BHARTIYA INSTITUTE OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

4CS4-24: Linux Shell Programming Lab Manual

1. Shell Programming: Shell script based on control structure- If-then-fi, if-thenelse-if, nested if-else, to find:

Conditional Statements: There are total 5 conditional statements which can be used in bash programming

1. if statement
2. if-else statement
3. if..elif..else..fi statement (Else If ladder)
4. if..then..else..if..then..fi..fi..(Nested if)
5. switch statement

if statement

This block will process if specified condition is true.

Syntax:

```
if [ expression ]

then

  statement

fi
```

if-else statement

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

Syntax

```
if [ expression ]

then

  statement1

else
```

```
  statement2

fi
```

if..elif..else..fi statement (Else If ladder)

To use multiple conditions in one if-else block, then elif keyword is used in shell. If expression1 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

  statement1

  statement2

  .

  .

elif [ expression2 ]

then

  statement3

  statement4

  .

  .

else

  statement5

fi
```

if..then..else..if..then..fi..fi..(Nested if)

Nested if-else block can be used when, one condition is satisfies then it again checks another condition. In the syntax, if expression1 is false then it processes else part, and again expression2 will be check.

Syntax:

```
if [ expression1 ]

then

  statement1

  statement2

  .

else

  if [ expression2 ]

  then

    statement3

    .

  fi

fi
```

a) Shell Program to find Largest of Three Numbers

```
clear

echo "Enter first number: "

read a

echo "Enter second number: "

 read b

echo "Enter third number: "

read c
```

```
if [ $a -gt $b ] && [ $a -gt $c ]
then
 echo "$a is greater"
elif [ $b -gt $a ] && [ $b -gt $c ]
then
echo "$b is greater"
 elif [ $c -gt $a ] && [ $c -gt $b ]
then
echo "$c is greater"
fi
```

OUTPUT

Enter first number: 6

Enter second number:7

Enter third number:9

9 is greater

b) To find a year is leap year or not.

```
echo “Enter the year”
read year
chk=`expr $year % 4`
if [ $chk –eq 0 ]
then
echo “ $year is a leap year ”
else
echo “ $year is a not leap year ”
fi
```

Enter the year
1995
year is not leap year

c) Linux program to input angles of a triangle and find out whether it is valid triangle or not

```
echo "enter  angle A"
read A
echo "enter angle B"
read B
echo "enter angle C"
read C
# sum all three angles
d=$((A+B+C))
if [ $A -eq 0 -o $B -eq 0 -o $C -eq 0 ]
then
echo "Enter angles greater than zero"
else
   if [ $d == 180 ];
   then
   echo "valid traingle"
   else
   echo "not a valid traingle"
   fi
fi
```

OUTPUT

enter angle A 66

enter angle B 24

enter angle C 90

valid triangle

d) To check whether a character is alphabet, digit or special character.

```
char=""

echo -n "Enter a one character : "
read char
if [ -z $(echo $char | sed -e 's/[0-9]//g') ]
then
 echo "$char is Number/digit"
elif [ -z $(echo $char | sed  -e 's/[A-Z]//g') ] # find out if character is upper
then
      echo "$char is UPPER character"

elif [ -z $(echo $char | sed -e 's/[a-z]//g') ] # find out if character is lower
then
      echo "$char is lower character"
else
 echo "$char is Special symbol" # else it is special character
fi
```

e) Write a shell script to calculate profit or loss.

```
echo "input nthe cost price of an item"
read cop
echo "input the selling price of the item"
read sop
if test $cop -eq $sop
then
echo "no profit or no gain"

fi
if test $cop -gt $sop
then
s=`echo $cop - $sop |bc`
echo " u obtained loss of rs:$s"
else
s=`echo $sop - $cop |bc`
echo "u obtained profit of rs:$s"
fi
```

Output:

```
input the cost price of an item
100
input the selling price of the item
```

120
u obtained profit of rs:20

2. Shell Programming - Looping- while, until, for loops

Until Loop

The until statement is very similar in syntax and function to the while statement. The only real difference between the two is that the until statement executes its code block while its conditional expression is false, and the while statement executes its code block while its conditional expression is true.

```
syntax:



until expression

    do

      commands #body of the loop

    done
```

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 command

do
```

```
  Statement to be executed

done
```

for statement

The for loop operate on lists of items. It repeats a set of commands for every item in a list.
Here var is the name of a variable and word1 to wordN are sequences of characters separated by spaces (words). Each time the for loop executes, the value of the variable var is set to the next word in the list of words, word1 to wordN.
Syntax

```
for (( expr1; expr2; expr3 ))

      do

              commands

      done
```

a) Write a shell script to print all even and odd number from 1 to 10

```
echo “First Odd and Even number till 10 are”
n=1
while [ $n -lt 30 ]; do
  out=$(( $n % 2 ))
  if [ “$out” -eq 0 ] then
    echo “$n is even number”
  else
    echo “$n is ODD number”
  fi
  n=$(( $n + 1 ))
            done
```

b) Write a shell script to print table of a given number

```
clear
 echo "which number to generate multiplication table"
 read number
 i=1
 while [ $i -le 10 ]
 do
 echo " $number * $i =`expr $number \* $i ` "
 i=`expr $i + 1`
 done
```

c) Write a Shell script to find factorial of a given integer.

```
echo "enter a number"
read num
fact=1
while [ $num -ge 1 ]
do
fact=`expr  $fact\* $num`
num=’expr $num – 1’
done
echo "factorial of $n is $fact"
```

OUTPUT

```
enter a number
4
Factorial of 4 is 24
```

d) Write a shell script to print sum of all even numbers from 1 to 10.

```
echo "enter limit"
read n
i=2
while [ $i -lt $n ]
# Replace while loop with for loop condition for((i=1;i<=N;i++))
do
sum=$((sum+i))
i=$((i+2))
done
echo "sum:"$sum
```

e) Write a shell script to print sum of digit of any number.

```
echo -n "Input no : "
read no

length=`expr length $no`

while [ $length -ne 0 ]
do
   b=`expr substr $no $length 1`
   ans=`expr $ans + $b`
   length=`expr $length - 1`
done
echo "Sum of Digit is : $ans"
```

3. Shell Programming - case structure, use of break

CASE Statement

The syntax is as follows:

```
case  $variable-name  in
    pattern1)
            command1
      ...
      ....
      commandN
      ;;
    pattern2)
            command1
      ...
      ....
      commandN
      ;;
    patternN)
            command1
      ...
      ....
      commandN
      ;;
    *)
esac
```

OR

```
case  $variable-name  in
    pattern1|pattern2|pattern3)
              command1
       ...
       ....
       commandN
       ;;
    pattern4|pattern5|pattern6)
              command1
       ...
       ....
       commandN
       ;;
    pattern7|pattern8|patternN)
              command1
       ...
       ....
       commandN
       ;;
    *)
esac
```

BREAK statement

```
break
```

OR

```
break N
```

a) Write a shell script to make a basic calculator which performs addition, subtraction, Multiplication, division.

Program:

```
clear
echo Enter the a value
read a
echo  Enter the b value
read b
echo 1.Addition
```

```
echo 2.Subtraction
echo 3.Multiplication
echo 4.Division
echo 5.Modules
echo Enter your choice
read choice
case $choice in
       1)echo Addition       : $(expr $a + $b);;
       2)echo Suubtraction   : $(expr $a - $b);;
       3)echo Multiplication : $(expr $a \* $b);;
       4)echo Division       : $(expr $a / $b);;
       5)echo Modules        : $(expr $a % $b);;
       *)echo This is not a choice
esac
```

OUTPUT:

```
Enter the a value
45
Enter the b value
5
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Modules
Enter your choice
4
Division : 9
```

b) Write a shell script to print days of a week.

PROGRAM:

```
echo "enter a number"
read n
case $n in
1) echo "Sunday" ;;
2) echo "Monday" ;;
3) echo "Tuesday" ;;
4) echo "Wednesday" ;;
```

```
5) echo "Thursday" ;;
6) echo "Friday" ;;
7) echo "Saturday" ;;
*) echo "enter value between 1 to 7" ;;
esac
```

c) ASSIGNEMENT: Write a shell script to print starting 4 months having 31 days.

4. Shell Programming – Functions

FUNCTION Declaration

```
function_name ()
{
 commands
}
```

Single line version:

```
function_name ()
{ commands;
}
```

a) Write a shell script to find a number is Armstrong or not.

PROGRAM

```
echo "Program to check armstrong number"

echo "Enter a number: "

read c

function armstrong {

x=$1

sum=0

r=0
```

```
n=0

l=${#x}

while [ $x -gt 0 ]

do

r=`expr $x % 10`

n=$(bc <<< "$r^$l")

sum=`expr $sum + $n`

x=`expr $x / 10`

done

if [ $sum -eq $c ]

then

echo "It is an Armstrong Number."

else

echo "It is not an Armstrong Number."

Fi

}

result=`armstrong $c`

echo $result
```

b) Write a shell script to find a number is palindrome or not.

```
if [ $# -eq 1 ]

then

Num=$1

else

echo -n "Enter a Number : "

 read Num

 fi

 Fibonacci()

{

case $1 in

0|1) printf "$1 " ;;

*) echo -n "$(( $(Fibonacci $(($1-2)))+$(Fibonacci $(($1-
1))) )) ";;

Esac

}

 echo "The Fibonacci sequence for the number $Num is : "

 for (( i=0; i<=$Num; i++ ))

      do

         Fibonacci  $i                                  #Calling
        function Fibonacci

        done
```

c) Write a shell script to print Fibonacci series.

PROGRAM

```
prime_1=0
echo "enter the range"
read n
echo " Primenumber between 1 to $n is:"
echo "1"
echo "2"
for((i=3;i<=n;))
do
for((j=i-1;j>=2;))
do
if [  `expr $i % $j` -ne 0 ] ; then
prime_1=1
else
prime_1=0
break
fi
j=`expr $j - 1`
done
if [ $prime_1 -eq 1 ] ; then
echo $i
fi
i=`expr $i + 1`
done
```

ASSIGNEMENT:

d) Write a shell script to find prime number.
e) Write a shell script to convert binary to decimal and decimal to binary