CPSC 3500 Computing Systems

Project 2: CPU Scheduling Simulator

Assigned: 2:05PM, Thursday, 01/24/2019

Due: 11:59PM, Saturday, 02/02/2019

Note: you can form a group size of 2 for this project. You may also take it as an individual project!

1. Descriptions

In this project, you will implement a CPU scheduler that **simulates** three scheduling algorithms:
1) First Come First Served (**FCFS**)
2) Round Robin (**RR**)
3) Shortest Remaining Time First (**SRTF**).

Since the CPU scheduler is a simulator, it does not require any actual process creation or execution. Context switching time is also ignored.

The simulator first reads process information from an input file and stores all data in memory. Then it starts simulating scheduling algorithms in a **time-driven** manner (you can simulate a "system clock" that ticks every time unit. Simply put, it is an integer counter staring at 0). At each time unit, it adds any newly arrived process(es) based on **arrival_time** into the ready queue and calls a specific scheduler algorithm (FCFS, RR or SRTF) in order to select an appropriate process from the ready queue (e.g., if the scheduling algorithm is preemptive or current process finishes). When a process is chosen to run, the simulator prints out a message indicating what process ID is chosen to execute for this time slot. If no process is running, it prints out an "idle" message. Before advancing to the next time unit, the simulator should update information associated with a process and ready queue status. Even though you do not need to create and execute a real process, a PCB-like data structure associated with each process is highly suggested as it allows you to move processes to different queues based on their process states.

Process information

The process information will be read from an input text file. The text file includes N lines if there are N processes. Each line has three data fields, separated by space(s).

```
pid     arrival_time     burst_time
```

All the fields are integer type where

pid is a unique numeric process ID. The **pid** field may have leading space(s).
arrival_time is the time when the process arrives in the unit of milliseconds.
burst_time the is the CPU time requested by a process, in the unit of milliseconds.

Command-line Usage and Examples

You run your simulator ***schedsim*** using the following commands:

Usage: **`schedsim input_file [FCFS|RR|SRTF] [time_quantum]`**

Where **input_file** is the file name with process 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 both are **preemptive**. **The last argument is required only for RR.** (See more examples below)

Examples:

`./schedsim input.1 FCFS`

- FCFS scheduling with the data file "input.1"

`./schedsim input.1 RR 2`

- Simulate RR scheduling with time quantum 2 milliseconds (the 4th parameter is required even for quantum 1 millisecond) with the data file "input.1"

`./schedsim input.1 SRTF`

- Simulate SRTF scheduling with the data file "input.1"

2. 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**. Context switching time is ignored in the simulator.

- **Implement scheduling algorithm for FCFS, RR, and SRTF.** The program should schedule processes and print progress of process every unit time (millisecond) as shown in **Appendix sample outputs**.

- **Print statistical information**. As soon as all processes are completed, the program should display the following stats and then exits:
  1) average waiting time (show 2 digits after float point)
  2) average response time (show 2 digits after float point)
  3) average turnaround time (show 2 digits after float point)
  4) CPU usage (show 2 digits after float point)

3. Suggestions

Before you implement the three scheduling algorithms, you should understand how each scheduling algorithm works (Design goes first!). Given a set of processes with scheduling information, you should be able to use Gantt charts to simulate the scheduling algorithm and compute the scheduling stats (e.g.,

average waiting time, response time, turnaround time and overall CPU usage), as shown in the textbook and slides.

Using a PCB-like (does not have to be exactly same) data structure for each process and different queues (i.e., new, running, ready) to manage processes based on their states definitely make life much easier.

4. Submission

For a group project, only one submission is required.

Before submission, you must make sure that your code is working on Linux server cs1. You must run the submission command on cs1. You must make sure that your Makefile works properly!

The following files must be included in your submission:

- Readme
- *.h: all .h files
- *.c/cpp: all .c/cpp files
- Makefile

You should create a package p2.tar including the required files as specified above, by running the command:

```
tar -cvf p2.tar Readme *.h *.cpp Makefile
```

Then, use the following command to submit p1.tar:

```
/home/fac/zhuy/class/SubmitHW/submit3500 p2 p2.tar
```

If submission succeeds, you will see the message similar to the following one on your screen:

```
==========Copyright(C)Yingwu Zhu==========
Wed Jan 17 21:53:58 PST 2018
Welcome testzhuy!
You are submitting array.cpp for assignment p2.
Transferring file.....
Congrats! You have successfully submitted your assignment! Thank you!
Email: zhuy@seattleu.edu
==========
```

You can submit your assignment multiple times before the deadline. Only the most recent copy is saved.

The assignment submission will shut down automatically once the deadline passes. You need to contact me for instructions on your late submission. Do not email me your submission!

5. Grading Criteria

Label	Notes
2a. Compilability & Complete submission (1 pt)	--- Your Makefile works properly. --- All required files are included in your submission.
2b. Readme, and Coding Format & Style (1 pt)	Readme is a text file, including: • Strengths • Weakness (any problems) • Team member names and their respective contributions, if it is a team project Coding format & style: • Clean, well-commented code.
2c. Functionality (7 pts)	The CPU scheduling algorithms behave as specified. 1. 3 scheduling algorithms work properly 2. Required messages and stats are displayed
2d. Resource management, Outputs (1 pt)	No memory leaks (if applicable). Output: • No debugging/testing messages • No messy messages
2e. Overriding policy	If the code cannot be compiled or executed (segmentation faults, for instance), it results in zero point. If the submission is incomplete (e.g., missing files), it results in zero point.
2f. Late submission	Please refer to the late submission policy on Syllabus.

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

For FCFS, RR, and SRTF

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

% ./schedsim
Usage: schedsim  input_file  FCFS|RR|SRJF  [quantum]

% ./schedsim   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 ...................
============================================================
CPU usage : 100.00 %
Average waiting time : 14.17
Average response time : 14.17
Average turnaround time: 21.00
============================================================

% ./schedsim   input.1   RR 2
Schdeuling 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 ...................
============================================================
CPU usage : 100.00 %
Average waiting time : 22.50
Average response time : 4.00
Average turnaround time: 29.33
============================================================

% ./schedsim   input.1   SRTF
Scheduling algorithm: SRTF
Total 6 tasks are read from "input.1". press 'enter' to start...
============================================================
<system time 0> process 2 is running
<system time 1> process 2 is running
<system time 2> process 2 is running
<system time 3> process 3 is running
<system time 4> process 3 is running
<system time 5> process 3 is running
<system time 6> process 3 is running
<system time 7> process 3 is running
<system time 8> process 3 is finished.......
<system time 8> process 4 is running
<system time 9> process 4 is running
<system time 10> process 4 is running
<system time 11> process 4 is running
<system time 12> process 4 is finished.......
<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 finished.......
<system time 18> process 5 is running
<system time 19> process 5 is running
<system time 20> process 5 is running
<system time 21> process 5 is running
<system time 22> process 5 is running
<system time 23> process 5 is running
<system time 24> process 5 is finished.......
<system time 24> process 6 is running
<system time 25> process 6 is running
<system time 26> process 6 is running
<system time 27> process 6 is running
<system time 28> process 6 is running
<system time 29> process 6 is running
<system time 30> process 6 is running
<system time 31> process 6 is finished.......
<system time 31> process 1 is running
<system time 32> process 1 is running
<system time 33> process 1 is running
<system time 34> process 1 is running
<system time 35> process 1 is running
```

```
<system time 36> process 1 is running
<system time 37> process 1 is running
<system time 38> process 1 is running
<system time 39> process 1 is running
<system time 40> process 1 is running
<system time 41> process 1 is finished.......
<system time 41> All processes finish .....................
============================================================
CPU usage : 100.00 %
Average waiting time : 10.50
Average response time : 9.00
Average turnaround time: 17.33
============================================================
```