# Assignment 2: CPU Scheduling Simulation

CSC 139 Operating System Principles - Fall 2021

Posted on Oct. 18, due on Oct. 31 (11:59 pm)

## 1 Objectives

This programming project is to simulate a few CPU scheduling policies discussed in the class. You will write a C program to implement a simulator with different scheduling algorithms. The simulator selects a task to run from ready queue based on the scheduling algorithm. Since the project intends to simulate a CPU scheduler, so it does not require any actual process creation or execution. When a task is scheduled, the simulator will simply print out what task is selected to run at a time. It outputs the way similar to Gantt chart style.

The learning objectives of this assignment are

1. To describe various CPU scheduling algorithms; (CLO3; CLO7)
2. To implement a simulator for CPU scheduling algorithms; (CLO3; CLO7)
3. To assess CPU scheduling algorithms based on scheduling criteria. (CLO3)

## 2 Description

The selected scheduling algorithms to implement in this project are 1) First Come First Serve (FCFS), 2) Round Robin (RR), and 3) Shortest Remaining Time First (SRTF). The detailed algorithms are already described in class slides and textbook Chapter 5.

### 2.1 Task Information

The task information will be read from an input file. The format is
`pid arrival_time burst_time`

All of fields are integer type where
`pid` is a unique numeric process ID
`arrival_time` is the time when the task arrives
`burst_time` the is the CPU time requested by a task
The time unit for `arrival_time`, `burst_time` and `interval` is millisecond.
Note: the input file can have an arbitrary extension and can be called from any legal path. The task information is not necessarily ordered in terms of any of the three fields.

### 2.2 Command-line Usage and Examples

`Usage: proj2 input_file [FCFS|RR|SRTF] [time_quantum]`
where `input_file` is the file name with task information. FCFS, RR, and SRTF are names of scheduling algorithms. The `time_quantum` only applies to RR. FCFS is non-preemptive, while RR and SRTF are both preemptive. The last argument is needed only for RR. (See following table for more examples)

| Examples | Description |
|---|---|
| `proj2 input.1 FCFS` | FCFS scheduling with the data file "`input.1`" |
| `proj2 input.1 RR 2` | Simulate RR scheduling with time quantum 2 milliseconds (4th parameter is required even for quantum 1 millisecond) with the data file "`input.1`" |
| `proj2 input.1 SRTF` | Simulate SRTF scheduling with the data file "`input.1`" |

### 2.3 Sample Inputs and Outputs

Sample input files and expected outputs are shown in Appendix. You can use it to verify your results.

## 3 Requirements

The project requires to simulate FCFS, RR, and SRTF scheduling for given tasks and to compute the average waiting time, response time, turnaround time, and overall CPU usage. You can find their definitions in textbook.

1. Implement scheduling algorithm for FCFS, RR, and SRTF. The program should schedule tasks and print progress of task every unit time (millisecond) as shown in sample outputs.
2. Print statistical information. As soon as all tasks are completed, the program should compute and print 1) average waiting time, 2) average response time, 3) average turnaround time, and 4) overall CPU usage.

Note: if you use static array to implement ready queue structure, you can assume the maximum queue length is 20.

## 4 Suggested Methodology

1. Implement one scheduling algorithm at each step
2. You can use circular linked list as a queue structure
3. You can use a data structure similar to PCB for each task, though it will be much simpler

## 5 Deliverables

Make sure your code can be compiled and work on ECS PA Coding1 server correctly. Upload to Canvas the following:

- All source codes/files that you have added/modified and the demonstrative results.
- A `README.TXT` file that briefly describes each file, how to compile the file(s), and how to run the file.

- These files should be placed in a directory called `<SacLink username>-asgmt2`.
- Use tar command to place all the files in a single file called `<SacLink username>-asgmt2.tar`. Assuming you are in the directory `<SacLink username>-asgmt2` do the following:
  - Goto the parent directory: `cd ..`
  - tar the files:
    `tar -cvf <SacLink username>-asgmt2.tar ./<SacLink username>-asgmt2`
  - Verify the files have been placed in a tar file:
    `tar -tvf <SacLink username>-asgmt2.tar`
  - Compress the files using gzip: `gzip <SacLink username>-asgmt2.tar`
  - Verify that the gzipped file exists: `ls <SacLink username>-asgmt2.tar.gz`

Note: `<SacLink username>` is just a placeholder. Replace it with your SacLink username.