‘

CS376 – Ursinus College
Spring 2022
Programming Project #2

1. Objectives:
This goal of this project is to *simulate* a few CPU scheduling algorithms discussed in class. You will write a C++ or Java 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, it does **not** require any actual process creation or execution. When a task is scheduled, the simulator will print out what task is selected to run at a time. The output of your simulator will be similar to a Gantt chart.

2. Description:
The selected scheduling algorithms you will implement in this project are

1) First Come First Serve (**FCFS**)
2) Round Robin (**RR**)
3) Shortest Job First (**SJF)**

You program must be able to run all three algorithms. Which one your program will run is determined by a command line argument. The details of these algorithms are available in the text book

2.1 Process Information
The task information will be read from an input file. The format is

pid arrival_time burst_time

All of fields are of integer type where:

pid is a unique numeric process ID
arrival_time is the time when the task arrives in the unit of milliseconds
burst_time the is the CPU time requested by a task, in the unit of milliseconds

The time unit for **arrival_time**, **burst_time** and **interval** is millisecond.
Here is a sample input file:

1 0 10
2 0 9
3 3 5

4 7 4
5 10 6
6 10 7

2.2 Command-line Usage and Examples
Usage: **proj2 input_file [FCFS|RR|SJF] [time_quantum]**

Where *input_file* is the file name with task information described in section 2.1. FCFS and RR are the names of scheduling algorithms. The time quantum only applies to RR. FCFS is non-preemptive while SJF, RR are **preemptive**. **The last argument is needed only for RR.** If you are using an IDE, be sure to tell me which one and use its command line argument capability.

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 SJF Shortest Job First Scheduling with data file "input.1"

2.2 Design Hints
Here is a possible design logic you can refer to. The simulator first reads task information from the input file and stores all data in a data structure. Then it starts simulating a scheduling algorithm in a **time driven** manner. At each time unit (or slot), it adds any newly arrived task(s) into the ready queue and calls a specific scheduler algorithm in order to select appropriate task from ready queue. When a task is chosen to run, the simulator prints out a message indicating what process ID is chosen to execute for this time slot. If no task is running (i.e. empty ready queue), it prints out an "idle" message. Before advancing to the next time unit, the simulator should make all necessary changes in task and ready queue status. **For this assignment, you're not using actual clock time on the computer, you can instead have a loop ticking off time in the simulator.**

3. Requirements:
The project requires you to simulate FCFS, SJF and RR scheduling for given tasks and to compute the **average waiting time, turnaround time and overall CPU usage**. You can find the definitions of these metrics in the textbook as well as in the class slides.

· **Implement scheduling algorithm for FCFS, SJF and RR.** The program should schedule tasks and print progress of task every unit time (millisecond) as shown in sample outputs.
· **Print statistical information**. As soon as all tasks are completed, the program

should compute and print 1) **average waiting time**, 2) **average turnaround time** and 3) **overall CPU usage**.

4. Suggested Methodology:

· Implement one scheduling algorithm at each step.
· You can use a circular linked list as the queue structure.
· You can use a data structure similar to a Process Control Block (PCB) for each task, though it will be much simpler than an actual PCB

5. Submission:

1. All source code files
2. A README that briefly describes each file, and how to run the program including which IDE you used, if any. I would prefer to run your code on the command line, but if that's not possible, I can deal.

6. Grading Policy:

1. If the program cannot be compiled and run without substantial fixes to code, it will result in a score of zero. I do understand that sometimes there are minor occurrences (a hardcoded file name or such) that may need a minor tweak or a configuration detail, but I will not do major surgery on your code to get it to run.

2. All style references (comments, loop choice, etc…) should follow the Ursinus Java Style guide. If you are working in C++, follow the guide as closely as possible for that language.

2. Grading breakdown is as follows:

Documentation (5 points)

Code should be well documented including comments at the top of each source code file indicating class, student's name, institution, project title, project number, due date, and purpose of the project. This documentation should be clear and descriptive.

`

Code should also have comments for each function/method giving a description of the functionality, parameters, return values.

A README which describes any steps/configuration I need to do to run and test your project.

Readability and Structure (5 points)

Code should be well structured. I should be able to tell what it's doing by reading it. Of course you have comments, but those are secondary to the code. The project should make sensible use of class structure and good programming design in general.

Robustness (15 points)

Bad input such as forgetting to include a time quantum for round robin, having a malformed input file, or other such mishaps should not crash your project. The project should let the use know what the problem is and then exit gracefully.

Design (20 points)

As this program will be in an object oriented language, I expect at the least method modularity such that code is not copied and pasted throughout. As well, I expect judicious usage of classes, aggregation, and inheritance.

Functionality (55 points)

The project must implement FCFS, RR, and SJF algorithms correctly giving output as stated in this problem description. All statements describing program functionality must be followed for full points.

7. Appendix: about input file and output information by CPU scheduling simulator:

Here is a sample input file:

```
% more input.1
1 0 10
2 0 9
3 3 5
4 7 4
5 10 6
6 10 7
```

My program details its usage

```
% proj2
Usage: proj2 input_file FCFS|SJF|RR [quantum]
```

And here is a sample output:

```
% proj2 input.1 FCFS
Scheduling algorithm: FCFS
Total 6 tasks are read from "input.1". press 'enter' to start...
================================================================
<system time 0> process 1 is running
<system time 1> process 1 is running
<system time 2> process 1 is running
<system time 3> process 1 is running
<system time 4> process 1 is running
<system time 5> process 1 is running
<system time 6> process 1 is running
<system time 7> process 1 is running
<system time 8> process 1 is running
<system time 9> process 1 is running
<system time 10> process 1 is finished.......
<system time 10> process 2 is running
<system time 11> process 2 is running
<system time 12> process 2 is running
<system time 13> process 2 is running
<system time 14> process 2 is running
<system time 15> process 2 is running
<system time 16> process 2 is running
<system time 17> process 2 is running
<system time 18> process 2 is running
<system time 19> process 2 is finished.......
<system time 19> process 3 is running
<system time 20> process 3 is running
<system time 21> process 3 is running
<system time 22> process 3 is running
<system time 23> process 3 is running
```

<system time 24> process 3 is finished.....
<system time 24> process 4 is running
<system time 25> process 4 is running
<system time 26> process 4 is running
<system time 27> process 4 is running
<system time 28> process 4 is finished.....
<system time 28> process 5 is running
<system time 29> process 5 is running
<system time 30> process 5 is running
<system time 31> process 5 is running
<system time 32> process 5 is running
<system time 33> process 5 is running
<system time 34> process 5 is finished.....
<system time 34> process 6 is running
<system time 35> process 6 is running
<system time 36> process 6 is running
<system time 37> process 6 is running
<system time 38> process 6 is running
<system time 39> process 6 is running
<system time 40> process 6 is running
<system time 41> process 6 is finished.....
<system time 41> All processes finish .....
==========
Average cpu usage : 100.00 %
Average waiting time : 14.17
Average response time : 14.17
Average turnaround time: 21.00
==========
% proj2 input.1 RR 2
Scheduling algorithm: RR
Total 6 tasks are read from "input.1". press 'enter' to start...
==========
<system time 0> process 1 is running
<system time 1> process 1 is running
<system time 2> process 2 is running
<system time 3> process 2 is running
<system time 4> process 1 is running
<system time 5> process 1 is running
<system time 6> process 3 is running
<system time 7> process 3 is running
<system time 8> process 2 is running
<system time 9> process 2 is running
<system time 10> process 1 is running
<system time 11> process 1 is running

<system time 12> process 4 is running
<system time 13> process 4 is running
<system time 14> process 3 is running
<system time 15> process 3 is running
<system time 16> process 5 is running
<system time 17> process 5 is running
<system time 18> process 6 is running
<system time 19> process 6 is running
<system time 20> process 2 is running
<system time 21> process 2 is running
<system time 22> process 1 is running
<system time 23> process 1 is running
<system time 24> process 4 is running
<system time 25> process 4 is running
<system time 26> process 4 is finished.....
<system time 26> process 3 is running
<system time 27> process 3 is finished.....
<system time 27> process 5 is running
<system time 28> process 5 is running
<system time 29> process 6 is running
<system time 30> process 6 is running
<system time 31> process 2 is running
<system time 32> process 2 is running
<system time 33> process 1 is running
<system time 34> process 1 is running
<system time 35> process 1 is finished.....
<system time 35> process 5 is running
<system time 36> process 5 is running
<system time 37> process 5 is finished.....
<system time 37> process 6 is running
<system time 38> process 6 is running
<system time 39> process 2 is running
<system time 40> process 2 is finished.....
<system time 40> process 6 is running
<system time 41> process 6 is finished.....
<system time 41> All processes finish .....
==========
Average cpu usage : 100.00 %
Average waiting time : 22.50
Average response time : 4.00
Average turnaround time: 29.33
==========