بسم الله الرحمن الرحيم

Group Members:

Zeeshan Ali (197)

Shayan Hassan Abbasi (167)

Operating System
Project Documentation

PROCESS SCHEDULING SIMULATOR

Table Of Content:

Process Scheduling Simulator:

- In an operating system, process scheduling refers to the mechanism that determines the order in which processes are executed by the CPU. The scheduler is responsible for managing the execution of multiple processes to maximize system efficiency and responsiveness.
- These simulators often provide command-line interface where users can define and simulate various scenarios, such as different scheduling algorithms.

Types Of Scheduling Algorithms:

1. **First Come First Serve (FCFS):**
   First Come First Serve (FCFS) is one of the simplest and most straightforward process scheduling algorithms used in operating systems. In FCFS scheduling, the process that arrives first is the one that gets executed first, and so on. It follows the principle of "first come, first served." This means that the process that enters the ready queue earliest will be the one to be selected for execution first.

2. **Shortest Job First (SJF):**
   Shortest Job First (SJF) is a process scheduling algorithm that selects the process with the shortest total burst time to execute first. The burst time is the amount of time a process requires to complete its execution. The idea behind SJF is to minimize the total processing time and reduce the waiting time for processes.

3. **Priority Scheduling Algorithm:**
   Priority Scheduling is a process scheduling algorithm in which each process is assigned a priority, and the process with the highest priority is selected for execution first. The priority can be based on various factors, such as the process's importance, resource requirements, or any other criteria defined by the system or the user. The process with the lowest priority value is typically the one that gets executed first.

4. Round Robin (RR):

Round Robin (RR) is a preemption-based process scheduling algorithm commonly used in operating systems. It is designed to provide fair and equal access to the CPU for all processes in the system. In Round Robin scheduling, each process is assigned a fixed time slice or quantum, and the CPU scheduler cyclically allocates the CPU to each process for its quantum, moving to the next process in the queue when the time slice expires.

Formulas:

1. Waiting Time: $WT[i] = WT[i-1] + BT[i-1]$
2. Turnaround Time: $TAT[i] = WT[i] + BT[i]$
3. Average Waiting Time: $AWT = \sum WT / n$
4. Average Turnaround Time: $ATAT = \sum TAT / n$

Code:

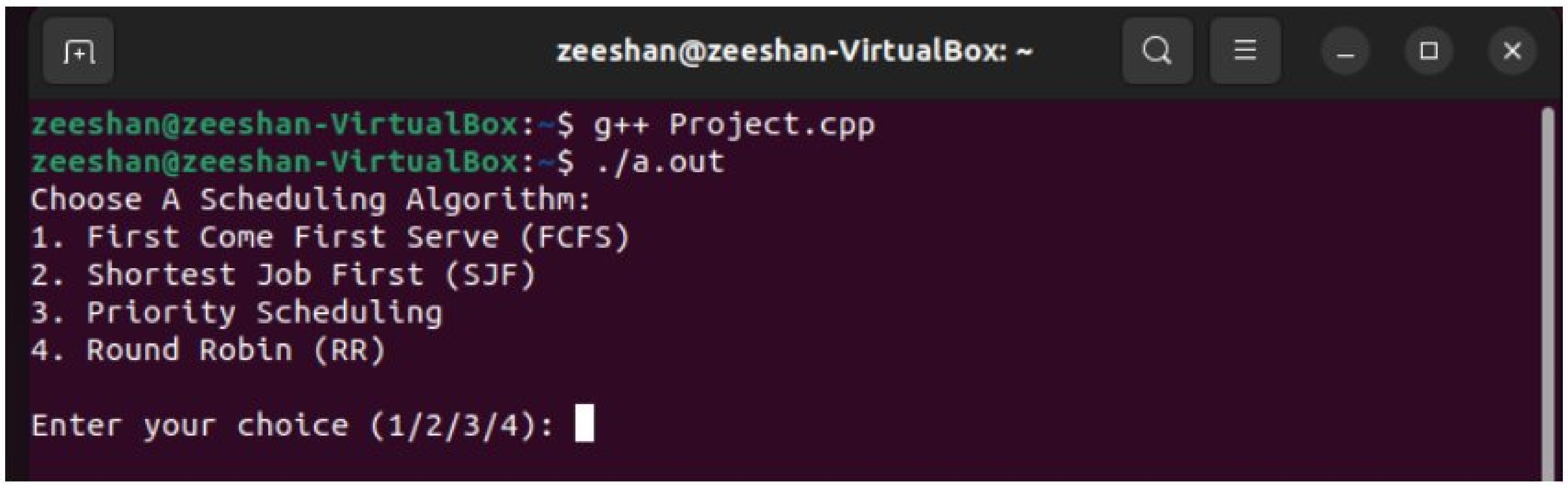
```
// OS FINAL PROJECT BY ZEESHAN ALI 197 & SHAYAN HASSAN 167
#include <iostream>
using namespace std;

// Function to swap two variables
void swap(int* a, int* b)
{
   int temp = *a;
   *a = *b;
   *b = temp;
}

int main()
{
   int choice;
   cout << "Choose A Scheduling Algorithm:\n";
   cout << "1. First Come First Serve (FCFS)\n";
   cout << "2. Shortest Job First (SJF)\n";
   cout << "3. Priority Scheduling\n";
   cout << "4. Round Robin (RR)\n";
   cout << "\nEnter your choice (1/2/3/4): ";
   cin >> choice;
   cout << endl;

   switch (choice)
   {
   case 1:
   {
      // FCFS Algorithm
```

```
        int n = 0;
        cout << "Enter Total Number Of Processes: ";
        cin >> n;

        int pid[n];
        int bt[n];
        int wt[n];
        int p[n];
        wt[0] = 0;
        int tat[n];
        float twt = 0;
        float ttat = 0;
        float awt;
        float atat;

        cout << "Enter Each Process ID:\n";
        for (int i = 0; i < n; i++)
        {
            cout << "P" << i + 1 << ":";
            cin >> pid[i];
            p[i] = i + 1;
        }
        cout << "\nEnter Burst Time Of Each Process:\n";
        for (int i = 0; i < n; i++)
        {
            cout << "P" << i + 1 << ":";
            cin >> bt[i];
            p[i] = i + 1;
        }

        for (int i = 1; i < n; i++)
        {
            wt[i] = wt[i - 1] + bt[i - 1];
            tat[i] = wt[i] + bt[i];
        }

        tat[0] = wt[0] + bt[0];

        cout << "\nPID\t Burst Time\t Waiting Time\t Turnaround Time\n";
        for (int i = 0; i < n; i++)
        {
            cout << pid[i] << "\t\t" << bt[i] << "\t\t" << wt[i] << "\t\t" << tat[i] << "\n";

            twt += wt[i];
            ttat += tat[i];
        }

        awt = twt / n;
        atat = ttat / n;
        cout << "Average Waiting Time = " << awt << "\n";
        cout << "Average Turnaround Time = " << atat << "\n";
        break;
    }

    case 2:
    {
        // SJF Algorithm
        int n;
        cout << "Enter Total Number Of Processes: ";
```

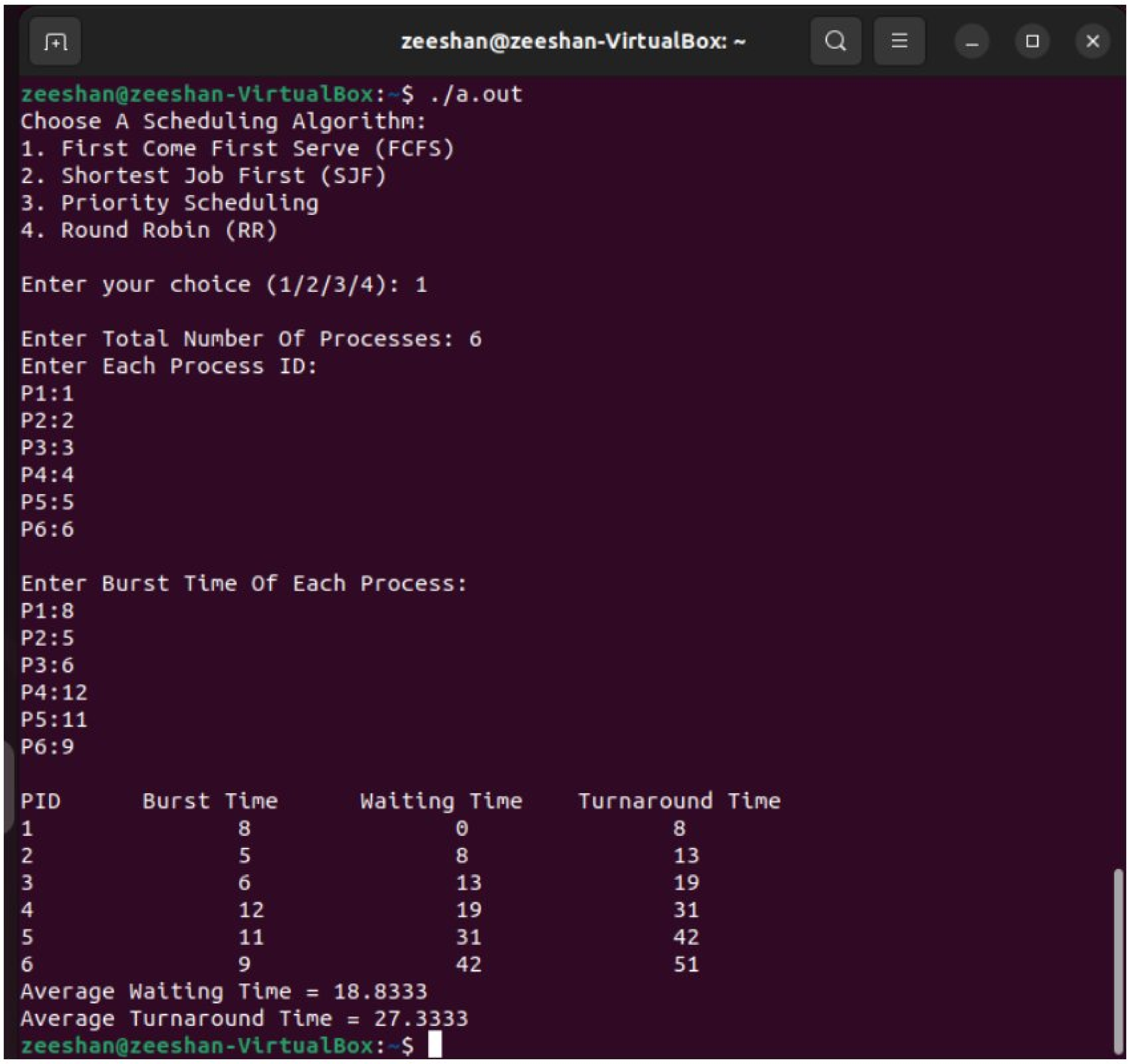
```
cin >> n;

int bt[n];
int p[n];
int wt[n];
int tat[n];
int total = 0;
int pos;
int temp;
float avg_wt;
float avg_tat;

cout << "Enter Burst Time:\n";
for (int i = 0; i < n; i++)
{
    cout << "P" << i + 1 << ":";
    cin >> bt[i];
    p[i] = i + 1;
}

// Sorting of burst times
for (int i = 0; i < n; i++)
{
    pos = i;
    for (int j = i + 1; j < n; j++)
    {
        if (bt[j] < bt[pos])
            pos = j;
    }
    temp = bt[i];
    bt[i] = bt[pos];
    bt[pos] = temp;
    temp = p[i];
    p[i] = p[pos];
    p[pos] = temp;
}

wt[0] = 0;
for (int i = 1; i < n; i++)
{
    wt[i] = 0;
    for (int j = 0; j < i; j++)
        wt[i] += bt[j];
    total += wt[i];
}

avg_wt = (float)total / n;
total = 0;

cout << "\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n";
for (int i = 0; i < n; i++)
{
    tat[i] = bt[i] + wt[i];
    total += tat[i];
    cout << "P" << p[i] << "\t" << bt[i] << "\t\t" << wt[i] << "\t\t" << tat[i] << "\n";
}

avg_tat = (float)total / n;
cout << "\nAverage Waiting Time = " << avg_wt << "\n";
```

```
    cout << "Average Turnaround Time = " << avg_tat << "\n";
    break;
}

case 3:
{
    // Priority Scheduling Algorithm
    int n;
    cout << "Enter Total Number of Processes: ";
    cin >> n;

    // b is array for burst time, p for priority, and index for process id
    int b[n], p[n], index[n];
    for (int i = 0; i < n; i++)
    {
        cout << "Enter Burst Time and Priority Value for Process " << i + 1 << ": ";
        cin >> b[i] >> p[i];
        index[i] = i + 1;
    }

    for (int i = 0; i < n; i++)
    {
        int a = p[i], m = i;

        // Finding out the highest priority element and placing it at its desired position
        for (int j = i; j < n; j++)
        {
            if (p[j] > a)
            {
                a = p[j];
                m = j;
            }
        }

        // Swapping processes
        swap(&p[i], &p[m]);
        swap(&b[i], &b[m]);
        swap(&index[i], &index[m]);
    }

    // T stores the starting time of the process
    int t = 0;

    // Printing the scheduled process and calculating waiting time and turnaround time
    cout << "\nOrder Of Process Execution is\n";
    int wait_time = 0;
    int tat[n];
    float avg_wt = 0, avg_tat = 0;

    for (int i = 0; i < n; i++)
    {
        cout << "P" << index[i] << " Is Executed From " << t << " to " << t + b[i] << "\n";
        wait_time += t;
        tat[i] = wait_time + b[i];
        t += b[i];
    }
    cout << "\n";
    cout << "\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n";
    for (int i = 0; i < n; i++)
    {
        cout << "P" << index[i] << "\t" << b[i] << "\t\t" << wait_time << "\t\t" << tat[i] << "\n";
        avg_wt += wait_time;
        avg_tat += tat[i];
        wait_time += b[i];
    }

    avg_wt /= n;
    avg_tat /= n;
    cout << "\nAverage Waiting Time = " << avg_wt << "\n";
    cout << "Average Turnaround Time = " << avg_tat << "\n";
    break;
}

case 4:
{
    // Round Robin (RR) Algorithm
    int n;
    int quantum;
    cout << "Enter Total Number of Processes: ";
    cin >> n;

    int bt[n];
    int wt[n];
    int tat[n];
    int p[n];
    int remainingTime[n];
    bool completed[n];

    cout << "Enter Burst Time For Each Process:\n";
    for (int i = 0; i < n; i++)
    {
        cout << "P" << i + 1 << ":";
        cin >> bt[i];
        p[i] = i + 1;
    }

    cout << "Enter Time Quantum: ";
    cin >> quantum;

    // Initializing remainingTime and completed arrays
    for (int i = 0; i < n; i++)
    {
        remainingTime[i] = bt[i];
        completed[i] = false;
    }

    // RR Algorithm
    int time = 0;
    while (true)
    {
        bool done = true;
        for (int i = 0; i < n; i++)
        {
            if (!completed[i])
            {
                done = false;
                if (remainingTime[i] > 0)
                {
                        if (remainingTime[i] > quantum)
                        {
                            time += quantum;
                            remainingTime[i] -= quantum;
                        }
                        else
                        {
                            time += remainingTime[i];
                            wt[i] = time - bt[i];
                            remainingTime[i] = 0;
                            completed[i] = true;
                        }
                    }
                }
            }
            if (done)
                break;
        }

        for (int i = 0; i < n; i++)
        {
            tat[i] = bt[i] + wt[i];
        }

        float avg_wt = 0, avg_tat = 0;
        for (int i = 0; i < n; i++)
        {
            avg_wt += wt[i];
            avg_tat += tat[i];
        }

        avg_wt /= n;
        avg_tat /= n;

        cout << "\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n";
        for (int i = 0; i < n; i++)
        {
            cout << "P" << i + 1 << "\t\t" << bt[i] << "\t\t" << wt[i] << "\t\t" << tat[i] << "\n";
        }

        cout << "\nAverage Waiting Time = " << avg_wt << "\n";
        cout << "Average Turnaround Time = " << avg_tat << "\n";
        break;
    }

    default:
        cout << "Invalid Choice\n";
    }

    return 0;
}
```

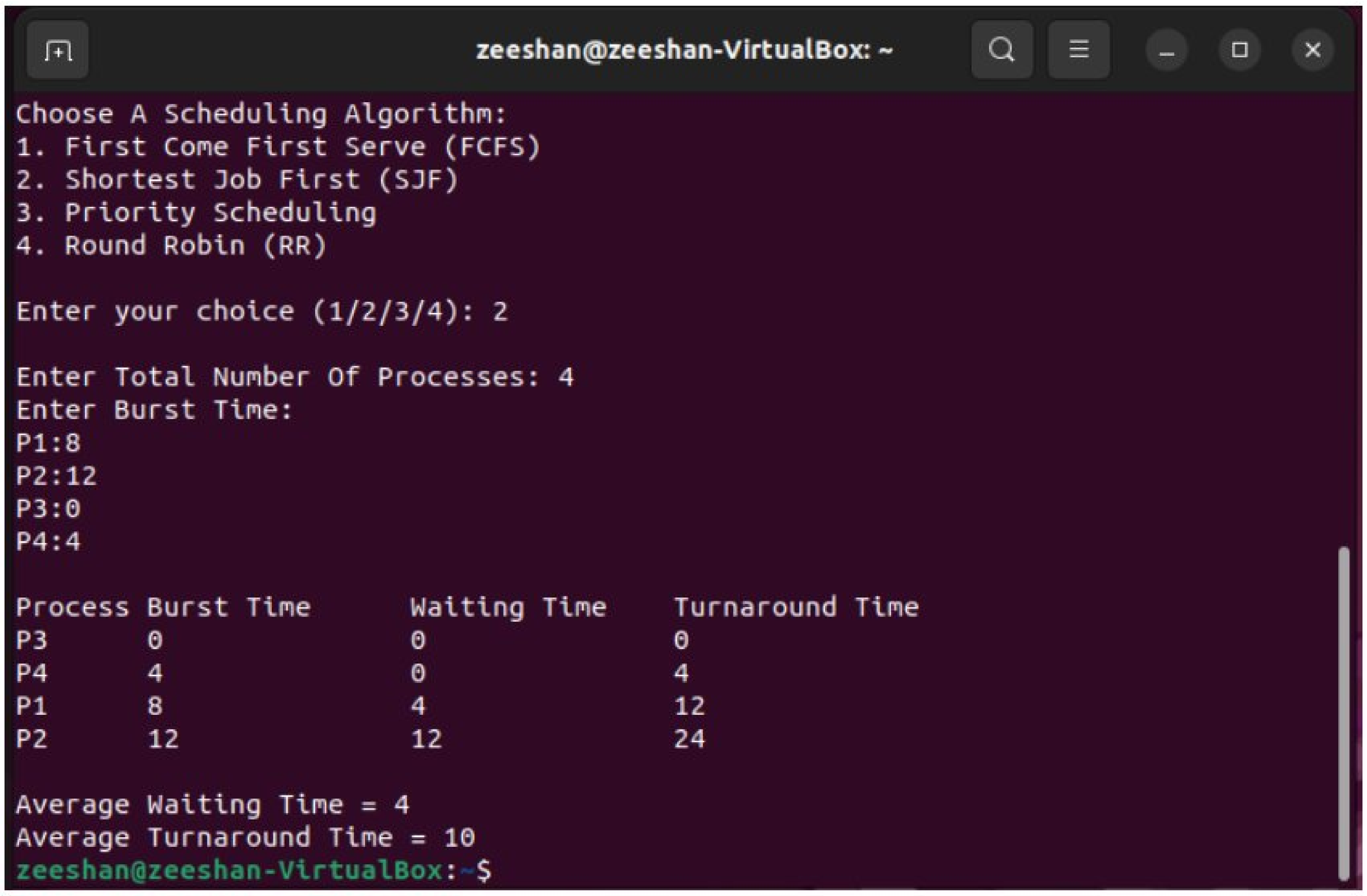
Output:



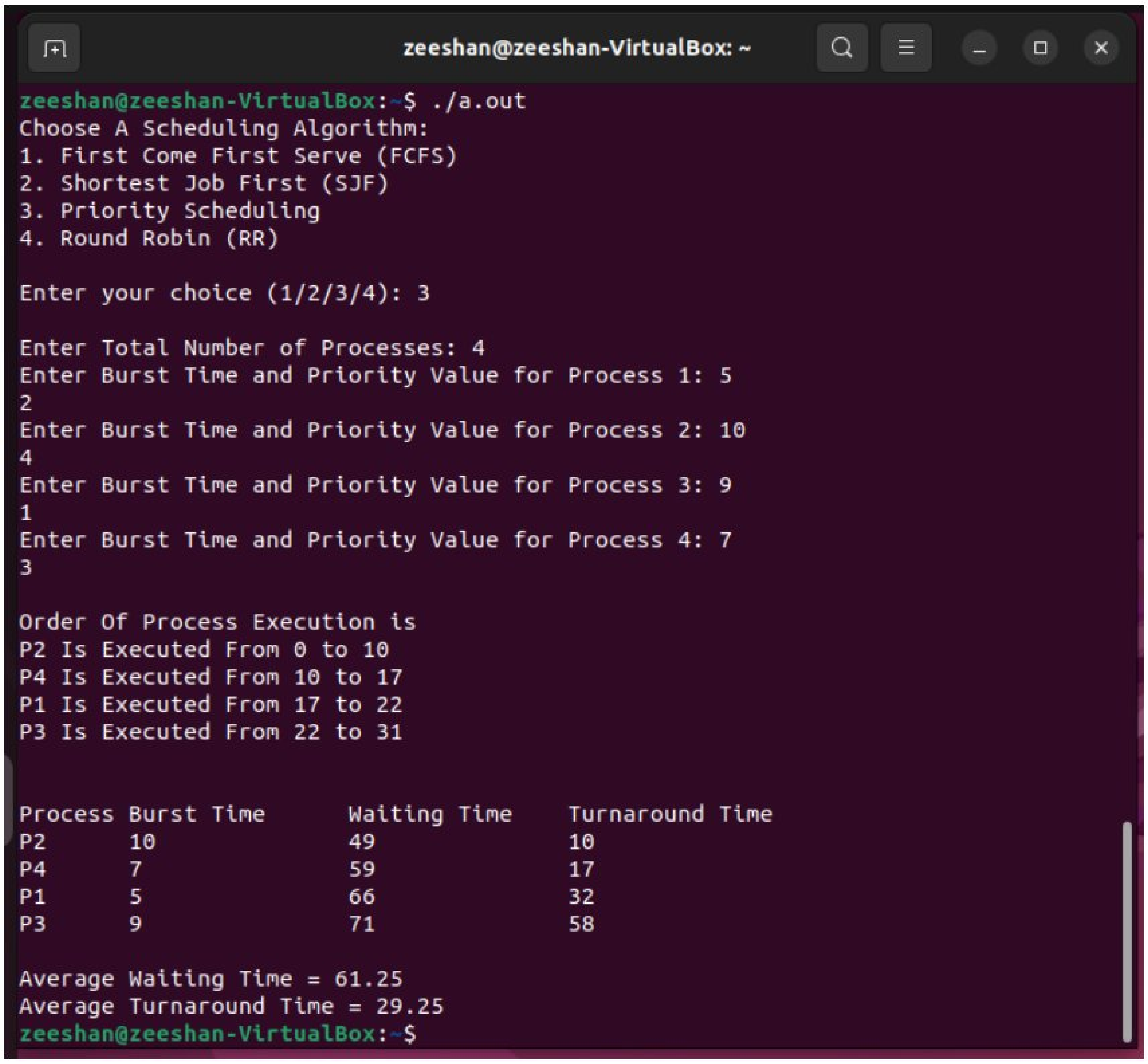
```
zeeshan@zeeshan-VirtualBox:~$ g++ Project.cpp
zeeshan@zeeshan-VirtualBox:~$ ./a.out
Choose A Scheduling Algorithm:
1. First Come First Serve (FCFS)
2. Shortest Job First (SJF)
3. Priority Scheduling
4. Round Robin (RR)

Enter your choice (1/2/3/4): 
```



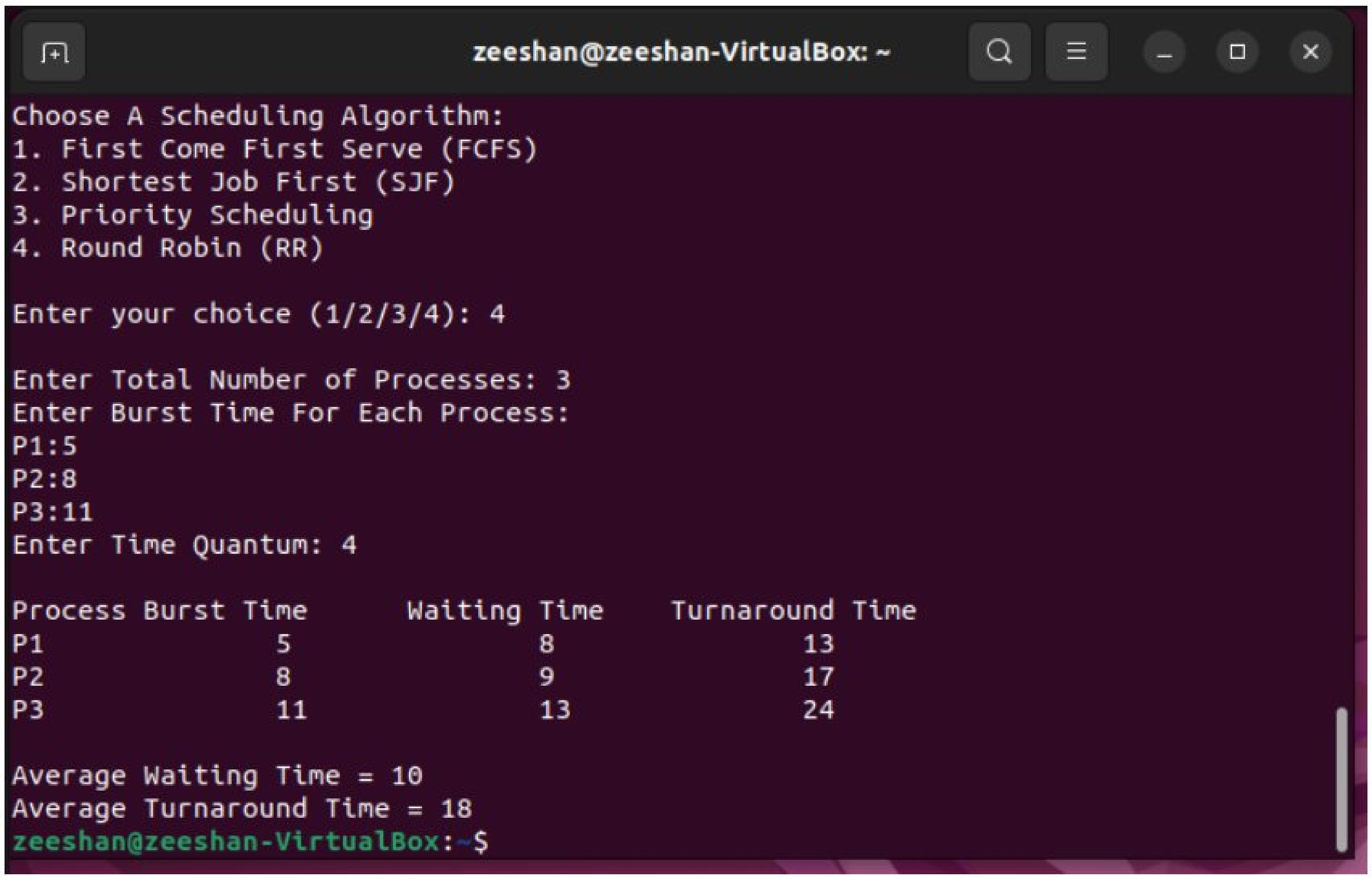
```
zeeshan@zeeshan-VirtualBox:~$ ./a.out
Choose A Scheduling Algorithm:
1. First Come First Serve (FCFS)
2. Shortest Job First (SJF)
3. Priority Scheduling
4. Round Robin (RR)

Enter your choice (1/2/3/4): 1

Enter Total Number Of Processes: 6
Enter Each Process ID:
P1:1
P2:2
P3:3
P4:4
P5:5
P6:6

Enter Burst Time Of Each Process:
P1:8
P2:5
P3:6
P4:12
P5:11
P6:9

PID      Burst Time      Waiting Time    Turnaround Time
1               8               0               8
2               5               8               13
3               6               13              19
4               12              19              31
5               11              31              42
6               9               42              51
Average Waiting Time = 18.8333
Average Turnaround Time = 27.3333
zeeshan@zeeshan-VirtualBox:~$ 
```



```
Choose A Scheduling Algorithm:
1. First Come First Serve (FCFS)
2. Shortest Job First (SJF)
3. Priority Scheduling
4. Round Robin (RR)

Enter your choice (1/2/3/4): 2

Enter Total Number Of Processes: 4
Enter Burst Time:
P1:8
P2:12
P3:0
P4:4

Process Burst Time      Waiting Time    Turnaround Time
P3      0               0               0
P4      4               0               4
P1      8               4               12
P2      12              12              24

Average Waiting Time = 4
Average Turnaround Time = 10
zeeshan@zeeshan-VirtualBox:~$
```



```
zeeshan@zeeshan-VirtualBox:~$ ./a.out
Choose A Scheduling Algorithm:
1. First Come First Serve (FCFS)
2. Shortest Job First (SJF)
3. Priority Scheduling
4. Round Robin (RR)

Enter your choice (1/2/3/4): 3

Enter Total Number of Processes: 4
Enter Burst Time and Priority Value for Process 1: 5
2
Enter Burst Time and Priority Value for Process 2: 10
4
Enter Burst Time and Priority Value for Process 3: 9
1
Enter Burst Time and Priority Value for Process 4: 7
3

Order Of Process Execution is
P2 Is Executed From 0 to 10
P4 Is Executed From 10 to 17
P1 Is Executed From 17 to 22
P3 Is Executed From 22 to 31


Process Burst Time      Waiting Time    Turnaround Time
P2      10              49              10
P4      7               59              17
P1      5               66              32
P3      9               71              58

Average Waiting Time = 61.25
Average Turnaround Time = 29.25
zeeshan@zeeshan-VirtualBox:~$
```



```
Choose A Scheduling Algorithm:
1. First Come First Serve (FCFS)
2. Shortest Job First (SJF)
3. Priority Scheduling
4. Round Robin (RR)

Enter your choice (1/2/3/4): 4

Enter Total Number of Processes: 3
Enter Burst Time For Each Process:
P1:5
P2:8
P3:11
Enter Time Quantum: 4

Process Burst Time      Waiting Time    Turnaround Time
P1              5               8               13
P2              8               9               17
P3              11              13              24

Average Waiting Time = 10
Average Turnaround Time = 18
zeeshan@zeeshan-VirtualBox:~$
```

Conclusion:

In summary, a process scheduling simulator is a technique that helps users explore and understand how different scheduling algorithms impact the performance of a computer system. It allows for the simulation of scenarios with various parameters, such as arrival times and priorities, and provides insights into metrics like turnaround time and CPU utilization. This tool is valuable for educational purposes, aiding students in learning the principles of process scheduling and assisting system administrators and developers in optimizing system performance.

Tools Used:

1. Ubuntu Linux
2. Visual Studio
3. Microsoft Word

... The END ...