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

**File type  
Number of links  
Time of last access  
Read, write, and execute permission**sriram@sriram-VirtualBox:~/Desktop/USP$ g++ fileinfo.cpp

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out ISE.txt

Regular file

No of links:1

Accessed time:Mon Nov 18 01:55:44 4461275

#include<stdio.h>

#include<sys/stat.h>

#include<limits.h>

#include<iostream>

#include<unistd.h>

#include<sys/types.h>

#include<time.h>

using namespace std;

void Read\_write(struct stat s){

if(s.st\_mode&R\_OK)

cout<<"Read"<<endl;

if(s.st\_mode&W\_OK){

cout<<"Write"<<endl;

}

if(s.st\_mode&X\_OK){

cout<<"Execute"<<endl;

}

}

void gettype(struct stat s){

if(S\_ISREG(s.st\_mode)){

cout<<"Regular file"<<endl;

}

else{

cout<<"Non regular file"<<endl;

}

}

void getlink(struct stat s){

cout<<"No of links:"<<s.st\_nlink<<endl;

}

void lasttime(struct stat attr) {

time\_t t=time\_t(&attr.st\_atim);

struct tm \*lt;

char buf[200];

localtime\_r(&t,lt);

strftime(buf,sizeof(buf),"%c",lt);

cout<<"Accessed time:"<<buf;

}

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

struct stat s;

for(int i=1;i<argc;i++){

if(stat(argv[i],&s)==-1){

perror("Error");

exit(0);

}

else{

gettype(s);

getlink(s);

Read\_write(s);

lasttime(s);

;

}

}

} **2.  Write a program in C that illustrates how to execute two commands concurrently with a command pipe.**

sriram@sriram-VirtualBox:~/Desktop/USP$ g++ pipecomd.cpp

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out

42 373 2352

sriram@sriram-VirtualBox:~/Desktop/USP$

#include<iostream>

#include<fcntl.h>

#include<stdlib.h>

#include<stdio.h>

#include<unistd.h>

using namespace std;

int main()

{

int pfd[2],p;

pipe(pfd);

p=fork();

if(p==0)

{

close(pfd[0]);

close(1);

dup(pfd[1]);

execlp("ls","ls","-l",(char\*)0);

}

else

{

close(pfd[1]);

close(0);

dup(pfd[0]);

execlp("wc","wc",(char\*)0);

}

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

sriram@sriram-VirtualBox:~/Desktop/USP$ cat ISE.txt

Dept of IDept of cssriram@sriram-VirtualBox:~/Desktop/USP$ g++ filecopier.cpp

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out ISE.txt copyISE.txt

5sriram@sriram-VirtualBox:~/Desktop/USP$ cat copyISE.txt

Dept of IDept of cssriram@sriram-VirtualBox:~/Desktop/USP$

#include<stdio.h>

#include<fcntl.h>

#include<stdlib.h>

#include<unistd.h>

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

int fd1,fd2;

char buf[50];

long int n;

fd1=open(argv[1],O\_RDONLY);

fd2=creat(argv[2],0700);

fd2=open(argv[2],O\_WRONLY|O\_TRUNC);

printf("%d",fd2);

if(fd1==-1 || fd2==1){

perror("errpr");

exit(1);

}

while((n=read(fd1,buf,50))>0){

if(write(fd2,buf,n)!=n){

perror("Writing issue");

exit(2);

}

if(n==-1){

perror("Reading issue");

exit(3);

}

}

close(fd2);

close(fd1);

exit(0);

} **4. Implement in C the following Unix commands using system calls  
a)Cat**

sriram@sriram-VirtualBox:~/Desktop/USP$ g++ commdcatfile.cpp

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out USP

error: No such file or directory

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out USP/ISE.txt

error: No such file or directory

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out ISE.txt

Dept of IDept of cssriram@sriram-VirtualBox:~/Desktop/USP$

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<fcntl.h>

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

int fd;

int i;

char buf[2];

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

if(fd<0){

perror("error");

exit(1);

}

while(read(fd,buf,1)>0){

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

}

}

**b)mv**

sriram@sriram-VirtualBox:~/Desktop/USP$ g++ commdmvfile.cpp

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out copyISE.txt CSE.txt

file1=copyISE.txt file2=CSE.txtsriram@sriram-VirtualBox:~/Desktop/USP$ cat copyISE.txt

cat: copyISE.txt: No such file or directory

sriram@sriram-VirtualBox:~/Desktop/USP$ cat CSE.txt

Dept of IDept of cssriram@sriram-VirtualBox:~/Desktop/USP$

#include<sys/types.h>

#include<sys/stat.h>

#include<stdio.h>

#include<fcntl.h>

#include<unistd.h>

using namespace std;

int 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);

}

**c)cp**

sriram@sriram-VirtualBox:~/Desktop/USP$ cat CSE.txt

Dept of IDept of cssriram@sriram-VirtualBox:~/Desktop/USP$ g++ commdcpfile.cpp

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out CSE.txt copyCSE.txt

5sriram@sriram-VirtualBox:~/Desktop/USP$ cat CSE.txt

Dept of IDept of cssriram@sriram-VirtualBox:~/Desktop/USP$ cat copyCSE.txt

Dept of IDept of cssriram@sriram-VirtualBox:~/Desktop/USP$

#include<stdio.h>

#include<fcntl.h>

#include<stdlib.h>

#include<unistd.h>

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

int fd1,fd2;

char buf[50];

long int n;

fd1=open(argv[1],O\_RDONLY);

fd2=creat(argv[2],0700);

fd2=open(argv[2],O\_WRONLY|O\_TRUNC);

printf("%d",fd2);

if(fd1==-1 || fd2==-1){

perror("errpr");

exit(1);

}

while((n=read(fd1,buf,50))>0){

if(write(fd2,buf,n)!=n){

perror("Writing issue");

exit(2);

}

if(n==-1){

perror("Reading issue");

exit(3);

}

}

close(fd2);

close(fd1);

exit(0);

}

**d)ln**

**#include<iostream>**

**#include<stdio.h>**

**#include<unistd.h>**

**using namespace std;**

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

if (argc!=3){

cerr<<"usage"<<argv[0]<<"<src\_file> <des\_file>"<<endl;

return 0;

}

if(link(argv[1],argv[2])==-1){

perror("link");

return 1;

}

return 0;

}

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

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out .

name:user\_details.cpp inode no:408006

name:locks.c inode no:394764

name:fifo\_files.cpp inode no:394726

name:signal2.cpp inode no:394658

name:locks2.c inode no:398507

name:commdcatfile.cpp inode no:393539

name:commdlnfile.cpp inode no:393540

name:pipecomd.cpp inode no:393527

name:file55 inode no:398887

name:inodenofinder.cpp inode no:393834

name:. inode no:443294

name:signal\_handle1.cpp inode no:394762

name:signal\_set2.cpp inode no:409753

#include<stdio.h>

#include<dirent.h>

#include<stdlib.h>

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

struct dirent \*de;

DIR \*dr;

if(dr==NULL){

perror("error");

exit(1);

}

dr=opendir(argv[1]);

while((de=readdir(dr))!=NULL){

printf("name:%s inode no:%ld\n",de->d\_name,de->d\_ino);

}

closedir(dr);

}

**6. Write a C Program that demonstrates redirection of standard output to a file .**

**EX:ls>f1.**

sriram@sriram-VirtualBox:~/Desktop/USP$ g++ outputredirect.cpp

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out redirect

sriram@sriram-VirtualBox:~/Desktop/USP$ cat redirect

a.out

commdcatfile.cpp

commdcpfile.cpp

commdlnfile.cpp

commdmvfile.cpp

copyCSE.txt

CSE.txt

demo.c

f1.txt

fifo\_files.cpp

file2.c

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int 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);

}

}

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

sriram@sriram-VirtualBox:~/Desktop/USP$ g++ childcreate.cpp

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out

PARENT PROCESS

sriram@sriram-VirtualBox:~/Desktop/USP$ CHILD PROCESS

#include<stdio.h>

#include<stdlib.h>

#include<fcntl.h>

#include<unistd.h>

int main(){

pid\_t pd;

pd=fork();

if(pd<0){

perror("error");

exit(1);

}

if(pd==0){

printf("CHILD PROCESS\n");

}

else{

printf("\nPARENT PROCESS\n");

}

}  
**8.Write a C program to create a Zombie process.**

sriram@sriram-VirtualBox:~/Desktop/USP$ g++ zombiecreate.cpp

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out

Child

sriram@sriram-VirtualBox:~/Desktop/USP$

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main(){

pid\_t child=fork();

if(child>0){

sleep(5);

printf("Child\n");

}

else{

exit(0);

}

} **9.Write a C program that illustrates how an orphan is created.**

sriram@sriram-VirtualBox:~/Desktop/USP$ g++ orphancreate.cpp

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out

Parent

sriram@sriram-VirtualBox:~/Desktop/USP$

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main(){

pid\_t child=fork();

if(child>0){

printf("Parent\n");

}

else if(child==0){

sleep(5);

printf("Child\n");

}

} **11)Write a C program that illustrates communication between two unrelated processes using a named pipe.**

sriram@sriram-VirtualBox:~/Desktop/USP$ g++ comm2.cpp

comm2.cpp: In function ‘int main()’:

comm2.cpp:10:13: warning: ISO C++ forbids converting a string constant to ‘char\*’ [-Wwrite-strings]

10 | char \*f="myfifo";

| ^~~~~~~~

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out

hello

USER1:hello

sriram@sriram-VirtualBox:~/Desktop/USP$ g++ comm1.cpp

comm1.cpp: In function ‘int main()’:

comm1.cpp:10:13: warning: ISO C++ forbids converting a string constant to ‘char\*’ [-Wwrite-strings]

10 | char \*f="myfifo";

| ^~~~~~~~

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out

hello

USER2:hello

#include<stdlib.h>

#include<sys/types.h>

#include<sys/fcntl.h>

#include<sys/stat.h>

#include<string.h>

#include<stdio.h>

#include<unistd.h>

int main(){

int fd;

char \*f="myfifo";

mkfifo(f,0666);

char arr1[80];

char arr2[80];

while(1){

fd=open(f,O\_WRONLY);

fgets(arr2,80,stdin);

write(fd,arr2,strlen(arr2)+1);

close(fd);

fd=open(f,O\_RDONLY);

read(fd,arr1,80);

printf("USER2:%s\n",arr1);

close(fd);

}

}

#include<stdlib.h>

#include<sys/types.h>

#include<sys/fcntl.h>

#include<sys/stat.h>

#include<string.h>

#include<stdio.h>

#include<unistd.h>

int main(){

int fd;

char \*f="myfifo";

mkfifo(f,0666);

char arr1[80];

char arr2[80];

while(1){

fd=open(f,O\_RDONLY);

read(fd,arr1,80);

printf("USER1:%s\n",arr1);

close(fd);

fd=open(f,O\_WRONLY);

fgets(arr2,80,stdin);

write(fd,arr2,strlen(arr2)+1);

close(fd);

}

}

**12.Write a C program that illustrates suspending and resuming processes using signals.**

sriram@sriram-VirtualBox:~/Desktop/USP$ g++ signalling.cpp

sriram@sriram-VirtualBox:~/Desktop/USP$ ./a.out

suspending

loop:0

loop:1

loop:2

loop:3

loop:4

loop:5

loop:6

loop:7

loop:8

loop:9

Main terminated

sriram@sriram-VirtualBox:~/Desktop/USP$

#include<stdlib.h>

#include<sys/types.h>

#include<sys/fcntl.h>

#include<sys/stat.h>

#include<string.h>

#include<stdio.h>

#include<unistd.h>

#include<signal.h>

using namespace std;

void sig\_al(int signo){

printf("Signal received\n");

}

int main(){

int s;

if(signal(SIGALRM,sig\_al)==SIG\_ERR){

perror("error");

}

else{

printf("suspending\n");

alarm(5);

for(int i=0;i<10;i++){

printf("loop:%d\n",i);

}

printf("Main terminated\n");

}

}

**13.Write a separate program using signal system call to catch the following signals.**

**o   SIGSEGV**

**o   SIGINT**

**o   SIGFPE**

**o   SIGALRM (use alarm system call)**

**o   SIGALRM (use setitimer system call)**

**o   SIGVTALRM (use setitimer system call)**

**o   SIGPROF (use setitimer system call)**

#include<iostream>

#include <stdio.h>

#include <signal.h>

using namespace std;

int main()

{

sigset\_t sigset;

sigemptyset(&sigset);

sigaddset(&sigset, SIGINT);

sigaddset(&sigset, SIGSEGV);

sigaddset(&sigset, SIGFPE);

sigaddset(&sigset, SIGALRM);

sigaddset(&sigset, SIGVTALRM);

sigaddset(&sigset, SIGPROF);

if (sigprocmask(SIG\_SETMASK, &sigset, 0) == -1)

perror("sigprocmask");

//initialize set

if (sigprocmask(0, NULL, &sigset) < 0)

perror("sigprocmask error");

if (sigismember(&sigset, SIGINT))  cout<<"SIGINT \n";

if (sigismember(&sigset, SIGSEGV))  cout<<"SIGSEGV \n";

if (sigismember(&sigset, SIGFPE))  cout<<"SIGFPE \n";

if (sigismember(&sigset, SIGVTALRM))  cout<<"SIGVTALRM \n";

if (sigismember(&sigset, SIGALRM))  cout<<"SIGALRM \n";

if (sigismember(&sigset, SIGPROF))  cout<<"SIGPROF \n";

}

SIGSEGV

#include <iostream>

#include <string.h>

#include <signal.h>

void handler(int nSignum, siginfo\_t\* si, void\* vcontext) {

  std::cout << "Segmentation fault" << std::endl;

  ucontext\_t\* context = (ucontext\_t\*)vcontext;

  context->uc\_mcontext.gregs[REG\_RIP]++;

}

int main() {

  std::cout << "Start" << std::endl;

  struct sigaction action;

  memset(&action, 0, sizeof(struct sigaction));

  action.sa\_flags = SA\_SIGINFO;

  action.sa\_sigaction = handler;

  sigaction(SIGSEGV, &action, NULL);

  int\* x = 0;

  int y = \*x;

  std::cout << "End" << std::endl;

  return 0;

}

SIGINT

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#define errExit(msg) do { perror(msg); exit(EXIT\_FAILURE); \

} while (0)

static void sigintHandler(int sig)

{

write(STDERR\_FILENO, "Caught SIGINT!\n", 15);

}

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

if (signal(SIGINT, sigintHandler) == SIG\_ERR)

errExit("signal SIGINT");

while (1) {

fprintf(stderr, "%d", 0);

sleep(3);

}

exit(EXIT\_SUCCESS);

}