

# INDEX

S.No.	LIST OF PROGRAMS	Page no.
WEEK 1	a) To Install Ubuntu Linux and <b>LINUX Commands</b> (File Handling utilities, Text processing utilities, Network utilities, Disk utilities, Backup utilities and Filters) b) Write a Shell Script that accepts a file name, starting and ending line numbers as arguments and displays all lines between the given line numbers. c) Write a shell script that deletes all lines containing the specified word in one or more filessupplied as arguments to it. d) Write a shell script that displays a list of all files in the current directory to which the userhas read, write and execute permissions.	1-
WEEK 2	a) Write a shell script that receives any number of file names as arguments checks if everyargument supplied is a file or directory and reports accordingly. whenever the argumentis a file it reports no of lines present in it b) Write a shell script that accepts a list of file names as its arguments, counts and reports theoccurrence of each word that is present in the first argument file on other argument files.	29-32
WEEK 3	a) Write a shell script to list all of the directory files in a directory b) Write a shell script to find factorial of a given number.	33-35
WEEK 4	a) write an awk script to count number of lines in a file that does not contain vowels b) write an awk script to find the no of characters ,words and lines in a file	36-38
WEEK 5	Implement in c language the following Unix commands using system calls a) cat    b) ls    c) mv	39-42
WEEK 6	Write a C program that takes one or more file/directory names as command line input and reports following information a) File Type                                         b) Number Of Links c) Time of last Access                                 d) Read ,write and execute permissions	43-44
WEEK 7	Write a C program to list every file in directory, its inode number and file name	45
WEEK 8	a) Write a C program to create child process and allow parent process to display “parent”and the child to display “child” on the screen b) Write a C program to create zombie process c) Write a C program to illustrate how an orphan process is created	46-48
WEEK 9	a) Write a C program that illustrate communication between two unrelated process usingnamed pipes b) Write a C program that receives a message from message queue and display them	49-52
WEEK 10	a) Write a C program to allow cooperating process to lock a resource for exclusive use( using semaphore) b) Write a C program that illustrate the suspending and resuming process using signal c) Write a C program that implements producer-Consumer system with two process usingsemaphore	53-56
WEEK11	Write client server programs using c for interaction between server and client process using Unix Domain sockets	57-60
WEEK 12	Write a C program that illustrates two processes communicating using Shared memory	61-62



## EXPERIMENT NO: 1. b)

**Aim:** Write a Shell Script that accepts a file name, starting and ending line numbers as Arguments and displays all lines between the given line numbers.

### **ALGORITHM:**

Step 1: Create a file with 5-6 lines of data

File can be created by vi sample.dat or cat sample.dat

Step 2: Now write a shell script with

vi 1.sh

Step 3: Check the no of arguments for shell script

if 0 arguments then print no arguments

else if 1 argument then print 1 argument

else if 2 arguments then print 2 arguments

else check for file is there or not(if file is not there print file does not exists)

1else sed -ne "\$2','\$3' p' \$1

sed is one of powerful filter(streameditor)

-e default option (script on command line)

-n suppresses automatic output

\$2 first line number passed \$3 2nd line number passed

p is a print command (prints current content to the pattern space).

\$1 is name of file from which we are printing data between the line numbers.

Step 4: top

### **Script Name: 1sh**

```
#!/bin/bash
if [ $# -lt 3 ]
then
    echo "To execute you have to enter 3 arguments in command line in following order..."
    echo " File Name ,starting line number and ending line number..."
else
    sed -n $2,$3p $1
fi
```

### **Commands used in the script:**

#### **Sed command:**

stream editor for filtering and transforming text

1. Replacing or substituting string

Sed command is mostly used to replace the text in a file. The below simple sed command replaces the word "unix" with "linux" in the file.

\$sed 's/unix/linux/' file.txt

2. Replacing the nth occurrence of a pattern in a line

\$sed 's/unix/linux/2' file.txt

Replaces 2<sup>nd</sup> occurrence

3. printing lines for a given range

\$sed -n 1,5p hello.txt Prints first 5 lines in the file hello.txt

**nl command:**

The nl utility in Linux is used to give number lines of a file on console.

Example:

```
$ nl sort.txt
 1      UK
 2      Australia
 3      Newzealand
 4      Brazil
 5      America
```

**Execution:**

check how many lines of data in the input file

```
root@localhost sh]# cat hello.txt | nl
1 abc
2 def
3 ghi
4 abc
5 abc
6 cccc
```

Executing Shell script:

```
run1:
[root@localhost sh]# sh 1.sh abc1.txt 2 4
def
ghi
abc
```

compare with the data in the file and output

**Viva Questions**

1. What is a shell script?
2. How to find current shell name
3. How to switch to another shell
4. How to execute shell Script

**Exercises:**

S.No.	Task
1	Write a shell script to count no of character in a file ,prompt for inputfile
2	Write a shell script to count no of character in a file name given in command prompt
3	Write a shell script to perform arithmetic operation using case statement

## EXPERIMENT NO: 1. c)

**AIM: Write a shell script that deletes all lines containing the specified word in one or more files Supplied as arguments to it.**

ALGORITHM:

Step 1: Create a file with 5-6 lines of data  
Create the 2 file f1 and f2 as vi s1 and vi s2

Step2: Now write a shell script with

vi 2.sh

step3: Check the no of arguments for shell script

```
if 0 arguments then print no arguments
else pattern=$1(word will be stored in pattern)
for fname in $*
for every filename in given files
if it is a file if [ -f $fname ] then
print DELETING $pattern FROM
$fname sed '/$pattern/d' $fname
sed acts as filter if word is a file in any line that will be deleted
'/' is used to represent regular expressions
'/d' is a delete command in sed
else print file NOT FOUND
```

**Script name: 2.sh**

```
#!/bin/bash
if [ $# -lt 2 ]then
echo "Enter atleast two files as input in command line"
else
printf "enter a word to find:"
read word
for f in $*
do
printf "\n In File $f:\n"
sed /$word/d $f
done
fi
```

Execution:

```
run1:
check data in input files
[root@localhost sh]# cat abc1.txt
abc
def
ghi
abc
abc
cccc
[root@localhost sh]# cat abc2.txt
abc
```

```

def
ghi
abc
abc
cccc
Executing shell script
[root@localhost sh]# sh 2.sh abc1.txt abc2.txt
enter a word to find:abc
In File abc1.txt:
def
ghi
cccc
In File abc2.txt:
def
ghi
cccc

```

**Expected output:**

Displays lines from files s1 s2 after deleting the word hi

**Viva Questions**

- 1.Explain various loops in shell script
- 2.Explain grep
- 3.Explain egrep
- 4.Explain fgrep
5. Explain sed

**Exercises:**

S.No.	Task
1	Write a shell script to count occurrence of a word in a file
2	Write a shell script to print line numbers in which a particular word has occurred where word is provided as input.

## EXPERIMENT NO: 1. d)

**Aim:** Write a shell script that displays a list of all files in the current directory to which the user has read, write and execute permissions.

**ALGORITHM:**

Step1: selects list of files from present working directory  
Step 2:check for each file whether it has read, write and execute permissions if true goto step 3  
Step 3: print file  
Step 4 :stop

**Script name:** 3.sh

```
#!/bin/bash
echo "List of Files which have Read, Write and Execute Permissions in Current Directory are.."
for file in *
do
    if [ -r $file -a -w $file -a -x $file ]
    then
        echo $file
    fi
done
```

**Execution:**

```
$sh 3.sh
```

**Expected output:**

by executing above shell script you will get all files which have read ,write and execute Permissions in current working directory

**sample output**

```
[root@localhost sh]# sh 3.sh
List of Files which have Read, Write and Execute Permissions in Current Directory are...
5.sh
a.out
```

**Viva Questions:**

- 1.Display all files in a directory
- 2.how to use chmod
- 3.How to change file permissions

**Exercises:**

S.No.	Task
1	Write a shell script to display all files with read or write or execute permissions provide a selection menu
2	Write a comparison report for using chmod using symbolic representation or octal number representation
3	Write a shell script to count no of files in current directory with full permissions

## EXPERIMENT NO: 2.a)

**Aim:-Write a shell script that receives any number of file names as arguments checks if every argument supplied is a file or directory and reports accordingly. Whenever the argument is a file it reports no of lines present in it**

**ALGORITHM:**

step 1: if arguments are less than 1 print Enter at least one input file name and goto step 9  
Step 2: selects list a file from list of arguments provided in command line  
Step 3: check for whether it is directory if yes print is directory and goto step 9  
step 4: check for whether it is a regular file if yes goto step 5 else goto step 8  
Step 5: print given name is regular file  
step 6: print No of lines in file  
step 7: goto step  
step 8: print not a file or a directory  
step 9: stop

**Script name: 4.sh**

```
#!/bin/bash
if[ $# -lt 1 ]
then
    echo "Enter at least one input file name"
else
    for i in $*
    do
        if[ -d $i ]
        then
            echo " given name is directory"
        elif[ -f$i ]
        then
            echo " given name is file: $i"
            echo " No of lines in file are : `wc -l $i`"
        else
            echo "given name is not a file or a directory"
        fi
    done
fi
```

**Execution:**

provide two file names as input one a regular file and other directory  
for example abc1.txt a text file as first argument and vazralu a directory as second argument

**Run1:**

```
[root@localhost sh]# sh 4.sh abc1.txt vazralu
given name is file: abc1.txt
No of lines in file are : 7 abc1.txt
```

vazralu is directory

**run 2:**[root@localhost sh]# sh 4.sh abc1.txt abc2.txt  
given name is file: abc1.txt  
No of lines in file are : 7 abc1.txt  
given name is file: abc2.txt  
No of lines in file are : 7 abc2.txt

#### Viva Questions:

1. What is an internal command in Linux?

Internal commands are also called shell built-in commands. Example: cd,fg. Since these are shell built-in, no process is created while executing these commands, and hence are considered to be much faster.

2. x and y are two variables containing numbers? How to add these 2 numbers?

\$ expr \$x + \$y

3. How to add a header record to a file in Linux?

\$ sed -i '1i HEADER' file

4. How to find the list of files modified in the last 30 mins in Linux?

\$ find . -mmin -30

5. How to find the list of files modified in the last 20 days?

\$ find . -mtime -20

#### Exercises:

S.No.	Task
1	Write a shell script to count no of regular files in the current workingdirectory
2	Write a shell script to display list of currently logged users

## EXPERIMENT NO: 2. b)

**Aim:-Write a shell script that accepts a list of file names as its arguments, counts and reports the occurrence of each word that is present in the first argument file on other argument files.**

**ALGORITHM:**

step1: Check the no of arguments for shell script  
    if 0 arguments then print no arguments

step2: else translate each word in the first file is to be on separate line  
        which will be stored in temp file

step3: for i in \$\*  
        for every filename in given files

step 4: translate each word in the file is to be on separate line  
        which will be stored in temp1 file

step5: count no of lines in temp file assign it to j

step6: initialize j=1

step 7: while i < j  
        extract the line that are common in both the file by using  
        head and tail commands  
        then apply the filter grep to count and print the lines  
        which are common to files  
        increment j

step 8: stop

**Script name:5.sh**

```
#!/bin/bash
echo "no of arguments $#"
if [ $# -le 2 ]
then
    echo "Error : Invalid number of arguments."
    exit
fi
str=`cat $1 | tr '\n' '''
for a in $str
do
    echo "in file $a"
    echo "Word = $a, Count = `grep -c \"$a\" $2`"
done
```

**Execution and output:**

check data in abc1.txt file

```
[root@localhost sh]# cat abc1.txt
abc
def
ghi
abc
abc
```

```
cccc
check data in abc1.txt file
[root@localhost sh]# cat abc2.txt
abc
def
ghi
abc
abc
cccc
```

### **executing script**

```
[root@localhost sh]# sh 5.sh abc1.txt abc2.txt
Word = abc, Count = 3
Word = def, Count = 1
Word = ghi, Count = 1
Word = abc, Count = 3
Word = abc, Count = 3
Word = cccc, Count = 1
```

### Viva Questions

#### 1. What is Shell Scripting ?

Shell scripting, in Linux or Unix, is programming with the shell using which you can automate your tasks. A shell is the command interpreter which is the interface between the User and the kernel. A shell script allows you to submit a set of commands to the kernel in a batch. In addition, the shell itself is very powerful with many properties on its own, be it for string manipulation or some basic programming stuff.

#### 2. The command "cat file" gives error message "--bash: cat: Command not found". Why?

It is because the PATH variable is corrupt or not set appropriately. And hence the error because the cat command is not available in the directories present PATH variable.

#### 3. How to find the length of a string in Linux?

```
$ x="welcome" $ echo ${#x} 7
```

#### 4. What are the different timestamps associated with a file?

Modification time:- Refers to the time when the file is last modified.

Access time :- The time when the file is last accessed.

Changed time :- The time when the attributes of the file are last changed.

#### 5. How to get the list of files alone in a directory in Linux?

```
$ ls -lrt | grep ^-
```

### **Exercises:**

S.No.	Task
1	Write a shell script to print prime numbers
2	Write a shell script to print Fibonacci numbers

## EXPERIMENT NO: 3. a)

**Aim:-Write a shell script to list all of the directory files in a directory.**

Algorithm:

Step1: enter the name of the directory

    Read dir

Step2: if it is a directory

    Then list the files present in that directory

    By using ls command with -p option to list all directory files in a given directory

Step 3: else enter the directory name

Step 4: stop

**Script name: 6.sh**

```
#!/bin/bash
echo " Enter dir name: "
read dir
if [ -d $dir ]
then
    printf " Files in Directory $dir are...\n`ls $dir`"
else
    echo " Dir does not exist"
fi
```

Execution and output:

```
[root@localhost sh]# sh 6.sh
```

Enter dir name:

japs

Files in Directory japs are...

abc1.txt

abc2.txt

ls-1.c

prg5

s1

Viva Questions

1. A string contains a absolute path of a file. How to extract the filename alone from the absolute path in Linux?

```
$ x="/home/guru/temp/f1.txt"
```

```
$ echo $x | sed 's^.*/^'
```

2. How to find all the files created after a pre-defined date time, say after 10th April 10AM?

This can be achieved in 2 steps:

1. Create a dummy file with the time stamp, 10th April 10AM.

2. Find all the files created after this dummy file.

```
$ touch -t 1004101000 file
```

```
$ find . -newer file
```

3. The word "Unix" is present in many .txt files which is present across many files and also files present in sub directories. How to get the total count of the word "Unix" from all the .txt files?

```
$ find . -name *.txt -exec grep -c Unix '{}' \; | awk '{x+=$0;}END{print x}'
```

### Exercises:

S.No.	Task
1	How to find the files modified exactly before 30minutes? \$ find . -mmin 30
2	How to print the contents of a file line by line in Linux?

## EXPERIMENT NO: 3. b)

**Aim:-Write a shell script to find factorial of a given number.**

### **ALGORITHM**

Step 1: read any number to find factorial

Step 2: initialize fact=1 and i=1

Step 3: while i less than

do

fact=fact\* i

i=i+1

done

step 4:print fact

step 5:stop.

**Script Name:7.sh**

```
#!/bin/bash
echo "Factorial Calculation Script. .."
echo "Enter a number: "
read f
fact=1
factorial=1
while [ $fact -le $f ]
do
    factorial=`expr $factorial \* $fact`
    fact=`expr $fact + 1`
done

echo "Factorial of $f = $factorial"
```

Execution and Output:

```
[root@localhost sh]# sh 7.sh
Factorial Calculation Script....
Enter a number: 4
Factorial of 4 = 24
```

### **Exercises:**

S.No.	Task
1	Write a shell script to find sum of first n natural numbers
2	Write a shell script to find largest of given three numbers

## EXPERIMENT NO: 4. a)

**Aim:-write an awk script to count number of lines in a file that does not contain vowels**

### **ALGORITHM**

Step 1: create a file with 5-10 lines of data

Step 2: write an awk script by using grep command to filter the lines  
that do not contain vowels

```
awk '$0 ~/aeiou/ {print $0}' file1
```

step3: count=count+1

step4:print count

step5:stop

**Awk script name:**nm.awk

```
BEGIN{}  
{  
If($0 !~/[aeiou AEIOU]/)  
wordcount+=NF  
}  
END  
{  
print "Number of Lines are", wordcount  
}
```

**input file for awk script:**data.dat

```
bcdfghj  
abcdfghj  
bcdfghj  
ebcdfghj  
bcdfghj  
ibcdfghj  
bcdfghj  
obcdfghj  
bcdfghj  
ubcdfghj
```

Executing the script:

```
[root@localhost awk]# awk -f nm.awk data.dat
```

```
bcdfghj  
bcdfghj  
bcdfghj  
bcdfghj  
bcdfghj
```

Number of lines are 5

### **Exercises:**

S.No.	Task
1	Write an awk script to find square root of a given number
2	Write an awk script to find maximum of two numbers , read input from keyboard

## **EXPERIMENT NO: 4. b)**

**Aim:-write an awk script to find the no of characters ,words and lines in a file**

### **ALGORITHM**

Step 1: create a file with 5 to 10 lines of data

Step 2: write an awk script

    find the length of file

    store it in chrcnt

step3: count the no of fields (NF), store it in wordcount

step4: count the no of records (NR), store it in NR

step5: print chrcnt,NRwordcount

step6: stop

**Awk script name:nc.awk**

```
BEGIN{}  
{  
    print len=length($0),"\\t",$0  
    wordcount+=NF  
    chrcnt+=len  
}  
END {  
    print "total characters",chrcnt  
    print "Number of Lines are",NR  
    print "No of Words count:",wordcount  
}
```

**input data file name:data.dat**

```
bcdfghj  
abcdfghj  
bcdfghj  
ebcdfghj  
bcdfghj  
ibcdfghj  
bcdfghj  
obcdfghj  
bcdfghj  
ubcdfghj
```

Executing the script:

```
[root@localhost awk]# awk -f nc.awk data.dat
```

```
7 bcdfghj  
8 abcdfghj  
7 bcdfghj  
8 ebcdfghj  
7 bcdfghj
```

```
8 ibcdfghj
7 bcdflghj
8 obcdfghj
7 bcdflghj
8 ubcdfghj
total characters 75
Number of Lines are 10
No of Words count: 10
```

#### VIVA QUESTIONS:

1. How to find the last modified file or the newest file in a directory?

```
$ ls -lrt | grep ^- | awk '{print $NF}'
```

2. How to access the 10th command line argument in a shell script in Linux?

```
$1 for 1st argument, $2 for 2nd, etc... For 10th argument, ${10}, for 11th, ${11} and so on.
```

3. How to find the sum of all numbers in a file inLinux?

```
$ awk '{x+=$0}END{print x}' file
```

4. How to delete a file which has some hidden characters in the file name?

Since the rm command may not be able to delete it, the easiest way to delete a file with some hidden characters in its name is to delete it with the find command using the inode number of the file.

```
$ ls -li
total 32
9962571 -rw-r--r-- 1 guru users 0 Apr 23 11:35
$ find . -inum 9962571 -exec rm '{}' \;
```

5. Using the grep command, how can you display or print the entire file contents?

```
$ grep '*' file
```

6. What is the difference between a local variable and environment variable in Linux?

A local variable is the one in which the scope of the variable is only in the shell in which it is defined. An environment variable has scope in all the shells invoked by the shell in which it is defined.

## EXPERIMENT NO: 5

**Aim:** Implement in c language the following Unix commands using systemcalls    a)cat    b)ls    c)mv

a) AIM:-Write a c program to implement **cat command** using system calls

**Description:**

cat COMMAND:      cat linux command concatenates files and print it on the standard output.

**SYNTAX:**

cat [OPTIONS] [FILE]...

**OPTIONS:**

- A Show all.
- b Omits line numbers for blank space in the output.
- e A \$ character will be printed at the end of each line prior to a new line.
- E Displays a \$ (dollar sign) at the end of each line.
- n Line numbers for all the output lines.
- s If the output has multiple empty lines it replaces it with one empty line.
- T Displays the tab characters in the output.
- v Non-printing characters (with the exception of tabs, new-lines & form-feeds) are printed visibly.

Operations With cat Command:

1. To Create a new file:

**\$cat > file1.txt**

This command creates a new file file1.txt. After typing into the file press control+d (^d) simultaneously to end the file.

2. To Append data into the file:

**\$cat >> file1.txt**

To append data into the same file use append operator >> to write into the file, else the file will be overwritten (i.e., all of its contents will be erased).

3. To display a file:

**\$cat file1.txt**

This command displays the data in the file.

4. To concatenate several files and display:

**\$cat file1.txt file2.txt**

The above cat command will concatenate the two files (file1.txt and file2.txt) and it will display the output in the screen. Some times the output may not fit the monitor screen. In such situation you can print those files in a new file or display the file using less command.

cat file1.txt file2.txt | less

5. To concatenate several files and to transfer the output to another file.

**\$cat file1.txt file2.txt > file3.txt**

In the above example the output is redirected to new file file3.txt. The cat command will create new file file3.txt and store the concatenated output into file3.txt.

**Algorithm:**

Step 1:Start  
Step 2:read arguments from keyboard at command line  
Step 3:if no of arguments are less than two print ENTER CORRECT ARGUMENTS  
Else goto step 4  
Step4:read the date from specified file and write it to destination file  
Step 5 :stop

**Program file name:catdemo.c**

```
#include<stdio.h>
#include<sys/types.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/stat.h>
int main(int argc,char *argv[])
{
    int fd,n;
    char buff[512];
    if(argc!=2)
        printf("ENTER CORRECT ARGUMENTS :");
    if((fd=open(argv[1],4))<0)
    {
        printf("ERROR");
        return 0;
    }
    while(n=read(fd,buff,sizeof(buff))>0)
        write(1,buff,n);
}
```

b) **AIM:-**Write a c program to implement **ls command** using systemcalls  
**Description:**

**ls command** is used to list the files present in a directory

**Algorithm:**

Step 1. Start.  
Step 2. open directory using opendir( ) system call.  
Step 3. read the directory using readdir( ) system call.  
Step 4. print dp.name and dp.inode .  
Step 5. repeat above step until end of directory.  
Step 6: Stop.

**Program name: lsdemo.c**

```
#include<stdio.h>
#include<dirent.h>
void quit(char*,int);
int main(int argc,char **argv)
{
```

```

DIR *dirop;
struct dirent *dired;
if(argc!=2)
{
    printf("Invalid number of arguments\n");
}
if((dirop=open(dir[1]))==NULL)
    printf("Cannot open directory\n");
while((dired=readdir(dirop))!=NULL)
    printf("%10d %s\n",dired->d_ino,dired->d_name);
closedir(dirop);
}

```

c) **AIM:** write a c program that simulates **mv command** (using system calls)

**Description:**

**mv command** is used to move or rename a file

synatax:

mv file1 file2

here file1 is renamed as file2

**Algorithm:**

Step 1: Start

Step 2: open an existed file and one new open file using open() system call

Step 3: read the contents from existed file using read( ) system call

Step 4: write these contents into new file using write system call using write( ) system call

Step 5: repeat above 2 steps until eof

Step 6: close 2 file using fclose( ) system call

Step 7: delete existed file using using unlink( ) system

Step 8: Stop.

**Program File name:** mvdemo.c

```

#include<stdio.h>
#include<string.h>
int main(int argc ,char *argv[])
{
int r,i;
char p[20],q[20];
if(argc<3)
    printf("improper arguments\n file names required\n");
else
if( argc==3)
{
    printf("\n%s\n",argv[1],argv[2]);
    r=link(argv[1],argv[2]);
    printf("%d\n",r);
    unlink(argv[1]);
}

```

```
    }
else
{
    for(i=1;i<argc-1;i++)
    {
        strcpy(p,argv[argc-1]);
        strcat(p,"/");
        strcat(p,argv[i]);
        printf("%s%s\n",argv[i],p);
        link(argv[i],p);
        unlink(argv[i]);
    }
}
```

## EXPERIMENT NO: 6

**Aim:Write a C program that takes one or more file/directory names as command line input and reports following information**

- |                      |                                       |
|----------------------|---------------------------------------|
| A) File Type         | B)Number Of Links                     |
| c) Time oflast Acces | D) Read,write and execute permissions |

Algorithm:

Step 1:start  
Step 2:Declare struct stat a  
Step 3:read arguments at command line  
Step 4: set the status of the argument using stat(argv[i],&a);  
Step 5:Check whether the given file is Directory file by using S\_ISDIR(a.st\_mode)  
    if it is a directory file print Directory file  
    Else  
        print is Regular file  
Step6: print number of links  
Step 7:print last time access  
Step 8:Print Read,write and execute permissions  
Step 9:stop

**Program File name: 13.c**

```
#include<stdio.h>
#include<sys/stat.h>
#include<time.h>
int main(int argc,char *argv[])
{
int i,j;
struct stat a;
for (i=1;i<argc;i++)
{
    printf("%s : ",argv[i]);
    stat(argv[i],&a);
    if(S_ISDIR(a.st_mode))
    {
        printf("is a Directory file\n");
    }
    else
    {
        printf("is Regular file\n");
    }

printf("*****File Properties*****\n");
printf("Inode Number:%d\n",a.st_ino);
printf("UID:%o\n",a.st_uid);
printf("GID:%o\n",a.st_gid);
printf("No of Links:%d\n",a.st_nlink);
printf("Last Access time:%s",asctime(localtime(&a.st_atime))));
```

```

        printf("Permission flag:%o\n",a.st_mode%512);
        printf("size in bytes:%d\n",a.st_size);
        printf("Blocks Allocated:%d\n",a.st_blocks);
        printf("Last modification time %s\n",ctime(&a.st_atime));
    }
}

```

**Exercises:**

S.No.	Task
1	write a c program that simulates mkdir command using systemcalls
2	write a c program that simulates rmdir command using systemcalls

## EXPERIMENT NO: 7

**Aim:** Write a C program to list every file in directory, its inode number and file name

**Algorithm:**

- Step 1:Start
- Step 2:Read Directory name
- Step 3:open the directory
- Step 4: print file name and Inode number of each file in the directory
- Step 5:Stop

**Program file name:**inode.c

```
#include<fcntl.h>
#include<stdio.h>
#include<dirent.h>
#include<sys/stat.h>
int main(int argc,char*argv[])
{
DIR *dirop;
struct dirent *dired;
if(argc!=2)
{
    printf("Invalid number of arguments\n");
}
else if((dirop=open(dir(argv[1]))==NULL)

        printf("Cannot open Directory\n");
else
{
printf("%10s %s \n","Inode","File Name");
    while((dired=readdir(dirop))!=NULL)
        printf("%10d %s\n ",dired->d_ino,dired->d_name);
    closedir(dirop);
}
return 0;
}
```

**Exercises:**

S.No.	Task
1	Write a c program to test whether the given file is seekable or not
2	Write a c program to for requesting and releasing lock

## **EXPERIMENT NO: 8 a)**

**Aim:** Write a C program to create child process and allow parent process to display “parent” and the child to display “child” on the screen

**Algorithm:**

- Step 1: start
- Step2: call the fork() function to create a child process
  - fork function returns 2 values
- step 3: which returns 0 to child process
- step 4:which returns process id to the parent process
- step 5:stop

**Program file name:16.c**

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main()
{
    int pid,pid1,pid2;
    pid=fork();
    if(pid== -1)
    {
        printf("ERROR IN PROCESS CREATION \n");
        exit(0);
    }
    if(pid!=0)
    {
        pid1=getpid();
        printf("\n the parent process ID is %d", pid1);
    }
    else
    {
        pid2=getpid();
        printf("\n the child process ID is %d\n", pid2);
    }
}
```

**Execution:**

```
[root@dba ~]# cc -o 16 16.c
[root@dba ~]# ./16
```

```
the child process ID is 4485
the parent process ID is 4484
```





## EXPERIMENT NO: 9 a)

**Aim:- Write a C program that illustrate communication between two unrelated process using named pipes**

**Algorithm for server :**

```
step 1:Start  
step 2:Create a first named pipe by using mkfifo system call  
        Pipe1=mkfifo(NP1,0666).  
step 3:if mkfifo returns -1 then  
        print a message that error in creating the pipe.  
step 4:Create a second named pipe by using mkfifo system call  
        Pipe2=mkfifo(NP2,0666).  
step 5:if mkfifo returns -1 then  
        print a message that error in creating the pipe.  
step 6:Open the first pipe for reading by open system call by setting  
        O_RDONLY Fd=open(NP1,O_RDONLY)  
step 7: Open the second pipe for writing by open system call by setting  
        O_WRONLY Fd=open(NP2,O_WRONLY)  
step 8:read the data from the first pipe by using read system call  
        numread=Read(fd,buf,MAX_BUF-SIZE) buf*numread+='\0'  
step 9:print the data that we have read from pipe  
step 10:convert the data to the upper case.  
step 11:write the converted string back to second pipe by write(fd,buf, strlen(buf))  
step 12:stop.
```

**Algorithm for client :**

```
Step 1:start  
Step 2:check whether the no of arguments specified were correct or not  
Step 3:if no of arguments are less then print error message  
Step 4:Open the first named pipe for writing by open system call by setting  
        O_WRONLY Fd=open(NP1,O_WRONLY)  
Step 5: .Open the second named pipe for reading by open system call by setting  
        O_RDONLY Fd=open(NP2,O_RDONLY)  
Step 6: write the data to the pipe by using write system call  
        write(fd,argv[1],strlen(argv[1]))  
Step 7: read the data from the first pipe by using read system call  
        numread=Read(fd,buf,MAX_BUF_SIZE) buf*numread+='\0'  
Step 8: print the data that we have read from pipe  
Step 9:stop
```

### **Program file name:named\_pipe.c**

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<string.h>
#include<fcntl.h>
void server(int,int);
void client(int,int);
int main()
{
    int p1[2],p2[2],pid;
    pipe(p1);
    pipe(p2);
    pid=fork();
    if(pid==0)
    {
        close(p1[1]);
        close(p2[0]);
        server(p1[0],p2[1]);
        return 0;
    }
    close(p1[0]);
    close(p2[1]);
    client(p1[1],p2[0]);
    wait();
    return 0;
}

void client(int wfd,int rfd)
{
    int i,j,n;
    char fname[2000];
    char buff[2000];
    printf("ENTER THE FILE NAME :");
    scanf("%s",fname);
    printf("CLIENT SENDING THE REQUEST ....PLEASE WAIT\n");

    sleep(10);
    write(wfd,fname,2000);
    n=read(rfd,buff,2000);
    buff[n]='\0';
    printf("THE RESULTS OF CLIENTS ARE .....\\n");
    write(1,buff,n);
}
void server(int rfd,int wfd)
{
    int i,j,n; char fname[2000];
    char buff[2000];
```

```

n=read(rfd, fname, 2000);
fname[n]='\0';
int fd=open(fname,O_RDONLY);
sleep(10);
if(fd<0)
    write(wfd,"can't open",9);
else
    n=read(fd,buff,2000);
write(wfd,buff,n);
}

```

**Exercises:**

S.No.	Task
1	Write a program to demonstrate the function of a pipe
2	Write a program to demonstrate the pipe function using dup() system call



## EXPERIMENT NO: 10 a)

**Aim:-Write a C program to allow cooperating process to lock a resource for exclusive use using, a) Semaphore**

```
#include<stdio.h>
#include<stdlib.h>
#include<error.h>
#include<sys/types.h>
#include<sys/IPC.h>
#include<sys/sem.h>
int main(void)
{
key_t key;
int semid;
union semun arg;
if((key==ftok("sem demo.c","j"))== -1)
{
perror("ftok");
exit(1);
}
if(semid=semget(key,1,0666|IPC_CREAT)== -1)
{
perror("semget");
exit(1);
}
arg.val=1;
if(semctl(semid,0,SETVAL,arg)== -1)
{
perror("smctl");
exit(1);
}
return 0;
}
```

**Exercises:**

S.No.	Task
1	Write a program using the simpler semaphore operation
2	Write a program to create a semaphore

## EXPERIMENT NO: 10 b)

**Aim:-Write a C program that illustrate the suspending and resuming process using signal**

### **Algorithm:**

- Step 1: call the signal function to generate the signal
- Step 2: execution of process will be started
- Step 3: call alarm function to suspend the execution of current process
- Step 4: then it will execute the signal function
- Step 5: again the process will be resumed
- Step 6: stop

### **Program**

```
#include<stdio.h>
int main()
{
    int n;
    if(signal(SIGALRM,sig_alarm)==SIG_ERR)
        printf("Signal error");
    alarm(5);
    for(n=0;n<=15;n++)
        printf("from for loop n=%d",n);
    printf("main program terminated");
}

void sig_alarm(int signo)
{
    printf("from sigalarm function");
}
```

### **Exercises:**

S.No.	Task
1	Write a program using kill and rise functions
2	Write a program using abort()

## EXPERIMENT NO: 10 c)

**Aim:-Write a C program that implements producer –consumer system with two processes using semaphores**

**Algorithm for producer :**

step 1:Start  
step 2:Create a named pipe by using mkfifo system call Pipe1=mkfifo(NP1,0666)  
step 3:if mkfifo returns -1 then print a message that error in creating the pipe  
step 4:Open the pipe for reading by open system call by setting O\_RDONLY Fd=open(NP1,O\_RDONLY)  
step 5:read the data from the pipe by using read system call  
    numread=Read(fd,buf,MAX-BUF-SIZE)  
step 6:print the data that we have read from pipe  
step 7:convert the data to the upper case.  
step 8:print the converted data  
step 9:stop.

**Algorithm for consumer:**

Step 1:start  
step 2:check whether the no of arguments specified were correct or not  
step 3:if no of arguments are less then print error message  
step 4:Open the pipe for writing by open system call by setting O\_WRONLY  
    Fd= open (NP1, O\_WRONLY )  
step 5: write the data to the pipe by using write system call write(fd,argv[1],strlen(argv[1]))  
step 6:stop

**Consumer:**

```
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#define MAXSIZE 10
#define FIFO_NAME "myfifo"
int main()
{
    int fifoid; int fd, n; char *r;
    system("clear");
    r=(char *)malloc(sizeof(char)*MAXSIZE); int open_mode=O_RDONLY;
    if( (fd=open(FIFO_NAME, open_mode)) < 0 )
    {
        printf("\nError: Named pipe cannot be opened\n"); exit(0);
    }
    while(1)
    {
        n=read(fd, r, MAXSIZE); if(n > 0)
            printf("\nConsumer read: %s", r);
    }
} /*main close*/
```

**Producer program:**

```
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#define MAXSIZE 10

#define FIFO_NAME "myfifo"
```

```

int main()
{
int fifoid; int fd, n; char *w;
int open_mode;
    system("clear");
    w=(char *)malloc(sizeof(char)*MAXSIZE);
    open_mode=O_WRONLY;
    fifoid=mkfifo(FIFO_NAME, 0755);
if(fifoid== -1)
{
    printf("\nError: Named pipe cannot be Created\n"); exit(0);
}
if( (fd=open(FIFO_NAME, open_mode)) < 0 )
{
    printf("\nError: Named pipe cannot be opened\n");
    exit(0);
}
while(1)
{
    printf("\nProducer :"); fflush(stdin);
    read(0, w, MAXSIZE);
    n=write(fd, w, MAXSIZE);
    if(n > 0)
        printf("\nProducer sent: %s", w);

}
} /*main close*/

```

Output:

```

$ cc -o producer producer.c          #first window
$ cc -o consumer consumer.c         # second window
$ ./producer      #first window
$./consumer      # second window
Producer:
Producer sent: hai      #first window
Consumer read: hai      # second window
Producer sent: good morning #first window
Consumer read: good morning # second window
Producer sent: welcome   #first window
Consumer read: welcome   # second window

```



```

int connection_handler(int connection_fd)
{
    int nbytes;
    char buffer[256];
    nbytes = read(connection_fd, buffer, 256);
    buffer[nbytes] = 0;
    printf("MESSAGE FROM CLIENT: %s\n", buffer);
    nbytes = snprintf(buffer, 256, "hello from the server");
    write(connection_fd, buffer, nbytes);
    close(connection_fd);
    return 0;
}

int main(void)
{
    struct sockaddr_un address;
    int socket_fd, connection_fd;
    socklen_t address_length;
    pid_t child;
    socket_fd = socket(PF_UNIX, SOCK_STREAM, 0);
    if(socket_fd < 0)
    {
        printf("socket() failed\n");
        return 1;
    }

    unlink("./demo_socket");

    /* start with a clean address structure */
    memset(&address, 0, sizeof(struct sockaddr_un));

    address.sun_family = AF_UNIX;
    snprintf(address.sun_path, UNIX_PATH_MAX, "./demo_socket");

    if(bind(socket_fd,
           (struct sockaddr *) &address,
           sizeof(struct sockaddr_un)) != 0)
    {
        printf("bind() failed\n");
        return 1;
    }
}

```

```

if(listen(socket_fd, 5) != 0)
{
    printf("listen() failed\n");
    return 1;
}

while((connection_fd = accept(socket_fd,
                             (struct sockaddr *) &address,
                             &address_length)) > -1)
{
    child = fork();
    if(child == 0)
    {
        /* now inside newly created connection handling process */
        return connection_handler(connection_fd);
    }
    /* still inside server process */
    close(connection_fd);
}
close(socket_fd);
unlink("./demo_socket");
return 0;
}

```

### **Client.c**

```

#include <stdio.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
#include <string.h>

int main(void)
{
    struct sockaddr_un address;
    int socket_fd, nbytes;
    char buffer[256];

    socket_fd = socket(PF_UNIX, SOCK_STREAM, 0);
    if(socket_fd < 0)
    {
        printf("socket() failed\n");
        return 1;
    }

```

```

/* start with a clean address structure */
memset(&address, 0, sizeof(struct sockaddr_un));

address.sun_family = AF_UNIX;
snprintf(address.sun_path, UNIX_PATH_MAX, "./demo_socket");

if(connect(socket_fd,
           (struct sockaddr *) &address,
           sizeof(struct sockaddr_un)) != 0)
{
    printf("connect() failed\n");
    return 1;
}
nbytes = snprintf(buffer, 256, "hello from a client");
write(socket_fd, buffer, nbytes);

nbytes = read(socket_fd, buffer, 256);
buffer[nbytes] = 0;

printf("MESSAGE FROM SERVER: %s\n", buffer);

close(socket_fd);
return 0;
}

```

### **Exercises:**

S.No.	Task
1	Write a program to demonstrate getting and setting the socket options through socket related system call
2	Write a program to demonstrate bind system call.

## EXPERIMENT NO: 12

**Aim:-Write a C program that illustrates two processes communicating using Shared memory**

**Algorithm:-**

step1.Start

step 2.Include header files required for the program are

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
```

step 3.Declare the variable which are required as

```
pid_t pid
int *shared /* pointer to the shm */
int shmid
```

step 4.Use shmget function to create shared memory

```
#include <sys/shm.h>
int shmget(key_t key, size_t size, int shmflg)
```

The shmget() function shall return the shared memory identifier associated with key The

argument key is equal to IPC\_PRIVATE. so that the operating system selects the next

available key for a newly created shared block of memory. Size represents size of shared memory block Shmflg shared memory permissions which are represented by octalinteger

```
shmid = shmget (IPC_PRIVATE, sizeof(int), IPC_CREAT | 0666);
```

print the shared memory id

step 5.if fork()==0 Then

begin

```
shared = shmat(shmid, (void *) 0, 0)
print the shared variable(shared) *shared=2
print *shared sleep(2)
print *shared
```

end

step 6.else

begin

```
shared = shmat(shmid, (void *) 0, 0)
print the shared variable(shared)
print *shared sleep(1) *shared=30
printf("Parent value=%d\n", *shared);
sleep(5)
shmctl(shmid, IPC_RMID, 0)
```

end

step 7.stop.

**Program:**

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
```