Lab0

II.SHELL PROGRAMMING

Shell programs are files that holds one or more linux and unix commands. Before you execute a file, you should give execution permission to the file

A.Input and Output

- **echo:**used for standard output, You can command to display text or value of variable.

  echo [options] [string variables..]
  Options:
  -n Do not output the trailing new line.
  -e Enable interpretation of the following backslash escaped characters in the strings:
  \a alert (bell)
  \b backspace
  \c suppress trailing new line
  \n new line
  \r carriage return
  \t horizontal tab
  \\ backslash

  Ex.Script that gives the name of the user, date and calender
  $cat >first
  echo "Hello $USER"
  echo –e "Today is \c ";date
  echo "Calendar"
  cal
  ^D
  $chmod 100 first
  $./first

  In shell programming declaring a varible before using is not necessary. The values of the variable can be seen using $ sign before its name

  Ex.A script for writing Hello Linux
  $ cat > Hello
  msg=”Hello Linux”
  echo $msg
  ^D
  $chmod 700 Hello
  $ ./Hello
  Hello Linux
  $

- **read:**get input from the user

  Ex.A simple program to get a number and a chacter from the user
  $cat >readExample

```
echo input a number
read num
echo input a character
read char
echo your number is $num
echo your char is $char
^D
$chmod 700 readExample
$./readExample
```

B.Arithmetic Operations

Operations are / ,- ,+ ,% ,* ,= ,>= ,<= ,< ,> ,& ,| ,(,), a,o,!

expr op1 operator op2

Ex:
$expr 342 + 321
$ expr 100 / 10
$ expr 20 % 3
$ echo `expr 6 \* 30`
Note:For multiplication use *

Ex:
$x='expr 13 * 30 '
$echo $x
$390

- **test:** You can use to test, if the test condition true, it returns true

  test –d filename: filename is a directory or not
  test –f filename: filename is a file or not
  test –r filename: is there a read permission or not
  test –w filename: is there a write permission or not
  test –x filename: is there a execution permission or not

  test str1=str2 : is str1 equal to str2
  test str1!=str2 : is str1 not equal to str2
  test n1 –eq n2 : is n1 equal to n2
  test n1 –le n2 : is n1 less than or equal to n2
  test n1 –ge n2 : is n1 greater than or equal to n2
  test n1 –ne n2 : is n1 not equal n2
  test n1 –lt n2 : is n1 less than n2

  Note: you can not directly use test under the shell, test should be part of a program

C.If-Else Statement and Control Operations

Sytax of the If statement

```
if condition
```

then

command1

command2

elif

then

command3

command4

else

command5

fi

Ex.

```
if test –x filename

then

    echo execution permission

else

    echo no execution permission

fi
```

Ex.

```
Echo enter a number
Read num1
Echo enter another number
Read num2
if test $num1 –lt $num2
then
        echo $num1 is less than $num2
else
        echo $num1 is greater than or equal to $num2
fi
```

D.Case Statement

Syntax of the case statement:

```
case variable-name in
```

```
        option1 )
            command1
            command2
        ;;
        option2 )
            command3
            command4
        ;;
        *)
            command5
        ;;
        esac
```

Ex

```
clear
echo “1.Clean the Screen”
echo “2.Show the files with details”
echo “3.Show the files with hidden files”
echo “4.Show the files with details and hidden files”
echo –n “Enter your choice”
read choice
case $choice in
1)
        clear ;;
2)
        ls –l ;;
3)
        ls –a ;;
4)
        ls –al ;;
*)
        echo “wrong choice”
esac
```

E. Loops

- **while-do loop**

While-do Statement

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

Ex::

```
var=0
while test $var –lt 100
do
    echo “var=$var”
    var =’expr $var+1’
done
```

- **For-Do Loop**

For Statement

```
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
```

Ex.

```
for car in ford fiat bmw
do
    echo $car
done
```

Ex

```
for i in 10 9 8 7 6 5 4 3 2 1 fire
do
    echo “t minus $i sec”
done
```

Ex. List all worlds in the file called myfile

```
for i in ‘cat myfile’
do
    echo $i
done
```

Ex. List all files in the current directory

```
for i in *
do
    echo $i
done
```