Loops in Shell Scripts

Loop defined as:
"*Computer can repeat particular instruction again and again, until particular condition satisfies. A group of instruction that is executed repeatedly is called a loop.*"
Bash supports:

- for loop
- while loop

Note that in each and every loop,
(a) First, the variable used in loop condition must be initialized, then execution of the loop begins.
(b) A test (condition) is made at the beginning of each iteration.
(c) The body of loop ends with a statement that modifies the value of the test (condition) variable.

for Loop

Syntax:

```
            for { variable name } in { list }
            do
            execute one for each item in the list until the list is not
finished (And repeat all statement between do and done)
            done
```

Before try to understand above syntax try the following script:

```
$vi for2.sh
for i in 1 2 3 4 5
do
echo "Welcome $i times"
done
```

Run above script as follows:

/bin/sh for1.sh

How it works:
The for loop first creates i variable and assigned a number to i from the list of number from 1 to 5, The shell execute echo statement for each assignment of i. (This is usually know as iteration) This process will continue until all the items in the list were not finished, because of this it will repeat 5 echo statements. To make you idea more clear try following script:

Even you can use following syntax:

Syntax:

```
         for (( expr1; expr2; expr3 ))
         do
               .....
                          ...
               repeat all statements between do and
               done until expr2 is TRUE
         Done
```

In above syntax BEFORE the first iteration, ***expr1*** is evaluated. This is usually used to initialize variables for the loop.
All the statements between do and done is executed repeatedly UNTIL the value of ***expr2*** is TRUE.
AFTER each iteration of the loop, ***expr3*** is evaluated. This is usually use to increment a loop counter.

```
$vi for2.sh
for ((  i = 0 ;  i <= 5;  i++  ))
do
  echo "Welcome $i times"
done
```

Run the above script as follows:
$ /bin/sh for2.sh

Welcome 0 times
Welcome 1 times
Welcome 2 times
Welcome 3 times
Welcome 4 times
Welcome 5 times

In above example, first expression (i = 0), is used to set the value variable **i** to zero.
Second expression is condition i.e. all statements between do and done executed as long as expression 2 (i.e continue as long as the value of variable **i** is less than or equel to 5) is TRUE.
Last expression **i++** increments the value of **i** by 1 i.e. it's equivalent to i = i + 1 statement.

Nesting of for Loop

As you see the if statement can nested, similarly loop statement can be nested. You can nest the for loop. To understand the nesting of for loop see the following shell script.

```
$ vi nestedfor.sh
for (( i = 1; i <= 5; i++ ))      ### Outer for loop ###
do

    for (( j = 1 ; j <= 5; j++ )) ### Inner for loop ###
    do
          echo -n "$i "
    done

  echo "" #### print the new line ###
```

```
done
```

Run the above script as follows:
$ /bin/sh nestefor.sh

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

Here, for each value of **i** the inner loop is cycled through 5 times, with the varible **j** taking values from 1 to 5. The inner for loop terminates when the value of **j** exceeds 5, and the outer loop terminets when the value of **i** exceeds 5.

while loop

Syntax:

```
            while [ condition ]
            do
                  command1
                  command2
                  command3
                  ..
                  ....
             done
```

Loop is executed as long as given condition is true. For e.g.. Above for loop program (shown in last section of for loop) can be written using while loop as:

WRITE A SHELL SCRIPT TO GENERATE FIBONACCI NUMBERS FROM 1 TO N

```
$ vi fibonacci.sh
echo "Enter n for Fibonacci series: - "
read n
echo "Fibonacci series is: - "
echo "0"
echo "1"
i=0
j=1
cnt=2
while [ $cnt –le $n ]
do
k=`expr $i + $j`
i= $j
```

```
j= $k
echo $k
cnt=`expr $cnt + 1`
done
```

Run the above script as follows:

$ /bin/sh `fibonacci.sh`

<u>Output –</u>

```
Enter n for Fibonacci Series:-
10
Fibonacci Series is:-
0
1
1
2
3
5
8
```

Reference:

Beginning Linux Programming by NEIL MATTHEW published by SPD