CPSC 3500 Computing Systems
Winter 2018

Programming Assignment 1: CPU Scheduler Simulator
Due: Friday, January 26th at 2:05pm

Description

In this programming assignment, you will create a simulator that will simulate the execution of different CPU scheduling algorithms discussed in class.

In particular, your simulator must implement the following scheduling algorithms

a. Shortest Job First (SJF)
b. Shortest Remaining Time First (SRTF)
c. Nonpreemptive Priority (NP)
d. Preemptive Priority (PP)

Program Flow

1. The program will be invoked by the command line. The command line will consist of two command line parameters in the order given:
- *Input file:* Name of the input file (more details on the input file format below)
- *Scheduling algorithm:* Either 'SJF', 'SRTF', 'NP', or 'PP'.

Example command lines are given later in the document.

2. Open the input file. The input file consists of lines with the following fields:
- *Process id:* A unique process identifier.
- *Arrival time:* Time (in milliseconds) when the task arrives.
- *Burst time:* Time (in milliseconds) the task takes on the CPU.
- *Priority*: A nonnegative integer representing the priority. Lower numbers mean higher priorities – zero is the highest priority.

More information about the input file format:
- All fields are integers and separated by one or more spaces.
- The process id is guaranteed to be unique.
- You may not assume anything about the order of line in the input file. The lines are not necessarily ordered by process id and/or their arrival times.

3. Carry out the simulation. Print out a message each time a different process is scheduled on the CPU like this:

```
Time 30 Process 4
```

If the CPU becomes idle, print out this message:

```
Time 50 Idle
```

4. Once there are no more processes to execute, print out the following statistics in the specified format:
   - *CPU utilization:* Display as a percentage, rounded to the nearest integer, from 0-100%.
   - *Average waiting time:* Display as a floating point number with at most two digits past the decimal point.
   - *Worst case waiting time:* Display as an integer.

5. Exit the program.

Additional Requirements

- You must use the command line to get the file name, scheduling algorithm, and time quantum (when needed). Do NOT prompt the user for these – submissions that do this will incur a 10 point penalty. If you need help on using command line parameters in C++, please consult the brief tutorial posted on the course website.
- If there is a tie, select the process with the lower process id.
- Make sure the statistics printed out at the end of the program are in the properly specified format.

Error Checking

- Your program must detect invalid command lines. If the command line is invalid, abort the program with an error message.
- If the input file does not exist (or does not open), abort the program with an error message.

Note: Be sure to test these scenarios! They will be tested during grading.

Assumptions

- You may assume the input file is well formatted:
  - Each line consists of three integers.
  - The process id is unique for each line (and is greater than or equal to zero).
  - The arrival time is greater than or equal to zero.
  - The burst time is strictly greater than zero.
  - The priority is greater than or equal to zero.
- The input file will contain at least one line. Beyond this, you cannot make any assumptions on how long the input file is.
- Assume that context switch overhead is zero.

Note: You do NOT have to check that the input file meets these assumptions. The instructor will not use improperly formatted or empty files during grading.

Implementation Hint

One implementation strategy is to implement a ready queue structure (keeping in mind that the ready queue does not necessarily behave like a queue in all cases) and a separate queue that stores the processes that are yet to arrive. You may use STL if desired.

Example

Here is a sample input file (location on cs1: `/home/fac/elarson/cpsc3500/pa1/sample.txt`) followed by the output for each of the scheduling algorithms.

```
0 0 5 3
1 10 20 3
2 10 10 5
3 15 3 2
4 20 30 1
5 25 5 4
```

Output for: `./sim sample.txt SJF`

```
Time 0 Process 0
Time 5 Idle
Time 10 Process 2
Time 20 Process 3
Time 23 Process 1
Time 43 Process 5
Time 48 Process 4
CPU Utilization: 94%
Average waiting time: 10.67
Worst-case waiting time: 28
```

Output for: `./sim sample.txt SRTF`

```
Time 0 Process 0
Time 5 Idle
Time 10 Process 2
Time 15 Process 3
Time 18 Process 2
Time 23 Process 1
Time 25 Process 5
Time 30 Process 1
Time 48 Process 4
CPU Utilization: 94%
Average waiting time: 8.17
Worst-case waiting time: 28
```

Output for: `./sim sample.txt NP`

```
Time 0 Process 0
Time 5 Idle
Time 10 Process 1
Time 30 Process 4
Time 60 Process 3
Time 63 Process 5
Time 68 Process 2
CPU Utilization: 94%
Average waiting time: 25.17
Worst-case waiting time: 58
```

Output for: `./sim sample.txt PP`

```
Time 0 Process 0
Time 5 Idle
Time 10 Process 1
Time 15 Process 3
Time 18 Process 1
Time 20 Process 4
Time 50 Process 1
Time 63 Process 5
Time 68 Process 2
CPU Utilization: 94%
Average waiting time: 21.50
Worst-case waiting time: 58
```

Grading

The assignment will be graded in accordance with the programming assignment expectation handout.

Grading is broken down as follows:

Shortest Job First (SJF)	16 points
Shortest Remaining Time First (SRTF)	16 points
Nonpreemptive Priority (NP)	16 points
Preemptive Priority (NP)	16 points
CPU Utilization Stat	5 points
Average Waiting Time Stat	5 points
Worst Case Waiting Time Stat	5 points
Stats formatting	5 points
Error Checking	6 points
Style	10 points
TOTAL	100 points

Functionality is graded for each of the four scheduling algorithms as follows (16 points each):

- all tests pass (no penalty)
- all tests pass except one -4
- all tests pass except 2-3 tests (sample input passes) -7
- all tests pass except 2-3 tests (sample input fails) -10
- at least one test is correct but more than 3 fail -13
- no tests are correct -16

Functionality is graded for each of the three stats as follows (5 points each):

- all tests pass (no penalty)
- all tests pass except one -2
- all tests pass except 2-3 tests -3
- failed all the tests for one scheduling algorithms -3
  (but the other algorithms were correct)
- failed more than 3 tests spanning more than one algorithm -5

The stats formatting is graded as follows (5 points total):

- CPU utilization displayed as a percentage, rounded to the nearest integer, from 0-100%?
  YES: no penalty NO: -2
- Average waiting time displayed as a floating point number with at most two digits past the decimal point?
  YES: no penalty NO: -2
- Worst case waiting time displayed as an integer?
  YES: no penalty NO: -1

Error checking is graded using three tests. Each test is two points and is graded as follows:

- Suitable error message is printed and program aborts (no penalty)
- Suitable error message is not printed -1
- Program does not abort -1

Style will be inspected manually based on the guidelines within the programming assignment expectation handout. Meeting all guidelines will result in full credit.

If the submission does not use the command line but instead prompts the user for the file name, your program will receive an additional 10 point penalty.

Submitting your Program

On `cs1`, run the following script in the directory with your program:

```
/home/fac/elarson/submit/cpsc3500/pa1_submit
```

This will copy all source code files (must have a `.cpp` or `.h` suffix). in the current directory to a directory that accessible by the instructor.

In addition it will look for a makefile (with the file name `Makefile`). If a makefile is not present, the submission script will reject your program. The submission program will then attempt to compile your program using `make`. The executable produced by the `Makefile` must be called `sim`. If `make` fails, the submission program will reject your submission. If you are unfamiliar with using make and/or creating make files, please consult the tutorial on the class website. A sample `Makefile` is available on `cs1` at:

```
/home/fac/elarson/cpsc3500/pa1/Makefile
```

Failure to submit a successful submission before the due date and time will result in a zero. Programs that fail to compile, even if make is successful, will result in a zero. Only the last assignment submitted before the due date and time will be graded. *Late submissions are not accepted and result in a zero.*