Unix / Linux - Types de boucles Shell

Dans ce chapitre, nous discuterons des boucles shell dans Unix. Une boucle est un puissant outil de programmation qui vous permet d'exécuter un ensemble de commandes à plusieurs reprises. Dans ce chapitre, nous examinerons les types de boucles suivants disponibles pour les programmeurs shell −

- La boucle while
- La boucle for
- La boucle jusqu'au
- La boucle de sélection

Vous utiliserez différentes boucles en fonction de la situation. Par exemple, le **pendant** la boucle exécute les commandes données jusqu'à ce que la condition donnée reste vraie; le **jusqu'au** la boucle s'exécute jusqu'à ce qu'une condition donnée devienne vraie.

Une fois que vous avez de bonnes pratiques de programmation, vous acquerrez l'expertise et, par conséquent, commencerez à utiliser une boucle appropriée en fonction de la situation. Ici, **pendant** et **pour** les boucles sont disponibles dans la plupart des autres langages de programmation comme **C**, **C + +** et **PERL**, etc.

Boucles de nidification

Toutes les boucles prennent en charge le concept de nidification, ce qui signifie que vous pouvez mettre une boucle à l'intérieur d'une autre ou de boucles différentes. Cette nidification peut aller jusqu'à un nombre illimité de fois en fonction de vos besoins.

Voici un exemple de nidification **pendant** boucle. Les autres boucles peuvent être imbriquées en fonction de l'exigence de programmation de la même manière −

Nidification pendant les boucles

Il est possible d'utiliser une boucle while dans le corps d'une autre boucle while.

Syntaxe

```
while command1 ; # this is loop1, the outer loop
do
   Statement(s) to be executed if command1 is true

   while command2 ; # this is loop2, the inner loop
   do
```

```
      Statement(s) to be executed if command2 is true
   done

   Statement(s) to be executed if command1 is true
done
```

Exemple

Voici un exemple simple de nidification en boucle. Ajoutons une autre boucle de compte à rebours à l'intérieur de la boucle que vous comptiez jusqu'à neuf −

```
#!/bin/sh

a=0
while [ "$a" -lt 10 ]    # this is loop1
do
   b="$a"
   while [ "$b" -ge 0 ]  # this is loop2
   do
      echo -n "$b "
      b=`expr $b - 1`
   done
   echo
   a=`expr $a + 1`
done
```

Cela produira le résultat suivant. Il est important de noter comment **echo -n** travaille ici. Ici **-n** option permet d'éviter d'imprimer un nouveau caractère de ligne.

```
0
1 0
2 1 0
3 2 1 0
4 3 2 1 0
5 4 3 2 1 0
6 5 4 3 2 1 0
7 6 5 4 3 2 1 0
8 7 6 5 4 3 2 1 0
9 8 7 6 5 4 3 2 1 0
```