Assignment 4

Georgiy Krylov

October 14, 2024

ASSIGNMENT IS TO BE COMPLETED INDIVIDUALLY BY ALL STUDENTS!

1 Description

This assignment is to practice scheduling algorithms on the example of the Round-Robin algorithm. **The assignment is Due by 11:59 p.m. (midnight) on Friday, 25th of October 2024, one minute before Saturday.**

2 Task

In this assignment you're to implement a simulation of the **Round Robin** scheduling algorithm. It is a preemptive algorithm that makes scheduling decisions when a time quantum allotted for the current job expires, or the CPU has finished executing the current job. The next job to be scheduled is selected from the list of already known jobs following the Round Robin algorithm. In the case when two jobs have the same arrival time, the order of input is the tiebreaker.

CPU time is an integer that starts from one and is used for scheduling as well as for output.

Your program to read the jobs : username, job ID, arrival time and duration.

- Username is a sequence of characters not longer than 100 symbols.
- Job IDs are unique capital characters of English alphabet.
- Duration is a positive nonzero integer.

The jobs should be read before starting any simulation (hint: main thread can do that). Your program should print what the CPU was executing during each time unit (see in section 3), starting from one.

2.1 Algorithm

1. Every unit of time, see if the CPU time matches a job's arrival time, to add the jobs to the scheduling queue from the queue containing the inputs.
2. Every unit of time, your CPU should check if the current job was just finished (remaining time is zero)
   (a) If that is the case
      i. The summary should be updated
      ii. The job from the scheduling queue should be selected for execution.
   (b) If that is not the case, your CPU should check if the quantum of time has expired.
      i. The job from the queue head should be scheduled next (in the FIFO order).
3. If there are no jobs currently on CPU
   (a) The CPU should print current time and "-"
4. Otherwise (if there are jobs on the CPU)
   (a) The CPU should then print the current time, and the job that is currently assigned to CPU.
   (b) The CPU should decrease the current job's remaining time
5. The CPU should increase its current time and go to Step 1 again, until all jobs are finished (queue is empty).

In the end, the program should output a per-user summary. The summary should contain information about all users and when the last job submitted by this user was finished. Remember, you cannot execute jobs that did not arrive yet. In summary, users should be sorted by the order of their arrival.

2.2 Design Suggestions

This assignment simulates behavior of one processor, so it is not threaded. The next one will be focused on multiple processors, so it will be using multiple threads simulating CPUs, potentially with jobs migration and processor affinity. To make it easier to get ready for the next assignment, consider using your data structures (if it was a LL) from Assignment 1 for solving this assignment. Hints:

- To turn a linked list into a queue, you can implement an enqueue method that add elements to the very end of the linked list.
- You can use your `hasItem` function to avoid creating duplicate usernames in your summary queue (I'd advise using three queues, one to store the input, one for making scheduling decisions, and the third one for summary).

- You can remove the job from the ready queue and enqueue it again to simulate FCFS order.

3 Input/Output format

The first line of input will be the time quantum. The jobs are always sorted in the order of their arrival See the attached sample_intput.txt and sample_output.txt for clarification.

4 Submission instructions

Please submit your .c file(s) and .h files (if any) to D2L Assignment box. Make sure your code compiles and runs. Make sure your code follows the specified input/output format. You must use C programming language to solve this assignment. Important to note in this course: if your program produces correct output, it doesn't necessarily mean you have properly managed the resources and penalties are possible. This assignment focuses on threads management and communication, and therefore the TA will be asked to make sure the threads are properly created and terminated.

NOTE: THE INPUT AND OUTPUT OF YOUR PROGRAM IS SPECIFIED SUCH THAT THE MARKER CAN AUTO TEST YOUR SUBMISSIONS. PLEASE FOLLOW THE SPECIFICATIONS! INPUT WILL BE READ VIA stdin (E.G. KEYBOARD). OUTPUT SHOULD BE PRINTED TO stdout (E.G. MONITOR).