AWK - LOOPS

Like conditional statements, AWK also provides looping statements. It is used to execute set of action in repeated manner. The loop execution continues as long as the loop condition is true. This tutorial explain AWK's loops with suitable example.

For Loop

Below is syntax of the for loop:

```
for (initialisation; condition; increment/decrement)
    action
```

Initially for statement performs initialisation action, then it checks condition; if condition is true then it executes actions, after that it performs increment or decrement operation. The loop execution continues as long as the condition is true. For instance below example prints 1 to 5 numbers using for loop:

```
[jerry]$ awk 'BEGIN { for (i = 1; i <= 5; ++i) print i }'
```

On executing the above code, you get the following result:

```
1
2
3
4
5
```

While Loop

While loop keeps executing the action until a particular logical condition evaluates to true. Given below is syntax of the while loop:

```
while (condition)
    action
```

AWK first checks condition, if the condition is true then it executes action, this process repeats as long as the loop condition evaluates to true. For instance below example prints 1 to 5 numbers using while loop:

```
[jerry]$ awk 'BEGIN {i = 1; while (i < 6) { print i; ++i } }'
```

On executing the above code, you get the following result:

```
1
2
3
4
5
```

Do-While Loop

The do-while loop is similar to the while loop, except that the test condition is evaluated at the end of the loop. Given below is the syntax of the do while loop:

```
do
    action
while (condition)
```

In do-while loop action statement gets executed at least once even when condition statement evaluates to false. For instance below example prints 1 to 5 numbers using do-while loop:

```
[jerry]$ awk 'BEGIN {i = 1; do { print i; ++i } while (i < 6) }'
```

On executing the above code, you get the following result:

```
1
2
3
4
5
```

Break Statement

As name suggest it is used to end the loop execution. Here is the example which ends the loop when sum becomes greater that 50.

```
[jerry]$ awk 'BEGIN {sum = 0; for (i = 0; i < 20; ++i) { sum += i; if (sum > 50) break;
else print "Sum =", sum } }'
```

On executing the above code, you get the following result:

```
Sum = 0
Sum = 1
Sum = 3
Sum = 6
Sum = 10
Sum = 15
Sum = 21
Sum = 28
Sum = 36
Sum = 45
```

Continue Statement

Continues statement is used inside loop to skip to next iteration of a loop. It is useful when we wish to skip processing of some data inside the loop. For instance below example uses continue statement to print the even number between 1 to 20.

```
[jerry]$ awk 'BEGIN {for (i = 1; i <= 20; ++i) {if (i % 2 == 0) print i ; else continue}
}'
```

On executing the above code, you get the following result:

```
2
4
6
8
10
12
14
16
18
20
```

Exit Statement

It is used to stop the execution of the script. It accepts an integer as an argument which will be the exit status code for the AWK process. If no argument is supplied, exit returns status zero. Here is the example which stops the execution when sum becomes greater that 50.

```
[jerry]$ awk 'BEGIN {sum = 0; for (i = 0; i < 20; ++i) { sum += i; if (sum > 50)
exit(10); else print "Sum =", sum } }'
```

On executing the above code, you get the following result:

```
Sum = 0
Sum = 1
Sum = 3
Sum = 6
Sum = 10
Sum = 15
Sum = 21
Sum = 28
Sum = 36
Sum = 45
```

Let us check the return status of script.

```
[jerry]$ echo $?
```

On executing the above code, you get the following result:

```
10
```