Operating System Practical

SET-1

Name : Akshat Kushwah

Uni. Roll no. : 20020570004

Practical 1: Implement SJF with specified arrival time and burst time. Compute waiting time, turnaround and completion time.

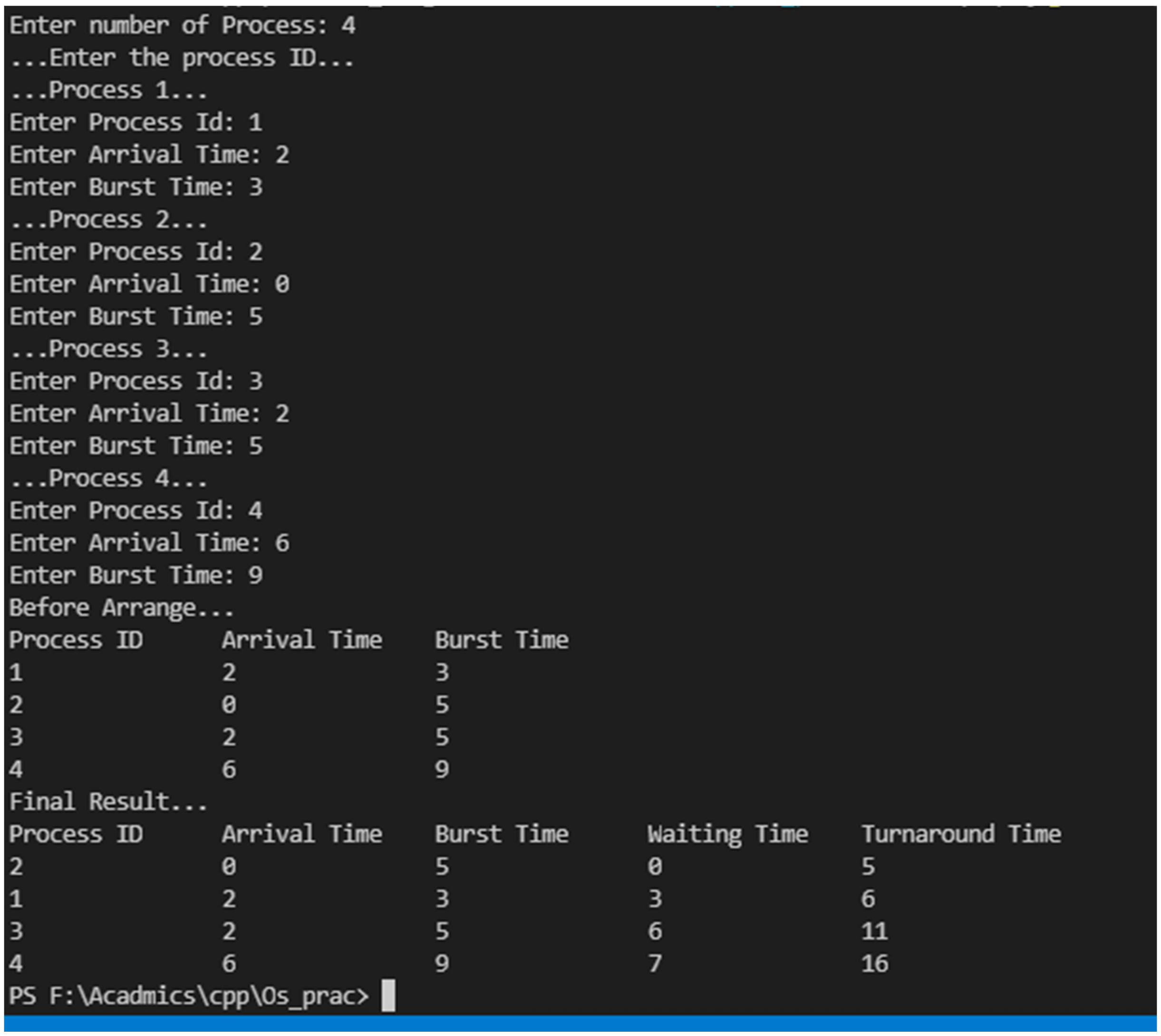
CODE:

```
// NAME                : Akshat Kushawah
// Colleg RollNo.      : 20201403
// University RollNO : 20020570004


// SET - 1
// OPERATING SYSTEM Practical -- 1

// C++ program to implement Shortest Job first with Arrival Time


// Description :
// This Algo is associated with NEXT CPU BRUST
// When CPU is avilable , it assigned to the process that has smallest next Cpu
brust
// For equal next CPU brust ,FCFS is used

#include <iostream>
using namespace std;
int mat[10][6];

void swap(int* a, int* b)
{
    int temp = *a;
```

```c
    *a = *b;
    *b = temp;
}

void arrangeArrival(int num, int mat[][6])
{
    for (int i = 0; i < num; i++) {
        for (int j = 0; j < num - i - 1; j++) {
            if (mat[j][1] > mat[j + 1][1]) {
                for (int k = 0; k < 5; k++) {
                    swap(mat[j][k], mat[j + 1][k]);
                }
            }
        }
    }
}

void completionTime(int num, int mat[][6])
{
    int temp, val;
    mat[0][3] = mat[0][1] + mat[0][2];
    mat[0][5] = mat[0][3] - mat[0][1];
    mat[0][4] = mat[0][5] - mat[0][2];

    for (int i = 1; i < num; i++) {
        temp = mat[i - 1][3];
        int low = mat[i][2];
        for (int j = i; j < num; j++) {
            if (temp >= mat[j][1] && low >= mat[j][2]) {
                low = mat[j][2];
                val = j;
            }
        }
        mat[val][3] = temp + mat[val][2];
        mat[val][5] = mat[val][3] - mat[val][1];
        mat[val][4] = mat[val][5] - mat[val][2];
        for (int k = 0; k < 6; k++) {
            swap(mat[val][k], mat[i][k]);
        }
    }
}

int main()
{
    int num, temp;
```

```cpp
    cout << "Enter number of Process: ";
    cin >> num;

    cout << "...Enter the process ID...\n";
    for (int i = 0; i < num; i++) {
        cout << "...Process " << i + 1 << "...\n";
        cout << "Enter Process Id: ";
        cin >> mat[i][0];
        cout << "Enter Arrival Time: ";
        cin >> mat[i][1];
        cout << "Enter Burst Time: ";
        cin >> mat[i][2];
    }

    cout << "Before Arrange...\n";
    cout << "Process ID\tArrival Time\tBurst Time\n";
    for (int i = 0; i < num; i++) {
        cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"
            << mat[i][2] << "\n";
    }

    arrangeArrival(num, mat);
    completionTime(num, mat);
    cout << "Final Result...\n";
    cout << "Process ID\tArrival Time\tBurst Time\tWaiting "
            "Time\tTurnaround Time\n";
    for (int i = 0; i < num; i++) {
        cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"
            << mat[i][2] << "\t\t" << mat[i][4] << "\t\t"
            << mat[i][5] << "\n";
    }
}
```

OUTPUT:



```
Enter number of Process: 4
...Enter the process ID...
...Process 1...
Enter Process Id: 1
Enter Arrival Time: 2
Enter Burst Time: 3
...Process 2...
Enter Process Id: 2
Enter Arrival Time: 0
Enter Burst Time: 5
...Process 3...
Enter Process Id: 3
Enter Arrival Time: 2
Enter Burst Time: 5
...Process 4...
Enter Process Id: 4
Enter Arrival Time: 6
Enter Burst Time: 9
Before Arrange...
Process ID      Arrival Time    Burst Time
1               2               3
2               0               5
3               2               5
4               6               9
Final Result...
Process ID      Arrival Time    Burst Time      Waiting Time    Turnaround Time
2               0               5               0               5
1               2               3               3               6
3               2               5               6               11
4               6               9               7               16
PS F:\Acadmics\cpp\Os_prac>
```

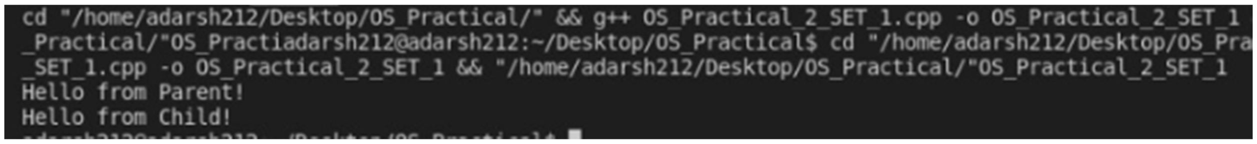
Practical 2: Write a program to demonstrate fork where parent and child run different codes and parent process should be executed first.

CODE:

```
// NAME               : Akshat Kushawah
// Colleg RollNo.     : 20201403
// University RollNO : 20020570004

// SET - 1
// OPERATING SYSTEM Practical -- 2

// C++ program to implement fork where parent and
// child run different codes and parent process should
// be executed first

// Description:
// Fork system call is used for creating a new process (child process),
// which runs concurrently with the process that makes the fork() call (parent
process).
// After a new child process is created, both processes will execute
// the next instruction following the fork() system call.
// It takes no parameters and returns an integer value.

It takes no parameters and returns an integer value.



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

void forkexample()
{

    // child process because return value zero

    if (fork() == 0)
```

```
        printf("Hello from Child!\n");



    // parent process because return value non-zero.

    else

        printf("Hello from Parent!\n");
}

int main()
{

    forkexample();

    return 0;
}
```

OUTPUT



```
cd "/home/adarsh212/Desktop/OS_Practical/" && g++ OS_Practical_2_SET_1.cpp -o OS_Practical_2_SET_1
_Practical/"OS_Practiadarsh212@adarsh212:~/Desktop/OS_Practical$ cd "/home/adarsh212/Desktop/OS_Pra
_SET_1.cpp -o OS_Practical_2_SET_1 && "/home/adarsh212/Desktop/OS_Practical/"OS_Practical_2_SET_1
Hello from Parent!
Hello from Child!
[illegible]
```
]