Content

- Conditional statements in Unix
- If statement
- If else statement
- Break and Continue statement

Introduction to Conditional Statement

- Decision making is one of the most fundamental concepts of computer programming.
- Like in any other programming language, if, if..else, if..elif..else and nested if statements in Shell can be used to execute code based on a certain condition.
- Conditional statement is the next form of control statement that allows Shell to execute statements in a controlled way and make the right choice.

Conti…

- Unix provides a number of ways for conditionally executing the other commands.

✓ if statement

✓ if-else statement

Unix provides a number of relational operators. These can be used to compare numeric values.

-lt: less than

-le: less than or equal to

-gt: greater than

-ge: greater than or equal to

-eq: equal to

-ne: not equal to

if statement

Syntax:

```
if [ expression/control command ] ;
then
statements
fi
```

- The Shell *expression* is evaluated in the above syntax. If the resulting value is *true*, given *statement(s)* are executed. If the *expression* is *false*, then no statement will be executed.

Conti…

- **If** statements (closely related, **case** statements) allow us to make decisions in our Shell scripts.
- They allow us to decide whether or not to run a piece of code based upon conditions that we may set.
- If statements, combined with loops allow us to make much more complex scripts which may solve larger tasks.

Conti…

```
#Initializing two variables (test.sh)
a=10
b=20

#Check whether they are equal
if [ $a == $b ] ;
then
    echo "a is equal to b"
fi

#Check whether they are not equal
if [ $a != $b ] ;
then
    echo "a is not equal to b"
fi
```

Output:

```
$bash -f test.sh
a is not equal to b
```

if-else statement

- If specified condition is not true in if part then else part will be execute.

Syntax

```
if [ expression/control command ];
 then
 statements
 else
Statements
 fi
```

Conti…

```
#Initializing two variables
a=20
b=20

if [ $a == $b ]
then
    #If they are equal then print this
    echo "a is equal to b"
else
    #else print this
    echo "a is not equal to b"
fi
```

Output

```
$bash -f test.sh
a is equal to b
```

Break statement

- All statement inside the loop executed as long as some condition are true.
- If break placed inside the loop, when loop reach the break statement it will terminated out from the loop.

The syntax of the break statement takes the following form:

break [n]

Conti…

Example for break:

```
i=0;
while [ $i -lt 5 ]
do
        echo "Number: $i"
        i=`expr $i + 1`
if [ $i -eq 2 ];
then
        break
fi
done
echo "All Done!“
```

output:

```
Number: 0
Number: 1
All Done!
```

Continue statement

- If continue placed inside the loop, when loop reach the continue statement it will not execute next lines of the loop and it will go to the next iteration.   OR
- The continue statement skips the remaining commands inside the body of the enclosing loop for the current iteration and passes program control to the next iteration of the loop.
  - ➢ The syntax of the continue statement is as follows:

continue [n]

Conti…

Example for continue:

```
i=0;
while [ $i -lt 5 ]
do
        if [ $i -eq 2 ];
        then
                i=`expr $i + 1`
                continue
        fi
 echo "Number: $i"
 i=`expr $i + 1`
 done
            echo "All Done!"
```

```
Output:
Number: 1
Number: 3
Number: 4
Number: 5
All Done!
```

For loop

```
for var1 in 1 2 3
do
        echo $var1
done
```

```
a="1 2 3 4 5"
for NUM in $a
do
        for NUM1 in $a
        do
                echo -n "*"
        done
        echo
done
```

Case statement

```
ch=0
while [ $ch -le 5 ]
do
echo "1 : Addition"
echo "2 : subtraction"
echo "3 : multiplication"
echo "4 : division"
echo "5 : modulus"
echo "6 : exit"
echo "enter your choice:"
read ch
        if [ $ch -le 5 ] ; then
                echo -n "enter 1st no:"
                read no1
                echo -n "enter 2nd no:"
                read no2
        else
                echo "invalid choice"
        fi
Clear
case "$ch" in
        1)
echo " addition "
echo " ans = " ` expr $no1 + $no2 ` ;;
2)
                echo " subtration "
                echo " ans = " ` expr $no1 - $no2
` ;;
        3)
                echo " multiplication "
                echo " ans = " ` expr $no1 \*
$no2 ` ;;
        4)
                echo " division "
                echo " ans = " ` expr $no1 /
$no2 ` ;;
        5)
                echo " modulus "
                echo " ans = " ` expr $no1 %
$no2 ` ;;
        6)
                exit
        esac
Done
```

Until loop

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

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

- To execute expression
- `expr $var1 + $ var2`

Vi editor

- I for insert text – to go to insert mode
- Esc for going back to command mode

- : end
- w - to save
- q – quit

- :wq - Save and Quit