Operating Systems Laboratory

B.Tech. 5th Semester

Name	**: K Srikanth**
Roll Number	**: 17ETCS002124**
Department	**: Computer Science and Engineering**

Faculty of Engineering & Technology

Ramaiah University of Applied Sciences

Ramaiah University of Applied Sciences

Private University Established in Karnataka State by Act No. 15 of 2013

Faculty	Engineering & Technology
Programme	B. Tech. in Computer Science and Engineering
Year/Semester	2nd Year / 5th Semester
Name of the Laboratory	Operating Systems Laboratory
Laboratory Code	19CSL308A

List of Experiments

1. Programs using process management system calls
2. Programs using file management system calls
3. Programs based on multithreaded programming
4. Programs for process scheduling algorithms
5. Solution to producer consumer problem using Mutex and Semaphore
6. Solutions to Dining philosopher problem using Semaphore
7. Programs for deadlock avoidance algorithm
8. Programs for memory management algorithms

Name: K Srikanth   Roll Number: 17ETCS002124

Index Sheet

No.	Lab Experiment	Performing the experiment (7)	Document (7)	Viva (6)	Total Marks (20)
1	Programs using process management system calls				
2	Programs using file management system calls				
3	Programs based on multithreaded programming				
4	Programs for process scheduling algorithms				
5	Solution to Producer Consumer Problem using Semaphore and Mutex				
6	Solution to Dining Philosopher problem using Semaphore				
7	Programs for deadlock avoidance algorithm				
8	Programs for memory management algorithms				
9	Lab Internal Test conducted along the lines of SEE and valued for 50 Marks and reduced for 20 Marks				
	Total Marks				

Component 1 = Lab Internal Marks =

Signature of the Staff In-charge

Name: K Srikanth Roll Number: 17ETCS002124

Laboratory 1

Title of the Laboratory Exercise: Programs using process management system calls

1. Introduction and Purpose of Experiment

   A system call is a programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. There are different types of system calls developed for various purposes. They are mainly classified as process management, file management, directory management. By solving the problems students will be able to apply process management system calls

   Aim and Objectives

   Aim

   - To develop programs involving process management system calls

   Objectives

   At the end of this lab, the student will be able to

   - Use different process management system calls
   - Apply different system calls wherever required
   - Create C programs using process management system calls

2. Experimental Procedure

   i. Analyse the problem statement

   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code

   iii. Implement the algorithm in C language

   iv. Compile the C program

   v. Test the implemented program

   vi. Document the Results

   vii. Analyse and discuss the outcomes of your experiment

3. Questions

   Implement the following operations in C

Create four different processes (with different process ID) and assign four different tasks (addition, subtraction, Multiplication, division) to each process. All processes should display the result along with its process ID and parent process ID.

4. Calculations/Computations/Algorithms

Flowchat

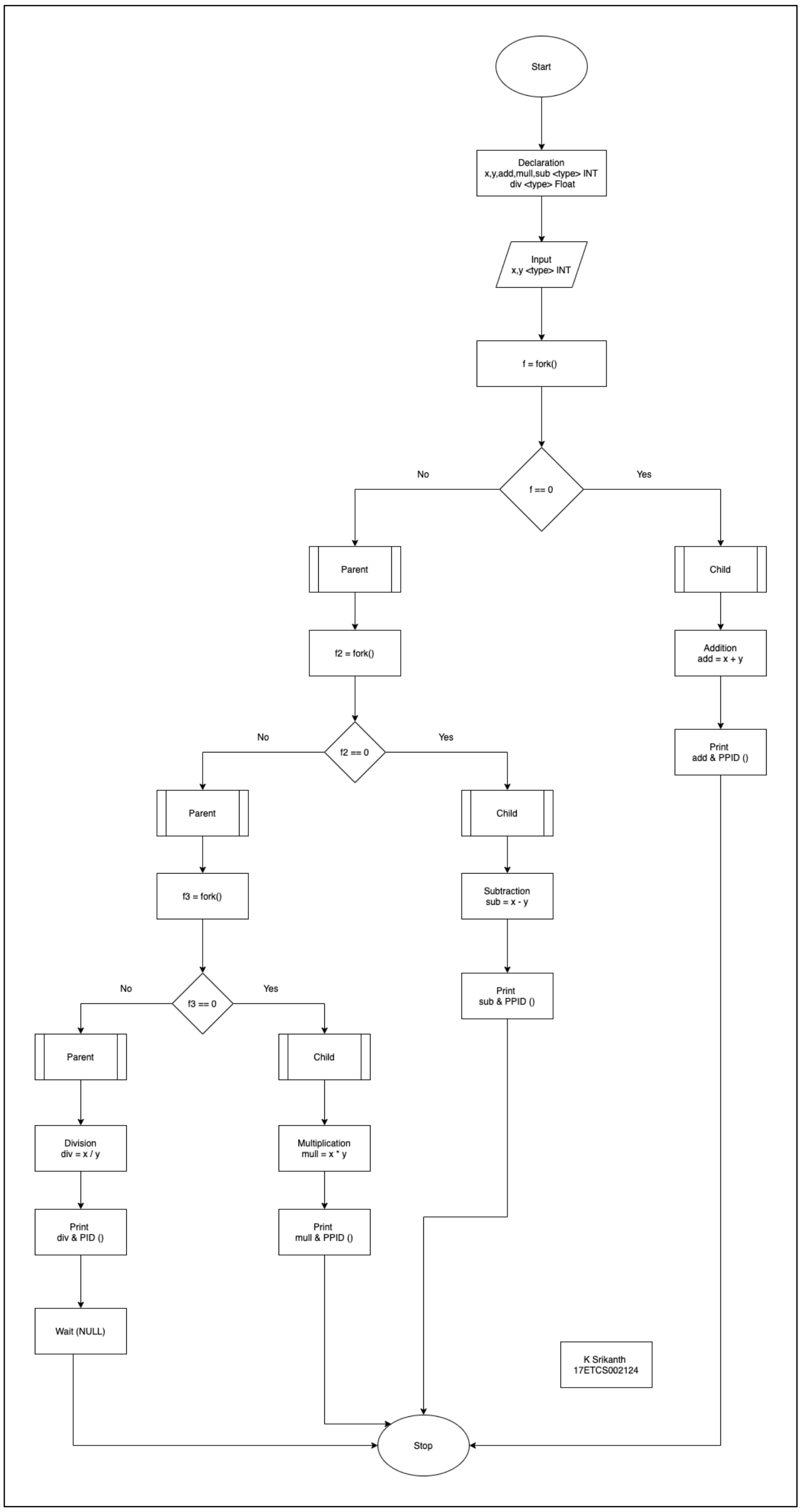
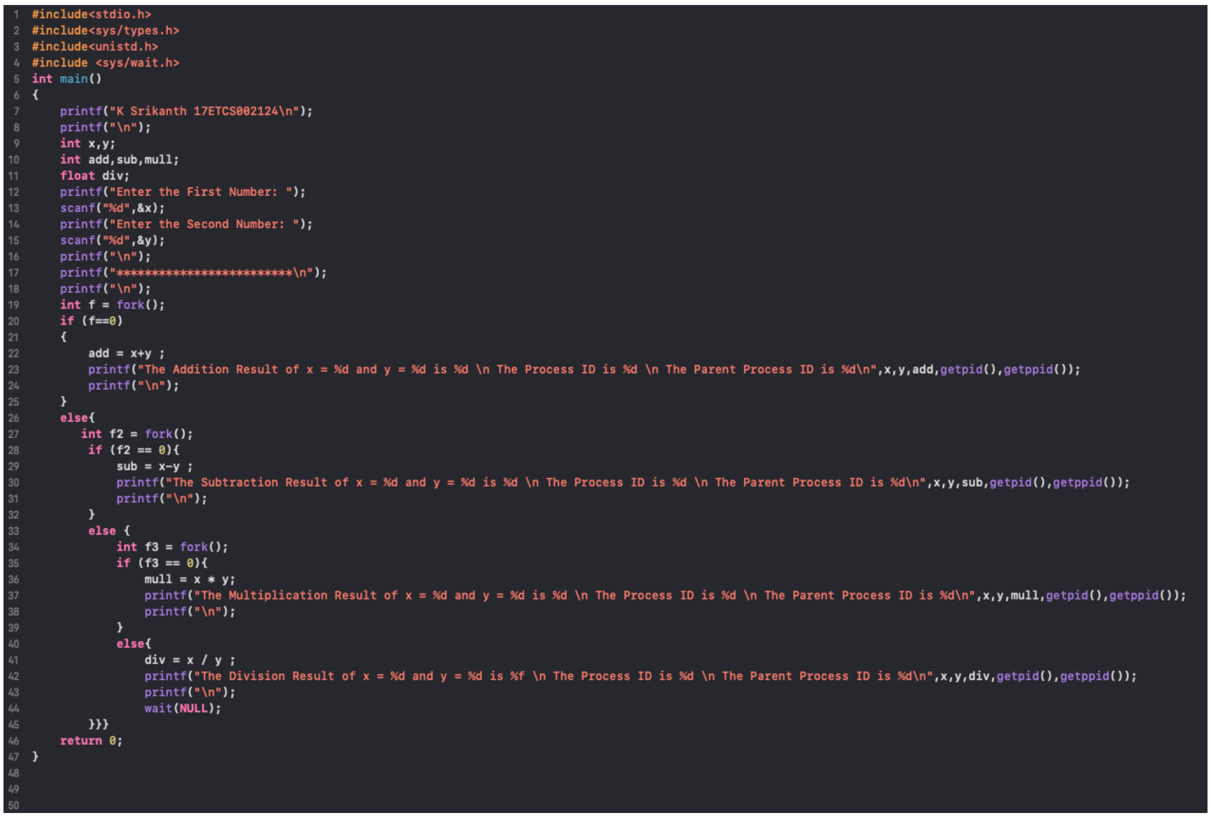


Figure 1.1 Flowchart of the given problem statement

Name: K Srikanth   Roll Number: 17ETCS002124

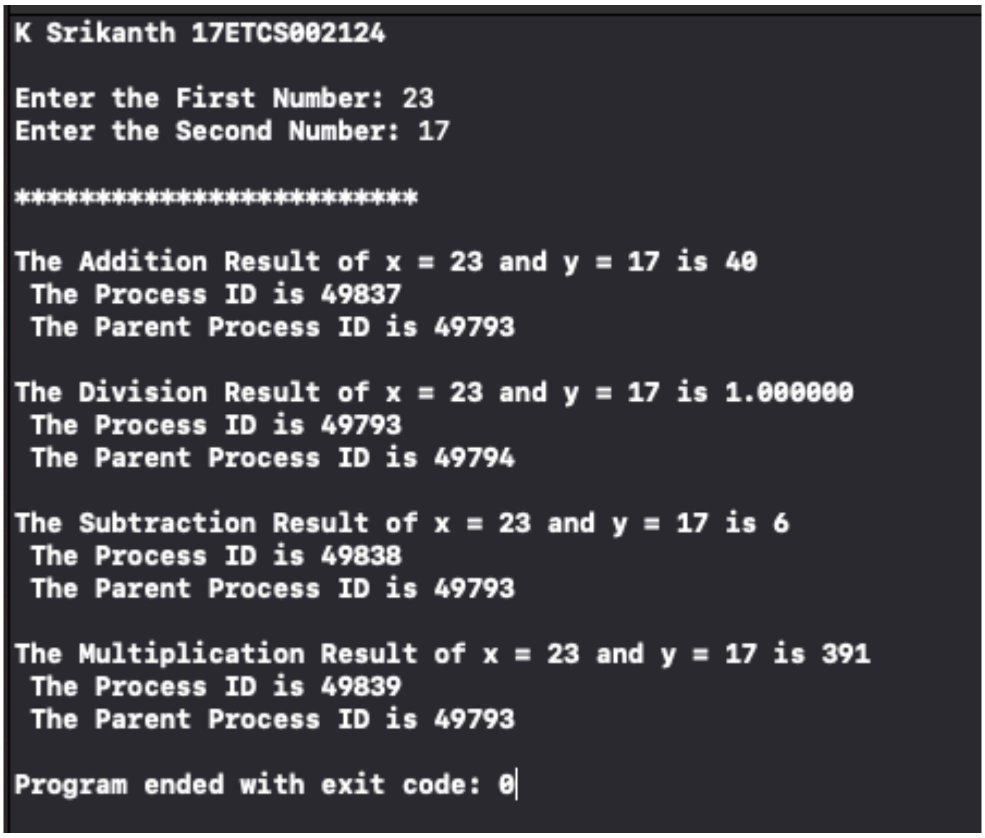
Code



```c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include <sys/wait.h>
int main()
{
    printf("K Srikanth 17ETCS002124\n");
    printf("\n");
    int x,y;
    int add,sub,mull;
    float div;
    printf("Enter the First Number: ");
    scanf("%d",&x);
    printf("Enter the Second Number: ");
    scanf("%d",&y);
    printf("\n");
    printf("*************************\n");
    printf("\n");
    int f = fork();
    if (f==0)
    {
        add = x+y ;
        printf("The Addition Result of x = %d and y = %d is %d \n The Process ID is %d \n The Parent Process ID is %d\n",x,y,add,getpid(),getppid());
        printf("\n");
    }
    else{
       int f2 = fork();
        if (f2 == 0){
            sub = x-y ;
            printf("The Subtraction Result of x = %d and y = %d is %d \n The Process ID is %d \n The Parent Process ID is %d\n",x,y,sub,getpid(),getppid());
            printf("\n");
        }
        else {
            int f3 = fork();
            if (f3 == 0){
                mull = x * y;
                printf("The Multiplication Result of x = %d and y = %d is %d \n The Process ID is %d \n The Parent Process ID is %d\n",x,y,mull,getpid(),getppid());
                printf("\n");
            }
            else{
                div = x / y ;
                printf("The Division Result of x = %d and y = %d is %f \n The Process ID is %d \n The Parent Process ID is %d\n",x,y,div,getpid(),getppid());
                printf("\n");
                wait(NULL);
        }}}
    return 0;
}
```

Figure 1.2 C Program for the given problem statement

5. Presentation of Results



```
K Srikanth 17ETCS002124

Enter the First Number: 23
Enter the Second Number: 17

*************************

The Addition Result of x = 23 and y = 17 is 40
 The Process ID is 49837
 The Parent Process ID is 49793

The Division Result of x = 23 and y = 17 is 1.000000
 The Process ID is 49793
 The Parent Process ID is 49794

The Subtraction Result of x = 23 and y = 17 is 6
 The Process ID is 49838
 The Parent Process ID is 49793

The Multiplication Result of x = 23 and y = 17 is 391
 The Process ID is 49839
 The Parent Process ID is 49793

Program ended with exit code: 0
```

Figure 1.3 C Program Output for the given problem statement

6. Analysis and Discussions

System calls is a way for a program to interact with operating system. When a program makes a system call at that time it makes a request to OS kernel. There are different types of system calls. **Fork()**, **getpid()** and **getppid()** are such process management requests.

Fork ( ) : It is a function which clones a flow to a child flow, so that both parent flow and child flow run parallel among different processors, thus enabling a program to utilising multiprocessing.

Each process has a unique reference known as process ID. And parent process ID is the ID of the process from which the current (child) process has been cloned from.

Using this system calls, we are supposed to find out the basic arithmetic operational results for two values, all running side by side. So first fork a to produce a child flow with a=0 (usually fork values of child flows would be Zero), but for parent process, A=non negative number.

If selection is arranged such a way child flow does addition and parent flow does division. Fork B inside child flow to further split flows, child one for difference and parent for product.

Process IDs and Parent Process IDs are there to check the relation between flows, like we see the addition process is the child of division.

Wait ( ) : This function is used when parent process is terminated before all the child processes are executed so if we use **Wait ( )** System call the parent process waits for the child processes to be finished and get terminated

7. Conclusions

A system call is a programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. **FORK, GETPID , GETPPID** and **WAIT** can be utilised to make program work effectively.

8. Comments

1. Limitations of Experiments

Program is simple illustration of system calls

2. Limitations of Results

When you try to print parent process ID of a parent ID its gives us a ID but we can't verify If the ID exists or not cause the parent process ID of the parent ID doesn't exist (in our case).

3. Learning happened

Importance of multiprocessors in system and role of Operating system and system calls .