**MULTIPLE FILE COPYING**

**Lab no# 07**

****

**Fall 2021**

**CSE-302 System Programming Lab**

Submitted by: **Ashfaq Ahmad**

Registration No: **19PWCSE1795**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr: Ma’am Madeha sheer**

**February** 3, 2022

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Lab Objective(s):**

* Understand and implement the select function.

**Task # 01: Write a program that copies two files sequentially in a single process.**

**Source Code:**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <fcntl.h>

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

{

if(argc!=5) //four argument + this file itself =5

{

printf("Sorry! invalid Numbers of arguments:\n");

return -1;

}

int fd[4]; //4 file descripters

for(int i=0; i<argc-1; i+=2) //loop will iterate two time.

{

fd[i]=open(argv[i+1],O\_RDONLY); //1 and 3 will opened for read

if(fd[i]==-1)

{

printf("Sorry! %s can't opened successfully\n",argv[i+1]);

perror("Reason");

return -1;

}

fd[i+1]=open(argv[i+2],O\_WRONLY|O\_CREAT|O\_TRUNC,S\_IRWXU,S\_IRWXG,S\_IRWXO); //2 and 4 will opened for write.

if(fd[i+1]==-1)

{

printf("Sorry! %s can't opened successfully\n",argv[i+2]);

perror("Reason");

return -1;

}

}

char buffer[100];

int bytesread;

int byteswritten;

for(int i=0; i<argc-1; i+=2)

{

do

{

bytesread=read(fd[i],buffer,sizeof(buffer)); //read from 1 and 3

//printf("%d",bytesread);

if(bytesread==-1)

{

printf("Sorry! Data can't read from %s\n",argv[i+1]);

perror("Reason");

return -1;

}

byteswritten=write(fd[i+1],buffer,bytesread); //write on 2 and 4

//printf("%d",byteswritten);

if(byteswritten==-1)

{

printf("Sorry! Data can't written on %s\n",argv[i+2]);

perror("Reason");

return -1;

}

} while(bytesread!=0);

}

for (int i=0; i<argc-1; i++) //close 1 2 3 4 files

{

int x=close(fd[i]);

if(x==-1)

{

printf("Sorry! %s can't closed successfully\n",argv[i+1]);

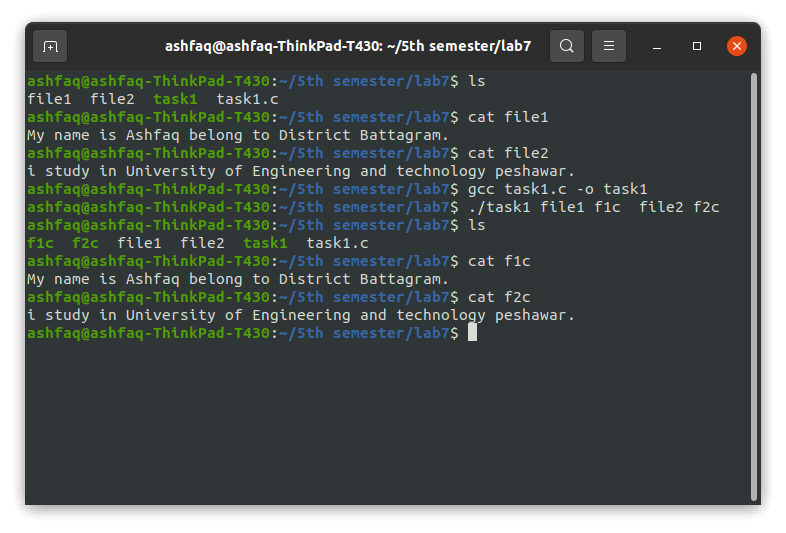
perror("Reason");

}

}

}

**Output:**



**Discussion:**

While sequentially copying files in a single process works fine in our case, it may lead to a problem where the process gets blocked even though one of the two files is ready for copying.

**Task # 02: Write a program that monitors two files by forking a child**

**Source Code:**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <sys/wait.h>

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

{

if(argc!=5)

{

printf("Sorry! Invalid Numbers of arguments\n");

return -1;

}

int x;

for(int i=1; i<argc; i+=2) //run two time

{

x=fork(); //syscall for child creation

if(x==-1)

{

printf("Sorry! child %d can't created succesfully\n",i+1);

perror("Reason");

return -1;

}

if(x==0) //fan condtion. no of child= no of fork syscall. here is 2 childs

//Each child will open read and write files then read and write and then close both file.

{

int fd1=open(argv[i],O\_RDONLY); //1 and 3 will opened for read

if(fd1==-1)

{

printf("Sorry! %s can't opened successfully\n",argv[i]);

perror("Reason");

return -1;

}

int fd2=open(argv[i+1],O\_WRONLY|O\_CREAT|O\_TRUNC,S\_IRWXU,S\_IRWXG,S\_IRWXO); //2 and 4 will opened for write.

if(fd2==-1)

{

printf("Sorry! %s can't opened successfully\n",argv[i+1]);

perror("Reason");

return -1;

}

char buffer[100];

int bytesread;

int byteswritten;

do

{

bytesread=read(fd1,buffer,sizeof(buffer)); //read from 1 and 3

if(bytesread==-1)

{

printf("Sorry! Data can't read from %s\n",argv[i]);

perror("Reason");

return -1;

}

byteswritten=write(fd2,buffer,bytesread); //write on 2 and 4

if(byteswritten==-1)

{

printf("Sorry! Data can't written on %s\n",argv[i+1]);

perror("Reason");

return -1;

}

} while(bytesread!=0);

int cfd1=close(fd1); //close read file

if(cfd1==-1)

{

printf("Sorry! %s can't closed successfully\n",argv[i+1]);

return -1;

}

int cfd2=close(fd2); //close write file.

if(cfd2==-1)

{

printf("Sorry! %s can't closed successfully\n",argv[i+2]);

return -1;

}

}

}

if(x>0) //parent process . return x is child id which is terminated.

for(int i=0; i<(argc-1)/2; i++) //will run two times.

{

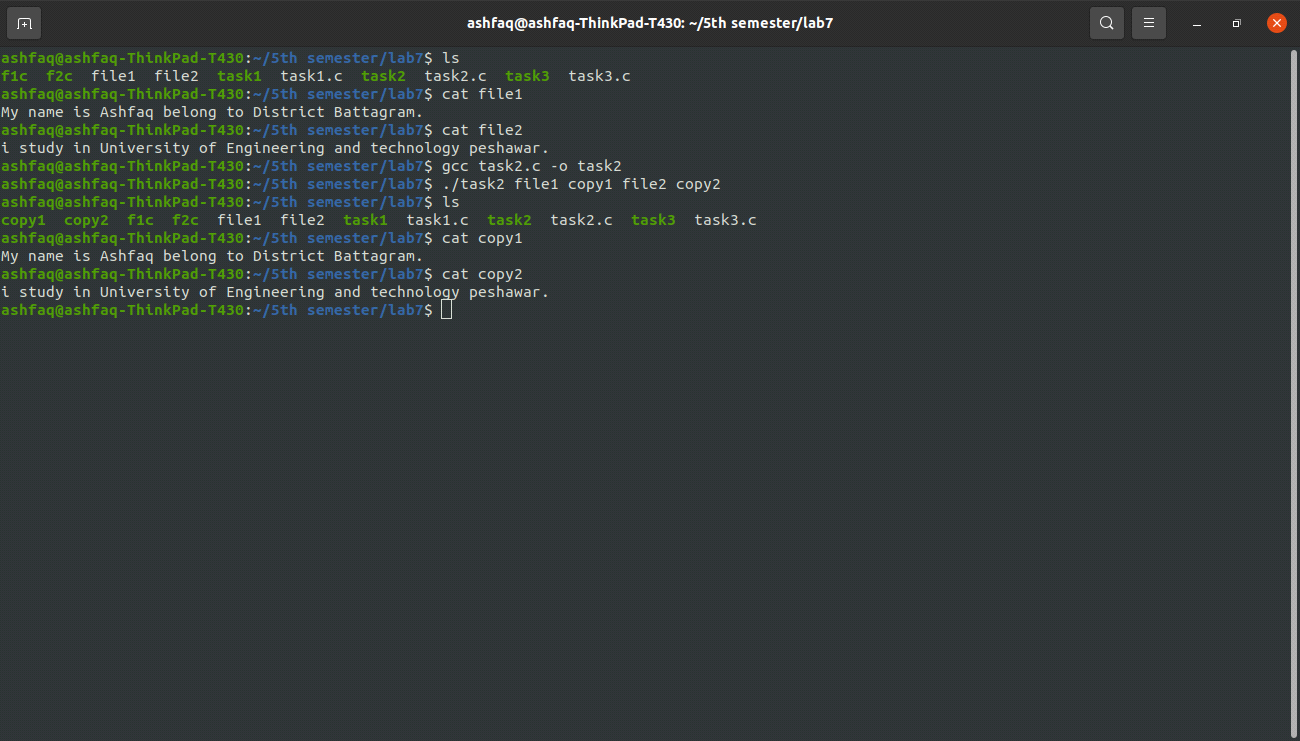
int y=wait(NULL); //as two childs created so two times wait function will call.

}

return 0;

}

**Output:**



**Discussion:**

Copying files in parallel using multiple child processes is a good solution to our problem but it also has its limitations. The parent and child processes have separate address spaces and it is difficult for them to communicate.

**Task # 03: Write a program that monitors two files for reading using 'select'.**

**Source Code:**

#include <stdio.h>

#include <unistd.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <sys/select.h>

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

{

if(argc<3) //argument must be two for writing while for reading i passed their names.

{

printf("Sorry! invalid numbers of arguments\n");

return -1;

}

int fd1=open("file1",O\_RDONLY); //open file1 for reading

if(fd1==-1)

{

printf("Sorry! file1 can't opened successfully\n");

perror("Reason");

return -1;

}

int fd2=open("file2",O\_RDONLY); //opened file2 for reading

if(fd2==-1)

{

printf("Sorry! file2 can't opened successfully\n");

perror("Reason");

return -1;

}

int fd3=open(argv[1],O\_WRONLY|O\_CREAT|O\_TRUNC,S\_IRWXU,S\_IRWXG,S\_IRWXO); //open file3 for writing

if(fd3==-1)

{

printf("Sorry! %s can't opened successfully\n",argv[1]);

perror("Reason");

return -1;

}

int fd4=open(argv[2],O\_WRONLY|O\_CREAT|O\_TRUNC,S\_IRWXU,S\_IRWXG,S\_IRWXO); //open file4 for writing

if(fd4==-1)

{

printf("Sorry! %s can't opened successfully\n",argv[2]);

perror("Reason");

return -1;

}

fd\_set readfds; //create a set of read files

FD\_ZERO(&readfds); //first ste contian garbig values we need to clear it using FD\_ZERO function.

FD\_SET(fd1,&readfds); //set fd1 for read

FD\_SET(fd2,&readfds); //set fd2 for read.

int nfds; //maximum no of files to be monitered.

if(fd1>fd2) //check which one is maximum

{

nfds=fd1;

}

else

nfds=fd2;

int z=select(nfds+1,&readfds,NULL,NULL,NULL); //select function

//we will add with fds because file descripter start from zero.

//it return -1 in error case. greater than zero in success case.

//greater than zero is the numbers of files ready for read, write and error check.

//it return zero in time argument case when no file is ready and time is over then return 0.

//if time argument is NULL and no device is ready then it will block untill it leat 1 member of the set become ready.

if(z==-1) //error

{

printf("Sorry! an error occured in select function\n");

perror("Reason");

return -1;

}

else if(z>0) //select function successed.

{

char buffer[100];

int bytesread;

int byteswritten;

if(FD\_ISSET(fd1,&readfds)) //FD\_ISSET check either fd1 is set to 1 or not if yes its mean fd1 ready for reading.

{

printf("file1 is ready for reading\n");

do

{

bytesread=read(fd1,buffer,sizeof(buffer)); //read from file1

if(bytesread==-1)

{

printf("Sorry! Data can't read from file1\n");

perror("Reason");

return -1;

}

byteswritten=write(fd3,buffer,bytesread); //write on file3

if(byteswritten==-1)

{

printf("Sorry! Data can't written on %s\n",argv[1]);

perror("Reason");

return -1;

}

} while(bytesread!=0);

}

else

printf("Sorry! file1 is not ready for reading\n");

if(FD\_ISSET(fd2,&readfds))

{

printf("file2 is ready for reading\n");

do

{

bytesread=read(fd2,buffer,sizeof(buffer)); //read from file2

if(bytesread==-1)

{

printf("Sorry! Data can't read from file1\n");

perror("Reason");

return -1;

}

byteswritten=write(fd4,buffer,bytesread); //write on file4

if(byteswritten==-1)

{

printf("Sorry! Data can't written on %s\n",argv[2]);

perror("Reason");

return -1;

}

}while(bytesread!=0);

}

else

printf("Sorry! file2 is not ready for reading\n");

}

int cfd1=close(fd1); //close file 1

if(cfd1==-1)

{

printf("Sorry! file1 can't closed successfully\n");

perror("Reason");

return-1;

}

int cfd2=close(fd2); //close file 2

if(cfd2==-1)

{

printf("Sorry! file2 can't closed successfully\n");

perror("Reason");

return-1;

}

int cfd3=close(fd3); //close file3

if(cfd3==-1)

{

printf("Sorry! %s can't closed successfully\n",argv[1]);

perror("Reason");

return-1;

}

int cfd4=close(fd4); //close file 4

if(cfd4==-1)

{

printf("Sorry! %s can't closed successfully\n",argv[2]);

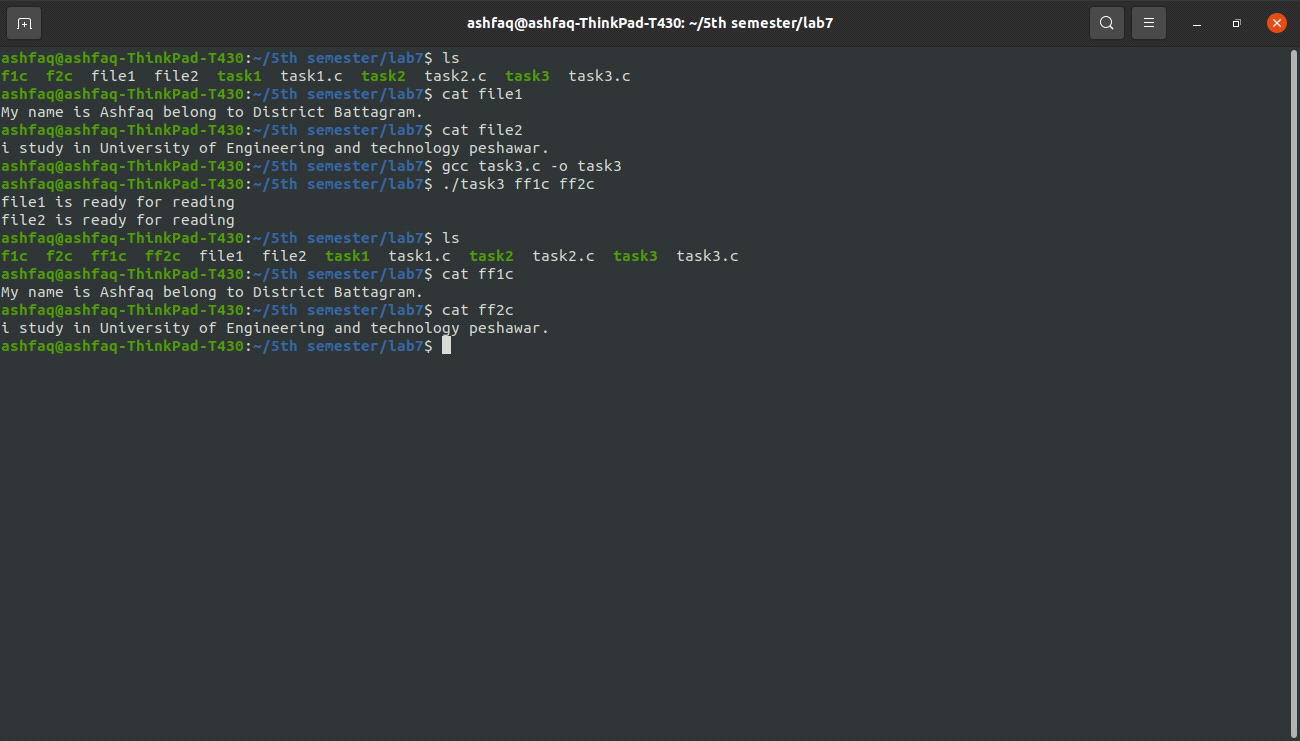
perror("Reason");

return-1;

}

}

**Output:**



**Discussion:**

The select function is a complete solution to our problem. It prevents the process from being blocked if a file is not ready and it can also communicate easily with the calling process using its return values.

**Task # 04: Write a program that monitors N files for reading using 'select'.**

**Source Code:**

#include <stdio.h>

#include <unistd.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <sys/select.h>

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

{

if(argc<2 || argc%2==0) //arguments must be even +this file=odd and argument must >2.

{

printf("Sorry! invalid Numbers of arguments\n");

return -1;

}

int fd[32]; //file descripter table.

for(int i=1; i<argc; i+=2) //run argc-1/2 time.

{

fd[i]=open(argv[i],O\_RDONLY); //odd will opened for read.

if(fd[i]==-1)

{

printf("Sorry! %s can't opened successfully\n",argv[i]);

perror("Reason");

return -1;

}

fd[i+1]=open(argv[i+1],O\_WRONLY|O\_CREAT|O\_TRUNC,S\_IRWXU,S\_IRWXG,S\_IRWXO);//even will opened for write.

if(fd[i+1]==-1)

{

printf("Sorry! %s can't opened successfully\n",argv[i+1]);

perror("Reason");

return -1;

}

}

fd\_set readset; //set of files for reading.

fd\_set writeset; //set of files for writing

FD\_ZERO(&readset); //set table to 0.

FD\_ZERO(&writeset);

for(int i=1; i<argc; i+=2) //run argc-1/2 time.

{

FD\_SET(fd[i],&readset); //set odd file for reading

FD\_SET(fd[i+1],&writeset); //set even files for writing

}

int nfds=fd[1];

for(int i=1; i<argc; i++) //run argc-1 time will find maximum file descripter.

{

if(fd[i]>nfds)

{

nfds=fd[i+1]; //maximun no of files for monitering.

}

}

int x=select(nfds+1,&readset,&writeset,NULL,NULL); //select function. here it will no return 0 b/c time argument==NULL.

if(x==-1) //select function failed

{

printf("Sorry! an error occured with select function\n");

perror("Reason");

return -1;

}

else if(x>0) //select function success

{

printf("The number of files ready are: %d\n",x);

char buffer[100];

int bytesread;

int byteswritten;

for(int i=1; i<argc; i+=2) //run argc-1/2 time.

{

if(FD\_ISSET(fd[i],&readset) && FD\_ISSET(fd[i+1],&writeset))

//if both fd[i] and fd[i+1] are set i,e ready

{

printf("%s is ready for reading\n",argv[i]);

printf("%s is ready for writing\n",argv[i+1]);

do

{

bytesread=read(fd[i],buffer,sizeof(buffer)); //read from fd[i]

if(bytesread==-1)

{

printf("Sorry! Data can't read from %s\n",argv[i]);

perror("Reason");

return -1;

}

byteswritten=write(fd[i+1],buffer,bytesread); //write on fd[i+1].

if(byteswritten==-1)

{

printf("Sorry! Data can't be written on %s\n",argv[i+1]);

perror("Reason");

return -1;

}

}while(bytesread!=0);

int c1=close(fd[i]); //close fd[i]

if (c1==-1)

{

printf("Sorry! %s can't closed successfully\n",argv[i]);

printf("Reason");

return -1;

}

int c2=close(fd[i+1]); //close fd[i+1]

if (c2==-1)

{

printf("Sorry! %s can't closed successfully\n",argv[i+1]);

printf("Reason");

return -1;

}

}

else //if both fd[i] and fd[i+1] or one of them is not ready.

{

if(FD\_ISSET(fd[i],&readset))

//if fd[i] is ready print fd[i+1] is not ready.

{

printf("%s is not read for writing\n",argv[i+1]);

}

else if(FD\_ISSET(fd[i+1],&writeset))

//if fd[i+1] is ready print fd[i] is not ready.

{

printf("%s is not read for reading\n",argv[i]);

}

else //or both are not ready.

{

printf("%s is not read for reading\n",argv[i]);

printf("%s is not read for writing\n",argv[i+1]);

}

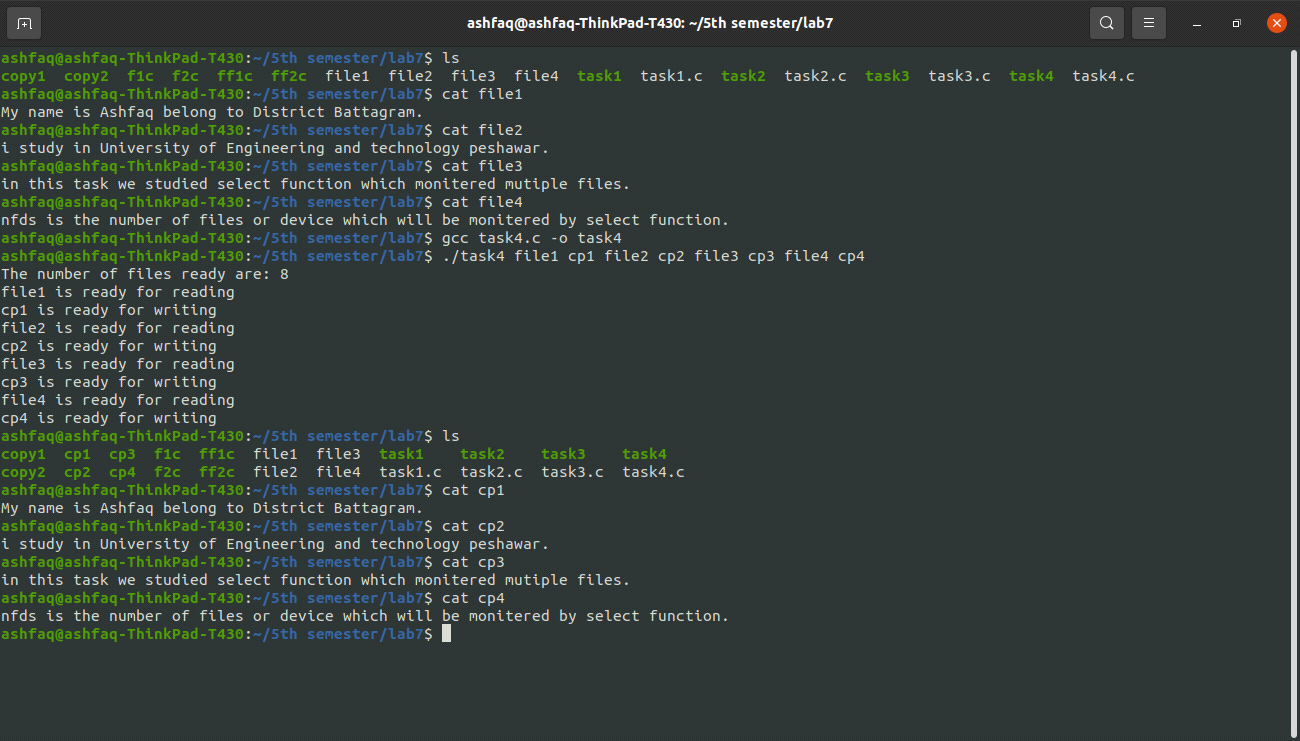
}

}

}

}

**Output:**



**Discussion:**

One of the main advantages of select function is that it is not limited to monitoring just two files, it can be used to monitor as many files as we want.

**Task # 05: Write a function that creates a delay of N seconds using select function. Pass N as an argument to the function.**

**Source Code:**

#include <stdio.h>

#include <unistd.h>

#include <sys/select.h>

#include <sys/time.h> //or time.h

#include <stdlib.h>

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

{

printf("select function is wait %ds and %dms at least one member of set is ready\n",atoi(argv[1]),atoi(argv[2]));

struct timeval waittime; //timeval is structure consist of two members. tv\_sec and tv\_usec.

waittime.tv\_sec=atoi(argv[1]);

waittime.tv\_usec=atoi(argv[2]);

int x=select(1,NULL,NULL,NULL,&waittime);

if(x==-1) //select function failed

{

printf("Sorry! an error occured with select function\n");

perror("Reason");

return -1;

}

if(x>0) //success

{

printf("Awakened\n");

}

if(x==0) //if time out and no device or file is ready.

//in this program no set is passed to select function so no device is ready.after wait time it will return 0.

{

printf("Time out\n");

}

}

**Output:**

