Assignment 3

Georgiy Krylov

February 10, 2022

ASSIGNMENT IS TO BE COMPLETED INDIVIDUALLY BY ALL STUDENTS!

1 Description

This assignment to introduce the class to the process of POSIX threads creation, parameters passing, barrier synchronization. **The assignment is Due by 11:59 p.m. (midnight) on Thursday, 17th of February 2022.**

2 Task

In this assignment you're to implement a simulation of **Shortest Remaining Time First** scheduling algorithm. It is a preemptive algorithm that makes scheduling decisions when new jobs arrive or the CPU is finished execution of the current job. The next job to be scheduled is selected from the list of already known jobs (i.e. the jobs that have arrival time less or equal the current CPU time). In the case when two jobs have the same remaining time, the order of input is the tiebreaker. **CPU time** is an integer that starts from zero and is used for scheduling as well as for output.

Your program to read the jobs : user name, job id, arrival time, and duration. Job names 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 zero.

2.1 Algorithm

1. 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 with shortest remaining time and arrival time less or equal to the current time should be scheduled.

(b) If that is not the case, your CPU should check if there are new jobs arrived that have shorter (strictly shorter, not shorter or equal) remaining time.

i. The job with shortest remaining time and arrival time less or equal to the current time should be scheduled.

2. If there is no jobs currently on CPU

   (a) The CPU should print current time and "-"

3. Otherwise (if three are jobs on the CPU)

   (a) The CPU should then print the current time, and the job that is currently assigned on CPU.

   (b) The CPU should decrease the current job's remaining time

4. 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 when the last job by this user was finished. 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 linked lists from Assignment 0 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 contains function to avoid creating duplicate user names in your summary queue (I'd advice to use two queues, one for making scheduling decisions, another for summary).
- You can write a function that will find shortest remaining time job by iterating through all elements in the list.
- You can use modified version of find and replace function to reduce the remaining time on an instance of a job.
- You can use your delete function to remove jobs that were finished by id (the letter).

3 Input/Output format

See the attached sample_intput.txt and sample_output.txt for clarification.

4 Submission instructions

Please submit your C file 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).