Shell Programming - Loop

Amit Patel

LOOP

- Loops are a powerful programming tool that enable you to execute a set of commands repeatedly.
- We would examine the following types of loops available to shell programmers –
  - The while loop
  - The for loop
  - The until loop
  - The select loop

LOOP

- You would use different loops based on different 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++

WHILE LOOP

- The while loop enables you to execute a set of commands repeatedly until some condition occurs. It is usually used when you need to manipulate the value of a variable repeatedly.
- Syntax:

```
while command
do
   Statement(s) to be executed if command is true
done
```

- Here Shell command is evaluated. If the resulting value is true, given statement(s) are executed. If command is false then no statement would be not executed and program would jump to the next line after done statement.

WHILE LOOP

- The statement within the loop would executed till the exit status of the control command remains true. When the exit status of the control command turns out to be false, control passes to the first command that the follow the body of the while loop.
- The control command can be any valid unix command.
- The statement within the loop may be a single command or a group of command. It is necessary to put them between do and done.
- As a rule, while must have a control command that will return an exit status 1 (false), otherwise loop would be executed forever means becomes infinite loop.

WHILE LOOP

```
#!/bin/sh
a=0
while [ $a -lt 5]
do
   echo $a
   a=`expr $a + 1`
done
```

This will produce following result –

```
1
2
3
4
5
```

- Each time this loop executes, the variable a is checked to see whether it has a value that is less than 10.
- If the value of a is less than 10, this test condition has an exit status of 0. In this case, the current value of a is displayed and then a is incremented by 1.

FOR LOOP

- The for loop operate on lists of items. It repeats a set of commands for every item in a list.
- Syntax

```
for var in word1 word2 ... wordN
do
   Statement(s) to be executed for every word.
done
```

- Here var is the name of a variable and word1 to wordN are sequences of characters separated by spaces (words). Each time the for loop executes, the value of the variable var is set to the next word in the list of words, word1 to wordN.

FOR LOOP

```
#!/bin/sh
for var in  1 2 3 4 5
do
   echo $var
done
```

This will produce following result −

```
1
2
3
4
5
```

UNTIL LOOP

- The while loop is perfect for a situation where you need to execute a set of commands while some condition is true. Sometimes you need to execute a set of commands until a condition is true.
- Syntax

```
until command
do
   Statement(s) to be executed until command is true
done
```

- Here Shell command is evaluated. If the resulting value is false, given statement(s) are executed. If command is true then no statement would be not executed and program would jump to the next line after done statement.

UNTIL LOOP

```
#!/bin/sh
a=1
until [ ! $a -lt 5 ]
do
   echo $a
   a=`expr $a + 1`
Done
```

This will produce following result –

```
1
2
3
4
5
```

SELECT LOOP

- The select loop provides an easy way to create a numbered menu from which users can select options. It is useful when you need to ask the user to choose one or more items from a list of choices.

```
select var in word1 word2 ... wordN
do
   Statement(s) to be executed for every word.
done
```

- Here var is the name of a variable and word1 to wordN are sequences of characters separated by spaces (words). Each time the for loop executes, the value of the variable var is set to the next word in the list of words, word1 to wordN.
- For every selection a set of commands would be executed with-in the loop.

SELECT LOOP

```
#!/bin/ksh
select DRINK in tea cofee water juice appe all none
do
  case $DRINK in
    tea|cofee|water|all)
      echo "Go to canteen"
      ;;
    juice|appe)
      echo "Available at home"
    ;;
    none)
      break
    ;;
    *) echo "ERROR: Invalid selection"
    ;;
  esac
```

SELECT LOOP

```
$./test.sh
1) tea
2) cofee
3) water
4) juice
5) appe
6) all
7) none
#? juice
Available at home
#? none
$
```

SELECT LOOP

```
$PS3="Please make a selection => " ; export PS3
$./test.sh
1) tea
2) cofee
3) water
4) juice
5) appe
6) all
7) none
Please make a selection => juice
Available at home
Please make a selection => none
$
```