Conditional Statements

The conditional statements is used in any programming language to do any decision making tasks.
There are total 5 conditional statements statements which can be used in bash programming

i. if statement
ii. if-else statement
iii. if-elif-else-fi statement (Else if ladder)
iv. if-then-else-if-then-fi-fi (Nested if)
v. Switch statement.

i. if statement :- This block will process if specified cond^{n} is true if specified condition is not true in if part then else part will be execute.

Syntax

```
if [expression]
then
    Statement 1
else
fi
```

ii. if-else statement :- if specified condition is not true true in if part then else part will be execute.

Syntax

```
if [expression]
then
    Statement 1
else
    Statement 2
fi
```

iii. Else if ladder :- To use multiple conditions in one if-else block, then elif keyboard is used in shell. If expression 1 is true then it executes statement 1 and 2, and this process continues. If none of the condition is true then it processes else part.

Syntax

```
if [expression1]
then
    Statement 1
    Statement 2
    .
    .
    .
else [expression 2]
then
    statement 3
    statement 4
    .
    .
else
    statement 5
fi
```

iv. Nested if :- Nested if-else block can be used when, one condition is satisfies then it again checks another condition. In the syntax, if expression 1 is false then it processes else part, and again expression 2 will be check.

Syntax :-

```
if [expression1]
then
    statement1
    Statement 2
    :
else
    if [expression2]
    then
        Statement 3
        .
    fi
fi
```

v. Switch statements :- case statement works as a switch statement if specified value match with a the pattern then it will execute a block of that particular pattern.

When a match is found all of the associated statements until the double semicolon (;;) is executed.

A case will be terminated when the last command is executed.

If there is no match, the exit status of the case is zero.

Syntax

```
case in
    Pattern 1) statement1 ;;
    Pattern n) statement2 ;;
esac
```

Conditional operators

-eq	equal equivalent
-lt	less than
-gt	greater than
==	equals to
!=	not equals to
!	false
-d	check existence of a directory
-e	check existence of file
-r	check existence of file and read permission
-w	check existence of file & write permission
-x	check existence of file & execute permission

Looping statement in Linux Shell scripting

There are 3 total looping statements which can be used in bash programming.

1. while statement
2. for statement
3. until statement

1. **while statement** :- Here command is evaluated and based on the result loop will executed, if command raise to false then loop will be terminated.

Syntax

```
while [condition]
do
    statement 1
    statement 2
done
```

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

Syntax :-

```
for [variable_name] in ...
do
    Statement 1
    Statement 2
    Statement n
done
```

iii. until statement :- The until loop is useful when you need to execute a set of commands until a condition is true.

Syntax

```
until [condition]
do
    Statement 1
    Statement 2
done
```

Loop control statements

i. break statement
ii. continue statement

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

the end of the loop.

Syntax

- break (to come out of a loop)

to come out of nested loop

break n

(n specifies the n^{th} enclosing loop to exit from)

ii. continue statement

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

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

Syntax

i. continue

ii. continue n

(n specifies the n^{th} enclosing loop to continue from)

Examples

i. Implementing while loop (printing from 0 to 9)

```
a=0
while [$a -lt 10]
do
    echo $a

    a=`expr $a + 1`
done
```

ii. Implementing for loop with break statement

```
for a in 1 2 3 4 5 6 7 8 9 10
do
    if [$a == 5]
    then
        continue
    fi
    echo "Iteration no $a"
done
```

iii. Implementing until loop (print 0 to 10)

```
a=0
until [ $a -gt 10]
do
    echo $a
    a=`expr $a + 1`
done
```