**UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA**

****

OPERATING SYSTEM PROJECT

**SUBJECT CODE: CS693**

**c program to implement the Process system calls & I/O system calls**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **GROUP MEMBER DETAILS:** | | | |
|  | **GROUP NO: 1** | | |  |
|  |  |  |  |  |
|  | **NAME** | |  | **ROLL NO** |
|  | Rajdeep Karmakar |  |  | 02 |
|  |  |  |  |
|  | Anirban Dey | |  | 82 |
|  |  |  |

INDEX

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **TOPIC** | **PAGE** |
| 1. | CERTIFICATE | 2 |
|  |  |  |
| 2. | ACKNOWLWDGEMENT | 3 |
|  |  |  |
| 3. | INTRODUCTION | 4 |
|  |  |  |
| 4. | SYSTEM CALLS | 5 |
|  |  |  |
| 5. | CODE | 7 |
|  |  |  |
| 6. | CONCLUSION | 13 |
|  |  |  |
| 7. | BIBLIOGRAPHY | 14 |
|  |  |  |

CERTIFICATE

This is to certify that **Rajdeep Karmakar and Anirban Dey** has

successfully completed the project **Implementation of Process System Calls and I/O System Calls using C** under the guidance of **Prof. Sukanya Roy and Prof Satyam Raha** during the year **2019** in partialfulfilment of **Operating System Practical** **Examination** conducted by **UEM, Kolkata.**

SIGNATURE

***ACKNOWLEDGEMENT***

We would like to express my special thanks of gratitude to our **Prof. Sukanya Roy and Prof Satyam Raha** as well as our HOD **Prof. Sukalyan Goswami** who gave us the golden opportunity to do this wonderful project on **Creating Implementation of process system calls and i/o system calls**

**Using C**, which also helped us indoing a lot of Research and we came to know about so many new things. We are really thankful to them.

***SIGNATURE***

***INTRODUCTION***

In this project we have **implemented Process System Calls and I/O System Calls using C.**

**SOFTWARE USED:** Dev C++

Dev-C++ is a free full-featured integrated development environment distributed under the GNU General Public License for programming in C and C++. It is written in Delphi. It is bundled with, and uses, the MinGW or TDM-GCC 64bit port of the GCC as its compiler.

**SOFTWARE REQUIREMENTS:**

**CPU:** Intel Pentium 4, 2.53 GHz or equivalent

**OS:** Microsoft Windows 7, 8.1, 10, Ubuntu Linux 16.04 LTS (Ubuntu12.04 or 14.04 LTS not supported anymore)

**RAM:** 2 GB

**Storage:** 1.4 GB of free disk space

**Display resolution:** 1024 x 768

***System Calls***

In computing, a **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to **interact with the operating system**. A computer program makes a system call when it makes a request to the operating system’s kernel. System call **provides** the services of the operating system to the user programs via Application Program Interface (API). It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

**Services Provided by System Calls:**

1. Process creation and management
2. Main memory management
3. File Access, Directory and File system management
4. Device handling(I/O)
5. Protection
6. Networking, etc.

**Types of System Calls:** There are 5 different categories of system calls –

* 1. **Process control:** end, abort, create, terminate, allocate and free memory.
  2. **File management:** create, open, close, delete, read file etc.
  3. Device management
  4. Information maintenance
  5. Communication

**Examples of Windows and Unix System Calls –**

|  |  |  |
| --- | --- | --- |
|  | **WINDOWS** | **UNIX** |
| Process Control | CreateProcess() ExitProcess() WaitForSingleObject() | fork() exit() wait() |
| File Manipulation | CreateFile() ReadFile() WriteFile() CloseHandle() | open() read() write() close() |
| Device Manipulation | SetConsoleMode() ReadConsole() WriteConsole() | ioctl() read() write() |
| Information Maintenance | GetCurrentProcessID() SetTimer() Sleep() | getpid() alarm() sleep() |
| Communication | CreatePipe() CreateFileMapping() MapViewOfFile() | pipe() shmget() mmap() |
|  |  |  |

Source Code

#include<stdio.h>

#include<fcntl.h>

#include<errno.h>

#include<string.h>

extern int errno;

void main()

{

int ch1;

char ch2,ch3;

char name[20],buffer[102];

int fd=0,val;

while(1)

{

printf("1. I/O Calls\n");

printf("2. Process Calls\n");

printf("3. Exit.\n");

printf("Enter your choice:");

scanf("%d",&ch1);

switch(ch1)

{

case 1:

{

while(1)

{

printf("I/O calls.\n");

if(fd>0)

printf("\*\*A file is open.\*\*\n\n");

printf("a. Create/Open file.\n");

printf("b. Close file.\n");

printf("c. Read file.\n");

printf("d. Write in file.\n");

printf("e. Exit.");

scanf("%c",&ch2);

switch(ch2)

{

case 'a':

{

if(fd>0)

printf("A file is already open. Close it first to create/open a new file.\n");

else

{ printf("Enter the name of the file to create/open (20 characters):"); 7

gets(name);

fd = open("foo.txt", O\_RDWR | O\_CREAT,0);

if(fd>0)

printf("%s File creation/opening successful.\n");

else

printf("Error Number % d\n", errno);

}

}

break;

case 'b':

{

if(fd>0)

{

if(close(fd)==0)

{

printf("File closed successfully.\n");

fd=0;

}

else

printf("File closing unsuccessful.\n");

}

else

printf("No file is currently open.\n");

}

break;

case 'c':

{

if(fd>0)

{

if(read(fd,buffer,100)<0)

printf("File reading error.\n");

else

{

printf("First 100 characters in the file is :\n");

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

}

}

else

printf("Open a file first to read from.\n");

}

break;

case 'd':

{

if(fd>0)

{

printf("Enter what you want to write to file (max 100 characters): ");

gets(buffer); 8

if(write(fd,buffer, strlen(buffer))<0)

printf("Writing to file error.\n");

else

printf("Writing to file is successful.\n");

}

else

printf("Open a file first to write to.\n");

}

break;

case 'e':

break;

default:

printf("Enter correct choice.\n");

}

if(ch2=='e')

break;

}

}

break;

case 2:

{

while(1)

{

printf("Process Calls.\n");

printf("i. fork()\n");

printf("j. exec()\n");

printf("k. exit.\n");

scanf("%c",&ch3);

switch(ch3)

{

case 'i':

{

printf("Executing fork system call.\n");

val=fork();

if(val==0)

{

printf("This string is printed from the child process.\nAnd the child process will terminate now.\n");

exit(1);

}

else

printf("This string is printed from the current/parent process.\nThe parent process will continue.\n");

}

break;

case 'j':

{ 9

printf("Executing exec system call will terminate this program.\nDo you want to continue(y/n):");

scanf("%c",&ch2);

if(ch2=='y'||ch2=='Y')

{

printf("Executing exec system call to execute ls function to print all the files in the current directory.\n\n");

execl("/bin/ls", "ls", 0);

}

else

printf("exec system call cancelled.\n");

}

break;

case 'k':

break;

default:

printf("Enter correct choice.\n");

}

if(ch3=='k')

break;

}

}

break;

case 3:

exit(1);

break;

default:

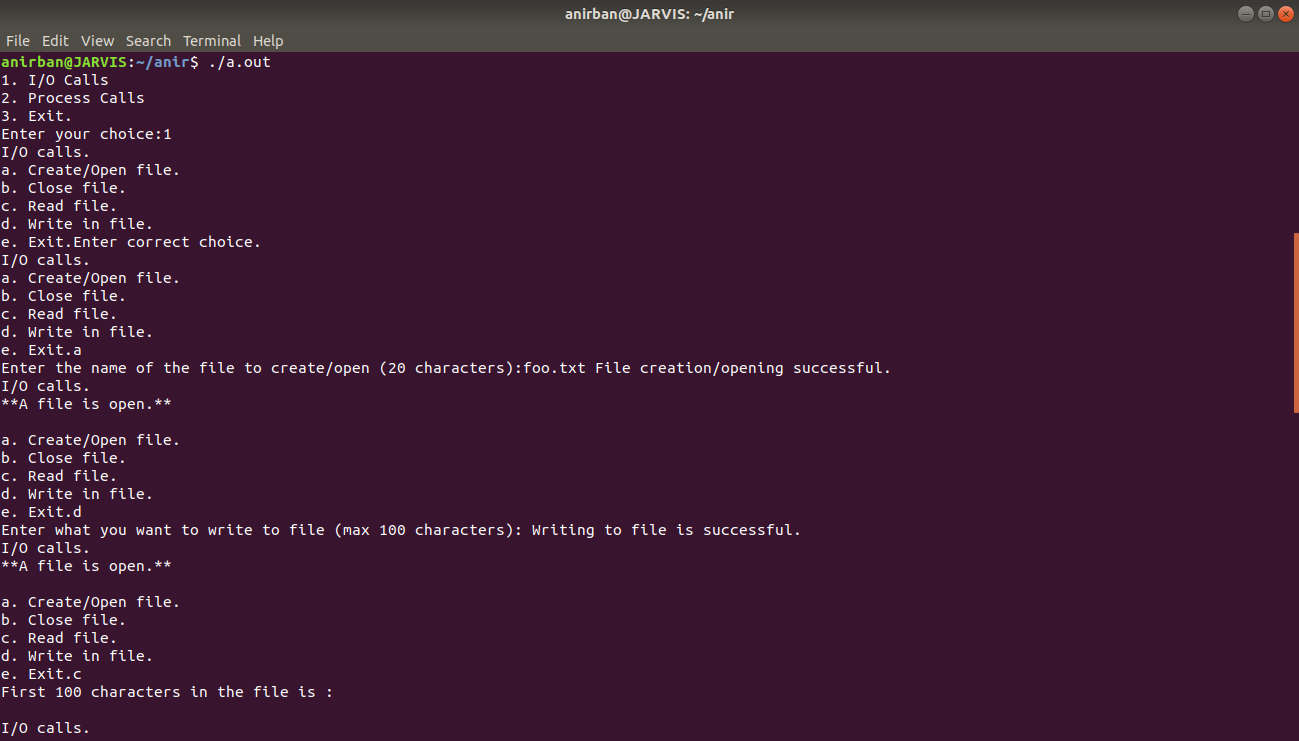
printf("Please enter a valid choice\n");

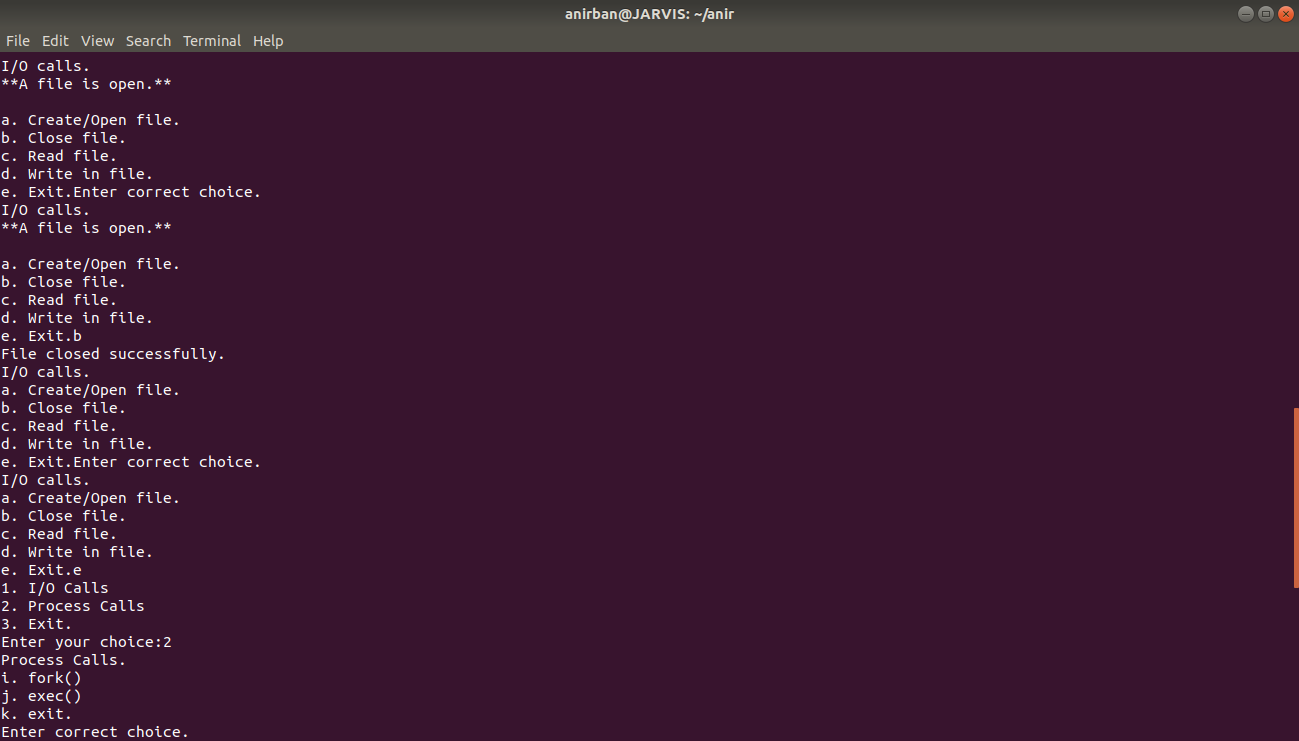
}

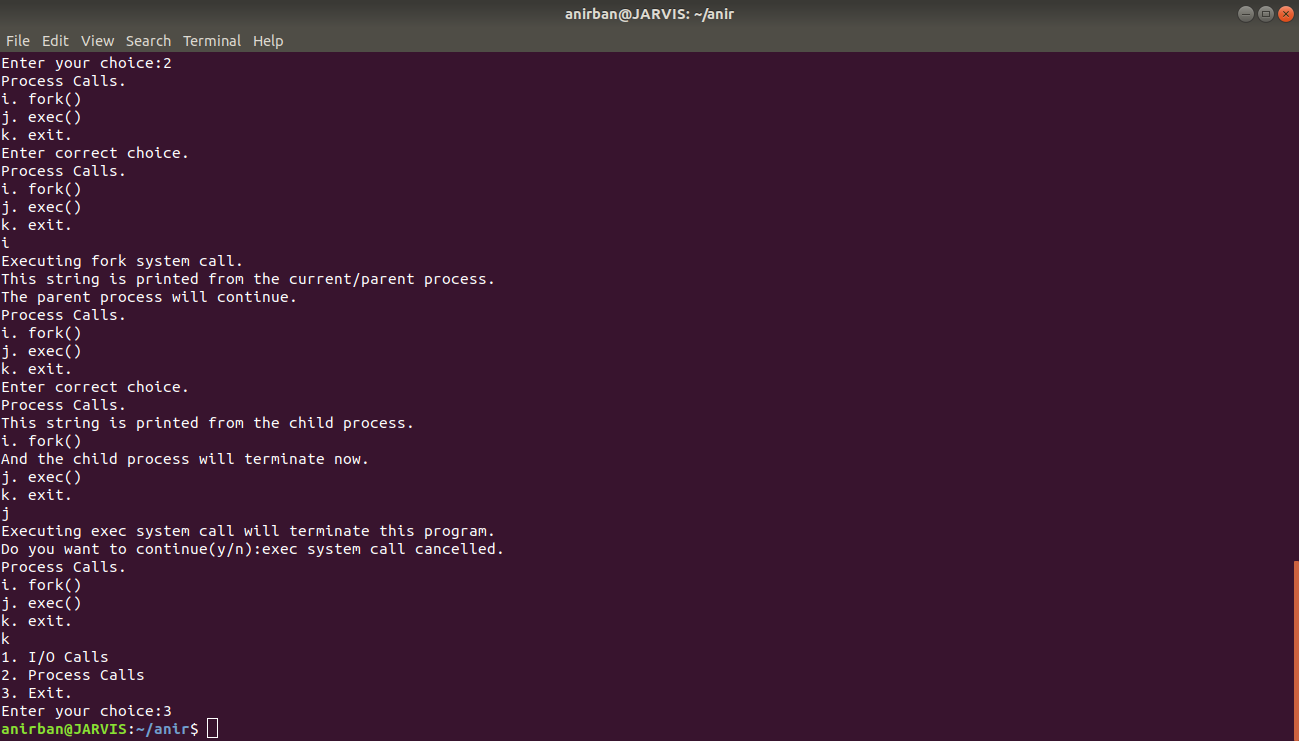
}

}

Output







Conclusion

Thus the Process System Calls and I/O System Calls were successfully implemented with C language.

All the functions gave correct results

Bibliography

System Calls and Other Theory

Geeks For Geeks

<https://www.geeksforgeeks.org/>

Google

<https://www.google.com/>

Reference for Code

Stackoverflow

<https://stackoverflow.com/>