**Design and Algorithm**

This program implements a multilevel feedback queue scheduling algorithm. And compares execution stats of the multilevel feedback queue scheduling and round robin scheduling. All new threads start of by executing in queue0 which has a time quantum of timeslice/2. If a thread does not finish executing in queue0, it is then moved to queue1. Queue1 has a time quantum of timeslice. A thread is executed in a time of timeslice/2 and periodically needs to check queue0 to make sure no new thread have arrived. All processes need to be executed in higher level queues before lower level queues can run. If a thread does not complete in queue1, it is then moved to queue2 which has a time quantum of timeslice\*2. After running the thread for timeslice/2 the higher level queues are checked for new processes to execute before the thread in queue2 can finish. If a thread has not finished execution in queue2, then the thread is removed from queue2 and added to the tail end of queue2.

**Flow Chart of MFQS Algorithm**

Diagram

Description automatically generated

**Outputs**

**Running Test2 with MFQS**

Text

Description automatically generated

**Running Test2 with RR**

A screen shot of a computer

Description automatically generated with low confidence

**Running Test2 with pri**

A black screen with white text

Description automatically generated with low confidence

**Performance**

**Running Test2b with MFQS**

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generatedText

Description automatically generated

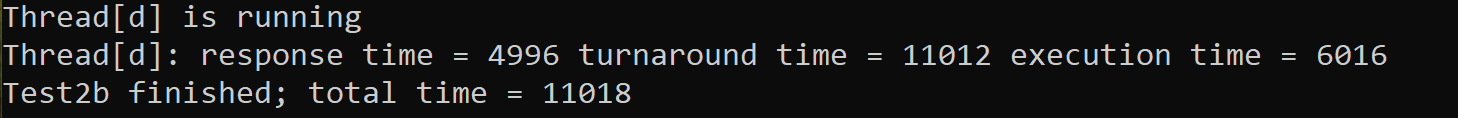
**Running Test2b with RR**

Graphical user interface, text

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidenceText

Description automatically generated



Both MFQS and RR performed about the same. It would make sense for MFQS to perform better than RR as there are multiple queues allowing threads with shorter CPU burst time to execute first. It could make sense that threads in queue 2 would wait a long time to execute but should still perform better than RR as RR is based on a first-come-first-serve algorithm, just with preemption.

**Files**

**Schedular\_rr:** contains the implementation of a round robin scheduling algorithm.

**Schedular\_mfq:** contains the implementation of a multilevel feedback queue scheduling algorithm.

**Schedular\_pri:** contains the implementation of a priority scheduling algorithm.

**Test2 and Test2b:** spawn 5 child threads from TestThread2 and TestThread2b. respectively, each names Thread[a], Thread[b], Thread[c], Thread[d], and Thread[e]. Threads spawned from ThreadTest2b, print out a heartbeat message every 0.1 seconds in the form “Thread[name] is running.” tests spawned from TestThread2 do not print the heartbeat message. At the end, both tests print out execution stats which include response time, turnaround time, and execution time. These execution stats are used for performance comparison between the 2 scheduling algorithms.

**Compilation**

The program is compiled using Linux.

* Start of by locating the program in your directory
* cp filename.java Scheduler.java where filename is one of the following
  + Sheduler\_mfqs.java
  + Sheduler\_rr.java
  + Scheduler\_pri.java
* javac \*.java
* java Boot
* Load whichever test from the shell with the command
  + l Test2
  + l Test2b
* once test is completed, exit the shell by typing q