Unix Shell Scripting

Loops

Loops helps us to repeat a block of code several times.
They are 4 types of loops which can be used in Shell scripting and they are

1. The while loop
2. The for loop
3. The until loop
4. The select loop

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.

While Loop

The while loop enables you to execute a set of commands repeatedly as long as the condition remains true.

Syntax

```
While [   <Condition>   ]
do
Statement(s) to be executed if condition is true
Done
```

Eg:

```
#!/bin/bash
a=0
while [ $a -lt 10 ]
do
echo $a
a=`expr $a + 1`
done
```

Until Loop

The until loop enables you to execute a set of commands repeatedly as long as the condition remains false.

Syntax

```
Until  [   <condition>   ]
 do
Statement(s) to be executed  as long as condition is false
Done
```

Example

Here is a simple example that uses the until loop to display the numbers zero to nine −

```
#!/bin/bash
a=0
until [  $a -gt 10 ]
do
echo $a
a=`expr $a + 1`
done
```

For Loop

For Loop iterates for each item given in the list and stop.

Syntax

```
for var in <list of values (items) separated  by space>
do
Statement(s) to be executed for every item in the list.
Done
```

Example1

```
#!/bin/sh
for  N in 0 1 2 3 4 5 6 7 8 9
do
echo $N
Done
```

Example2

```
#!/bin/sh
for FILENAME in $HOME/
do
echo $FILENAME
done
```

Select Loop

The *select* loop provides an easy way to create a numbered menu system 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.

Syntax

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

Example

```
#!/bin/bash

select CAR in  BMW LIMO TAYOTA none
do

read -p "Enter Kilometers " KM

case $CAR in
        BMW)  rate=50 ;;
        LIMO)  rate=70 ;;
        TAYOTA)  rate=40 ;;
        none)  break ;;
        *)
          echo invalid choice
          rate=0
            ;;
esac
AMT=`expr $rate \* $KM`
echo "Bill Amt is  $AMT"
done
```

Loop Control Statements

Loop control statements help in altering the iterations. There are 2 loop control statements

1. Break
2. Continue

The break statement

The **break** statement is used to terminate the execution of the entire loop

Syntax

Break

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.

Syntax

continue

ARRAYS

An array variable can hold multiple values at the same time. Arrays provide a method of grouping a set of data. Instead of creating a new name for each variable that is required, you can use a single array variable that stores multiple values.

Initializing an Array

array_name=(value1   value2   ... valueN)

Accessing a single element

${array_name[index]}

Eg :   echo $array_name[0]}   it will print the first element from array

echo   **${name[*]}**   prints all elements from array

echo   **${array_name[@]}** prints all elements from array same as above