Lab2. Shell Programming

Shell programs are a file that holds the one or more unix or linux commands.

A. Input and Output

echo: this command is used for standard output

Use echo command to display text or value of variable.

echo [options] [string, variables...]
Displays text or variables value on screen.
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

For e.g. **$ echo -e "An apple a day keeps away \a\t\tdoctor\n"**

Ex1.

```
clear
echo "Knowledge is Power"
```

ex2. Script to print user information who currently login , current date & time

```
clear
echo "Hello $USER"
echo –e "Today is \c ";date
echo –e "Number of user login : \c" ; who | wc -l
echo "Calendar"
cal
```

B. Variables:

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

```
$ cat > myprog
message="Hello world"
echo $message
^D
$ ./myprog
Hello world
$
```

read: this command is used for standard input

```
$ cat > myprog
echo input number
read number
echo your number is: $number
^D
$chmod 700 myprog
$ ./myprog
input number
5
your number is: 5
$
```

ex2.

```
cat > myprog2
echo input a character
read ch
echo "your number is: $a"
^D
```

C. Arithmetic Operations

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

expr op1 math-operator op2

Examples:

```
$ expr 1 + 3
$ expr 2 - 1
$ expr 10 / 2
$ expr 20 % 3
$ expr 10 \* 3
$ echo `expr 6 + 3`
```

ex.

```
typeset i result
a=100;b=50
result=$a \*  $b
echo $result
```

ex.

```
a=3.749
b=22.34
echo "$a \* $b" | bc
```

ex.

```
x=`expr 3 \* 5`
echo $x
```

D. Testing

test: test command is used for testing operations. If test result is true, it returns true.

test –d myfile : test the file called myfile whether it is a directory or not

test –f myfile : test the file called myfile whether it is a file or not

test –r myfile : test the file called my file whether it has read permission or not

test –w myfile : test the file called my file whether it has write permission or not

test –x myfile : test the file called my file whether it has 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

E. If Statements

Syntax of the if statement:

```
if condition
then
    command1
    command2
elif
then
    command3
    command4
else
     command5
fi
```

example:

```
if test –r file1
then
    echo read permission
else
    echo no read permission
fi
```

example:

```
echo "select a number between 0 and 20"
read number
if test $number -lt 10
then
      echo "Selected number is less than 10"
elif
      echo "Selected number is greater than or equal to 10"
fi
```

F. Case Statement

Syntax of the case statement:

```
case variable-name in
option1)
    command1
    command2
;;
option2)
    command3
    command4
;;
*)
    command5
;;
esac
```

example:

```
clear
echo "1. Clear screan"
echo "2. List logined users"
echo "3. List the files"
echo -n "Enter your choice:"
read choice
case $choise in
1)
   clear  ;;
2)
   who
   ;;
3)
   ls -al  ;;
*)
   echo "wrong choice"
esac
```

G. 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
```

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

ex: Lists all words in the file called mylist

```
for i in 'cat mylist'
do
       echo $i
done
```

example: List all files in the current directory

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

H. While Statement

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

```
i=1
while test $i –le 10
do
          echo "i= $i"
          i=`expr $i + 1`
done
```

Example:

```
clear
choice=0;
while $choice -ne -1
do
   echo "1. Clear screan"
   echo "2. List logined users"
   echo "3. List the files"
   echo -n "Enter your choice:"
   read choice
   case $choise in
   1)
      clear  ;;
   2)
      who
      ;;
   3)
      ls -al  ;;
   *)
      echo "wrong choice"
   esac
done
```

Question: Write a script to find out biggest number from the given three numbers. Numbers are supplies as command line argument. (Hw: Print error if sufficient arguments are not supplied.)