**PROGRAM CODE:**

#include<stdio.h>

#include<unistd.h>

#include<sys/wait.h>

void main()

{

int pid,ppid,n,status,tid;

n=fork();

if(n!=0)

printf("\nThis is the parent");

else

printf("\nThis is the child");

pid=getpid();

ppid=getppid();

printf("\nPID: %d\n",pid);

printf("PARENT'S PID: %d\n",ppid);

tid=wait(&status);

if(tid!=-1)

{

printf("\nPID OF TERMINATED CHILD: %d\n",tid);

if(WIFEXITED(status))

printf("Child terminated normally\tExit status: %d\n",WEXITSTATUS(status));

else if(WIFSIGNALED(status))

printf("Child process was terminated by a signal\tNumber

of the signal: %d",WTERMSIG(status));

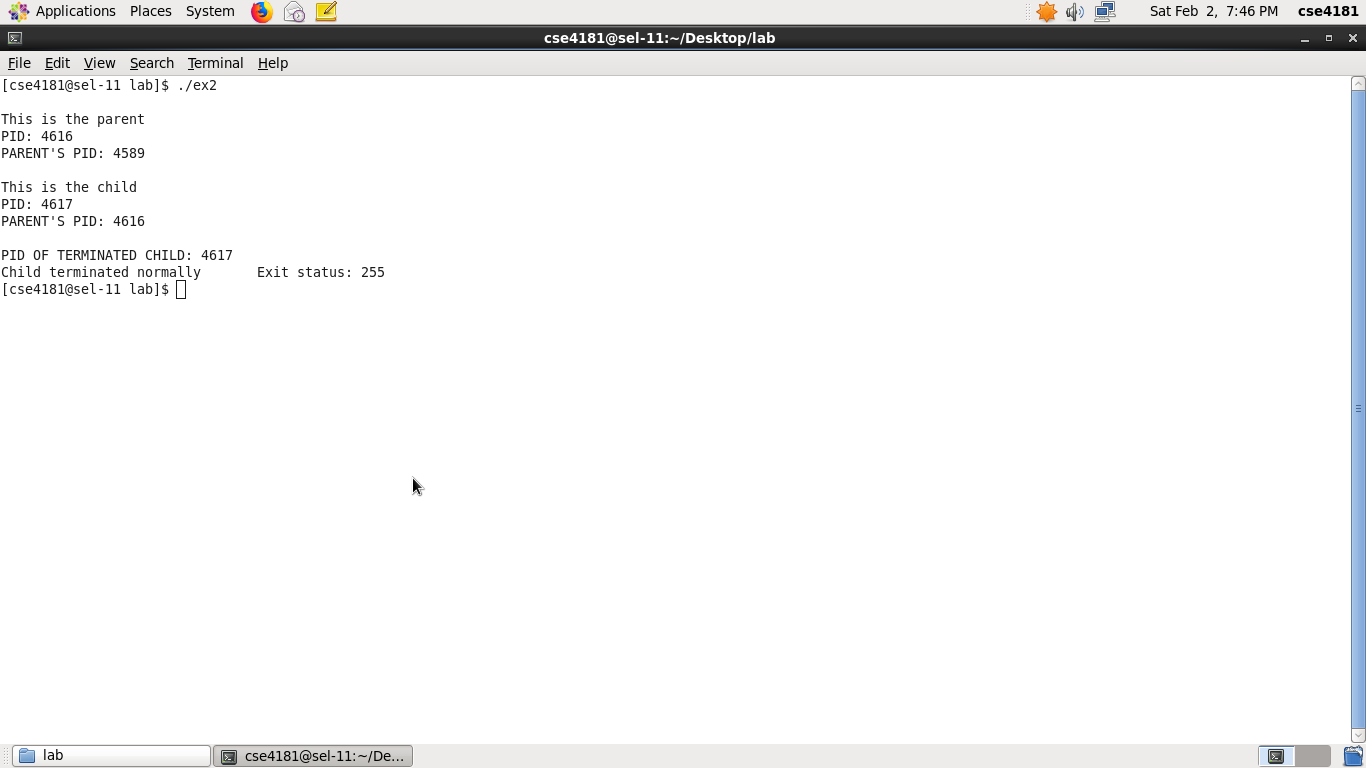
else if(WIFSTOPPED(status))

printf("Child process was stopped by delivery of a signal\tNumber of the stop signal: %d",WSTOPSIG(status));

}

}

**OUTPUT:**



**PROGRAM CODE:**

#include<stdio.h>

#include<unistd.h>

#include<sys/wait.h>

int fact(int n)

{

if(n<=1)

return 1;

else

return n\*fact(n-1);

}

void main()

{

int a,b,c,d,n,i;

a=fork();

if(a==0)

{ printf("\nThis is the process A.\nEnter a number: ");

scanf("%d",&n);

b=fork();

if(b==0)

{

printf("\nThis is the process B.\nThe sum of first %d natural nos is: %d\n",n,n\*(n+1)/2);

c=fork();

if(c==0)

{

printf("\nThis is the process C.\nThe %dth multiplication table is: \n",n);

for(i=1;i<=10;i++)

printf("%d \* %d = %d\n",n,i,n\*i);

d=fork();

if(d==0)

printf("\nThis is the process D.\nThe factorial of %d is: %d\n",n,fact(n));

} //end if c

} //end if b

} //end if a

wait(NULL);

}

**OUTPUT:**

G:\oslab\ss1\Screenshot-2.png

**PROGRAM CODE:**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<string.h>

#include <fcntl.h>

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

{

char \*source,

char \*dest;

char \*temp=(char \*) malloc(sizeof(char));

char \*str=(char \*) malloc(sizeof(char)\*400);

int fd1,fd2,n;

if(argc<3)

printf("\nNot enough arguments!");

else

{

source=argv[1];

dest=argv[2];

fd1=open(source,O\_RDONLY);

if(fd1<0)

{ printf("\nUnable to open %s file",source); exit(1); }

while(read(fd1,temp,sizeof(char))!=0)

strcat(str,temp);

close(fd1);

fd2=open(dest,O\_WRONLY | O\_CREAT);

if(fd2<0)

{ printf("\nUnable to open %s file",dest); exit(1); }

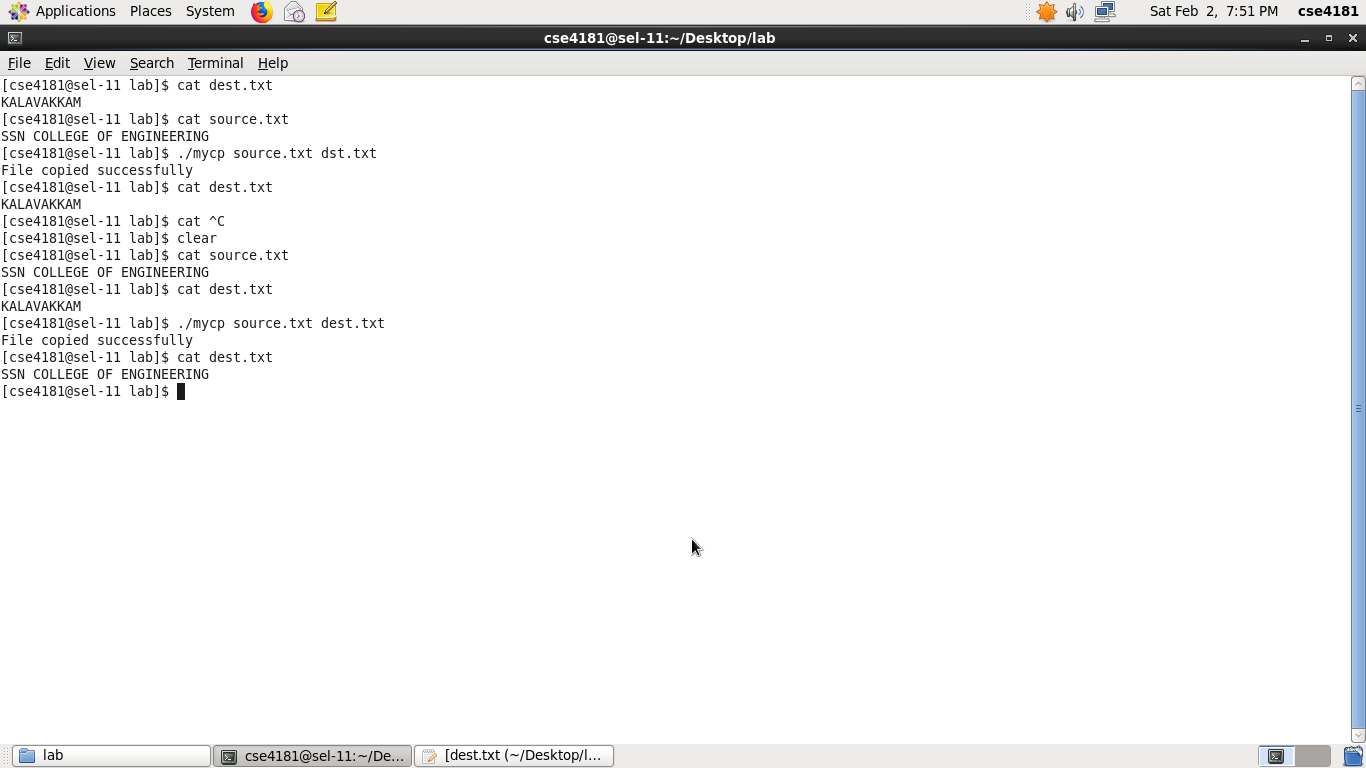
write(fd2,str,strlen(str)); close(fd2);

printf("File copied successfully\n");

}

}

**OUTPUT:**



**PROGRAM CODE:**

#include<stdio.h>

#include<unistd.h>

#include <fcntl.h>

#include<string.h>

#include<stdlib.h>

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

{

char \*file;

char \*pat;

int fd1;

char \*temp=(char \*) malloc(sizeof(char));

char \*str=(char \*) malloc(sizeof(char)\*200);

if(argc<3)

{ printf("\nNot enough arguments!"); exit(1); }

pat=argv[1];

file=argv[2];

fd1=open(file,O\_RDONLY);

if(fd1<0)

{ printf("\nUnable to open %s file",file); exit(1); }

while(read(fd1,temp,sizeof(char))!=0)

{ if(\*temp!='\n')

strcat(str,temp);

else

{ if(strstr(str,pat))

printf("%s\n",str);

strcpy(str, "");

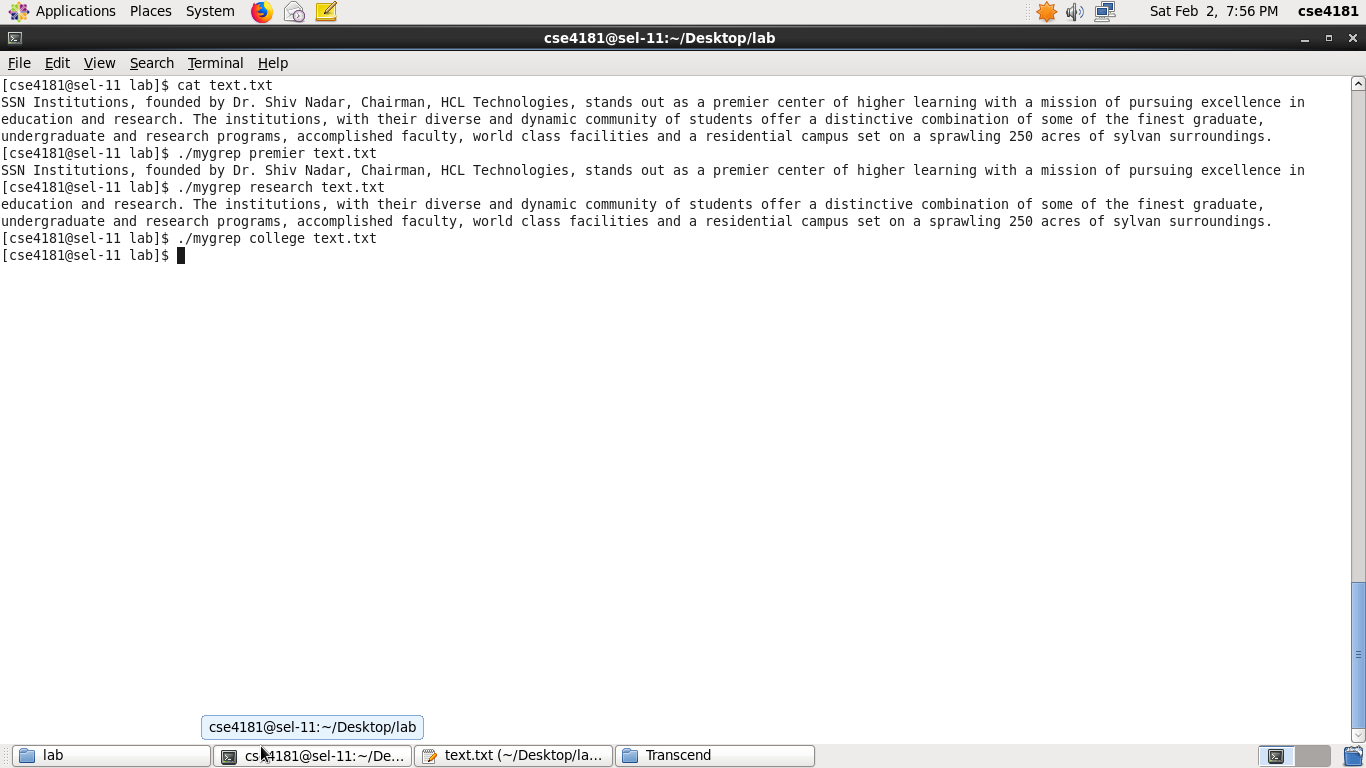
}

}

close(fd1);

}

**OUTPUT:**



**PROGRAM CODE:**

#include<unistd.h>

#include<stdio.h>

#include<dirent.h>

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

{

char \*dname;

struct dirent \* d;

if(argc<2)

printf("\nNot enough arguments!");

else

{

dname=argv[1];

DIR\* d1=opendir(dname);

if(d1==NULL)

printf("\nCant find directory");

else

while(d=readdir(d1))

printf("%s\n",d->d\_name);

close(d1);

}

}

//d1 represents a directory stream

**OUTPUT:**

