CSC 343 Operating Systems, Spring 2023, Dr. Dale E. Parson

Assignment 3, Implementing Multilevel Queue context scheduling.

This assignment is due via **make turnitin** from the MultiQCPUschedSpring2023 directory by **11:59 PM on Friday April 14**. There is a 10% penalty for each day it is late, and I will not accept solutions after I go over my solution in class. I will go over this handout on March 28. The recent class Zoom recordings augment this handout.

The goal of this assignment is to start with my supplied preemptive round-robin and shortest-remaining-time-first context schedulers as starting points for you to implement multilevel context-scheduling queues that send iobound processes to one queue and preemptively-scheduled set of contexts, and send cpubound processes to a different FIFO queue for non-preemptive first-come-first-served context scheduling. There is a README.txt file with questions for you to answer after you have completed the code. **Answers in README.txt are worth 30% of this assignment, so remember that working code is not the end of the requirements**.

Perform the following steps to get my handout. You will code and test on **mcgonagall**, to which you can **ssh mcgonagall** from acad.

```
cd  $HOME               # or start out in your login directory
mkdir  OpSys  # All of this semester's work goes under here, skip if you did it before.
cd  ./OpSys
cp  ~parson/OpSys/MultiQCPUschedSpring2023.problem.zip
MultiQCPUschedSpring2023.problem.zip
unzip MultiQCPUschedSpring2023.problem.zip
cd ./MultiQCPUschedSpring2023
make clean test
```

Testing passes fcfs.stm, rr.stm, and srtf.stm but fails MLQrr.stm and MLQsrtf.stm which you must complete. Some subsequent failures may hang the compiler or simulation, requiring a control-C to abort. Successful **make clean test** finishes in about six seconds.

STEPS:

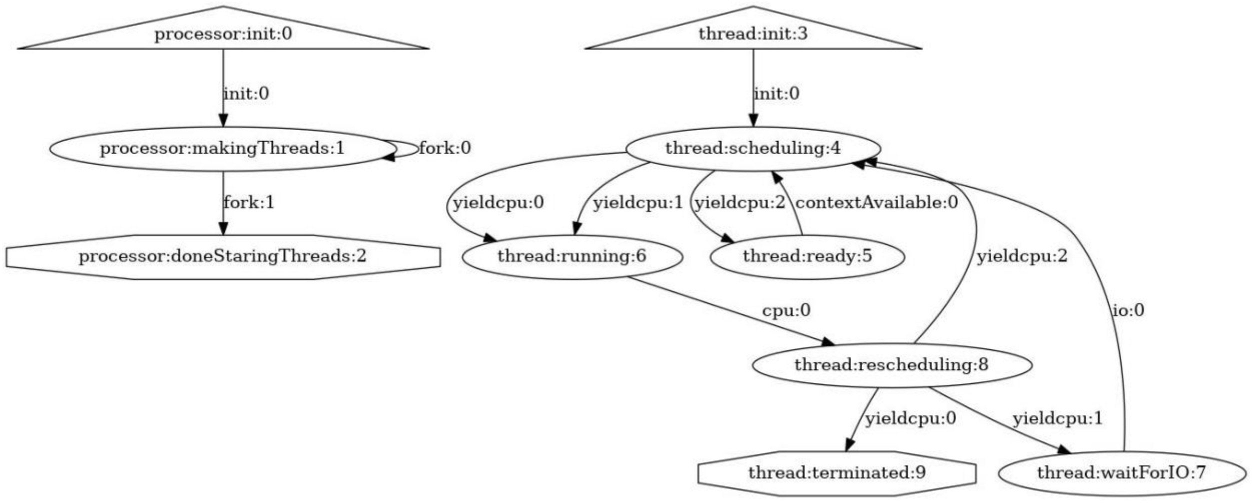
1. Edit **MLQrr.stm** and complete all STUDENT-tagged steps that we will go over March 25.
2. Run **make testMLQrr** to test this step.
3. Edit **MLQsrtf.stm** and complete all STUDENT-tagged steps that we will go over March 25.
4. Run **make testMLQsrtf** to test this step.
5. Run **make clean test** after everything works and after any subsequent change, and then **make turnitin** as before by the due date.

Any time a COMPILE succeeds, you can look at the graph for your state machine by running **make graphs** and then inspecting https://kuvapcsitrd01.kutztown.edu/~STUDENT/MLQrr.jpg or https://kuvapcsitrd01.kutztown.edu/~STUDENT/MLQsrtf.jpg, where STUDENT is your login ID. If you can't get at it with a browser this way, use WinSCP or FileZilla to copy the JPEG file from your project directory to your local machine. Below are the final, correct graphs.

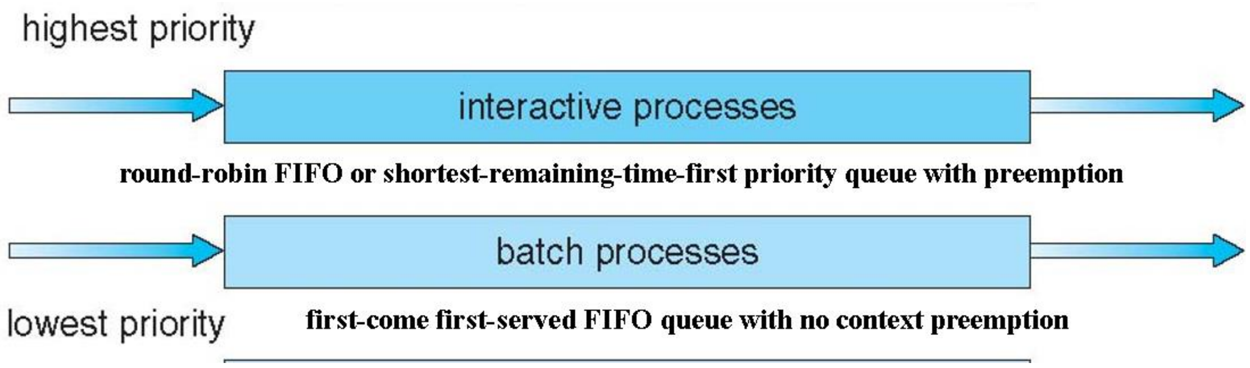
Once **make clean test** passes, **ANSWER THE QUESTIONS IN FILE README.txt** included in this project directory. Follow all instructions in README.txt.

Finally, turn it in by entering **make turnitin** and following the prompt as in previous projects.

I will distribute grades via email before the next class after the due date.



MLQrr.stm and MLQsrtf.stm have the same graph but different queuing for iobound processes


highest priority
interactive processes
round-robin FIFO or shortest-remaining-time-first priority queue with preemption
batch processes
lowest priority
first-come first-served FIFO queue with no context preemption

Adapted from textbook Chapter 6 slide

IO-bound process threads schedule via the interactive processes, preemptive queue, borrowing the batch queue when all IO-bound processors are busy and there is a CPU-bound processor free.

CPU-bound process queue schedule via the batch processes, non-preemptive FCFS / FIFO queue.

There are two queues on for the MLQ STMs. Single-queue rr.stm and srtf.stm use a single preemptive scheduler for all process threads.