien sûr, la plupart des scripts sont plus complexes que celui ci-dessus.

Le shell est, après tout, un véritable langage de programmation, avec des variables, des structures de contrôle, etc. Peu importe la complexité d'un script, il ne s'agit toujours que d'une liste de commandes exécutées séquentiellement.

Le script suivant utilise le **lire** commande qui prend l'entrée du clavier et l'attribue comme valeur de la variable PERSON et l'imprime enfin sur STDOUT.

```
#!/bin/sh

# Author : Zara Ali
# Copyright (c) Tutorialspoint.com
# Script follows here:

echo "What is your name?"
read PERSON
echo "Hello, $PERSON"
```

Voici un exemple d'exécution du script −

```
$./test.sh
What is your name?
Zara Ali
Hello, Zara Ali
$
```