TABLE OF CONTENTS

TITLE	**BASIC UNIX COMMANDS**		
Ex. No.	**1**	**Date of Exercise**	**05-12-2016**

Link: https://www.youtube.com/watch?v=_AqkHXYHnRY&feature=youtu.be

Aim : To perform basic unix commands using shell scripting.

Description:

Unix uses shells to accept commands given by the user. A Unix shell is a command-line interpreter or shell that provides a traditional Unix-like command line user interface. Users direct the operation of the computer by entering commands as text for a command line interpreter to execute, or by creating text scripts of one or more such commands.

Used commands :

- ls - list directory contents
- grep - print lines matching a pattern
- mkdir - make directories
- rmdir - remove empty directories
- cut - remove sections from each line of files
- cal - displays a calendar and the date of Easter
- tail - output the last part of files
- mv - move (rename) files
- > - output redirection
- | - pipeline operator, output redirected as input

Commands :

1. List the contents of user's home directory including the hidden files

```
ls -a
```

2. List the content of /var directory?

```
ls /var
```

3. Create two directories named dir1 & dir2

```
mkdir dir dir1
```

4. Create a hidden directory with your name?

```
mkdir .floura
```

5. Change into directory dir1?

```
cd dir
```

6.Copy the file /etc/passwd file to current directory with sample.txt as the filename

```
cp /etc/passwd ./sample.txt
```

7.Change to your home directory. Use pwd command to check you are in home directory.

```
  cd ~
pwd
```

8. Create a file test1.txt using Vim editor with the following contents to it

```
vi test.txt
```

a) Display the student names who are having Research Interest as GridComputing

```
grep Grid test.txt | cut -f 1
```

b) List all the student names & RegNo in the class

```
cat test.txt | cut -f 1,2
```

9. Display the contents of the file test1.txt without any blanklines

```
grep -v '^$' test.txt
```

10. Move the file sample.txt from dir1 directory to dir2 directory

```
mv dir/sample.txt dir1/
```

11. Change directory into dir2 directory

```
cd dir1
```

12. Check whether the file sample.txt is present their

```
ls | grep sample.txt
```

13.Rename the file sample.txt to new.txt and check whether sample.txt is there or not?

```
mv sample.txt new.txt
ls
```

14. Display the calendar of December 2020.

```
cal dec 2020
```

15. Remove the directory dir1

```
rmdir ../dir
```

16. Display last 3 lines of the file test1.txt

```
tail -3 new.txt
```

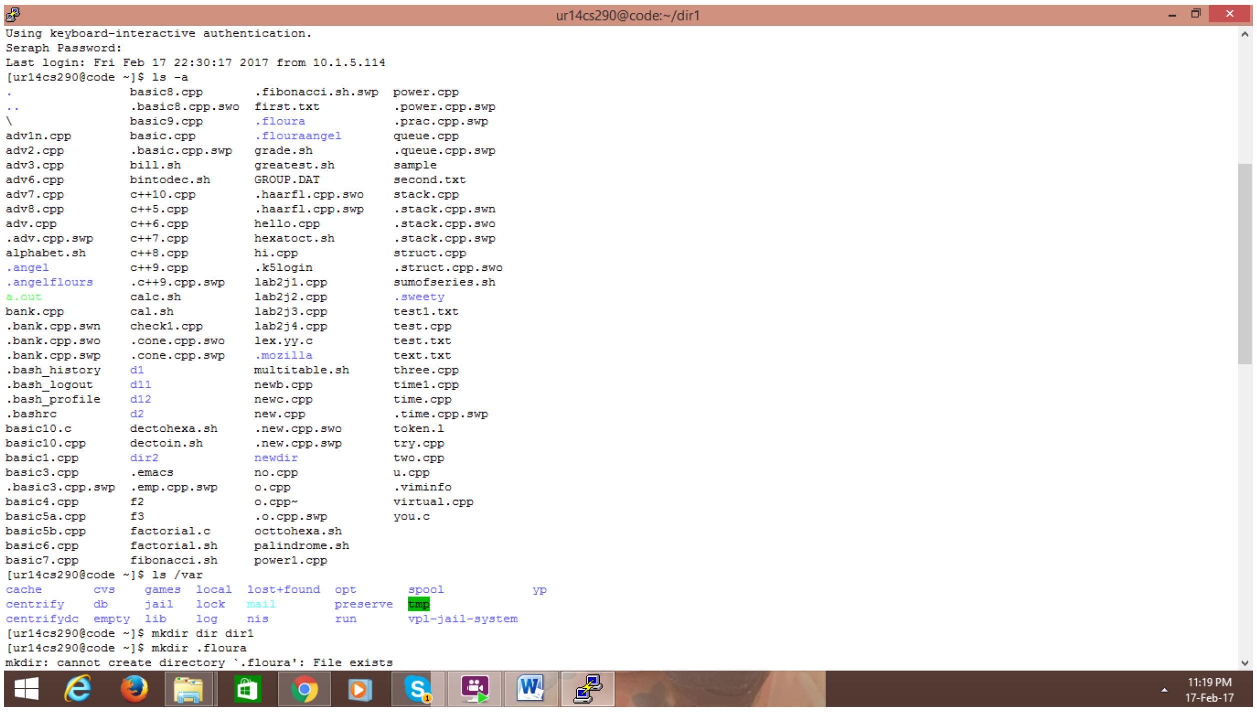
17.Display all the commands you have executed so far and save the list into a file named todayshistory.txt

```
history > todayhistory.txt
```

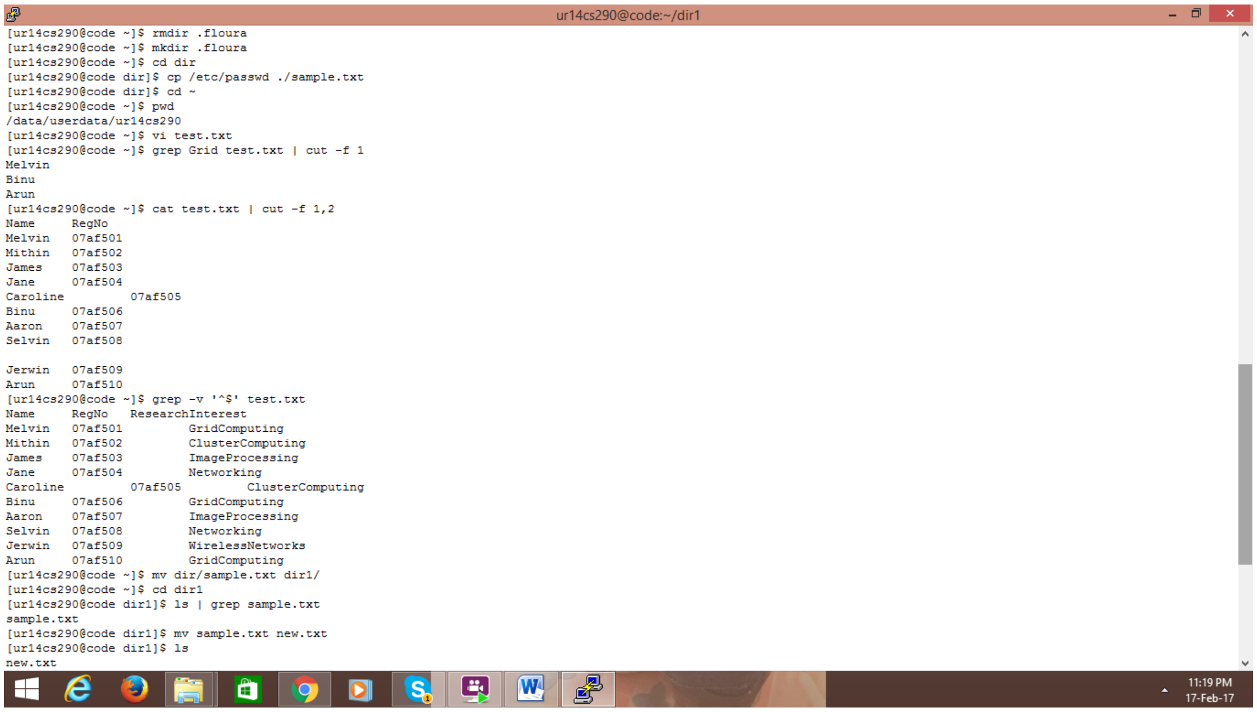
18. How many files are present under your home directory

```
ls ~ | wc -w
```

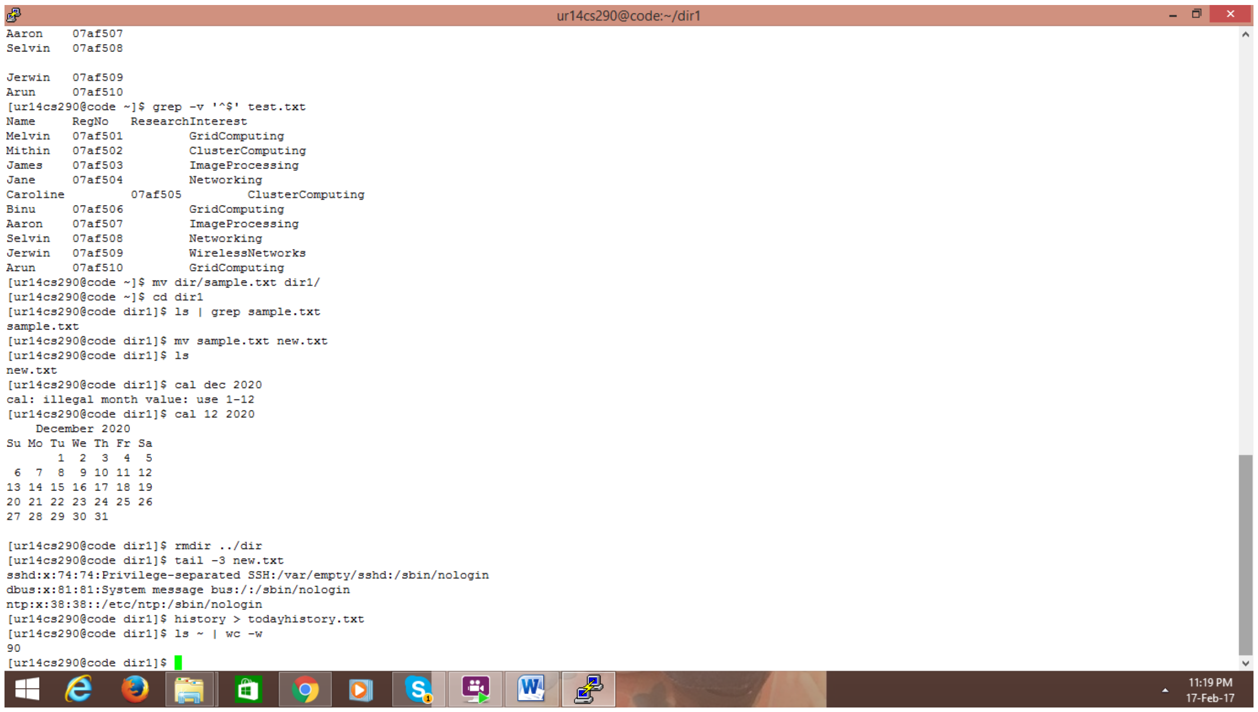
Output:



```
Using keyboard-interactive authentication.
Seraph Password:
Last login: Fri Feb 17 22:30:17 2017 from 10.1.5.114
[ur14cs290@code ~]$ ls -a
.                basic8.cpp       .fibonacci.sh.swp  power.cpp
..               .basic8.cpp.swo  first.txt          .power.cpp.swp
\                basic9.cpp       .floura            .prac.cpp.swp
adv1n.cpp        basic.cpp        .flouraangel       queue.cpp
adv2.cpp         .basic.cpp.swp   grade.sh           .queue.cpp.swp
adv3.cpp         bill.sh          greatest.sh        sample
adv6.cpp         bintodec.sh      GROUP.DAT          second.txt
adv7.cpp         c++10.cpp        .haarfl.cpp.swo    stack.cpp
adv8.cpp         c++5.cpp         .haarfl.cpp.swp    .stack.cpp.swn
adv.cpp          c++6.cpp         hello.cpp          .stack.cpp.swo
.adv.cpp.swp     c++7.cpp         hexatoct.sh        .stack.cpp.swp
alphabet.sh      c++8.cpp         hi.cpp             struct.cpp
.angel           c++9.cpp         .k5login           .struct.cpp.swo
.angelflours     .c++9.cpp.swp    lab2j1.cpp         sumofseries.sh
a.out            calc.sh          lab2j2.cpp         .sweety
bank.cpp         cal.sh           lab2j3.cpp         test1.txt
.bank.cpp.swn    check1.cpp       lab2j4.cpp         test.cpp
.bank.cpp.swo    .cone.cpp.swo    lex.yy.c           test.txt
.bank.cpp.swp    .cone.cpp.swp    .mozilla           text.txt
.bash_history    d1               multitable.sh      three.cpp
.bash_logout     d11              newb.cpp           time1.cpp
.bash_profile    d12              newc.cpp           time.cpp
.bashrc          d2               new.cpp            .time.cpp.swp
basic10.c        dectohexa.sh     .new.cpp.swo       token.l
basic10.cpp      dectoin.sh       .new.cpp.swp       try.cpp
basic1.cpp       dir2             newdir             two.cpp
basic3.cpp       .emacs           no.cpp             u.cpp
.basic3.cpp.swp  .emp.cpp.swp     o.cpp              .viminfo
basic4.cpp       f2               o.cpp~             virtual.cpp
basic5a.cpp      f3               .o.cpp.swp         you.c
basic5b.cpp      factorial.c      octtohexa.sh
basic6.cpp       factorial.sh     palindrome.sh
basic7.cpp       fibonacci.sh     power1.cpp
[ur14cs290@code ~]$ ls /var
cache       cvs    games  local  lost+found  opt       spool            yp
centrify    db     jail   lock   mail        preserve  tmp
centrifydc  empty  lib    log    nis         run       vpl-jail-system
[ur14cs290@code ~]$ mkdir dir dir1
[ur14cs290@code ~]$ mkdir .floura
mkdir: cannot create directory `.floura': File exists
```



```
[ur14cs290@code ~]$ rmdir .floura
[ur14cs290@code ~]$ mkdir .floura
[ur14cs290@code ~]$ cd dir
[ur14cs290@code dir]$ cp /etc/passwd ./sample.txt
[ur14cs290@code dir]$ cd ~
[ur14cs290@code ~]$ pwd
/data/userdata/ur14cs290
[ur14cs290@code ~]$ vi test.txt
[ur14cs290@code ~]$ grep Grid test.txt | cut -f 1
Melvin
Binu
Arun
[ur14cs290@code ~]$ cat test.txt | cut -f 1,2
Name     RegNo
Melvin   07af501
Mithin   07af502
James    07af503
Jane     07af504
Caroline        07af505
Binu     07af506
Aaron    07af507
Selvin   07af508

Jerwin   07af509
Arun     07af510
[ur14cs290@code ~]$ grep -v '^$' test.txt
Name     RegNo   ResearchInterest
Melvin   07af501         GridComputing
Mithin   07af502         ClusterComputing
James    07af503         ImageProcessing
Jane     07af504         Networking
Caroline        07af505         ClusterComputing
Binu     07af506         GridComputing
Aaron    07af507         ImageProcessing
Selvin   07af508         Networking
Jerwin   07af509         WirelessNetworks
Arun     07af510         GridComputing
[ur14cs290@code ~]$ mv dir/sample.txt dir1/
[ur14cs290@code ~]$ cd dir1
[ur14cs290@code dir1]$ ls | grep sample.txt
sample.txt
[ur14cs290@code dir1]$ mv sample.txt new.txt
[ur14cs290@code dir1]$ ls
new.txt
```



```
Aaron    07af507
Selvin   07af508

Jerwin   07af509
Arun     07af510
[ur14cs290@code ~]$ grep -v '^$' test.txt
Name     RegNo   ResearchInterest
Melvin   07af501         GridComputing
Mithin   07af502         ClusterComputing
James    07af503         ImageProcessing
Jane     07af504         Networking
Caroline         07af505         ClusterComputing
Binu     07af506         GridComputing
Aaron    07af507         ImageProcessing
Selvin   07af508         Networking
Jerwin   07af509         WirelessNetworks
Arun     07af510         GridComputing
[ur14cs290@code ~]$ mv dir/sample.txt dir1/
[ur14cs290@code ~]$ cd dir1
[ur14cs290@code dir1]$ ls | grep sample.txt
sample.txt
[ur14cs290@code dir1]$ mv sample.txt new.txt
[ur14cs290@code dir1]$ ls
new.txt
[ur14cs290@code dir1]$ cal dec 2020
cal: illegal month value: use 1-12
[ur14cs290@code dir1]$ cal 12 2020
    December 2020
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

[ur14cs290@code dir1]$ rmdir ../dir
[ur14cs290@code dir1]$ tail -3 new.txt
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
[ur14cs290@code dir1]$ history > todayhistory.txt
[ur14cs290@code dir1]$ ls ~ | wc -w
90
[ur14cs290@code dir1]$
```

Result: The unix commands are executed successfully.

TITLE	**CONDITIONAL STATEMENTS USING SHELL SCRIPTING**		
Ex. No.	**2**	**Date of Exercise**	**09-01-2017**

Link: https://www.youtube.com/watch?v=PCj7ZGMRIhk&t=10s

Aim :

To write programs to demonstrate conditional statements using shell scripting.

Description:

You can perform conditional statements operations on Bash shell variables. The bash shell has built-in arithmetic option. You can also use external command such as expr and bc calculator.

Syntax

While loop

```
while command

do

  Statement(s) to be executed if command is true

Done
```

If loop

```
if [ expression ]

then

  Statement(s) to be executed if expression is true

else

  Statement(s) to be executed if expression is not true

fi
```

Program:

1. **Greatest of three numbers**

```
echo "HELLO!! I'LL FIND THE GREATEST NUMBER FOR YOU "
echo "enter first number"
read first
echo "enter second number"
read sec
echo "enter third number"
read third
if [ $first -gt $sec ] ; then
if [ $first -gt $third ] ; then
echo  " $first is greatest number "
else
echo  " $third is greatest number "
fi
fi
if [ $sec -gt $third ] ; then
echo  " $sec is greatest number "
fi
if [ $third -gt $sec ] ; then
echo  " $third is greatest number "
fi
if [ $first -eq $sec ] ; then
if [ $first -eq $third ] ; then
echo  " All three are of same value "
```

```
fi
fi
```

2. **Student grade calculation**

```
echo "HELLO THIS IS FOR YOUR GRADE CALCULATION FOR 7
SUBJECTS"
echo "Enter the seven subject marks for the student(out of
100 - each)"
read  m1 m2 m3 m4 m5 m6 m7
sum1=`expr $m1 + $m2 + $m3 + $m4 + $m5 + $m6 + $m7 `
echo "Total Marks of 7 subjects are: " $sum1
per=`expr $sum1 / 7 `
echo " Percentage: " $per
if [ $per -ge 85 ]
then
output="O"
elif [ $per -ge 75 ]
then
output="S"
elif [ $per -ge  65 ]
then
output="A"
elif [ $per -ge  55 ]
then
output="B"
```

```
elif [ $per -ge  35 ]

then

output="P"

elif [ $per -lt 35 ]

then

output="F"

fi

if [ $output = "F" ] ; then

echo $output " - Fail "

else

echo "Congratulations,Your grade : "$output

fi
```

3. **Electricity bill calculator**

```
echo "I WILL CALCULATE THE ELECTRICITY BILL FOR YOU"

echo "LET ME TELL YOU THE COST OF EACH APPLIANCE"

echo "1.FAN - Rs 25 per hour"

echo "2.TUBELIGHT - Rs 23.5 per hour"

echo "3.FRIDGE- Rs 30 per hour"

echo "4.CHARGING POINT - Rs 17 per hour"

echo "NOW ENTER THE NUMBER OF HOUR EACH APPLIANCE WORKS"

echo "HOURS YOU USED FAN TODAY"

read fan

echo "HOURS YOU USED TUBELIGHT TODAY"

read light
```

```
echo "HOURS YOU USED FRIDGE TODAY"

read fridge

echo "HOURS YOU USED CHARGING POINT TODAY"

read charge

echo -n "TOTAL COST FOR TODAY:"

sum=`expr $fan*25+$light*23.5+$fridge*30+$charge*17|bc`

echo $sum

echo -n "ESTIMATED TOTAL COST FOR THIS MONTHS PROVIDED USAGE IS
CONSTANT :

"

sum1=`expr $sum*30|bc`

echo $sum1
```

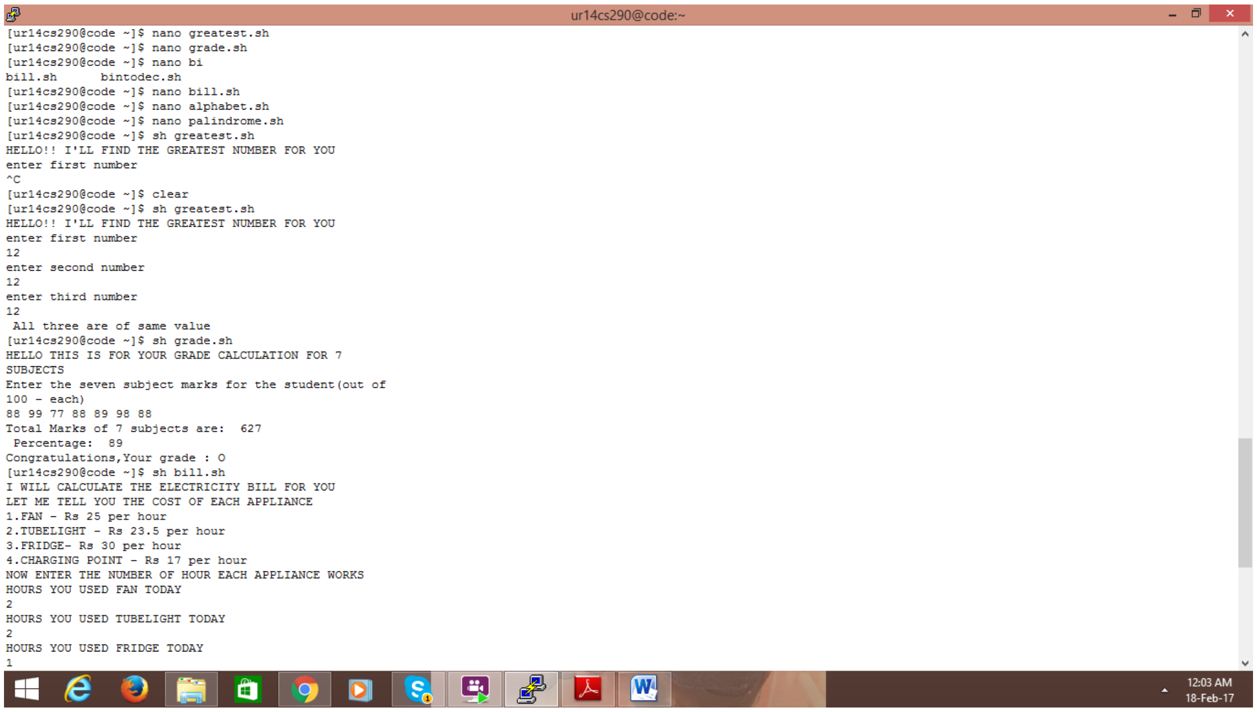
4. **Finding the alphabet is a vowel or consonant**

```
echo "I WILL TELL YOU WHETHER YOUR ALPHABET IS A VOWEL OR

CONSONANT"

 echo "Enter any character: "

read ch

case $ch in

"a") echo "It is a vowel.";;

"e") echo "It is a vowel.";;

"i") echo "It is a vowel.";;

"o") echo "It is a vowel.";;

"u") echo "It is a vowel.";;

*) echo "It is a consonant."

esac
```

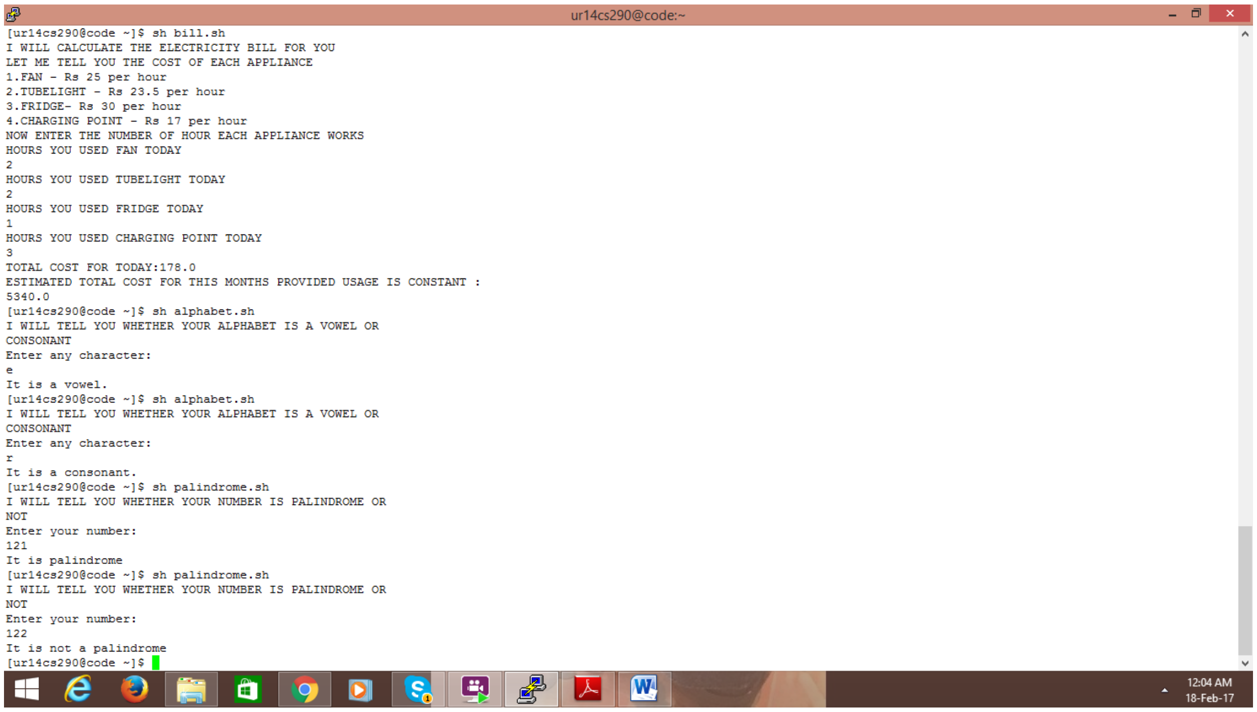
5. Given umber is palindrome or not

```
echo "I WILL TELL YOU WHETHER YOUR NUMBER IS PALINDROME OR
NOT"
echo "Enter your number:"
read s
rvs=`echo $s | rev`
if [ $s = $rvs ]
then
echo "It is palindrome"
else
echo "It is not a palindrome"
fi
```

Output:



```
[ur14cs290@code ~]$ nano greatest.sh
[ur14cs290@code ~]$ nano grade.sh
[ur14cs290@code ~]$ nano bi
bill.sh       bintodec.sh
[ur14cs290@code ~]$ nano bill.sh
[ur14cs290@code ~]$ nano alphabet.sh
[ur14cs290@code ~]$ nano palindrome.sh
[ur14cs290@code ~]$ sh greatest.sh
HELLO!! I'LL FIND THE GREATEST NUMBER FOR YOU
enter first number
^C
[ur14cs290@code ~]$ clear
[ur14cs290@code ~]$ sh greatest.sh
HELLO!! I'LL FIND THE GREATEST NUMBER FOR YOU
enter first number
12
enter second number
12
enter third number
12
 All three are of same value
[ur14cs290@code ~]$ sh grade.sh
HELLO THIS IS FOR YOUR GRADE CALCULATION FOR 7
SUBJECTS
Enter the seven subject marks for the student(out of
100 - each)
88 99 77 88 89 98 88
Total Marks of 7 subjects are:  627
 Percentage:  89
Congratulations,Your grade : O
[ur14cs290@code ~]$ sh bill.sh
I WILL CALCULATE THE ELECTRICITY BILL FOR YOU
LET ME TELL YOU THE COST OF EACH APPLIANCE
1.FAN - Rs 25 per hour
2.TUBELIGHT - Rs 23.5 per hour
3.FRIDGE- Rs 30 per hour
4.CHARGING POINT - Rs 17 per hour
NOW ENTER THE NUMBER OF HOUR EACH APPLIANCE WORKS
HOURS YOU USED FAN TODAY
2
HOURS YOU USED TUBELIGHT TODAY
2
HOURS YOU USED FRIDGE TODAY
1
```



```
[ur14cs290@code ~]$ sh bill.sh
I WILL CALCULATE THE ELECTRICITY BILL FOR YOU
LET ME TELL YOU THE COST OF EACH APPLIANCE
1.FAN - Rs 25 per hour
2.TUBELIGHT - Rs 23.5 per hour
3.FRIDGE- Rs 30 per hour
4.CHARGING POINT - Rs 17 per hour
NOW ENTER THE NUMBER OF HOUR EACH APPLIANCE WORKS
HOURS YOU USED FAN TODAY
2
HOURS YOU USED TUBELIGHT TODAY
2
HOURS YOU USED FRIDGE TODAY
1
HOURS YOU USED CHARGING POINT TODAY
3
TOTAL COST FOR TODAY:178.0
ESTIMATED TOTAL COST FOR THIS MONTHS PROVIDED USAGE IS CONSTANT :
5340.0
[ur14cs290@code ~]$ sh alphabet.sh
I WILL TELL YOU WHETHER YOUR ALPHABET IS A VOWEL OR
CONSONANT
Enter any character:
e
It is a vowel.
[ur14cs290@code ~]$ sh alphabet.sh
I WILL TELL YOU WHETHER YOUR ALPHABET IS A VOWEL OR
CONSONANT
Enter any character:
r
It is a consonant.
[ur14cs290@code ~]$ sh palindrome.sh
I WILL TELL YOU WHETHER YOUR NUMBER IS PALINDROME OR
NOT
Enter your number:
121
It is palindrome
[ur14cs290@code ~]$ sh palindrome.sh
I WILL TELL YOU WHETHER YOUR NUMBER IS PALINDROME OR
NOT
Enter your number:
122
It is not a palindrome
[ur14cs290@code ~]$
```

Result: The unix commands are executed successfully.

TITLE	**LOOPING STATEMENTS USING SHELL SCRIPTING**		
Ex. No.	**3**	**Date of Exercise**	**16-01-2017**

Link: https://www.youtube.com/watch?v=Sx4UyU69s8Y

Aim :

To write programs to demonstrate looping statements using shell scripting.

Description:

You can perform looping statements operations on Bash shell variables. The bash shell has built-in arithmetic option. You can also use external command such as expr and bc calculator.

Syntax

While loop

```
while command

do

   Statement(s) to be executed if command is true

Done
```

If loop

```
if [ expression ]

then

   Statement(s) to be executed if expression is true

else

   Statement(s) to be executed if expression is not true

fi
```

Program:

1. **Factorial of number**

```
echo "HELLO, I WILL FIND THE FACTORIAL FOR YOU"
echo ""
echo "Mention your number whose factorial you wish to find: "
read fact

ans=1
counter=0
while [ $fact -ne $counter ]
do
        counter=`expr $counter + 1`
        ans=`expr $ans \* $counter`
done
echo "Your factorial is $ans"
```

2. **Sum of given series**

```
echo "HELLO I WILL FIND THE SUM OF GIVEN SERIES"
echo ""
echo "ENTER ONE NUMBER AT A TIME"
echo ""
d=1
sum=0
while [ $d -eq 1 ]
do
echo "Enter your number : "
read num
sum=`expr $sum+$num|bc`
echo "Press 1 to enter another number"
read d
done
echo ""
echo "SUM OF YOUR GIVEN SERIES IS :$sum"
```

3. **Fibonacci series**

```
echo "I WILL PRINT FIBONACCI SERIES"

echo ""

echo "HOW MANY TERMS YOU WANT TO PRINT"

read n

a=0
```

```
b=1

count=0

echo "YOUR FIBONACCI SERIES IS : "

echo $a

echo $b

n=`expr $n - 2`

while [ $count -lt $n ]

do

c=`expr $b + $a`

echo $c

a=`expr $b`

b=`expr $c`

count=`expr $count + 1`

done
```

4. **Multiplication Table**

```
echo "I WILL PRINTING MULTIPLICATION TABLE OF YOUR DESIRED
NUMBER"
echo ""
echo "Enter the number to find its multiplication: "
read n

echo "Enter Range of your table:"
read r
i=0

while [ $i -le $r ]
do
   echo " $n x $i = `expr $n \* $i`"
   i=`expr $i + 1`
done
```

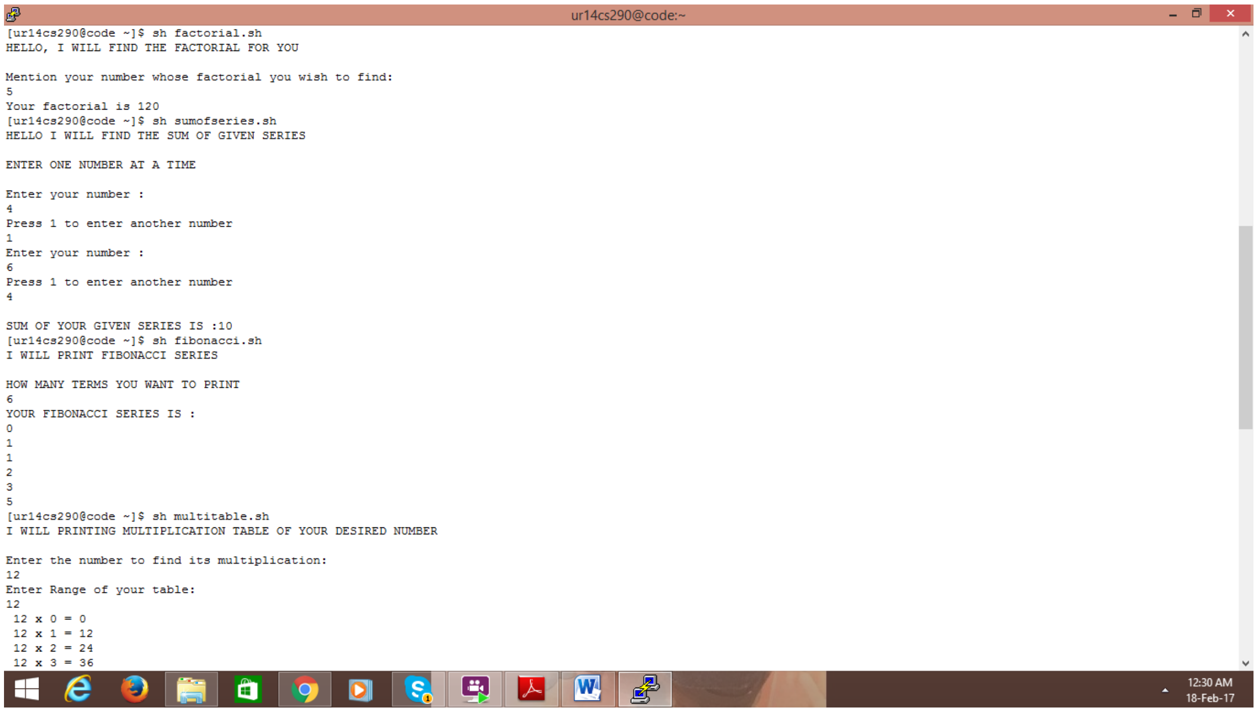
5. Scientific calculator

```
echo "Basic Calculator"

d=1

while [ $d -eq 1 ]

do

echo "Enter your choice of operation"

echo "1.Addition"

echo "2.Subtraction"

echo "3.Multiplication"

echo "4.Division"

echo "5.Modulus"

echo "6.Sin"

echo "7.Cosine"

echo "8.Tan"

echo "9.Square root of a number"

echo "10.Square of a number"

read input

if [ $input -le 5 ]

then

echo "Enter the first number a: "

read a

echo "Enter the second number b: "

read b

fi
```

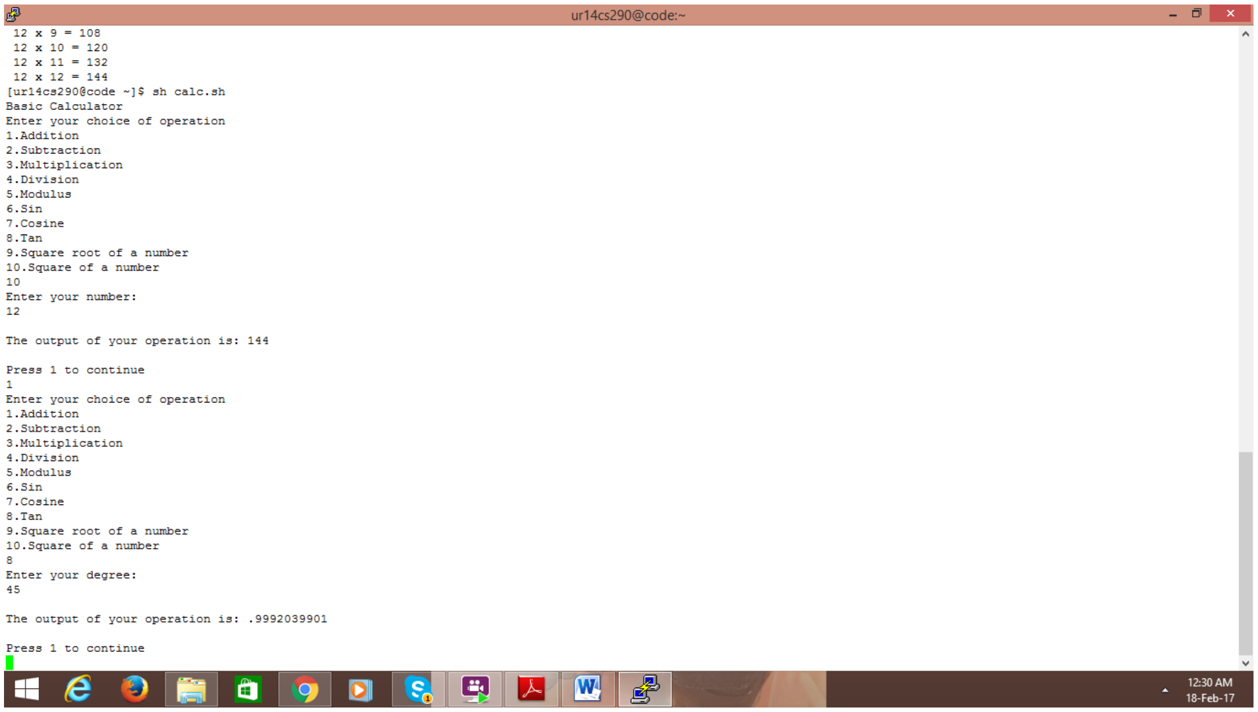
```
if [ $input -ge 6 ]
then
if [ $input -le 10 ]
then
if [ $input -le 8 ]
then
echo "Enter your degree: "
read degree
elif [ $input -gt 8 ]
then
echo "Enter your number: "
read degree
else
echo "Wrong input"
fi
else
echo "Wrong input"
fi
else
echo "Wrong input"
fi
echo ""
echo -n "The output of your operation is: "
case $input in
1)output=`echo $a+$b|bc`;;
```

```
2)output=`echo $a-$b|bc`;;

3)output=`echo $a\*$b|bc`;;

4)output=`echo $a\/$b|bc`;;

5)output=`echo $a%$b|bc`;;

6)out=`echo $degree \* 3.14|bc -l`

out1=`echo $out \/ 180|bc -l`

output=`echo "scale=10;s($out1)" |bc -l`

;;

7)out=`echo $degree \* 3.14|bc -l`

out1=`echo $out \/ 180|bc -l`

output=`echo "scale=10;c($out1)" |bc -l`

;;

8)out=`echo $degree \* 3.14|bc -l`

out1=`echo $out \/ 180|bc -l`

10)output=`echo $degree\*$degree|bc`;;

*)output="wrong choice";;

esac

echo $output

echo ""

echo "Press 1 to continue"

read d

done
```

Output:



```
[ur14cs290@code ~]$ sh factorial.sh
HELLO, I WILL FIND THE FACTORIAL FOR YOU

Mention your number whose factorial you wish to find:
5
Your factorial is 120
[ur14cs290@code ~]$ sh sumofseries.sh
HELLO I WILL FIND THE SUM OF GIVEN SERIES

ENTER ONE NUMBER AT A TIME

Enter your number :
4
Press 1 to enter another number
1
Enter your number :
6
Press 1 to enter another number
4

SUM OF YOUR GIVEN SERIES IS :10
[ur14cs290@code ~]$ sh fibonacci.sh
I WILL PRINT FIBONACCI SERIES

HOW MANY TERMS YOU WANT TO PRINT
6
YOUR FIBONACCI SERIES IS :
0
1
1
2
3
5
[ur14cs290@code ~]$ sh multitable.sh
I WILL PRINTING MULTIPLICATION TABLE OF YOUR DESIRED NUMBER

Enter the number to find its multiplication:
12
Enter Range of your table:
12
 12 x 0 = 0
 12 x 1 = 12
 12 x 2 = 24
 12 x 3 = 36
```



```
 12 x 9 = 108
 12 x 10 = 120
 12 x 11 = 132
 12 x 12 = 144
[ur14cs290@code ~]$ sh calc.sh
Basic Calculator
Enter your choice of operation
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Modulus
6.Sin
7.Cosine
8.Tan
9.Square root of a number
10.Square of a number
10
Enter your number:
12

The output of your operation is: 144

Press 1 to continue
1
Enter your choice of operation
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Modulus
6.Sin
7.Cosine
8.Tan
9.Square root of a number
10.Square of a number
8
Enter your degree:
45

The output of your operation is: .9992039901

Press 1 to continue
```

Result: The unix commands are executed successfully.

TITLE	**BASE CONVERSION USING SHELL SCRIPTING**		
Ex. No.	**4**	**Date of Exercise**	**14-02-2017**

Link: https://youtu.be/0oN0sG_lA8E or https://www.youtube.com/watch?v=0oN0sG_lA8E&feature=youtu.be

Aim :

To write programs to demonstrate base conversion using shell scripting.

Description:

You can perform base conversion operations on Bash shell variables. The bash shell has built-in arithmetic option. You can also use external command such as expr and bc calculator.

obase – output base

ibase – input base

Syntax

While loop

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

Example :

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

Program:

1. **Binary to Decimal**

```
echo "Enter your binary number"
read d
echo ""
Bnumber=$Binary
        Decimal=0
        power=1
while [ $d -ne 0 ]
        do
            rem=$(expr $d % 10 )
            Decimal=$((Decimal+(rem*power)))
            power=$((power*2))
            d=$(expr $d / 10)
        done
echo "Your Decimal number is : $Decimal"
```

2. **Decimal to binary**

```
echo "DECIMAL TO BINARY CONVERSION"
echo ""
echo "Enter you decimal number"
read d
Binary=
        while [ $d -ne 0 ]
        do
```

```
            Bit=$(expr $d % 2)
            Binary=$Bit$Binary
            d=$(expr $d / 2)
        done
echo  "Binary Number is : $Binary"
```

3. **Octal to Hexadecimal**

```
echo "Enter the octal number"
read d
echo "Your Hexadecimal number is "
echo "obase=16; ibase=8; $d" | bc
```

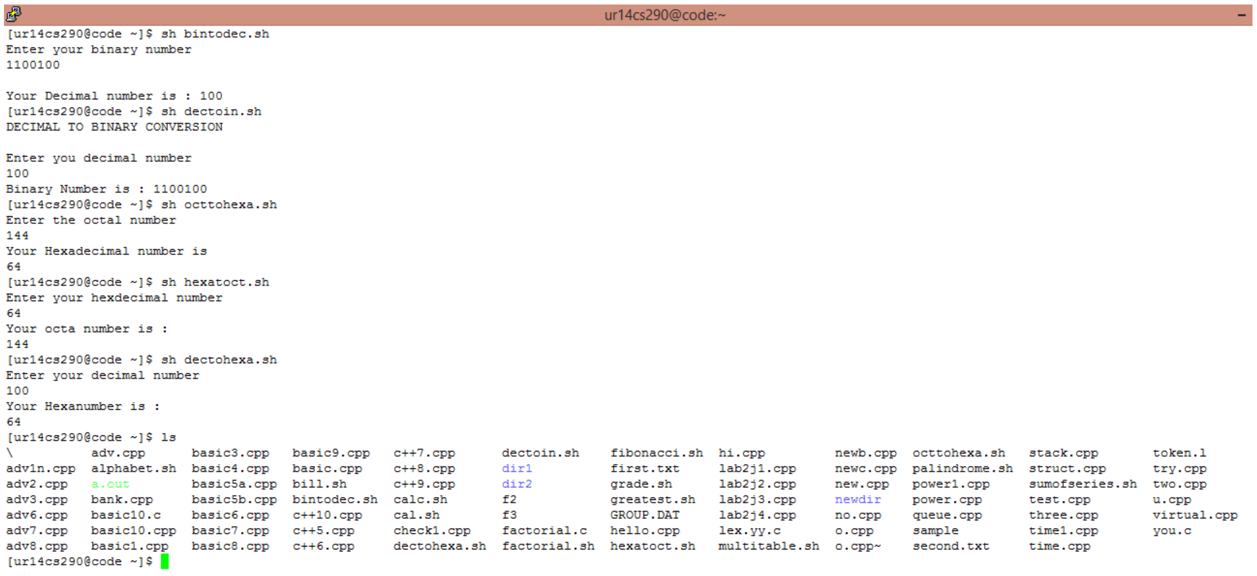
4. **Hexadecimal to Octal**

```
echo "Enter your hexdecimal number"
read d
echo "Your octa number is : "
echo "obase=8; ibase=16; $d" | bc
```

5. **Decimal to Hexadecimal**

```
echo "Enter your decimal number"
read d
echo "Your Hexanumber is : "
printf '%x\n' $d
```

Output:



ur14cs290@code:~

```
[ur14cs290@code ~]$ sh bintodec.sh
Enter your binary number
1100100

Your Decimal number is : 100
[ur14cs290@code ~]$ sh dectoin.sh
DECIMAL TO BINARY CONVERSION

Enter you decimal number
100
Binary Number is : 1100100
[ur14cs290@code ~]$ sh octtohexa.sh
Enter the octal number
144
Your Hexadecimal number is
64
[ur14cs290@code ~]$ sh hexatoct.sh
Enter your hexdecimal number
64
Your octa number is :
144
[ur14cs290@code ~]$ sh dectohexa.sh
Enter your decimal number
100
Your Hexanumber is :
64
[ur14cs290@code ~]$ ls
\           adv.cpp      basic3.cpp   basic9.cpp   c++7.cpp      dectoin.sh    fibonacci.sh  hi.cpp         newb.cpp  octtohexa.sh   stack.cpp       token.l
adv1n.cpp   alphabet.sh  basic4.cpp   basic.cpp    c++8.cpp      dir1          first.txt     lab2j1.cpp     newc.cpp  palindrome.sh  struct.cpp      try.cpp
adv2.cpp    a.out        basic5a.cpp  bill.sh      c++9.cpp      dir2          grade.sh      lab2j2.cpp     new.cpp   power1.cpp     sumofseries.sh  two.cpp
adv3.cpp    bank.cpp     basic5b.cpp  bintodec.sh  calc.sh       f2            greatest.sh   lab2j3.cpp     newdir    power.cpp      test.cpp        u.cpp
adv6.cpp    basic10.c    basic6.cpp   c++10.cpp    cal.sh        f3            GROUP.DAT     lab2j4.cpp     no.cpp    queue.cpp      three.cpp       virtual.cpp
adv7.cpp    basic10.cpp  basic7.cpp   c++5.cpp     check1.cpp    factorial.c   hello.cpp     lex.yy.c       o.cpp     sample         time1.cpp       you.c
adv8.cpp    basic1.cpp   basic8.cpp   c++6.cpp     dectohexa.sh  factorial.sh  hexatoct.sh   multitable.sh  o.cpp~    second.txt     time.cpp
[ur14cs290@code ~]$ █
```

Result: The unix commands are executed successfully.

Ex. No. 5	**File Operation using cut & grep**
Date of Exercise	**20-02-2017**

Link: https://www.youtube.com/watch?v=3dmvznquUlA

Aim: To write a program to understand the usage of cut and grep commands

Description:

Cut: **cut** is a **Unix** command line utility which is used to extract sections from each line of input usually from a file. cut –[Options]

Grep: **grep**, which stands for "global regular expression print," processes text line by line and prints any lines which match a specified pattern.

grep [*OPTIONS*] *PATTERN* [*FILE*...]

Program:

1. Write a shell script that given a person's uid, tells you how many times that person is logged on. (use who,grep, wc)

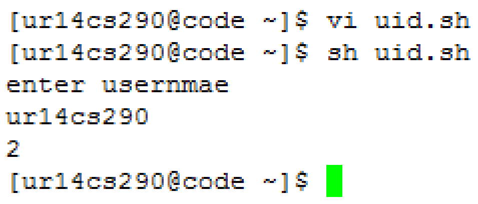
   echo "enter

   usernmae" read

   uid

   uercnt=`who | grep $uid

   |wc -l` echo "$uercnt"



```
[ur14cs290@code ~]$ vi uid.sh
[ur14cs290@code ~]$ sh uid.sh
enter usernmae
ur14cs290
2
[ur14cs290@code ~]$ 
```

2. Write a shell script that takes a uid as an argument and prints out that person's name, home directory, shell and group number. Print out the name of the group corresponding to the group number, and other groups that person may belong to (details are available in
/etc/passwd and /etc/group

files) echo Displaying the

log of users

echo "Users:" `cat /etc/passwd |cut -d ':'

-f 1` echo select the user from the

above list:

read user

echo "username:" `grep $user /etc/passwd|cut -d ':'

-f 1` echo "Password:" `grep $user /etc/passwd|cut

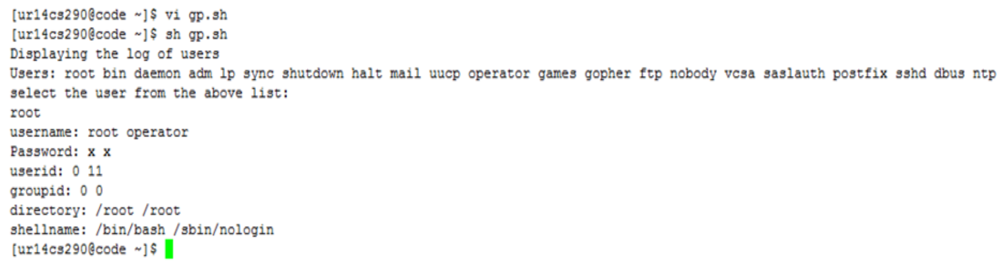
-d ':' -f 2` echo "userid:" `grep $user

/etc/passwd|cut -d ':' -f 3` echo "groupid:" `grep

$user /etc/passwd|cut -d ':' -f 4` echo "directory:"

`grep $user /etc/passwd|cut -d ':' -f 6` echo

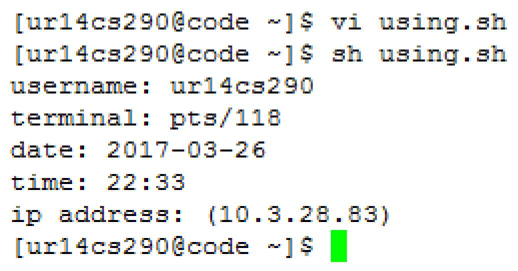
"shellname:" `grep $user /etc/passwd|cut -d ':' -f 7`



```
[ur14cs290@code ~]$ vi gp.sh
[ur14cs290@code ~]$ sh gp.sh
Displaying the log of users
Users: root bin daemon adm lp sync shutdown halt mail uucp operator games gopher ftp nobody vcsa saslauth postfix sshd dbus ntp
select the user from the above list:
root
username: root operator
Password: x x
userid: 0 11
groupid: 0 0
directory: /root /root
shellname: /bin/bash /sbin/nologin
[ur14cs290@code ~]$
```

3. Write a shell script to check whether a particular user is using the system or not. If using, print his pseudo terminal number and his IP address (use finger,grep,cut)

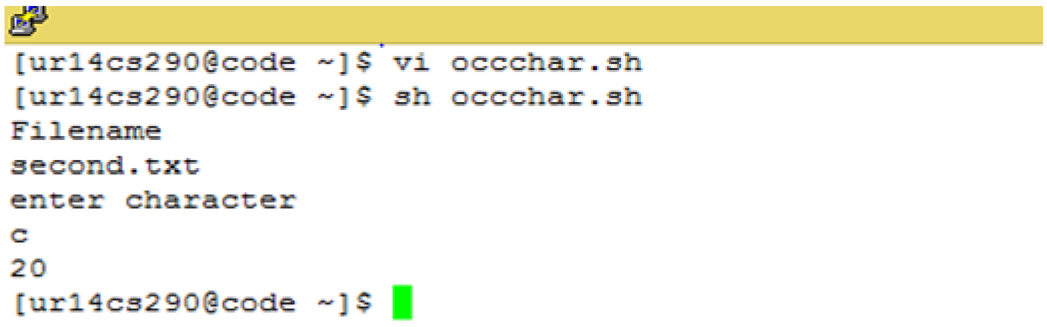
```
c=`who am i`

echo "username:" `who am i |cut -d ' ' -

f 1` echo "terminal:" `who am i |cut -d '

' -f 2` echo  "date:" `who am i |cut -d ' '

-f 8`

echo  "time:" `who am i |cut -d ' ' -f 9`

echo  "ip address:" `who am i |cut -d ' ' -f 10`
```



```
[ur14cs290@code ~]$ vi using.sh
[ur14cs290@code ~]$ sh using.sh
username: ur14cs290
terminal: pts/118
date: 2017-03-26
time: 22:33
ip address: (10.3.28.83)
[ur14cs290@code ~]$
```

4. Write a shell script to print the number of occurances for the given word within the file /etc/passwd

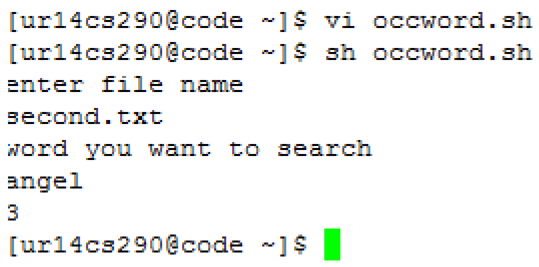
```
echo enter file

name read file

echo word you want to

search read f1

grep $f1 $file |wc –w
```



```
[ur14cs290@code ~]$ vi occchar.sh
[ur14cs290@code ~]$ sh occchar.sh
Filename
second.txt
enter character
c
20
[ur14cs290@code ~]$
```

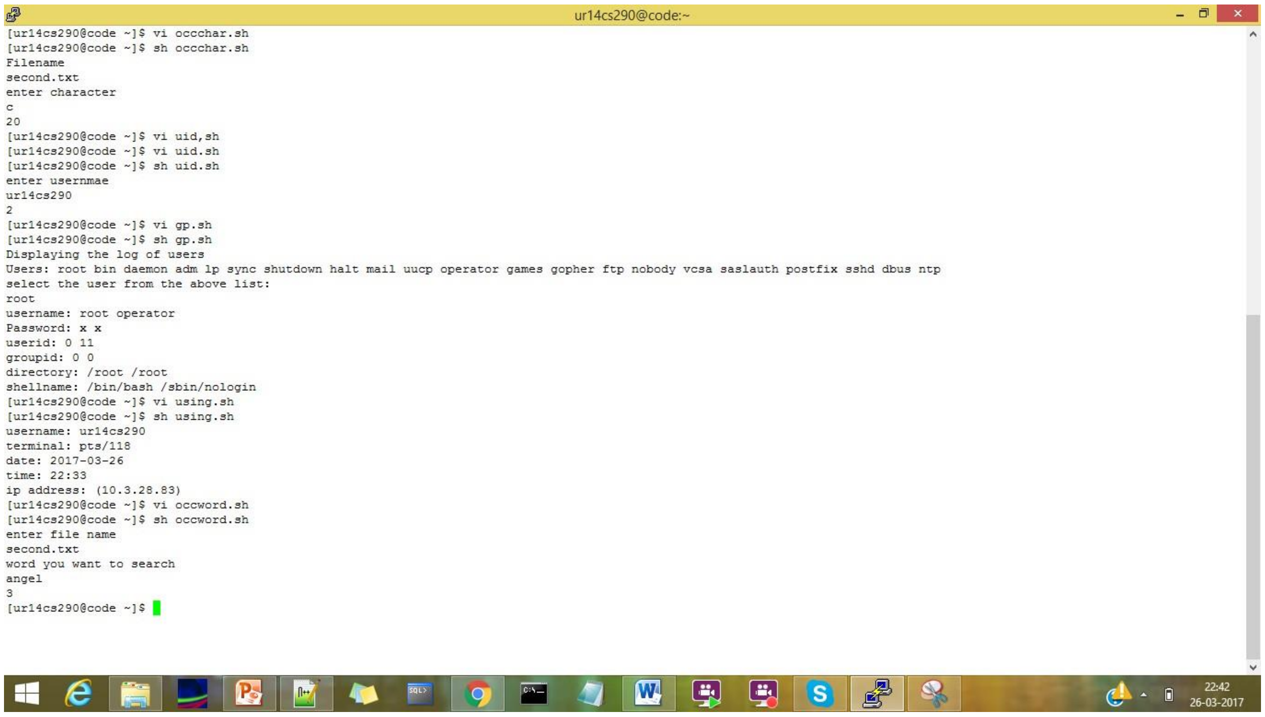
5. Write a shell script to print the number of occurances for the given word within the file
   /etc/passwd

```
echo enter file

name read file

echo word you want to

search read f1

grep $f1 $file |wc –w
```



```
[ur14cs290@code ~]$ vi occword.sh
[ur14cs290@code ~]$ sh occword.sh
enter file name
second.txt
word you want to search
angel
3
[ur14cs290@code ~]$
```

Output:



ur14cs290@code:~

```
[ur14cs290@code ~]$ vi occchar.sh
[ur14cs290@code ~]$ sh occchar.sh
Filename
second.txt
enter character
c
20
[ur14cs290@code ~]$ vi uid,sh
[ur14cs290@code ~]$ vi uid.sh
[ur14cs290@code ~]$ sh uid.sh
enter usernmae
ur14cs290
2
[ur14cs290@code ~]$ vi gp.sh
[ur14cs290@code ~]$ sh gp.sh
Displaying the log of users
Users: root bin daemon adm lp sync shutdown halt mail uucp operator games gopher ftp nobody vcsa saslauth postfix sshd dbus ntp
select the user from the above list:
root
username: root operator
Password: x x
userid: 0 11
groupid: 0 0
directory: /root /root
shellname: /bin/bash /sbin/nologin
[ur14cs290@code ~]$ vi using.sh
[ur14cs290@code ~]$ sh using.sh
username: ur14cs290
terminal: pts/118
date: 2017-03-26
time: 22:33
ip address: (10.3.28.83)
[ur14cs290@code ~]$ vi occword.sh
[ur14cs290@code ~]$ sh occword.sh
enter file name
second.txt
word you want to search
angel
3
[ur14cs290@code ~]$ 
```

Result: The unix commands are executed successfully.

Ex. No. 6	**User Defined Function and Arrays**
Date of Exercise	**27-02-2017**

Link: **https://www.youtube.com/watch?v=u08kIcfCvI0**

Aim: To write a C program to implement system calls

Description:

User-defined function:

```
function_name () {
  list of commands
}
```

Arrays:

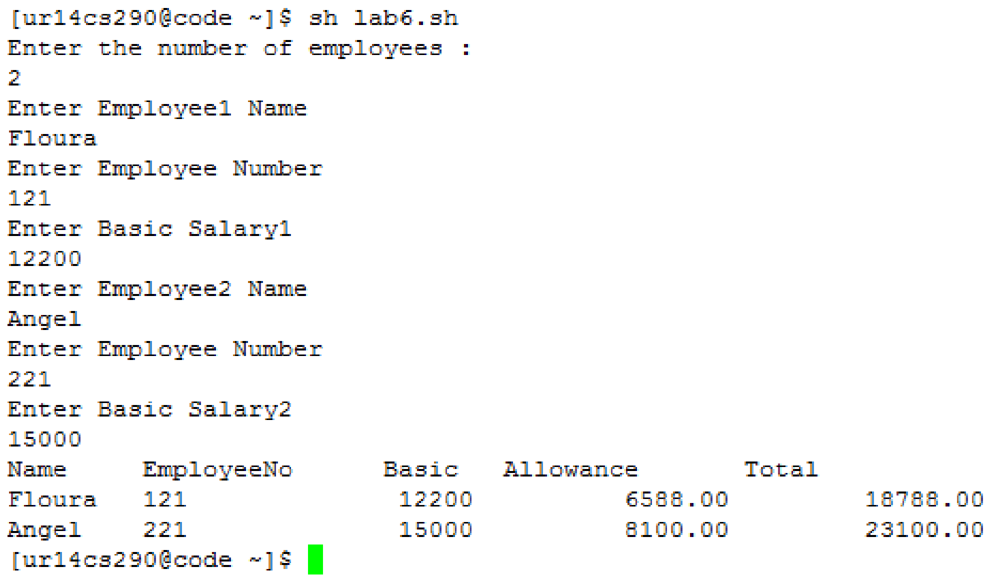
Arrays provide a method of grouping a set of variables.

array_name[index]=value

Program:

```
getdetails() {
echo "Enter Employee$i Name"
read name[$i]
echo "Enter Employee Number"
read empno[$i]
echo "Enter Basic Salary$i"
read basic[$i]
}


calculate() {
allowance[$i]=`echo ${basic[$i]}*0.54|bc`
total[$i]=`echo ${allowance[$i]}+${basic[$i]}|bc`

}

display() {

echo -e "${name[$i]} \t ${empno[$i]} \t  ${basic[$i]} \t ${allowance[$i]} \t ${total[$i]}"

}

echo "Enter the number of employees : "

read n

for i in `seq 1 $n`

do

getdetails

done

for i in `seq 1 $n`

do

calculate

done

echo -e "Name \t EmployeeNo \t Basic \t Allowance \t Total"

for i in `seq 1 $n`

do

display

done
```

Output:



```
[ur14cs290@code ~]$ sh lab6.sh
Enter the number of employees :
2
Enter Employee1 Name
Floura
Enter Employee Number
121
Enter Basic Salary1
12200
Enter Employee2 Name
Angel
Enter Employee Number
221
Enter Basic Salary2
15000
Name     EmployeeNo      Basic   Allowance       Total
Floura   121              12200          6588.00         18788.00
Angel    221              15000          8100.00         23100.00
[ur14cs290@code ~]$
```

Result: The Unix commands are executed successfully.

Ex. No. 7	**System Calls**
Date of Exercise	**13-03-2017**

Link: https://www.youtube.com/watch?v=T2dZrhbE0n0

Aim: To write a C program to implement system calls

Description:

A system call, sometimes referred to as a kernel call, is a request in a Unix-like operating system made via a software interrupt by an active process for a service performed by the kernel. A process (also frequently referred to as a task) is an executing (i.e., running) instance of a program.

System calls	**Function**
open	open an existing file or create a new file
read	Read data from a file
write	Write data to a file
lseek	Move the read/write pointer to the specified location
close	Close an open file
unlink	Delete a file
chmod	Change the file protection attributes
stat	Read file information from inodes

Flag	**Description**
O_RDONLY	open for reading only
O_WRONLY	open for writing only
O_RDWR	open for reading and writing
O_NONBLOCK	do not block on open
O_APPEND	append on each write
O_CREAT	create file if it does not exist

O_TRUNC	truncate size to 0
O_EXCL	error if create and file exists
O_SHLOCK	atomically obtain a shared lock
O_EXLOCK	atomically obtain an exclusive lock
O_DIRECT	eliminate or reduce cache effects
O_FSYNC	synchronous writes
O_NOFOLLOW	do not follow symlinks

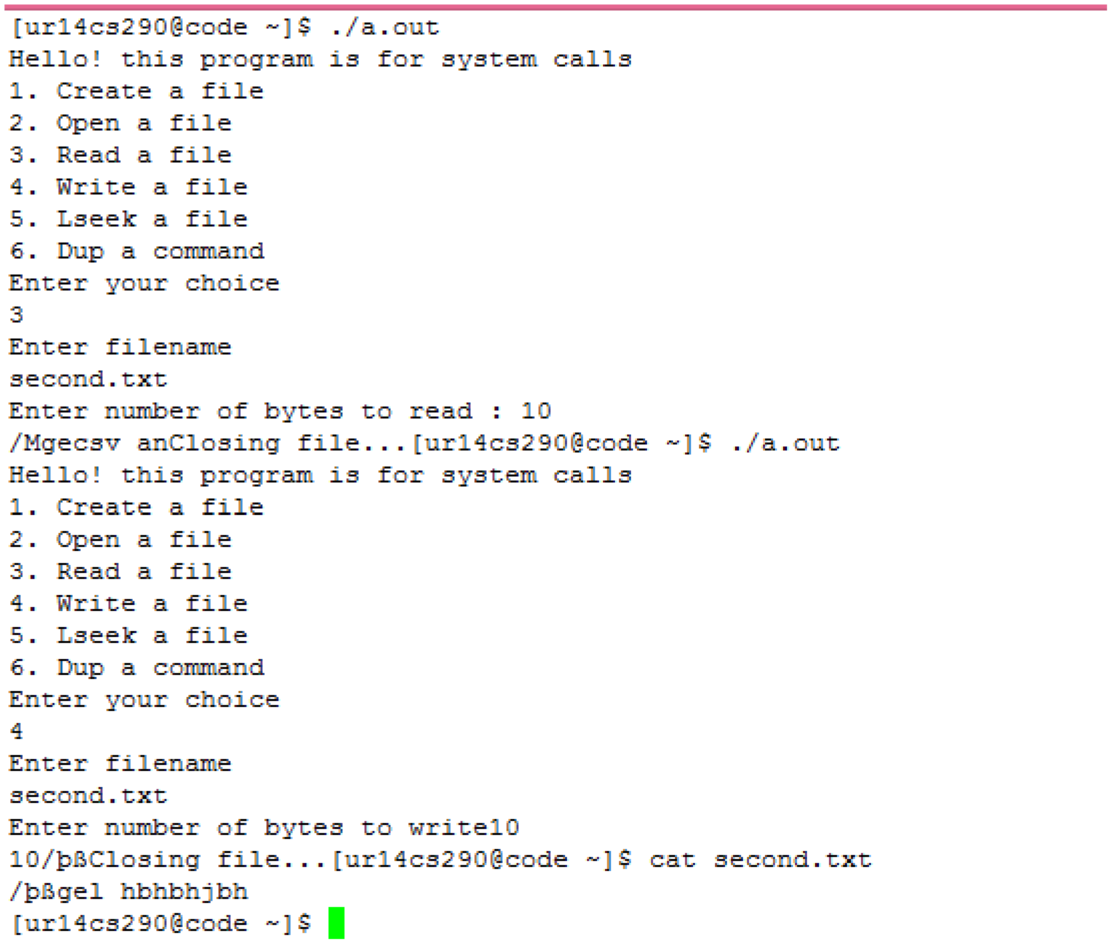
Program:

1. Use creat(), open(), write(),read(), lseek(), dup() and close()

```
#include <sys/types.h>
 #include <sys/stat.h>
 #include <fcntl.h>
 #include <stdio.h>
void main(){
int ch,offset;
int n,i;
int position;
char buffer[100];
char fname[100];
char count[100];
int count1;
int fd,fd1;
printf("Hello! this program is for system calls\n");
printf("1. Create a file\n");
printf("2. Open a file\n");
printf("3. Read a file\n");
printf("4. Write a file\n");
printf("5. Lseek a file\n");
printf("6. Dup a command\n");
printf("Enter your choice\n");
scanf("%d",&ch);
printf("Enter filename\n");
scanf("%s",&fname);
fd = open("second.txt",O_RDONLY);
if(fd == -1){
printf("File does not exist");
ch = 1;
}
switch(ch){
case 1:
printf("Creating a file .. ");
fd1 = creat(fname,07770);
break;
case 2:
if(fd == -1){
printf("File does not exist");
}
else{
printf("Opening a file .. ");
fd1 = open("second.txt",O_RDWR);
```

```
}
break;
case 3:
printf("Enter number of bytes to read : ");
scanf("%d",&count1);
n = read(fd,buffer,count1);
if(count1 >0){
for(i=0;i<count1;i++){
printf("%c",buffer[i]);
}
}
else{
printf("Error : no details");
}
break;
case 4:
printf("Enter number of bytes to write");
scanf("%d",&count1);
fd = open(fname,O_WRONLY);
n = write(fd,buffer,count1);
printf("%d",n);
if(n>0){
for(i=0;i<n;i++){
printf("%c",buffer[i]);
}
```

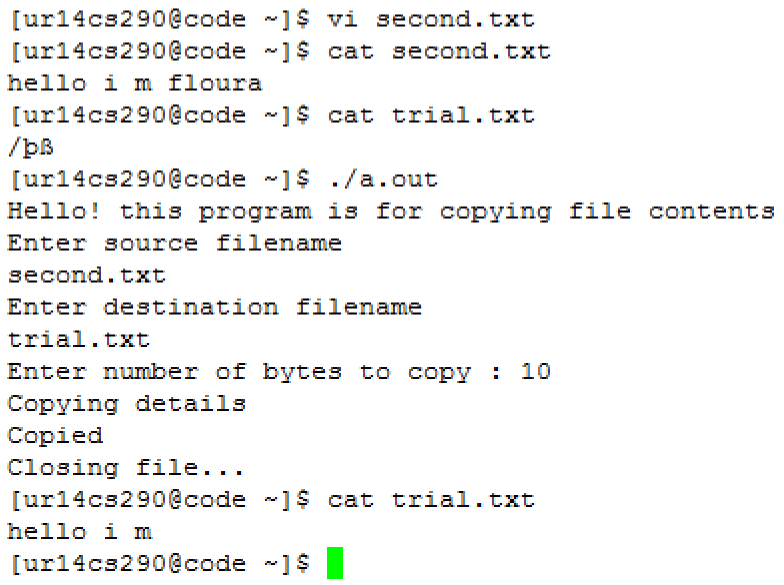
```
}

else{

printf("Error : no details");

}

break;

case 5:

printf("Enter your offset value");

scanf("%d",&offset);

position = lseek(fd,offset,0);

printf("Position:");

printf("%d",position);

break;

case 6:

fd1 = dup(fd);

printf("Original file : Duplicate file");

printf("%d  :  %d",fd,fd1);

break;

default:

printf("Default");

}

printf("Closing file...");

close(fd);

}
```



```
[ur14cs290@code ~]$ ./a.out
Hello! this program is for system calls
1. Create a file
2. Open a file
3. Read a file
4. Write a file
5. Lseek a file
6. Dup a command
Enter your choice
3
Enter filename
second.txt
Enter number of bytes to read : 10
/Mgecsv anClosing file...[ur14cs290@code ~]$ ./a.out
Hello! this program is for system calls
1. Create a file
2. Open a file
3. Read a file
4. Write a file
5. Lseek a file
6. Dup a command
Enter your choice
4
Enter filename
second.txt
Enter number of bytes to write10
10/þßClosing file...[ur14cs290@code ~]$ cat second.txt
/þßgel hbhbhjbh
[ur14cs290@code ~]$ 
```

2. Copy the contents of one file to another file

```
#include <sys/types.h>
 #include <sys/stat.h>
 #include <fcntl.h>
 #include <stdio.h>
void main(){
int ch,offset;
int n,i;
int position;
char buffer[100];
char fname[100];
char fname2[100];
char count[100];
int count1;
int fd2,fd,fd1;
printf("Hello! this program is for copying file contents\n");
printf("Enter source filename\n");
scanf("%s",&fname);
printf("Enter destination filename\n");
scanf("%s",&fname2);
fd = open("second.txt",O_RDONLY);
fd2 = open(fname2,O_WRONLY);
if(fd2 == -1){
printf("File does not exist\n");
ch = 1;
}
if(ch == 1){
printf("Creating a file ..\n ");
fd1 = creat(fname2,07770);
}
printf("Enter number of bytes to copy : ");
scanf("%d",&count1);
n = read(fd,buffer,count1);
if(n >0){
printf("Copying details\n");
}

else{

printf("Error1 : no details");

}

n = write(fd2,buffer,count1);

if(n>0){

printf("Copied\n");

}

else{

printf("Error : no details\n");

}

printf("Closing file...\n");

close(fd);

}
```



```
[ur14cs290@code ~]$ vi second.txt
[ur14cs290@code ~]$ cat second.txt
hello i m floura
[ur14cs290@code ~]$ cat trial.txt
/þß
[ur14cs290@code ~]$ ./a.out
Hello! this program is for copying file contents
Enter source filename
second.txt
Enter destination filename
trial.txt
Enter number of bytes to copy : 10
Copying details
Copied
Closing file...
[ur14cs290@code ~]$ cat trial.txt
hello i m
[ur14cs290@code ~]$
```

Result: The systemcalls commands are executed successfully.

Ex. No. 8	**PROCESS CREATION**
Date of Exercise	**20-03-2017**

Link: https://www.youtube.com/watch?v=LPEBaG1K5b4

Aim: To write a shell script to understand the usage of user defined functions and arrays

Description:

Fork: In computing, particularly in the context of the Unix operating system and its workalikes, fork is an operation whereby a process creates a copy of itself. It is usually a system call, implemented in the kernel. Fork is the primary (and historically, only) method of process creation on Unix-like operating systems.

Command: fork()

Functions: getpid() //process id

getppid() //parent process id

wait()

sleep()

Program:

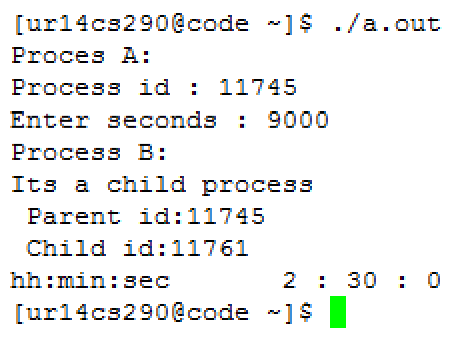
1. Write a C/C++ program to create the following process hierarchy and perform the following;

   • Process A gets seconds as an input from the user.

   • Process B calculates hours, mins and seconds and print it in hh:mm:sec format.

   • Print the process ID and parent process ID of each process.

```
#include <unistd.h>

#include<stdio.h>

#include<sys/types.h>

void main(){

int pid;
int seconds;
printf("Proces A: \n" );
int a ;
pid = getpid();
printf("Process id : %d",pid);
if(pid == 0){
printf("\nIts a child process\n : Parent id:");
a=getppid();
printf("%d",a);
printf("\n");
}
printf("\nEnter seconds : ");
scanf("%d",&seconds);
printf("Process B: \n");
pid = fork();
if(pid == 0){
printf("Its a child process\n Parent id:");
a=getppid();
printf("%d",a);
a=getpid();
printf("\n Child id:");
printf("%d",a);
printf("\n");
int hours = seconds/3600;
int minutes = seconds - (hours*3600);
minutes = minutes/60;

seconds = seconds - (minutes*60) - (hours*3600);

printf("hh:min:sec\t %d : %d : %d ",hours,minutes,seconds);

printf("\n");

}

}
```



```
[ur14cs290@code ~]$ ./a.out
Proces A:
Process id : 11745
Enter seconds : 9000
Process B:
Its a child process
 Parent id:11745
 Child id:11761
hh:min:sec       2 : 30 : 0
[ur14cs290@code ~]$
```

2. Write a C/C++ program to create the following process hierarchy and perform the following;

• Process A reads string as input in a char[]

• Process B reverses the string and displays

• Process C checks the string for palindrome

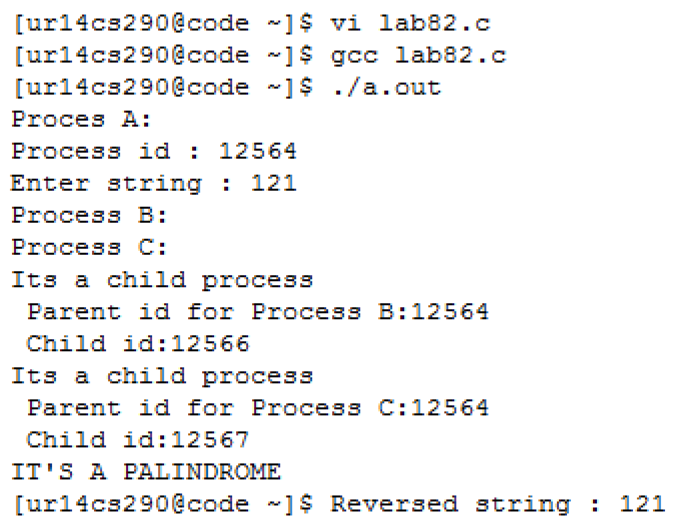
• Print the process ID and parent process ID of each process.

```
#include <unistd.h>

#include<stdio.h>

#include<sys/types.h>

#include<string.h>

#include  <signal.h>
```

```
#include  <sys/ipc.h>

#include  <sys/shm.h>


char *strrev(char *str)

{

    char *p1, *p2;


    if (! str || ! *str)

        return str;

    for (p1 = str, p2 = str + strlen(str) - 1; p2 > p1; ++p1, --p2)

    {

        *p1 ^= *p2;

        *p2 ^= *p1;

        *p1 ^= *p2;

    }

    return str;

}


void main(){

int pid;

char input[100];

char input2[100];

printf("Proces A: \n" );

int a ;

pid = getpid();
```

```
printf("Process id : %d",pid);
if(pid == 0){
printf("\nIts a child process\n : Parent id:");
a=getppid();
printf("%d",a);
printf("\n");
}
printf("\nEnter string : ");
scanf("%s",&input);
printf("Process B: \n");
pid = fork();
if(pid == 0){
printf("Its a child process\n Parent id:");
a=getppid();
printf("%d",a);
a=getpid();
printf("\n Child id:");
printf("%d",a);
printf("\n");
printf("Reversed string : %s",strrev(input));
sleep(1);
printf("\n");
kill(a, SIGQUIT);
}
printf("Process C: \n");
```

```
pid = fork();
if(pid == 0){
printf("Its a child process\n Parent id:");
a=getppid();
printf("%d",a);
a=getpid();
printf("\n Child id:");
printf("%d",a);
printf("\n");
if(input == strrev(input)){
printf("IT'S A PALINDROME");
printf("\n");
}
else{
printf("IT'S NOT A PALINDROME");
printf("\n");
}
sleep(1);
printf("\n");
kill(a, SIGQUIT);
}
}
```



```
[ur14cs290@code ~]$ vi lab82.c
[ur14cs290@code ~]$ gcc lab82.c
[ur14cs290@code ~]$ ./a.out
Proces A:
Process id : 12564
Enter string : 121
Process B:
Process C:
Its a child process
 Parent id for Process B:12564
 Child id:12566
Its a child process
 Parent id for Process C:12564
 Child id:12567
IT'S A PALINDROME
[ur14cs290@code ~]$ Reversed string : 121
```

3. Write a C/C++ program to create the following process hierarchy and perform the following;

• Process A reads an integer as input

• Process B finds the factorial of the number

• Process C checks Armstrong or not

• Process D prints the Fibonacci series

• Process E checks the number for prime

• Process F reverses the number

• Print the process ID and parent process ID of each process.

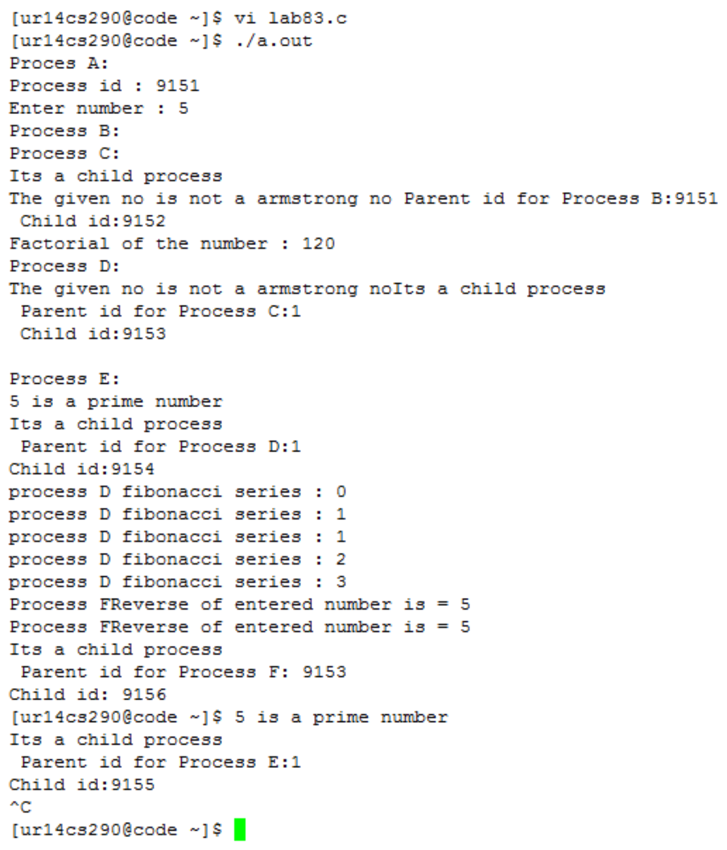
```
#include <unistd.h>

#include<stdio.h>

#include<sys/types.h>

#include<string.h>

#include  <signal.h>

#include  <sys/ipc.h>

#include  <sys/shm.h>
```

```
void main(){
int pid;
int i;
int input;
int input2 = 1;
printf("Proces A: \n" );
int a,a1,a2,a3,a4 ;
pid = getpid();
printf("Process id : %d",pid);
if(pid == 0){
printf("\nIts a child process\n : Parent id:");
a=getppid();
printf("%d",a);
printf("\n");
}
printf("\nEnter number : ");
scanf("%d",&input);
printf("Process B: \n");
pid = fork();
if(pid == 0){
for(i=1;i<=input;i++){
input2 *= i;
}
printf("Its a child process\n Parent id for Process B:");
a=getppid();
```

```
printf("%d",a);
a=getpid();
printf("\n Child id:");
printf("%d",a);
printf("\n");
printf("Factorial of the number : %d", input2);
printf("\n");
printf("Process D: \n");
pid = fork();
if(pid ==0){
printf("Its a child process\n Parent id for Process D:");
a1=getppid();
printf("%d",a1);
a1=getpid();
printf("\nChild id:");
printf("%d",a1);
printf("\n");
int c,next=0;
int first=0;
int second = 1;
input2 = input;
for ( c = 0 ; c <input2  ; c++ )
  {
    if ( c <= 1 )
      next = c;
```

```
        else
        {
          next = first + second;
          first = second;
          second = next;
        }
        printf("process D fibonacci series : %d\n",next);
      }

    kill(a1,SIGQUIT);
    printf("\n");
    }
    kill(a,SIGQUIT);
    }
    printf("Process C: \n");
    pid = fork();
    int temp = input;
    input2 = input;
    int rem=0;
    int cube,sum=0;
       while (input2 != 0)
       {
          rem = input2 % 10;
          cube = pow(rem,2);
          sum = sum + cube;
        input2 = input2 / 10;
    }
      if (sum == temp)
        printf ("The given no is armstrong no");
      else
        printf ("The given no is not a armstrong no");

    if(pid == 0){
    printf("Its a child process\n Parent id for Process C:");
    a2=getppid();
    printf("%d",a2);
    a2=getpid();
    printf("\n Child id:");
    printf("%d",a2);
    printf("\n");
    printf("\n");
    printf("Process E:\n");
    pid = fork();
    int flag = 0;
    int j;
    input2 = input;
      for (j = 2; j <= input2 / 2; j++)
      {
        if ((input2 % j) == 0)
        {
          flag = 1;
          break;
        }
      }
      if (flag == 0)
        printf("%d is a prime number \n", input);
       else
        printf("%d is not a prime number \n", input);
    if(pid ==0){
    printf("Its a child process \n Parent id for Process E:");
    a3=getppid();
    printf("%d",a3);
    printf("\nChild id:");
    a3=getpid();
    printf("%d",a3);
    printf("\n");
    kill(a3,SIGQUIT);
    }
    printf("Process F");
    pid = fork();
    int reverse = 0;
    input2 = input;
     while (input2 != 0)
      {
        reverse = reverse * 10;
```

```
    reverse = reverse + input2%10;
    input2  = input2/10;
  }
  printf("Reverse of entered number is = %d\n", reverse);
if(pid ==0){
printf("Its a child process \n Parent id for Process F: ");
a4 = getppid();
printf("%d",a4);
printf("\nChild id: ");
a4=getpid();
printf("%d",a4);
printf("\n");
kill(a4,SIGQUIT);
//kill(a2,SIGQUIT);
}
kill(a2,SIGQUIT);
}
}
```



```
[ur14cs290@code ~]$ vi lab83.c
[ur14cs290@code ~]$ ./a.out
Proces A:
Process id : 9151
Enter number : 5
Process B:
Process C:
Its a child process
The given no is not a armstrong no Parent id for Process B:9151
 Child id:9152
Factorial of the number : 120
Process D:
The given no is not a armstrong noIts a child process
 Parent id for Process C:1
 Child id:9153

Process E:
5 is a prime number
Its a child process
 Parent id for Process D:1
Child id:9154
process D fibonacci series : 0
process D fibonacci series : 1
process D fibonacci series : 1
process D fibonacci series : 2
process D fibonacci series : 3
Process FReverse of entered number is = 5
Process FReverse of entered number is = 5
Its a child process
 Parent id for Process F: 9153
Child id: 9156
[ur14cs290@code ~]$ 5 is a prime number
Its a child process
 Parent id for Process E:1
Child id:9155
^C
[ur14cs290@code ~]$ 
```

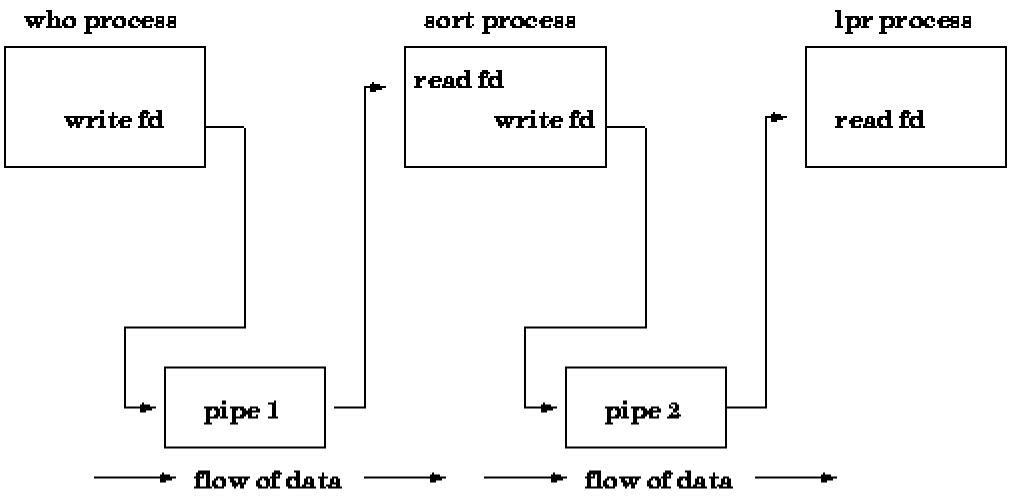
Result: The process creation using fork was executed successfully.

Ex. No. 9	**INTERPROCESS COMMUNICATION USING PIPES**
Date of Exercise	**27-03-2017**

Link: https://www.youtube.com/watch?v=oz3rJfVy_-M

Aim: To write a shell script to understand the usage of pipes in interprocess communication.

Description:



Pipes are a form of Inter-Process Communication (IPC) implemented on Unix and Linux variants. The kernel provides the synchronization between the processes accessing the same pipe. Data stored in the pipe is read on a First-In First-Out (FIFO) basis. The read /write operations are guaranteed to be atomic. The pipe is automatically removed by the OS when all the processes using it terminates. In fact, a pipe is a buffer managed by the kernel. It is a temporary storage of the data to be transferred between participating cooperative processes. The kernel takes care of the process synchronization.

Program:

1. Write a C/C++ program to implement named Pipes. Process1 creates a pipe and writes a string to the Pipe. Process2 opens the pipe reads that string and displays number of vowels in that string

   Writer.c :

   ```
   #include <fcntl.h>
   ```

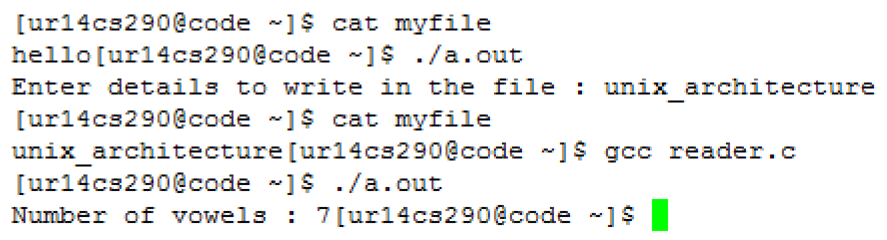
```
#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdio.h>

#include <string.h>


int main()

{

   int fd;

   char *myfifo = "/home/floura/Desktop/sem6/unix lab/labprograms/myfile";

   mkfifo(myfifo, 0666);

   fd = open("myfile", O_WRONLY);

   char c[100];

   printf("Enter details to write in the file : ");

   scanf("%s",&c);

   int length = strlen(c);

   write(fd, c, length);

   close(fd);

   unlink(myfifo);

   return 0;

}
```

Reader.c

```
#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>
```

```
#include <unistd.h>

#include <stdio.h>

#include <string.h>


#define MAX_BUF 1024


int main()

{

   int fd;

      int i=0;

   char * myfifo = "/home/floura/Desktop/sem6/unix lab/labprograms/myfile";

   char buf[MAX_BUF];

   fd = open("myfile",O_RDONLY);

   read(fd, buf, MAX_BUF);

   printf("Received: %s\n", buf);

   close(fd);

      int num=0;

   int len = strlen(buf);

   for( i=0;i<len;i++){

      if(buf[i] == 'a'||buf[i] == 'e'||buf[i] == 'i'||buf[i] == 'o'||buf[i] == 'u'){

            num++;

            }

      }

      printf("Number of vowels : %d",num);

   return 0;
```

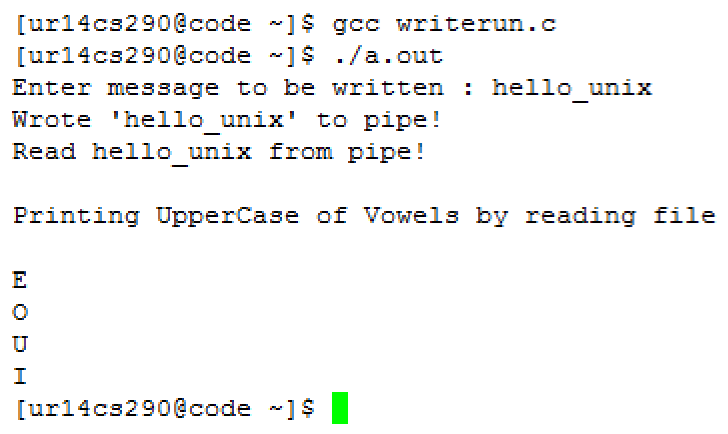
}



```
[ur14cs290@code ~]$ cat myfile
hello[ur14cs290@code ~]$ ./a.out
Enter details to write in the file : unix_architecture
[ur14cs290@code ~]$ cat myfile
unix_architecture[ur14cs290@code ~]$ gcc reader.c
[ur14cs290@code ~]$ ./a.out
Number of vowels : 7[ur14cs290@code ~]$
```

2. Write a C/C++ program to implement unnamed pipes. Parent process creates the pipe and writes a string to the pipe. The child process reads the pipe and makes all vowels in that string to uppercase and display

```
#include <stdio.h>

#include <string.h>

#define READ 0

#define WRITE 1


void main () {

  int fd[2], bytesRead;

  char message [100];

  char input[100];

  printf("Enter message to be written : ");

  scanf("%s",input);


  pipe ( fd );


  if ( fork ( ) == 0 ) {

    close (fd[READ]);
    write (fd[WRITE], input, strlen ( input) +1); //for null +1
    close (fd[WRITE]);
    printf("Wrote '%s' to pipe!\n", input);
  } else {
    close (fd[WRITE]);
    bytesRead = read ( fd[READ], message, 100);
    printf ( "Read %s from pipe!\n", message);
   int i=0;
        printf("\nPrinting UpperCase of Vowels by reading file\n\n");
        //printf("Message : %s",message);
   for(i=0;i<strlen(message);i++){
        if(message[i] == 'a'||message[i] == 'e'||message[i] == 'i'||message[i] == 'o'||message[i]
== 'u'){
        printf("%c\n",toupper(message[i]));
        }
        }
    close ( fd[READ]);
  }
}
```



```
[ur14cs290@code ~]$ gcc writerun.c
[ur14cs290@code ~]$ ./a.out
Enter message to be written : hello_unix
Wrote 'hello_unix' to pipe!
Read hello_unix from pipe!

Printing UpperCase of Vowels by reading file

E
O
U
I
[ur14cs290@code ~]$
```

Result: The inter process communication commands are executed successfully.

Ex. No. 10	**SHARED MEMORY USING INTER PROCESS COMMUNICATION**
Date of Exercise	**03-04-2017**

Link: https://www.youtube.com/watch?v=1eQrunVKmFk

Aim: To perform shared memory using inter-process communication.

Description:

A process creates a shared memory segment using shmget()|. The original owner of a shared memory segment can assign ownership to another user with shmctl(). It can also revoke this assignment. Other processes with proper permission can perform various control functions on the shared memory segment using shmctl(). Once created, a shared segment can be attached to a process address space using shmat()

Algorithm:

```
Step 1: create a structure simple with a[2][2] as an element of it

Step 2: prompt user for first matrix elements using the structure variable

Step 3: prompt user for first matrix elements using the structure variable

Step 4: perform addition of the matrices

Step 5:show the resulting matrix

Step 6: STOP
```

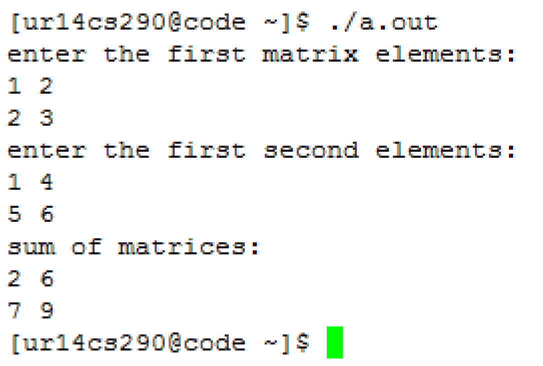
Program:

```
#define MAX_SIZE 4

#include<fcntl.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<stdio.h>

#include<sys/shm.h>

void main()
```

```
{
        struct simple
        {
                int a[2][2];
        }*m;
        int i,j,id,shmid;
        shmid=shmget(id,sizeof(struct simple)*2,IPC_CREAT|0666);
        m=shmat(shmid,0,0);
        printf("enter the first matrix elements:\n");
        for(i=0;i<2;i++){
                for(j=0;j<2;j++){
                        scanf("%d",&m[0].a[i][j]);
                                }
                        }
        printf("enter the first second elements:\n");
        for(i=0;i<2;i++){
                for(j=0;j<2;j++){
                        scanf("%d",&m[1].a[i][j]);
                                }
                        }
        for(i=0;i<2;i++){
                for(j=0;j<2;j++){
                        m[2].a[i][j]=m[0].a[i][j]+m[1].a[i][j];
                                }
                        }
```

```
    printf("sum of matrices: \n");
    for(i=0;i<2;i++){
        for(j=0;j<2;j++){
            printf("%d ",m[2].a[i][j]);
            }
            printf("\n");
            }
}
```

OUTPUT:



```
[ur14cs290@code ~]$ ./a.out
enter the first matrix elements:
1 2
2 3
enter the first second elements:
1 4
5 6
sum of matrices:
2 6
7 9
[ur14cs290@code ~]$ 
```

Result: The programs to perform system management using inter process communication is executed by using shared memory.