UNIX - SHELL LOOP TYPES

Loops are a powerful programming tool that enable you to execute a set of commands repeatedly. In this tutorial, you would examine the following types of loops available to shell programmers:

- The while loop
- The for loop
- The until loop
- The select loop

You would use different loops based on dfferent situation. For example while loop would execute given commands until given condition remains true where as until loop would execute until a given condition becomes true.

Once you have good programming practice you would start using appropriate loop based on situation. Here while and for loops are available in most of the other programming languages like C, C++ and PERL etc.

Nesting Loops:

All the loops support nesting concept which means you can put one loop inside another similar or different loops. This nesting can go upto unlimited number of times based on your requirement.

Here is an example of nesting **while** loop and similar way other loops can be nested based on programming requirement:

Nesting while Loops:

It is possible to use a while loop as part of the body of another while loop.

Syntax:

```
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
```

Example:

Here is a simple example of loop nesting, let's add another countdown loop inside the loop that you used to count to nine:

```
#!/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
```

This will produce following result. It is important to note how **echo -n** works here. Here **-n** option let echo to avoid printing a new line character.

```
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
```