Rajshahi University of Engineering & Technology

Course No.: CSE 3202
Course Title: Sessional Based on CSE 3201

Submitted To:

Mohiuddin Ahmed
Lecturer
Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology

Submitted By:

Name: Mondol Mridul Provakar
Roll: 1803062
Section: B
Department: Computer Science & Engineering

Date of submission: 12-12-2022

Problem Name: Implementation of process creation by using fork() function.

Theory: A process begins its life when it is created. A process goes through different states before it gets terminated. The first state that any process goes through is the creation of itself. Process creation happens through the use of fork() system call, which creates a new process(child process) by duplicating an existing one(parent process). The process that calls fork() is the parent, whereas the new process is the child.

Source Code:
Task 1:

Select mondol@DESKTOP-6QHQNRH: /mnt/g/WSL/Lab_4_Ext

GNU nano 4.8

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    int n1 = fork();
    if (n1 > 0)
    {
        printf("Root parent: %d\n", getpid());
        int n2 = fork();

        if (n2 == 0)
        {
            printf("Child 2: %d, parent id: %d\n", getpid(), getppid());
        }
    }
    else if (n1==0)
    {
        printf("Child 1: %d, parent id: %d\n", getpid(), getppid());
    }
    return 0;
}
```

Output:

```
mondol@DESKTOP-6QHQNRH:/mnt/g/WSL/Lab_4_Ext$ nano task1.c
mondol@DESKTOP-6QHQNRH:/mnt/g/WSL/Lab_4_Ext$ gcc task1.c -o task1
mondol@DESKTOP-6QHQNRH:/mnt/g/WSL/Lab_4_Ext$ ./task1
Root parent: 87
Child 1: 88, parent id: 87
Child 2: 89, parent id: 87
```

Task 2:

Select mondol@DESKTOP-6QHQNRH: /mnt/g/WSL/Lab_4_Ext

GNU nano 4.8

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{

    int n1 = fork();

    if (n1 > 0)
    {
        printf("Root parent: %d\n", getpid());

        int n2 = fork();
        if (n2 == 0)
        {
            printf("Child 2: pid %d, parent: %d\n", getpid(), getppid());

            int n3 = fork();
            if (n3 == 0)
            {
                printf("Child 5: pid %d, parent: %d\n", getpid(), getppid());
            }
            if (n3 > 0)
            {
                int n4 = fork();
                if (n4 == 0)
                {
                    printf("Child 6: pid %d, parent: %d\n", getpid(), getppid());
                }
            }
        }
        sleep(20);
    }
```

```
    if (n1 == 0)
    {
        printf("Child 1: pid %d, parent: %d\n", getpid(), getppid());

        int n5 = fork();

        if (n5 > 0)
        {
            int n6 = fork();
            if (n6 == 0)
            {
                printf("Child 4: pid %d, parent: %d\n", getpid(), getppid());
            }
        }
        if (n5 == 0)
        {
            printf("Child 3: pid %d, parent: %d\n", getpid(), getppid());
        }
        sleep(10);
    }
    return 0;
}
```

Output:

```
mondol@DESKTOP-6QHQNRH:/mnt/g/WSL/Lab_4_Ext$ nano task2.c
mondol@DESKTOP-6QHQNRH:/mnt/g/WSL/Lab_4_Ext$ gcc task2.c -o task2
mondol@DESKTOP-6QHQNRH:/mnt/g/WSL/Lab_4_Ext$ ./task2
Root parent: 188
Child 1: pid 189, parent: 188
Child 3: pid 191, parent: 189
Child 4: pid 192, parent: 189
Child 2: pid 190, parent: 188
Child 5: pid 193, parent: 190
Child 6: pid 194, parent: 190
```

Discussion:

The main problem was the speed of the implementation of the process. Sometimes the parent process worked so fast that the child process couldn't have the right parent id. That's why the 'sleep()' function was used. By using that function the problem was solved.