RAJSHAHI UNIVERSITY OF ENGINEERING & TECHNOLOGY

Department of Computer Science & Engineering

LAB REPORT

Topic: Process Creation

Course No: CSE 3202

Course Name: Sessional Based on Operating Systems

Submitted By

Saifur Rahman
Roll No: 1703018
Section: A
CSE, RUET

Submitted To

Mohiuddin Ahmed
Lecturer
CSE, RUET

Date of Lab: May 16, 2022

Date of Submission: May 30, 2022

Contents

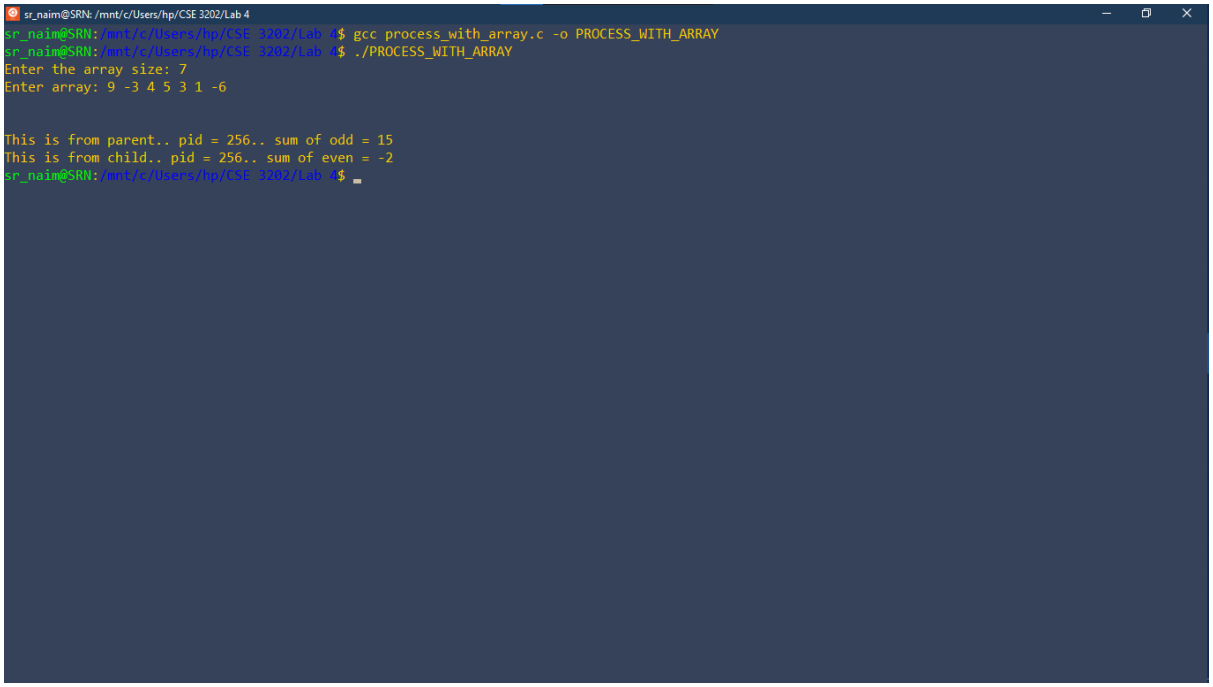
Program No. 1

Program Topic: A program that creates a process from another process, takes few integers as input, then sums up all the odd numbers of input if the parent process is running and sums up all the even numbers if child process is running.

CODE

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(){
        int n;
        printf("Enter the array size: ");
        scanf("%d", &n);
        int a[n];
        printf("Enter array: ");
        for(int i = 0; i < n; i++)
                scanf("%d", &a[i]);
        int sumEven = 0, sumOdd = 0;
        for(int i = 0; i < n; i++) {
                if(a[i] % 2 == 0)
                        sumEven += a[i];
                else
                        sumOdd += a[i];
        }
        int x = fork();
        if(x < 0)
                printf("Error Occurred\n");
        if(x > 0)
                printf("\nThis is from parent.. pid = %d.. sum of odd = %d\n",
getpid(), sumOdd);
        if(x == 0)
                printf("\nThis is from child.. pid = %d.. sum of even = %d\n",
getppid(), sumEven);
        return 0;
}
```

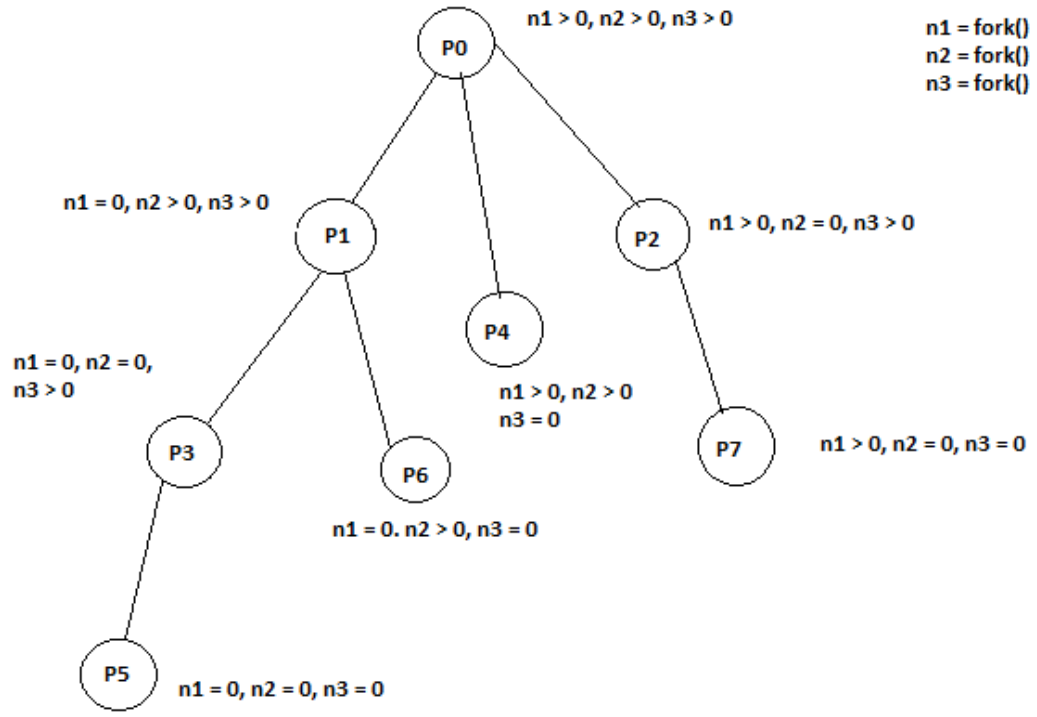
OUTPUT



Program No. 2

Program Topic: A program that creates a Process Tree with 3 'fork( )' instructions and determines the condition for accessing each process to do a specific operation.

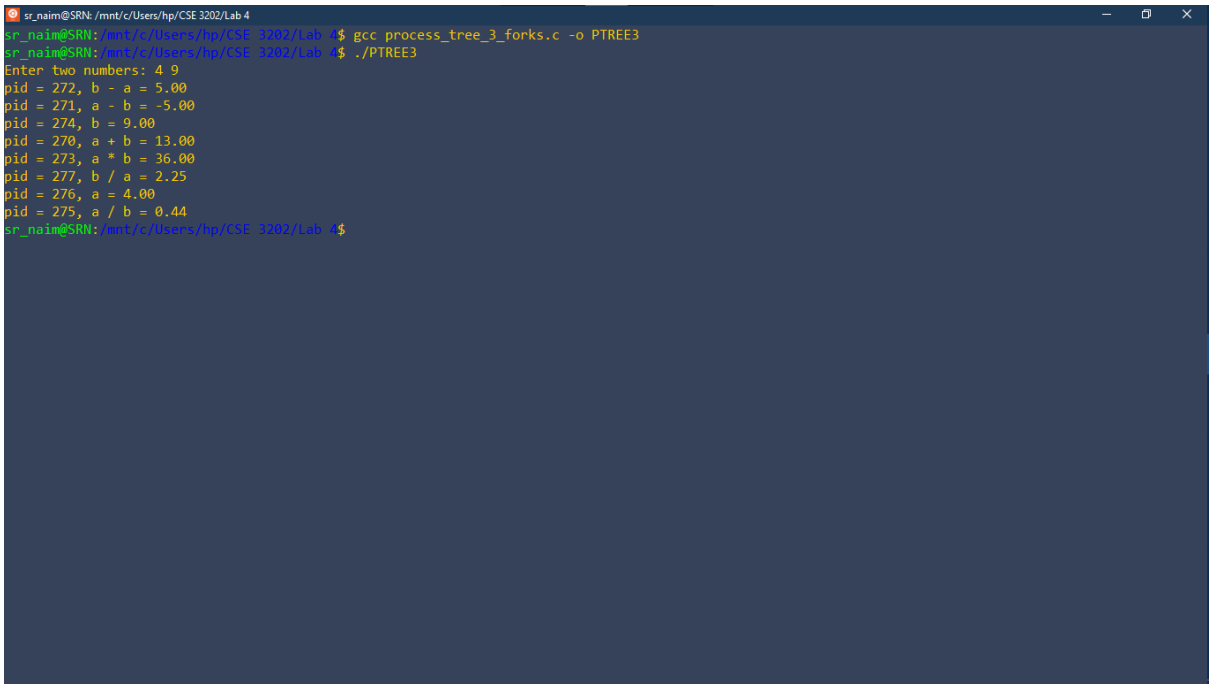
Process Tree



CODE

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main() {
        double a, b;
        printf("Enter two numbers: ");
        scanf("%lf %lf", &a, &b);
        int n1 = fork();
        int n2 = fork();
        int n3 = fork();
        if(n1 > 0 && n2 > 0 && n3 > 0)
                printf("pid = %d, a + b = %.2lf\n", getpid(), a+b);
        if(n1 == 0 && n2 > 0 && n3 > 0)
                printf("pid = %d, a - b = %.2lf\n", getpid(), a-b);
        if(n1 > 0 && n2 == 0 && n3 > 0)
                printf("pid = %d, b - a = %.2lf\n", getpid(), b-a);
        if(n1 == 0 && n2 == 0 && n3 > 0)
                printf("pid = %d, a * b = %.2lf\n", getpid(), a*b);
        if(n1 > 0 && n2 > 0 && n3 == 0)
                printf("pid = %d, a / b = %.2lf\n", getpid(), a/b);
        if(n1 == 0 && n2 == 0 && n3 == 0)
                printf("pid = %d, b / a = %.2lf\n", getpid(), b/a);
        if(n1 == 0 && n2 > 0 && n3 == 0)
                printf("pid = %d, a = %.2lf\n", getpid(), a);
        if(n1 > 0 && n2 == 0 && n3 == 0)
                printf("pid = %d, b = %.2lf\n", getpid(), b);
        return 0;
}
```

OUTPUT



Program No. 3

Program Topic: A program that calls another program via 'exec( )' instruction. The called program performs binary search.

CODE (Calling Binary Search.c)

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main() {
        char *args[]={"./SECOND",NULL};
        execv(args[0],args);
}
```

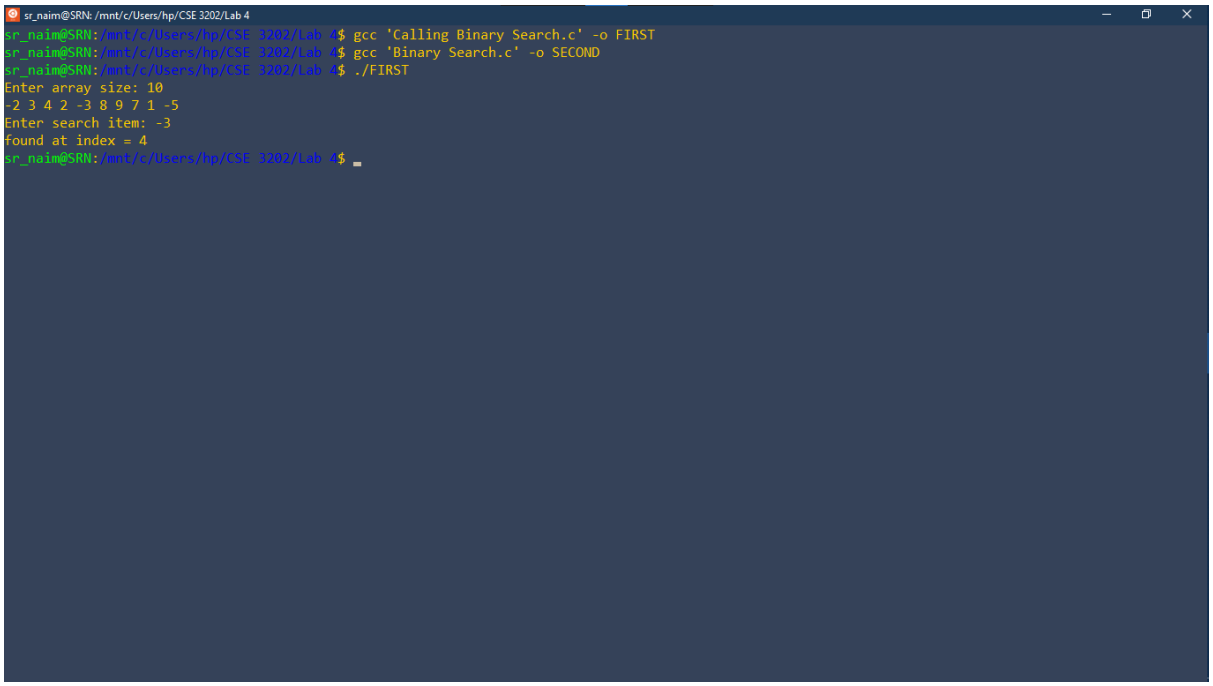
CODE (Binary Search.c)

```c
#include <stdio.h>
#include <unistd.h>
int main()
{
        int n;
        printf("Enter array size: ");
        scanf("%d", &n);
        int a[n];
        for(int i = 0; i < n; i++)
                scanf("%d", &a[i]);
        printf("Enter search item: ");
        int x;
        scanf("%d", &x);
        int l = 0;
        int h = n-1;
        while(l <= h) {
                int m = (l + h)/2;
                if(a[m] == x) {
                        printf("found at index = %d\n", m);
                        break;
                }
```

```
                else if(a[m] < x) {
                        l = m + 1;
                }
                else {
                        h = m - 1;
                }
        }

}
```

OUTPUT



Discussion:

- While creating a process and accessing it. The pid of parent and child processes may differ from one another.
  'getpid()' gives parent pid
  'getppid()' gives child pid

- While creating child process using fork(), only one recent fork number will change. Rest of the fork numbers will be inherited from parent process. Child will copy values from parent as 'fork()' creates same process.
  n = fork(); Here 'n' is the fork number

- While accessing another program from a program with exec(), one must be careful about the name given inside *args[]. The given name should be the name of called program's output file.

  In program 3, "SECOND" is the name given in *args[]. And the output file name of 'binary search.c' is also 'SECOND'.