**1)Write a program to display the file content in reverse order using lseek system call.**

#include<stdlib.h>

#include<stdio.h>

#include<fcntl.h>

#include<string.h>

#include<sys/stat.h>

#include<unistd.h>

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

{

int source, dest, n;

char buf;

int filesize;

int i;

if (argc != 3) {

fprintf(stderr, "usage %s <source> <dest>", argv[0]);

exit(-1);

}

if ((source = open(argv[1], O\_RDONLY)) < 0)

{ fprintf(stderr, "can't open source\n");

exit(-1);

}

if ((dest = open(argv[2], O\_WRONLY | O\_CREAT | O\_TRUNC)) < 0)

{ fprintf(stderr, "can't create dest\n");

exit(-1);

}

filesize = lseek(source, (off\_t) 0, SEEK\_END);

printf("Source file size is %d\n", filesize);

for (i = filesize - 1; i >= 0; i--)

{

lseek(source, (off\_t) i, SEEK\_SET);

if ((n = read(source, &buf, 1)) != 1) {

fprintf(stderr, "can't read 1 byte");

exit(-1);

}

if ((n = write(dest, &buf, 1)) != 1) {

fprintf(stderr, "can't write 1 byte");

exit(-1);

}

}

write(STDOUT\_FILENO, "DONE\n", 5);

close(source);

close(dest);

return 0;

}

**2)**Write a program

* 1. to read first 20 characters from a file
  2. seek to 10th byte from the beginning and display 20 characters from there
  3. seek 10 bytes ahead from the current file offset and display 20 characters
  4. display the file size

#include<stdio.h>

#include<unistd.h>

#include<fcntl.h>

#include<sys/types.h>

int main()

{

int file=0, n;

char buffer[25];

if((file=open("testfile.txt",O\_RDONLY)) < -1)

printf(“file open error\n”);

if(read(file,buffer,20) != 20)

printf(“file read operation failed\n”);

else

write(STDOUT\_FILENO, buffer, 20);

printf("\n");

if(lseek(file,10,SEEK\_SET) < 0)

printf(“lseek operation to beginning of file failed\n”);

if(read(file,buffer,20) != 20)

printf(“file read operation failed\n”);

else

write(STDOUT\_FILENO, buffer, 20);

printf("\n");

if(lseek(file,10,SEEK\_CUR) < 0)

printf(“lseek operation to beginning of file failed\n”);

if(read(file,buffer,20) != 20)

printf(“file read operation failed\n”);

else

write(STDOUT\_FILENO, buffer, 20);

printf("\n");

if((n = lseek(file,0,SEEK\_END)) <0)

printf(“lseek operation to end of file failed\n”);

printf("size of file is %d bytes\n",n);

close(file);

return 0;

}

**3)** Write a program to display various details of a file using stat structure(At least 5 fields)

#include <unistd.h>

#include <stdio.h>

#include <sys/stat.h>

#include <sys/types.h>

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

{

if(argc != 2)

return 1;

struct stat fileStat;

if(stat(argv[1],&fileStat) < 0)

return 1;

printf("Information for %s\n",argv[1]);

printf("File Size: \t\t%d bytes\n",fileStat.st\_size);

printf("Number of Links: \t%d\n",fileStat.st\_nlink);

printf("File inode: \t\t%d\n",fileStat.st\_ino);

printf("uid :%d",fileStat.st\_uid);

printf("File Permissions: \t");

printf( (S\_ISDIR(fileStat.st\_mode)) ? "d" : "-");

printf( (fileStat.st\_mode & S\_IRUSR) ? "r" : "-");

printf( (fileStat.st\_mode & S\_IWUSR) ? "w" : "-");

printf( (fileStat.st\_mode & S\_IXUSR) ? "x" : "-");

printf( (fileStat.st\_mode & S\_IRGRP) ? "r" : "-");

printf( (fileStat.st\_mode & S\_IWGRP) ? "w" : "-");

printf( (fileStat.st\_mode & S\_IXGRP) ? "x" : "-");

printf( (fileStat.st\_mode & S\_IROTH) ? "r" : "-");

printf( (fileStat.st\_mode & S\_IWOTH) ? "w" : "-");

printf( (fileStat.st\_mode & S\_IXOTH) ? "x" : "-");

printf("\n\n");

printf("The file %s a symbolic link\n", (S\_ISLNK(fileStat.st\_mode)) ? "is" : "is not");

return 0;

}

**4)Write a program to implement ls –li command which list the files in a specified directory. Your program should Print 5 attributes of files.**

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

#include <dirent.h>

#include <time.h>

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

{

struct dirent \*dir;

struct stat mystat;

DIR \*dp;

dp = opendir(".");

if(dp)

{

while(dir = readdir(dp))

{

stat(dir->d\_name,&mystat);

// inode mode uid guid access\_time

printf("%ld %o %d %d %s %s\n",

mystat.st\_ino,mystat.st\_mode,mystat.st\_uid,mystat.st\_gid,ctime(&mystat.st\_atime),dir->d\_name);

}

}

}

**Output:**

**gedit prog.c**

**cc prog.c**

**./a.out**

**3048649 100664 1000 1000 Thu Apr 19 15:23:17 2018**

**su2.txt**

**3048647 100664 1000 1000 Thu Apr 19 15:24:07 2018**

**prog.c**

**3048646 40775 1000 1000 Thu Apr 19 15:24:13 2018**

**.**

**3048648 100664 1000 1000 Thu Apr 19 15:23:09 2018**

**su1.txt**

**2892839 40755 1000 1000 Thu Apr 19 15:19:09 2018**

**..**

**3048650 100775 1000 1000 Thu Apr 19 15:24:13 2018**

**a.out**

**5)Write a program to remove empty files from the given directory.**

#include <stdio.h>

#include <fcntl.h>

#include <unistd.h>

#include <dirent.h>

int main()

{

DIR \*dp;

struct dirent \*dir;

int fd,n;

dp = opendir("."); //open current directory

if(dp)

{

while((dir = readdir(dp)) != NULL)

{

fd = open(dir->d\_name,O\_RDWR,0777);

n = lseek(fd,0,SEEK\_END);

if(!n)

{

unlink(dir->d\_name);

}

}

}

}

**Output:**

**gedit prog1.c**

**cc prog1.c**

**touch emptysu**

**ls -li emptysu**

**2894909 -rw-rw-r-- 1 msrit msrit 0 Apr 19 15:18 emptysu //observe size 0**

**touch emptysu.c**

**ls -li emptysu.c**

**2892236 -rw-rw-r-- 1 msrit msrit 0 Apr 19 15:18 emptysu.c //observe size 0**

**./a.out**

**Now when we type ls :files emptysu and emptysu.c aren’t there**

**6)Program to demonstrate the creation of hard links and the various properties of hard links**

#include<stdio.h>

#include<fcntl.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

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

{

        if(argc==3)

        {

        printf("Hard linking %s and %s",argv[1],argv[2]);

        if(link(argv[1],argv[2])==0)

                printf("\nHard link created");

        else

                printf("\nLink not created");

        }

        else if(argc==4)

        {

        printf("Soft linking %s and %s",argv[1],argv[2]);

        if(symlink(argv[1],argv[2])==0)

                printf("\nSoft link created");

        else

                printf("\nLink not created");

        }

}

**To run:**

vi progname.c

cc progname.c

./a.out **progname.c** nameofhardlink

When above three commands are run,o/p is:

Hard linking .... and nameofhardlink

Hard link created

Now execute command ls -li **progname.c** nameofhardlink

o/p:

The first number in two lines denotes inode and they are the same and link count is 2

This demonstrates hard link

Now if we do: gedit nameofhardlink

nameofhardlink and **progname.c** have same contents

For soft link:

./a.out **progname.c** nameofsoftlink dummyarg

\*note:dummyarg is just passes because we have written a condition if four args is given ,then create symbolic link between progname.c and nameofsoftlink

To see if a soft link is created execute

ls -li **progname.c** nameofsoftlink

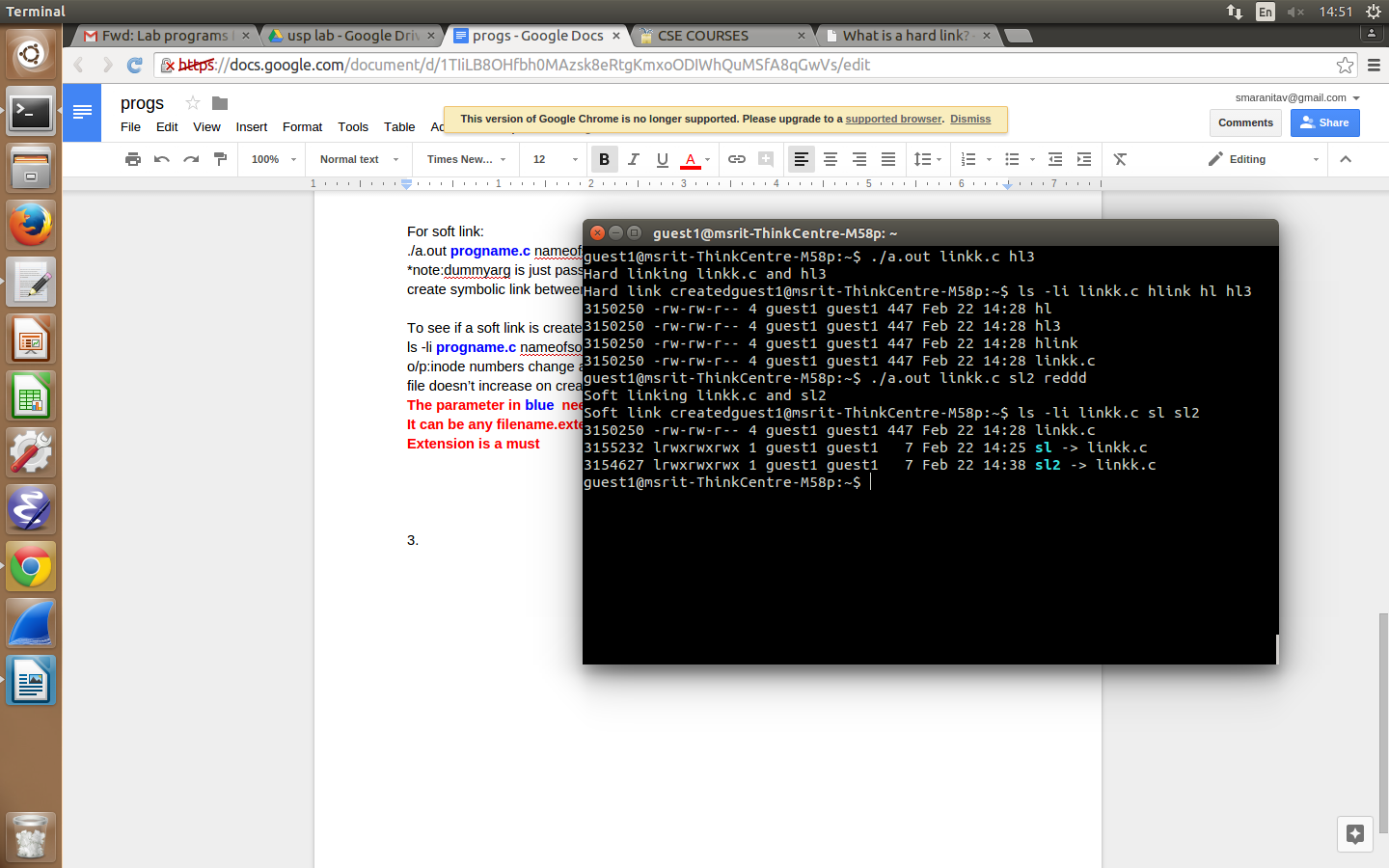
o/p:inode numbers change and link count for soft link remains 1 and link count for the original file doesn’t increase on creation of soft link

**The parameter in blue need not always be the progname.c.**

**It can be any filename.extension**

**Extension is a must**

**o/p:**

****

**7)Write a program to Copy access and modification time of a file to another file using utime function.**

#include <stdio.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#include <utime.h>

#include <time.h>

#include <fcntl.h>

int main(int argc,char\* argv[]) //copying ctime and mtime of argv[2] to argv[1]

{

int fd;

struct stat statbuf\_1;

struct stat statbuf\_2;

struct utimbuf times;

if(stat(argv[1],&statbuf\_1)<0)

printf("Error!\n");

if(stat(argv[2],&statbuf\_2)<0)

printf("Error!\n");

printf("Before Copying ...\n");

printf("Access Time %s\nModification Time%s\n",ctime(&statbuf\_1.st\_atime),ctime(&statbuf\_1.st\_mtime));

times.modtime = statbuf\_2.st\_mtime;

times.actime = statbuf\_2.st\_mtime;

if(utime(argv[1],&times)<0)

printf("Error copying time \n");

if(stat(argv[1],&statbuf\_1)<0)

printf("Error!\n");

printf("After Copying ...\n");

printf("Access Time %s\nModification Time%s\n",ctime(&statbuf\_1.st\_atime),ctime(&statbuf\_1.st\_mtime));

}

**Output:**

**gedit prog1.c**

**cc prog1.c -o prog1**

**./prog1 lab4.txt foo.html //give two filenames as command line args**

**Before Copying ...**

**Access Time Thu Apr 19 15:04:46 2018**

**Modification Time Thu Apr 19 15:04:46 2018**

**After Copying ...**

**Access Time Tue Feb 20 12:41:11 2018**

**Modification Time Tue Feb 20 12:41:11 2018**

**//here lab4.txt’s time is changed to foo.html’s time**

8)

#include <setjmp.h>  
#include<stdio.h>  
#include<stdlib.h>  
  
static void f1(int, int, int, int);  
static void f2(void);  
  
static jmp\_buf jmpbuffer;  
static int globval;  
  
int main(void)  
{  
 int autoval;  
 register int regival;   
 volatile int volaval;  
 static int statval;  
  
 globval = 1; autoval = 2; regival = 3; volaval = 4; statval = 5;  
 if (setjmp(jmpbuffer) != 0)   
 {  
 printf("after longjmp:\n");  
 printf("globval = %d, autoval = %d, regival = %d, volaval = %d, statval = %d\n", globval, autoval, regival, volaval, statval);  
 exit(0);  
 }  
/\*  
\* Change variables after setjmp, but before longjmp.  
\*/  
 globval = 95; autoval = 96; regival = 97; volaval = 98;  
 statval = 99;  
 f1(autoval, regival, volaval, statval); /\* never returns \*/  
 exit(0);  
}  
  
static void f1(int i, int j, int k, int l)  
{  
 printf("in f1():\n");  
 printf("globval = %d, autoval = %d, regival = %d, volaval = %d, statval = %d\n", globval, i, j, k, l);  
 globval=10000;  
 j=10000;  
 f2();  
}  
  
static void f2(void)  
{  
 longjmp(jmpbuffer, 1);  
}  
  
**output:**

**gedit prog1.c**

**cc prog1.c -o prog1**

**./prog1**

**//output:**

**in f1():**

**globval = 95, autoval = 96, regival = 97, volaval = 98, statval = 99**

**after longjmp:**

**globval = 10000, autoval = 96, regival = 97, volaval = 98, statval = 99**

12)

#include <stdio.h>  
#include <stdlib.h>  
#include <sys/stat.h>  
#include <sys/types.h>  
#include <unistd.h>  
  
void deamonize()  
{  
 pid\_t pid = fork();  
  
 if(pid < 0)  
 fprintf(stderr, "Error Forking\n");  
 else if(pid)  
 {  
 printf("PID of Child %d\n",pid);  
 exit(0); // kill the parent process , child is orphanded and runs in the bg  
 }  
  
 umask(0);  
 if(chdir("/") < 0)  
 printf("Error changing directory \n");  
 if(setsid() < 0) //make the child process the session leader  
 printf("Error creating session\n");  
   
 printf("Demon Created! \n");  
  
}  
int main()  
{  
 deamonize();  
 system("ps -axj");  
 return 0;  
}

**Output:**

**gedit prog1.c**

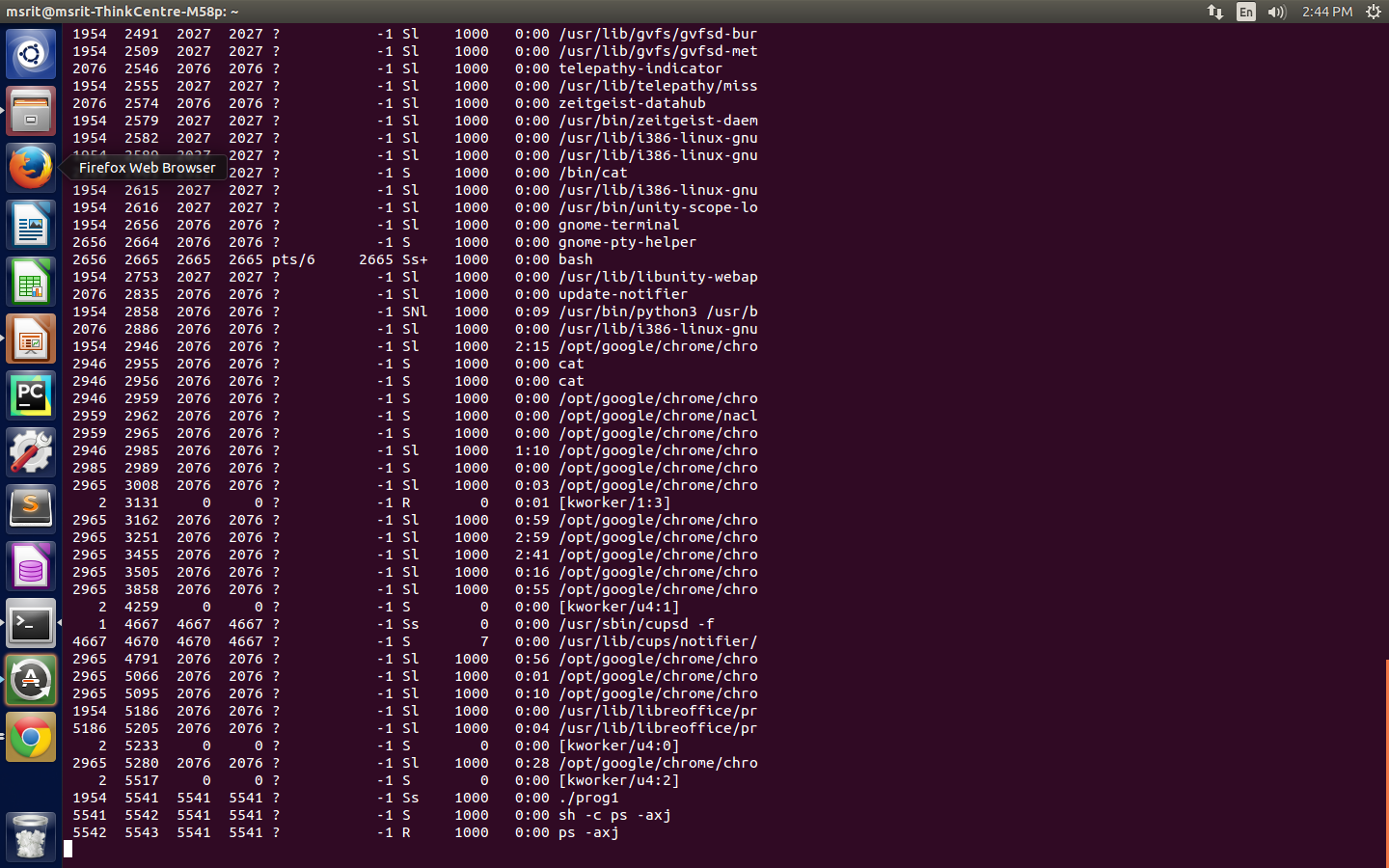
**cc prog1.c -o prog1**

**./prog1**

**//this is the output:**

**PID of Child 5541**

**Demon Created!**

****

**Observe the third line from the end : which demonstrates daemon process**

**PPID PID PGID SID TTY TPGID STAT UID TIME COMMAND**

**(these are the column names)**

**13)/\*Write a program using sigaction system call which calls a signal handler on SIGINT signal  
and then reset the default action of the SIGINT signal\*/**#include <stdio.h>  
#include <unistd.h>  
#include <signal.h>  
  
struct sigaction sig;  
  
void handler(int val)  
{  
 printf("Interrupt Received!\n");  
 sig.sa\_handler = SIG\_DFL;  
 sigaction(SIGINT,&sig,0);  
}  
  
int main()  
{  
 sig.sa\_flags = 0;  
 sigemptyset(&sig.sa\_mask);  
 sigaddset(&sig.sa\_mask,SIGINT); // listen only for SIGNIT  
 sig.sa\_handler = handler;  
  
 sigaction(SIGINT,&sig,0);  
  
 while(1)  
 {  
 printf("Progress is Happiness!\n");  
 sleep(1);  
 }  
}

**Output:**

**First time u type ctrl c ,interrupt received msg is seen**

**Next time u typre ctrl c the program stops**

**14)**

**/\*Write a program (use signal system call)  
i. which calls a signal handler on SIGINT signal and then reset the default action of the  
SIGINT signal  
ii. Which ignores SIGINT signal and then reset the default action of SIGINT signal\*/**  
#include <stdio.h>  
#include <unistd.h>  
#include <signal.h>  
  
void callback()  
{  
 printf("Interrupt Received !\n");  
 (void)signal(SIGINT,SIG\_DFL);  
}  
int main()  
{  
 int ch,i=0;  
 printf("Enter choice\n");  
 scanf("%d",&ch);  
  
 switch(ch)  
 {  
 case 1 : (void)signal(SIGINT,callback);  
 break;  
  
 case 2 : (void)signal(SIGINT,SIG\_IGN);  
 break;  
  
 }  
 while(1)  
 {  
 sleep(1);  
 printf("Press CTRL+C ...\n");  
 i++;  
 if(i == 10 && ch == 2)  
 (void) signal(SIGINT,SIG\_DFL);  
 }  
 return 0;  
}

**Output:if choice is 1 ,it works like prev program**

**If choice is 2 then before the tenth time if u press ctrl c it doesn’t interrupt**