Exp. No. 4 **Shell Programming**

Date:

Aim

To write simple shell scripts using shell programming fundamentals.

The activities of a shell are not restricted to command interpretation alone. The shell also has rudimentary programming features. Shell programs are stored in a file (with extension .**sh**). Shell programs run in interpretive mode. The original UNIX came with the Bourne shell (**sh**) and it is universal even today. C shell (**csh**) and Korn shell (**ksh**) are also widely used. Linux offers Bash shell (**bash**) as a superior alternative to Bourne shell.

Preliminaries

1. Comments in shell script start with **#**.
2. Shell variables are loosely typed i.e. not declared. Variables in an expression or output must be prefixed by **$**.
3. The **read** statement is shell's internal tool for making scripts interactive.
4. Output is displayed using **echo** statement.
5. Expressions are computed using the **expr** command. Arithmetic operators are `+ - * / %`. Meta characters *** ( )** should be escaped with a ****.
6. The shell scripts are executed

   $ `sh` *filename*

Decision-making

Shell supports decision-making using **if** statement. The **if** statement like its counterpart in programming languages has the following formats.

```
if [ condition ]
then
   statements
fi
```

```
if [ condition ]
then
   statements
else
   statements
fi
```

```
if [condition ]
then
   statements
elif [ condition ]
then
   statements
.. .
else
   statements
fi
```

The set of relational operators are `-eq -ne -gt -ge -lt -le` and logical operators used in conditional expression are `-a -o !`

Multi-way branching

The `case` statement is used to compare a variables value against a set of constants. If it matches a constant, then the set of statements followed after `)` is executed till a `;;` is encountered. The optional *default* block is indicated by `*`. Multiple constants can be specified in a single pattern separated by `|`.

```
case variable in
   constant1)
        statements ;;
   constant2)
        statements ;;
    ...
        *)
        statements
esac
```

Loops

Shell supports a set of loops such as **for**, **while** and **until** to execute a set of statements repeatedly. The body of the loop is contained between **do** and **done** statement.

The **for** loop doesn't test a condition, but uses a list instead.

```
for variable in list
do
        statements
done
```

The **while** loop executes the *statements* as long as the condition remains true.

```
while [ condition ]
do
        statements
done
```

The **until** loop complements the while construct in the sense that the *statements* are executed as long as the condition remains false.

```
until [ condition ]
do
        statements
done
```

A) Swapping values of two variables

```
# Swapping values – swap.sh
echo -n  "Enter value for A : "
read a
echo -n  "Enter value for B : "
read b
t=$a
a=$b
b=$t
echo "Values after Swapping"
echo "A Value is $a and B Value is $b"
```

Output

```
$ sh swap.sh
Enter value for A : 12
Enter value for B : 23
Values after Swapping
A Value is 23 and B Value is 12
```

B) Farenheit to Centigrade Conversion

```
# Degree conversion – degconv.sh
echo -n "Enter Fahrenheit : "
read f
c=`expr \( $f - 32 \) \* 5 / 9`
echo "Centigrade is : $c"
```

Output

```
$ sh degconv.sh
Enter Fahrenheit : 213
Centigrade is : 100
```

C) Biggest of 3 numbers

```
# Biggest – big3.sh
echo -n "Give value for A B and C: "
read a b c
if [ $a -gt $b -a $a -gt $c ]
then
   echo "A is the Biggest number"
elif [ $b -gt $c ]
then
   echo "B is the Biggest number"
else
   echo "C is the Biggest number"
fi
```

Output

```
$ sh big3.sh
Give value for A B and C: 4 3 4
C is the Biggest number
```

D) Grade Determination

```
# Grade – grade.sh
echo -n "Enter the mark : "
read mark
if [ $mark -gt 90 ]
then
   echo "S Grade"
elif [ $mark -gt 80 ]
then
   echo "A Grade"
elif [ $mark -gt 70 ]
then
   echo "B Grade"
elif [ $mark -gt 60 ]
then
   echo "C Grade"
elif [ $mark -gt 55 ]
then
   echo "D Grade"
elif [ $mark -ge 50 ]
then
   echo "E Grade"
else
   echo "U Grade"
fi
```

Output

```
$ sh grade.sh
Enter the mark : 65
C Grade
```

E) Vowel or Consonant

```
# Vowel  - vowel.sh
echo -n "Key in a lower case character : "
read choice
case $choice in
     a|e|i|o|u) echo "It's a Vowel";;
             *) echo "It's a Consonant"
esac
```

Output

```
$ sh vowel.
Key in a lower case character : e
It's a Vowel
```

F) Simple Calculator

```
# Arithmetic operations — calc.sh
echo -n "Enter the two numbers : "
read a b
echo " 1. Addition"
echo " 2. Subtraction"
echo " 3. Multiplication"
echo " 4. Division"
echo -n "Enter the option : "
read option
case $option in
   1) c=`expr $a + $b`
      echo "$a + $b = $c";;
   2) c=`expr $a - $b`
      echo "$a - $b = $c";;
   3) c=`expr $a \* $b`
      echo "$a * $b = $c";;
   4) c=`expr $a / $b`
      echo "$a / $b = $c";;
   *) echo "Invalid Option"
esac
```

Output

```
$ sh calc.sh
Enter the two numbers : 2 4
 1. Addition
 2. Subtraction
 3. Multiplication
 4. Division
Enter the option : 1
2 + 4 = 6
```

G) Multiplication Table

```
# Multiplication table – multable.sh
clear
echo -n "Which multiplication table? : "
read n
for x in 1 2 3 4 5 6 7 8 9 10
do
   p=`expr $x \* $n`
   echo -n "$n X $x = $p"
   sleep 1
done
```

Output

```
$ sh multable.sh
Which multiplication table? : 6
6 X 1 = 6
6 X 2 = 12
.....
```

H) Number Reverse

```
# To reverse a number – reverse.sh
echo -n "Enter a number : "
read n
rd=0
while [ $n -gt 0 ]
do
   rem=`expr $n % 10`
   rd=`expr $rd \* 10 + $rem`
   n=`expr $n / 10`
done
echo "Reversed number is $rd"
```

Output

```
$ sh reverse.sh
Enter a number : 234
Reversed number is 432
```

I) Prime Number

```
# Prime number – prime.sh
echo -n "Enter the number : "
read n
i=2
m=`expr $n / 2`
until [ $i -gt $m ]
do
   q=`expr $n % $i`
   if [ $q -eq 0 ]
   then
       echo "Not a Prime number"
       exit
   fi
   i=`expr $i + 1`
done
echo "Prime number"
```

Output

```
$ sh prime.sh
Enter the number : 17
Prime number
```

Result

Thus shell scripts were executed using different programming constructs