CS 332-104 Project   Spring 2017

Multi-Processing Scheduling Algorithm Simulation

For this semester's project, you will be simulating a few of the Uniprocessing scheduling algorithms by running them simultaneously on different threads. The goal here is to parse input from a text file, spawn 4 threads for each time unit, and run each algorithm on one of those threads.

Step 1: Generate your input text file(s)

Your input text file should be created manually and be in the following .csv format:

Process ID, execution time required, time entered
A,2,1
B,3,4
C,3,4 and so on…

You should create one file for each example you run, using the examples found in the Powerpoint slides **as well as** 2 examples you create yourself. I'll expect a comparison between your algorithm's result and the actual result, so make sure you do them all by hand as well.

Step 2: Create code to run algorithms simultaneously

The concept of these algorithms is that you look at what time a process comes in, how long it will take, and determine how it should be executed. For this reason, you'll need to create a program that simulates these "units" of time. This can just be a simple while loop, where each iteration is a time unit, and will end when all processes from the input file are "completed".

During each iteration, you will have to spawn threads that perform any necessary calculations for each algorithm to figure out what process should be running in the current time unit. You can keep an array or other data structure that tracks your current process for each scheduling algorithm, and pass that information to each thread on each iteration of the loop.

Step 3: Add the algorithm logic for each thread to call

The 4 algorithms you will be simulating are:

1. Round-robin
2. HRRN (Highest response ratio next)
3. Feedback
4. Shortest process next

For each iteration of the while loop, your output should look something like this:

```
RR
A, B, B, C , C
HRRN
A, C,B,D
Feedback
C,B,D,A
SPN
A,C,D,D and so on…
```

Each loop's print statement should print out more content on each line than the last, until all processes have "finished", i.e. the full schedule for each algorithm has been printed out. The number of iterations will equal the number of time units, AKA the total time it takes to complete all processes one-by-one.

Rubric

Languages Permitted: Python,Java,C,or C++
2 pts: Input files are properly formatted
2 pts: Output is properly formatted
4 pts: Each algorithm has been successfully implemented
2 pts: Multi-threading is functional based on the above requirements
2 pts: Work is shown for solving each example by hand and comparing to your code's results. Screenshots of successful execution must also be provided.
3 pts: Code has sufficient comments to explain how each major block works and what it's purpose is

15pts total credit, due the date of your final exam via Moodle submission

Keep in mind this rubric is a general guideline on how to get the most points for this project. If, for example, you give me code with no comments, no work shown, and no explanation of the output, you will receive a 0. On the other hand, if you give me well-documented code, show your work, but you can't quite get one of your algorithms to get the correct results, you might only lose 1 or 2 points. The more you document and the more work you show, the higher potential you have for points.

Cheating of course will not be tolerated. If I find anybody copying code from the Internet or their fellow classmates, all parties will receive a 0 and be subject to the NJIT Code of Conduct.

My office hours will be Mondays from 6-7pm in the PC mall under the parking deck. If you have any questions or concerns and cannot make that day, shoot me an email and we will work out alternative office hour arrangements. Best of luck and have fun!

- Ian Hall (igh2@njit.edu)