The Department of Electrical and Computer Engineering

The University of Alabama in Huntsville

CPE 435 Lab-07

Process Scheduling

Introduction

A multiprogramming operating system allows more than one process to be loaded into the executable memory at one time and allows for the loaded process to share the CPU using time multiplexing. One of the reasons for using multiprogramming is that the operating system itself is implemented as one or more processes, so there must be a way for the operating system and application processes to share the CPU. The scheduler is responsible for allocating the processor resources to the individual processes. There are different classes of scheduling that can be implemented for this purpose like Round-robin, Priority and First Come First Serve.

Important Terms

Quantum Time: It is the amount of time a process is allowed to run. After a process is run for Quantum time, the scheduler chooses another process to run interrupting the currently running process in *preemptive scheduling*. Based on the scheduling algorithm, the interrupted process will again be allowed to run.

Turn Around Time: It is the total amount of time taken by a process to complete its execution. Turn around time includes the time actually a process is run on the CPU plus the time it waits.

Burst Time: Time required by a process to complete. This is the actual time the process requires CPU for completion.

Wait Time: Time a process waits. *Wait Time = Turn Around Time - Burst Time*

Assignment 1

Round-robin scheduling algorithm is one of the simplest scheduling algorithms, where each process gets a small unit of CPU time (*time quantum*) . Write a program that implements this scheduling. Find the scheduling order, average waiting time for the processes and turn-around time (to each process) when round-robin scheduling is applied.

Your program will accept as input following items:

Number of processes, *n*
Quantum Time Unit
Id and Burst Time for each of *n* processes

You will create a structure as follows (This is just a hint, you may choose to do it in any way you like):

```
struct process{
        int pid; //Process ID
        int burst_time; //CPU Burst Time
        int working_time; //Time this process executed, if  working_time==burst_time, process is complete
        int t_round; //turn around time, time needed for the process to complete
 }
```

You can assume that all the processes arrive at the same time.

Your output must clearly show the order of execution for the processes. One sample example with quantum time 10ms is shown below. In addition you must also print the turn around time and wait time for each of the processes (you can create another table). Calculate and print average wait time also.

Time	**10ms**	**10ms**	**10ms**	**10ms**	**10ms**
PID	1	2	3	1	2

Sample Output

Given the input of 3 processes, process 1 with a burst time of 25, process 2 with a burst time of 7, process 3 with a burst time of 15, and a quantum time of 10 ms your output may look similar to the following:

```
-------------------------------------------------------------------------------------
| Time (ms) |     10    |      7    |     10    |     10    |      5    |      5    |
-------------------------------------------------------------------------------------
| PID       |      1    |      2    |      3    |      1    |      3    |      1    |
-------------------------------------------------------------------------------------

-------------------------------------------------------------
| PID                  |      1    |      2    |      3    |
-------------------------------------------------------------
| Wait Time (ms)       |     22    |     10    |     27    |
-------------------------------------------------------------
| Turn Around Time (ms) |     47    |     17    |     42    |
-------------------------------------------------------------

-------------------------------------
| Average Wait Time (ms) |     19    |
-------------------------------------
```

Assignment 2

Priority based-scheduling is one of the scheduling algorithms in which processes are allocated to the CPU on the basis of an externally assigned priority. The key to the performance of priority scheduling is in choosing priorities for the processes. Write a program that implements this scheduling. Find the scheduling order, average waiting time for the processes and turn-around time (to each process) when Priority scheduling is applied (assume that the processes are Non-Preemptive).

Your program will accept as input following items:

Number of processes, *n*
Id, Priority and Burst Time for each of *n* processes

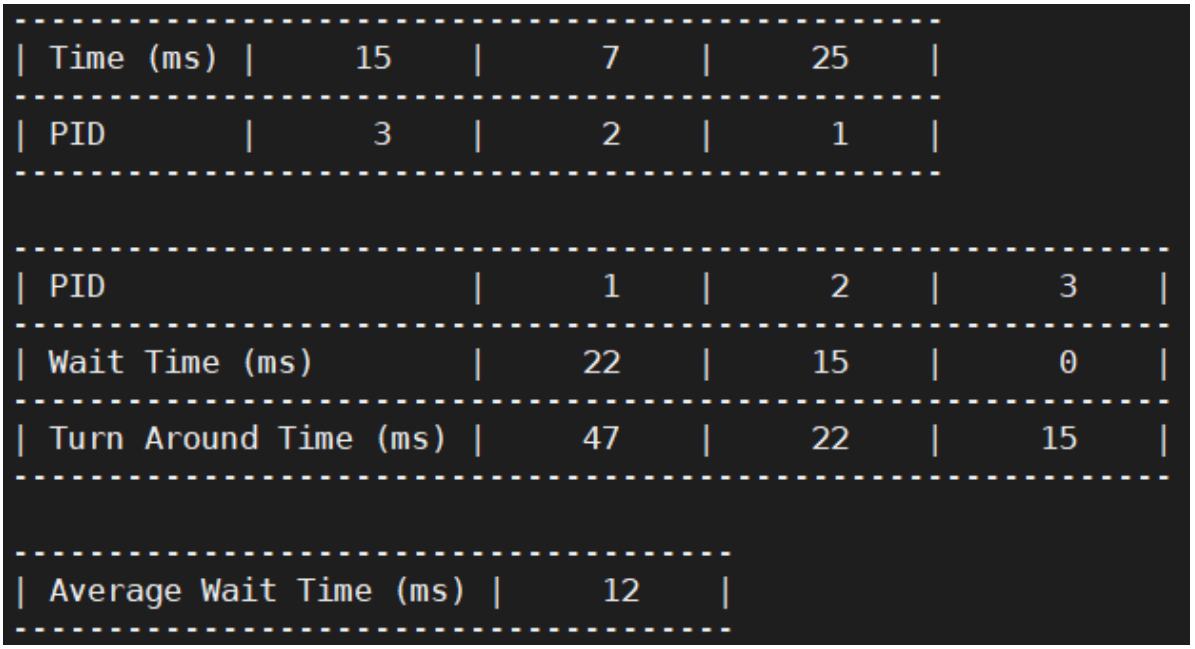
You will create a similar structure, but this time you will have priority for each of the processes.

```
struct process{
        int pid; //Process ID
        int priority;
        int burst_time; //CPU Burst Time
        int working_time; //Time this process executed, if  working_time==burst_time, process is complete
        int t_round; //turn around time, time needed for the process to complete
}
```

You must display the result in tabular form as in the previous assignment.

Sample Output

Given the input of 3 processes, process 1 with a burst time of 25 and a priority of 3, process 2 with a burst time of 7 and priority of 2, and process 3 with a burst time of 15 and a priority of 1 your output may look similar to the following:



Topics for theory

- Round Robin Scheduling
- Priority Based Scheduling
- Preemptive vs Non-Premptive Scheduling

Deliverables

Lab Report

The following material in each section is expected:

1. Cover page with your name, lab number, course name, and dates
2. Theory/Background (Material or methods relevant to the lab, a few sentences on each in your own words)
    a. Round Robin Scheduling
    b. Priority Based Scheduling
    c. Preemptive vs Non-Premptive Scheduling
3. Observations
    a. Round Robin Scheduling Output (screenshots of terminal output)
    b. Priority Based Scheduling Output (screenshots of terminal output)
4. Conclusion (Did your program work as expected, what can you take away from the lab?)
5. Appendix (for source code, submit the text in a table)

The report should be submitted as a single pdf document with the source code for your program within it.

Recorded Demonstration

The following material in each section is expected:

1. Introduce yourself and give the name of the lab
2. Compile the program
3. Show that the programs run similar to what was reported under observations

The demonstration may be in person or recorded and submitted as an mp4 file alongside the report.