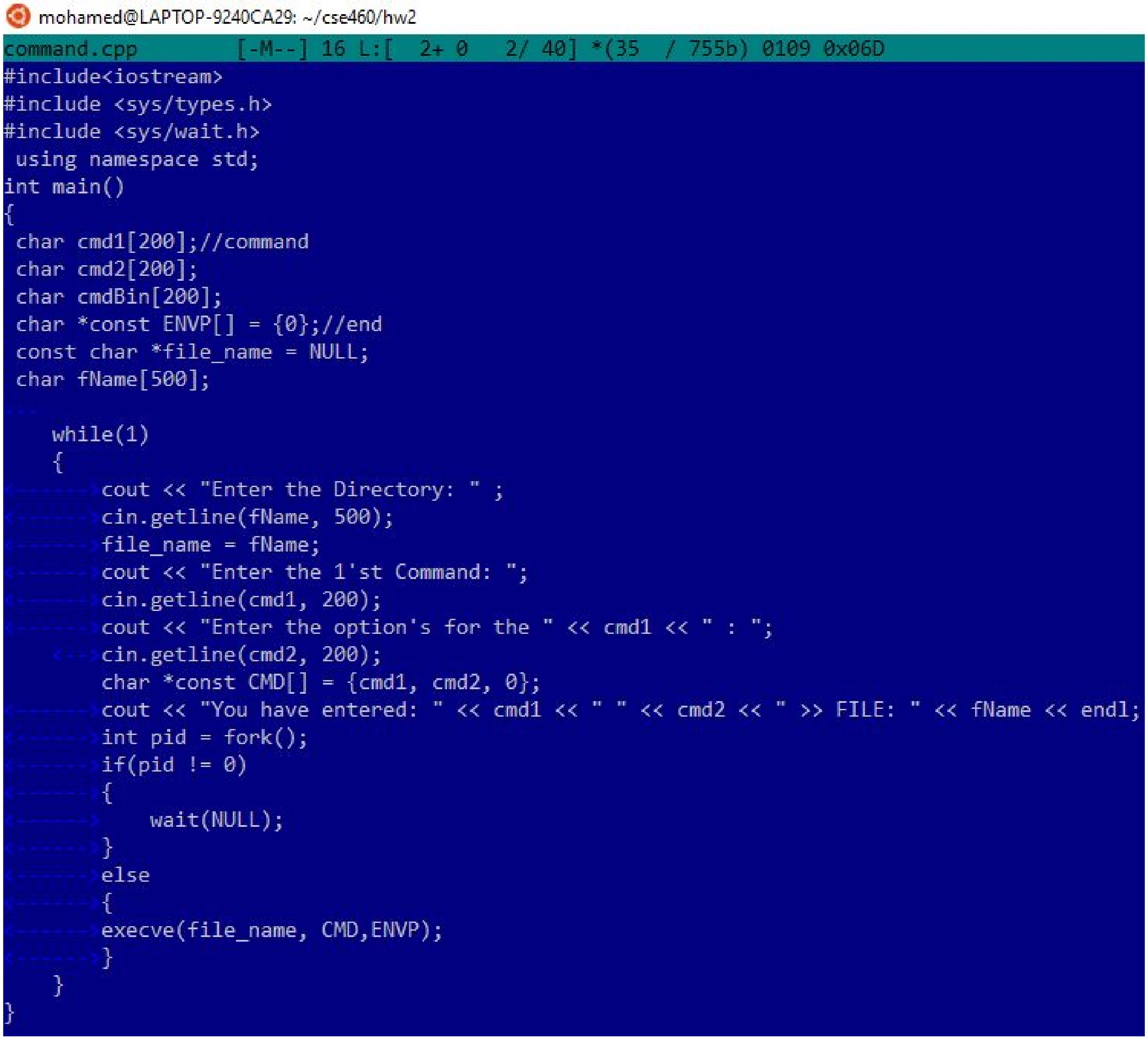
# CSE 460
# Homework 2
# Mohamed Abdul

1) **Write a simple shell that is similar to what we have discussed in class**

**Command.cpp //the code of the shell**

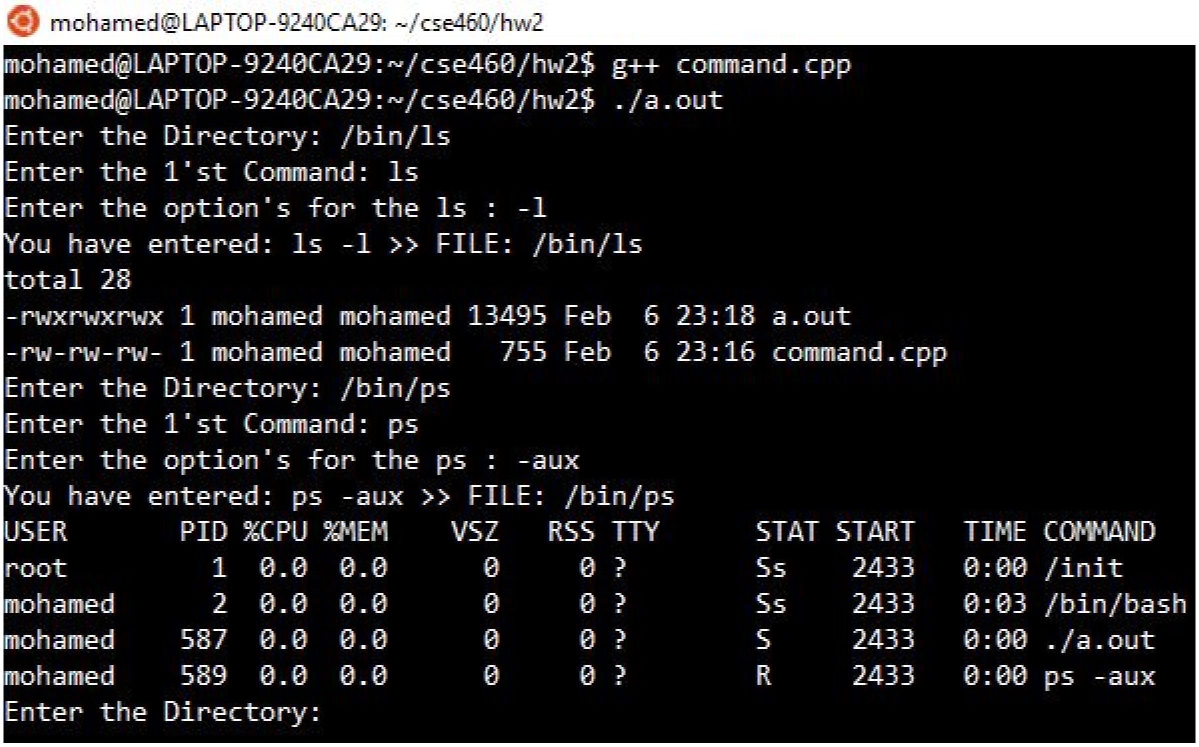


mohamed@LAPTOP-9240CA29: ~/cse460/hw2

```
command.cpp        [-M--] 16 L:[  2+ 0   2/ 40] *(35  / 755b) 0109 0x06D
#include<iostream>
#include <sys/types.h>
#include <sys/wait.h>
 using namespace std;
int main()
{
 char cmd1[200];//command
 char cmd2[200];
 char cmdBin[200];
 char *const ENVP[] = {0};//end
 const char *file_name = NULL;
 char fName[500];

    while(1)
    {
        cout << "Enter the Directory: " ;
        cin.getline(fName, 500);
        file_name = fName;
        cout << "Enter the 1'st Command: ";
        cin.getline(cmd1, 200);
        cout << "Enter the option's for the " << cmd1 << " : ";
        cin.getline(cmd2, 200);
        char *const CMD[] = {cmd1, cmd2, 0};
        cout << "You have entered: " << cmd1 << " " << cmd2 << " >> FILE: " << fName << endl;
        int pid = fork();
        if(pid != 0)
        {
            wait(NULL);
        }
        else
        {
        execve(file_name, CMD,ENVP);
        }
    }
}
```

## Output



mohamed@LAPTOP-9240CA29: ~/cse460/hw2

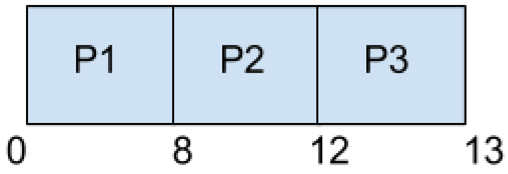
```
mohamed@LAPTOP-9240CA29:~/cse460/hw2$ g++ command.cpp
mohamed@LAPTOP-9240CA29:~/cse460/hw2$ ./a.out
Enter the Directory: /bin/ls
Enter the 1'st Command: ls
Enter the option's for the ls : -l
You have entered: ls -l >> FILE: /bin/ls
total 28
-rwxrwxrwx 1 mohamed mohamed 13495 Feb  6 23:18 a.out
-rw-rw-rw- 1 mohamed mohamed   755 Feb  6 23:16 command.cpp
Enter the Directory: /bin/ps
Enter the 1'st Command: ps
Enter the option's for the ps : -aux
You have entered: ps -aux >> FILE: /bin/ps
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0      0     0 ?        Ss    2433   0:00 /init
mohamed      2  0.0  0.0      0     0 ?        Ss    2433   0:03 /bin/bash
mohamed    587  0.0  0.0      0     0 ?        S     2433   0:00 ./a.out
mohamed    589  0.0  0.0      0     0 ?        R     2433   0:00 ps -aux
Enter the Directory:
```

**2) Suppose that the following processes arrive for execution at the times indicated. Each process will run the listed amount of time. In answering the questions, use nonpreemptive scheduling and base all decisions on the information that you have at the time the decision must be made.**

| Process | Arrival Time | Burst Time |
|---|---|---|
| P1 | 0.0 | 8 |
| P2 | 0.4 | 4 |
| P3 | 1.0 | 1 |

**a) What is the average waiting time for these processes with the FCFS scheduling algorithm?**

**Gantt Chart**

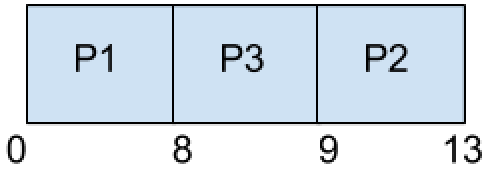


| **Process** | **Wait Time** |
|---|---|
| **P1** | **0** |
| **P2** | **7.6** |
| **P3** | **11** |

TOTAL = 18.6 Units Average Time = 18.6 / 3 = 6.2 Units

**b) What is the average waiting time for these processes with the SJF scheduling algorithm?**

**Gantt Chart**

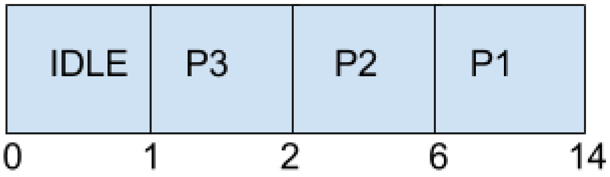


| **Process** | **Wait Time** |
|---|---|
| **P1** | **0** |
| **P2** | **8.6** |
| **P3** | **7** |

TOTAL = 15.6 Average Time = 15.6 / 3 = 5.2 Units

**c ) The SJF algorithm is supposed to improve performance, but notice that we chose to run process P1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average waiting time will be if the CPU is left idle for the first 1 unit, and then SJF scheduling is used. Remember that processes P1 and P2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knowledge scheduling.**

**Gantt Chart**



| Process | Wait Time |
|---|---|
| **P1** | **5** |
| **P2** | **1.6** |
| **P3** | **1** |

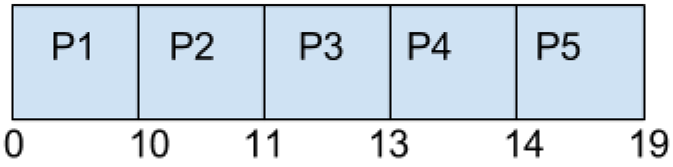
TOTAL = 7.6 Average Time = 7.6 / 3 = 2.53 Units

**3) Consider the following set of processes, with the length of the CPU-burst time given in milliseconds.**

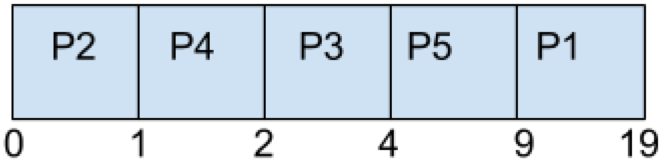
| Process | Burst Time | Priority |
|---|---|---|
| P1 | 10 | 3 |
| P2 | 1 | 1 |
| P3 | 2 | 3 |
| P4 | 1 | 4 |
| P5 | 5 | 2 |

a) Draw four Gantt charts that illustrate the execution of these processes using FCFS, SJF, a nonpreemptive priority ( a smaller number implies higher priority ), and RR ( quantum = 1 ) scheduling.

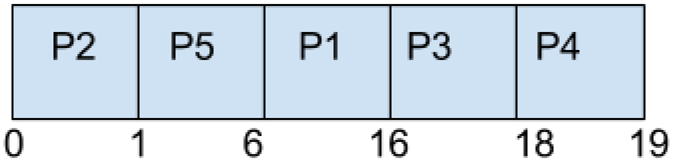
**FCFS Gantt Chart**



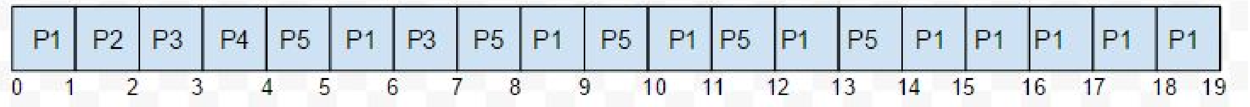
**SJF**



**Non-Preemptive**



**Round Robin (Quantum = 1)**



**b) Calculate the turnaround time of each process for each of the scheduling algorithms in part a).**

**FCFS**

| **Process** | **Turn Around Time** |
|---|---|
| **P1** | **10** |
| **P2** | **11** |
| **P3** | **13** |
| **P4** | **14** |

| | |
|---|---|
| **P5** | **19** |

TOTAL = 67 Units

Average Turnaround Time = 67/5 = 13.4 Units

### SJF

| **Process** | **Turn Around Time** |
|---|---|
| **P1** | **19** |
| **P2** | **1** |
| **P3** | **4** |
| **P4** | **2** |
| **P5** | **9** |

TOTAL = 35 Units

Average Turnaround Time = 35/5 = 7 Units

### Non-Preemptive

| **Process** | **Turn Around Time** |
|---|---|
| **P1** | **16** |
| **P2** | **1** |
| **P3** | **18** |
| **P4** | **19** |
| **P5** | **6** |

TOTAL = 60 Units

Average Turnaround Time = 60/5 = 12 Units

### RR

| **Process** | **Turn Around Time** |
|---|---|
| **P1** | **19** |
| **P2** | **2** |
| **P3** | **7** |
| **P4** | **4** |
| **P5** | **14** |

TOTAL = 46 Units

Average Turnaround Time = 46/5 = 9.2 Units

**c) Calculate the waiting time of each process for each of the scheduling algorithms in part a).**

1.

**FCFS**

| Process | Waiting Time |
|---|---|
| P1 | 0 |
| P2 | 10 |
| P3 | 11 |
| P4 | 13 |
| P5 | 14 |

TOTAL = 48 Units

Average Waiting Time = 48/5 = 9.6 Units

**SJF**

| Process | Waiting Time |
|---|---|
| P1 | 9 |
| P2 | 0 |
| P3 | 2 |
| P4 | 1 |
| P5 | 4 |

TOTAL = 16 Units

Average Waiting Time = 16/5 = 3.2 Units

**Non-Preemptive**

| Process | Waiting Time |
|---|---|
| P1 | 6 |
| P2 | 0 |
| P3 | 16 |
| P4 | 18 |
| P5 | 1 |

TOTAL = 41 Units

Average Waiting Time = 41/5 = 8.2 Units

**RR**

| **Process** | **Waiting Time** |
|---|---|
| **P1** | **9** |
| **P2** | **1** |
| **P3** | **5** |
| **P4** | **3** |
| **P5** | **9** |

TOTAL = 27 Units

Average Waiting Time = 27/5 = 5.4 Units

**d) Which of the schedules in part a) results in the minimal average waiting time ( over all processes**

I think, the **SJF** scheduling algorithm has the shortest waiting time compare to other scheduling algorithm.

**Evaluation:**

I give myself 60/60, because I have put lot of effort on this homework, I have provided the output for the program. I illustrated the Gantt chart.