Write a Shell script that accepts a filename, starting and ending line numbers as arguments and displays all the lines between the given line numbers.

AIM:

Write a Shell script that accepts a filename, starting and ending line numbers as arguments and displays all the lines between the given line numbers.

Program:

echo "enter the filename"

read fname

echo "enter the starting line number"

read s

echo "enter the ending line number"

read n

sed -n $s,$n\p $fname | cat > newline

cat newline

Write a Shell script that displays list of all the files in the current directory to which the user has read, Write and execute permissions.

#### AIM:

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

#### Program:

echo "enter the directory name"

read dir

if [ -d $dir ]

then

cd $dir

ls > f

exec < f

while read line

do

if [ -f $line ]

then

if [ -r $line -a -w $line -a -x $line ]

then

echo "$line has all permissions"

else

echo "files not having all permissions"

fi

fi

done

fi

Write a Shell script that receives any number of file names as arguments checks if every argument supplied is a file or a directory and reports accordingly. Whenever the argument is a file, the number of lines on it is also reported.

#### AIM:

Write a Shell script that receives any number of file names as arguments checks if every argument supplied is a file or a directory and reports accordingly. Whenever the argument is a file, the number of lines on it is also reported.

#### Program:

for x in $\*

do

if [ -f $x ]

then

echo " $x is a file "

echo " no of lines in the file are "

wc -l $x

elif [ -d $x ]

then

echo " $x is a directory "

else

echo " enter valid filename or directory name "

fi

done

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.

#### 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.

#### Program:

if [ $# -eq 0 ]

then

echo "no arguments"

else

tr " " "

" < $1 > temp

shift

for i in $\*

do

tr " " "

" < $i > temp1

y=`wc -l < temp`

j=1

while [ $j -le $y ]

do

x=`head -n $j temp | tail -1`

c=`grep -c "$x" temp1`

echo $x $c

j=`expr $j 1`

done

done

fi

Write a Shell script to list all of the directory files in a directory.

#### AIM:

Write a  Shell script to list all of the directory files in a directory.

#### Program:

# !/bin/bash

echo "enter directory name"

read dir

if[ -d $dir]

then

echo "list of files in the directory"

ls –l $dir|egrep ‘^d’

else

echo "enter proper directory name"

fi

Write a awk script to find the number of characters, words and lines in a file.

#### AIM:

Write a awk script to find the number of characters, words and lines in a file.

#### Program:

BEGIN{print "record.\t characters \t words"}

#BODY section

{

len=length($0)

total\_len =len

print(NR,":\t",len,":\t",NF,$0)

words =NF

}

END{

print("\n total")

print("characters :\t" total len)

print("lines :\t" NR)

}

#### Write  a C Program that makes a copy of a file using standard I/O  and system calls.

#### Program:

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

void typefile (char \*filename)

{

int fd, nread;

char buf[1024];

fd = open (filename, O\_RDONLY);

if (fd == -1) {

perror (filename);

return;

}

while ((nread = read (fd, buf, sizeof (buf))) > 0)

write (1, buf, nread);

close (fd);

}

int

main (int argc, char \*\*argv)

{

int argno;

for (argno = 1; argno < argc; argno )

typefile (argv[argno]);

exit (0);

}

Write in C the following Unix commands using system calls A). cat B). ls C). mv

#### AIM:

#### Implement in C the following Unix commands using system calls

A). cat       B). ls     C). mv

Program:

#### ****A) cat****

#include<sys/types.h>

#include<sys/stat.h>

#include<stdio.h>

#include<fcntl.h>

main( int argc,char \*argv[3] )

{

int fd,i;

char buf[2];

fd=open(argv[1],O\_RDONLY,0777);

if(fd==-argc)

{

printf("file open error");

}

else

{

while((i=read(fd,buf,1))>0)

{

printf("%c",buf[0]);

}

close(fd);

}

}

**B) ls**

#include <sys/types.h>

#include <sys/dir.h>

#include <sys/param.h>

#include <stdio.h>

#define FALSE 0

#define TRUE 1

extern int alphasort();

char pathname[MAXPATHLEN];

main() {

int count,i;

struct dirent \*\*files;

int file\_select();

if (getwd(pathname) == NULL )

{

printf("Error getting pathn");

exit(0);

}

printf("Current Working Directory = %sn",pathname);

count = scandir(pathname, &files, file\_select, alphasort);

if (count <= 0)

{

printf("No files in this directoryn");

exit(0);

}

printf("Number of files = %dn",count);

for (i=1;i<count 1; i)

printf("%s \n",files[i-1]->d\_name);

}

int file\_select(struct direct \*entry)

{

if ((strcmp(entry->d\_name, ".") == 0) ||(strcmp(entry->d\_name, "..") == 0))

return (FALSE);

else

return (TRUE);

}

**C) mv**

#include<sys/types.h>

#include<sys/stat.h>

#include<stdio.h>

#include<fcntl.h>

main( int argc,char \*argv[] )

{

int i,fd1,fd2;

char \*file1,\*file2,buf[2];

file1=argv[1];

file2=argv[2];

printf("file1=%s file2=%s",file1,file2);

fd1=open(file1,O\_RDONLY,0777);

fd2=creat(file2,0777);

while(i=read(fd1,buf,1)>0)

write(fd2,buf,1);

remove(file1);

close(fd1);

close(fd2);

}

Write a Program that takes one or more file/directory names as command line input and reports the following information on the file. A). File type B). Number of links. C). Time of last access. D). Read,Write and Execute permissions.

#### AIM:

Write  a Program that  takes one or more file/directory names as command line input and reports the following information on the file.                A). File type   B). Number of links.     C).Time of last access.    D).Read,Write and Execute permissions.

#### Program:

clear

for i in $\*

do

if [ -d $i ]

then

echo “Given directory name is found as $i”

fi

if [ -f $i ]

then

echo “Given name is a file as $i “

fi

echo “Type of file/directory $i”

file $i

echo “Last access time is:”

ls -l$i | cut-c 31-46

echo "no.of links"

ln $i

if [ -x $i –a -w $i-a –r $i ]

then

echo “$i contains all permission”

else

echo “$i does not contain all permissions”

fi

done

Write a C program to emulate the Unix ls-l command.

#### AIM:

Write a C program to emulate the Unix ls-l command.

#### Program:

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <stdlib.h>

int main()

{

int pid; //process id

pid = fork(); //create another process

if ( pid < 0 )

{ //fail

printf(“\nFork failed\n”);

exit (-1);

}

else if ( pid == 0 )

{ //child

execlp ( “/bin/ls”, “ls”, “-l”, NULL ); //execute ls

}

else

{ //parent

wait (NULL); //wait for child

printf(“\nchild complete\n”);

exit (0);

}

}

Write a C program to list for every file in a directory, its inode number and file name.

#### AIM:

14.Write  a C program to list for every file in a directory,its inode number and file name.

#### Program:

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

main(int argc, char \*argv[])

{

char d[50];

if(argc==2)

{

bzero(d,sizeof(d));

strcat(d,"ls ");

strcat(d,"-i ");

strcat(d,argv[1]);

system(d);

}

else

printf("\nInvalid No. of inputs");

}

Write a C Program that demonstrates redirection of standard output to a file .EX: ls > f1.

#### AIM:

Write  a C Program that demonstrates redirection of standard output to a file .EX:ls>f1.

#### Program:

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

main(int argc, char \*argv[])

{

char d[50];

if(argc==2)

{

bzero(d,sizeof(d));

strcat(d,"ls ");

strcat(d,"> ");

strcat(d,argv[1]);

system(d);

}

else

printf("\nInvalid No. of inputs");

}

#### output:

student@ubuntu:~$ gcc –o std.out std.c

student@ubuntu:~$ls

downloads documents listing.c listing.out std.c std.out

student@ubuntu:~$ cat > f1

^z

student@ubuntu:~$./std.out f1

student@ubuntu:~$cat f1

downloads

documents

listing.c

listing.out

std.c

std.out

Write a C program to create a Zombie process.

#### AIM :

Write  a C program to create a Zombie process.

#### Program:

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

int main ()

{

int pid\_t child\_pid;

child\_pid = fork ();

if (child\_pid > 0) {

sleep (60);

}

else {

exit (0);

}

return 0;

}

Design and Develop program for creating a private message queue.

#### AIM:

Design and Develop program for creating a private message queue.

#### Program:

#include<stdio.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/msg.h>

void main()

{

int queue\_id;

/\*create a private messaga queue,with access only to the owner\*/

queue\_id=msgget(IPC\_PRIVATE,0600);

if(queue\_id==-1)

{

perror(“msgget”);

exit(1);

}

printf(“message queue ID=%d\n”,queue\_id);

}

#### Output:

guest-b7wxtw@ubuntu:~$gcc –o message.out message.c

guest-b7wxtw@ubuntu:~$./message.out

message queue ID:32769

Design and Develop a program to demonstrate pipe function using dup system call.

#### AIM:

Design and Develop a program to demonstrate pipe function using  dup system call.

#### Program:

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

int ptd[2];

char buff[30];

if(pie(ptd)==-1)

{

perror(“pipe failed”);

exit(1);

}

if(!fork())

{

printf(“CHILD:writing to the pipe\n”);

write(ptd[1],”test”,5);

printf(“CHILD: exiting\n”);

exit(0);

}

else

{

printf(“PARENT:reading from pipe\n”);

read(ptd[0],buf,5);

printf(“PARENT: read \”%S “\n”,buff);

wait(NULL);

}

}

Write a C program that illustrates two processes communicating using shared memory.

#### AIM:

Write a C program that illustrates two processes communicating using shared memory.

#### Program:

#include<stdio.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/shm.h>

Struct country

{

Char name[30];

Char capital\_city [30];

Char currency[30];

Int population;

};

Int main(int argc,char\*argv[])

{

Int shm\_id;

Char\*shm\_addr;

Int\*countries\_num;

Struct country\*countries;

Struct shmid\_ds shm\_desc;

Shm\_id=shmget(100,2048,IPC\_CREAT|IPC\_EXCL\0600);

If(shm\_id==-1){

Perror(“main:shmget:”);

Exit(1);

}

Shm\_addr=shmat(shm\_id,NULL,0);

If(!shm\_addr){

Perror(“main:shmat:”);

Exit(1);

}

Countries\_num=(int\*)shm\_addr;

\*countries\_num=0;

Countries=(struct country\*)((void\*)shm\_addr sizeof(int));

Strcpy(countries[0],name,”U.S.A”);

Strcpy(countries[0],capital\_city,”WASHINGTON”);

Strcpy(countries[0],currency,”U.S.DOLLAR”);

Countries[0].population=250000000;

( countries\_num) ;

Strcpy(countries[1].name,”israel”);

Strcpy(countries[1].capital\_city,”jerushalem”);

Strcpy(countries[1].currency,”NEW ISRAEL SHEKED”);

Countries[1].population=6000000;

(\*countries\_num) ;

Strcpy(countries[2].name,”France”);

Strcpy(countries[2].capital\_city,”paris”);

Strcpy(countries[2].currency,”Frank”);

Countries[2].population=60000000;

(\*countries\_num) ;

For(i=0;i<(\*countries\_num);i )

{

Printf(“country%d:\n”,i 1);

Printf(“name:%d:\n”,i 1);

Printf(“currency:%s:\n”,countries[i].currency);

Printf(“population:%d:\n”,countries[i].population);

}

If(shmdt(shm\_addr)==-1){

Perror(“main:shmdt:”);

}

If(shmctl(shm\_id,IPC\_RMID,&SHM\_DESC)==-1)

{

Perror(“main:shmctl:”);

}

return 0;

}