UNIX - SHELL LOOP CONTROL

So far you have looked at creating loops and working with loops to accomplish different tasks. Sometimes you need to stop a loop or skip iterations of the loop.

In this tutorial you will learn following two statements used to control shell loops:

1. The **break** statement
2. The **continue** statement

The infinite Loop:

All the loops have a limited life and they come out once the condition is false or true depending on the loop.

A loop may continue forever due to required condition is not met. A loop that executes forever without terminating executes an infinite number of times. For this reason, such loops are called infinite loops.

Example:

Here is a simple example that uses the while loop to display the numbers zero to nine:

```
#!/bin/sh

a=10

while [ $a -lt 10 ]
do
   echo $a
   a=`expr $a + 1`
done
```

This loop would continue forever because a is alway greater than 10 and it would never become less than 10. So this true example of infinite loop.

The break statement:

The **break** statement is used to terminate the execution of the entire loop, after completing the execution of all of the lines of code up to the break statement. It then steps down to the code following the end of the loop.

Syntax:

The following **break** statement would be used to come out of a loop:

```
break
```

The break command can also be used to exit from a nested loop using this format:

```
break n
```

Here **n** specifies the nth enclosing loop to exit from.

Example:

Here is a simple example which shows that loop would terminate as soon as a becomes 5:

```
#!/bin/sh

a=0

while [ $a -lt 10 ]
do
   echo $a
   if [ $a -eq 5 ]
   then
      break
   fi
   a=`expr $a + 1`
done
```

This will produce following result:

```
0
1
2
3
4
5
```

Here is a simple example of nested for loop. This script breaks out of both loops if var1 equals 2 and var2 equals 0:

```
#!/bin/sh

for var1 in 1 2 3
do
   for var2 in 0 5
   do
      if [ $var1 -eq 2 -a $var2 -eq 0 ]
      then
         break 2
      else
         echo "$var1 $var2"
      fi
   done
done
```

This will produce following result. In the inner loop, you have a break command with the argument 2. This indicates that if a condition is met you should break out of outer loop and ultimately from inner loop as well.

```
1 0
1 5
```

The continue statement:

The **continue** statement is similar to the break command, except that it causes the current iteration of the loop to exit, rather than the entire loop.

This statement is useful when an error has occurred but you want to try to execute the next iteration of the loop.

Syntax:

```
continue
```

Like with the break statement, an integer argument can be given to the continue command to skip commands from nested loops.

```
continue n
```

Here n specifies the nth enclosing loop to continue from.

Example:

The following loop makes use of continue statement which returns from the continue statement and start processing next statement:

```
#!/bin/sh

NUMS="1 2 3 4 5 6 7"

for NUM in $NUMS
do
   Q=`expr $NUM % 2`
   if [ $Q -eq 0 ]
   then
      echo "Number is an even number!!"
      continue
   fi
   echo "Found odd number"
done
```

This will produce following result:

```
Found odd number
Number is an even number!!
Found odd number
Number is an even number!!
Found odd number
Number is an even number!!
Found odd number
```