Sean Skinner
CptS 370
3/15/2022
Program 3

Algorithm for MFQS
- To start I replaced the single vector representing the queue to a list of vectors representing the 3 levels of the MFQS. Q0 being highest priority followed by Q1 then finally Q2.

- Then I went into each constructor and modified them to instantiate the three queues whenever a Scheduler is created.

- Next I edited the getMyTcb() method to iterate through all the queues since there is no longer only a single queue being used.

- I also edited the addThread() method to add threads to queue 0 since that is where all the TCBs will start before being pushed to other threads if they do not execute in time.

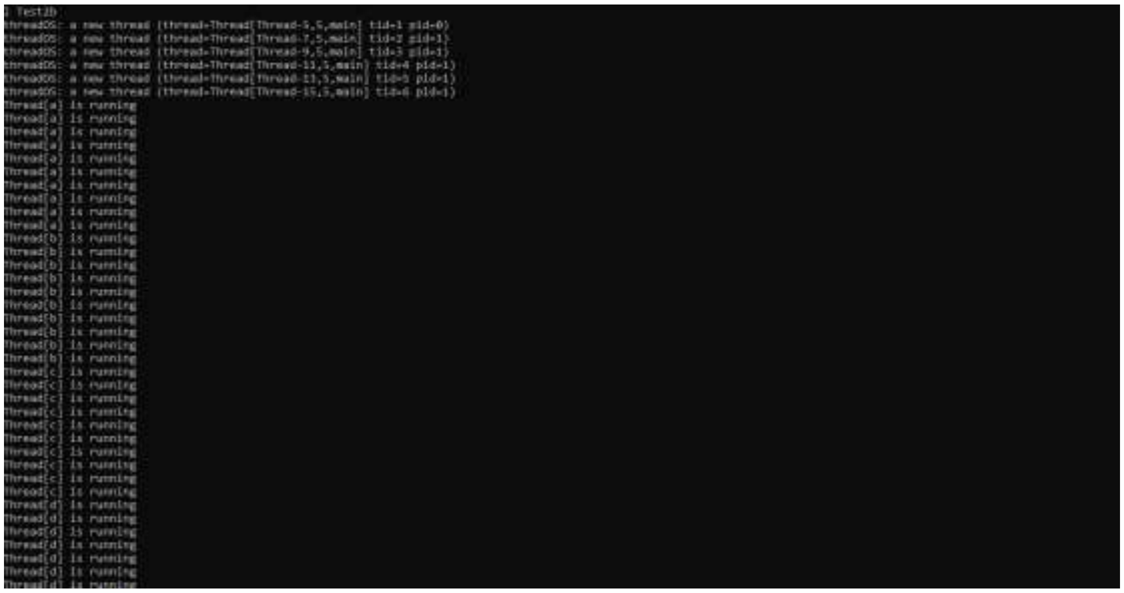
- Then I started to work on the run() method and in the while loop used if/else statements to check which queues were empty, if q0 was not empty it would work on q0, if q0 was empty and q1 was not empty it would work on q1, and if q0 and q1 were empty it would work on q2.

- Next Whenever threads were resumed I decided to do two things, first was to sleep the thread for one half of the time slice, as each queue executes a TCB for that long then looks at the other queues with higher priority to see if they have TCBs to execute, if not it will continue to execute in 500ms intervals. However, the other thing I decided to keep tack of was the number of times a queue had gone though an interval (for queues 1 and 2 as queue 0 would not need this feature) Later when I would check how many iterations a TCB had gone through on its current queue I could tell when to remove it and add it to another queue. If it reached the synchronized portion of the run code in queue 0 it would always be sent to queue1, if queue 1 had a multiple of 2 as its counter I would remove it from queue 1 and send it to queue 2, and if queue2's counter was a multiple of 4 it would send the TCB to the back of queue 2.

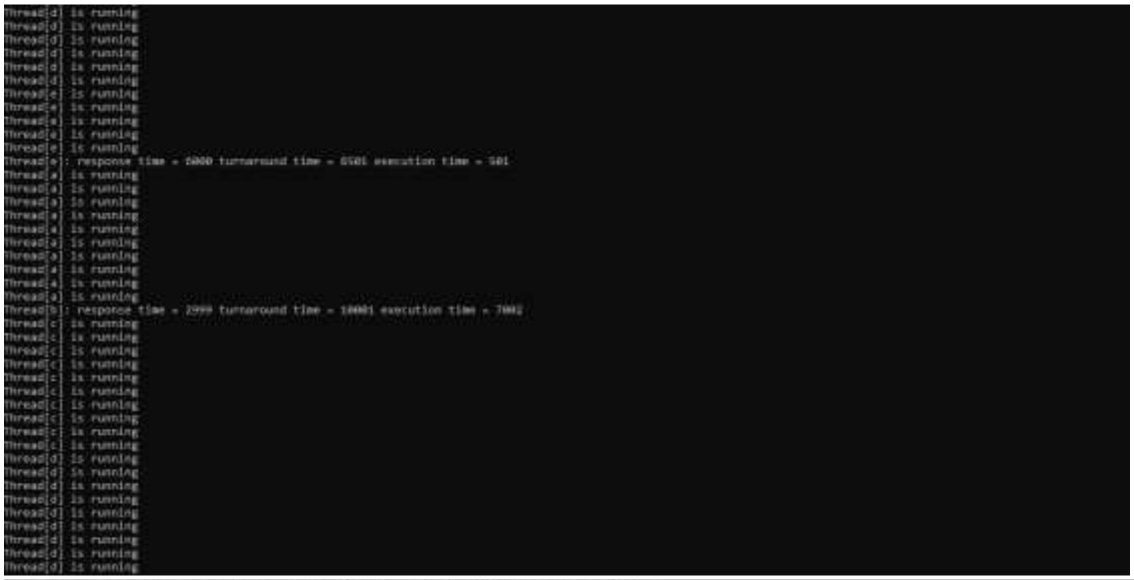
- Finally in order to send TCBs to different queuese I created a switchQueue() helper function that took in the current queue the TCB was in and the TCB itself, If the current queue was 0 it sent the TCB to queue 1, if the current queue was 1 it sent the TCB to queue 2, and if the current queue was 2 it sent the TCB to the back of queue2.

(NOTE: for the outputs I did notinclude all the outputs, instead I focused on including all the threads being terminated.)

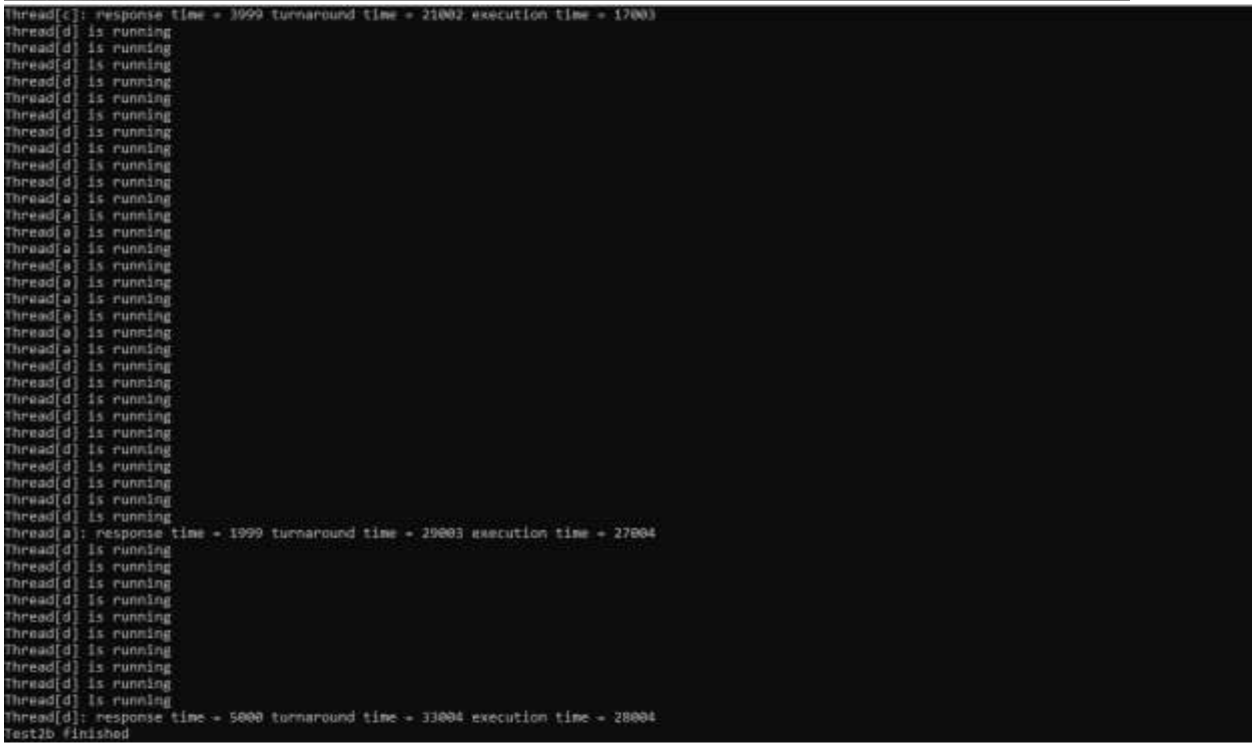
Round Robin Output:



```
l Test2b
threadOS: a new thread (thread=Thread[Thread-5,5,main] tid=1 pid=0)
threadOS: a new thread (thread=Thread[Thread-7,5,main] tid=2 pid=1)
threadOS: a new thread (thread=Thread[Thread-9,5,main] tid=3 pid=1)
threadOS: a new thread (thread=Thread[Thread-11,5,main] tid=4 pid=1)
threadOS: a new thread (thread=Thread[Thread-13,5,main] tid=5 pid=1)
threadOS: a new thread (thread=Thread[Thread-15,5,main] tid=6 pid=1)
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
```

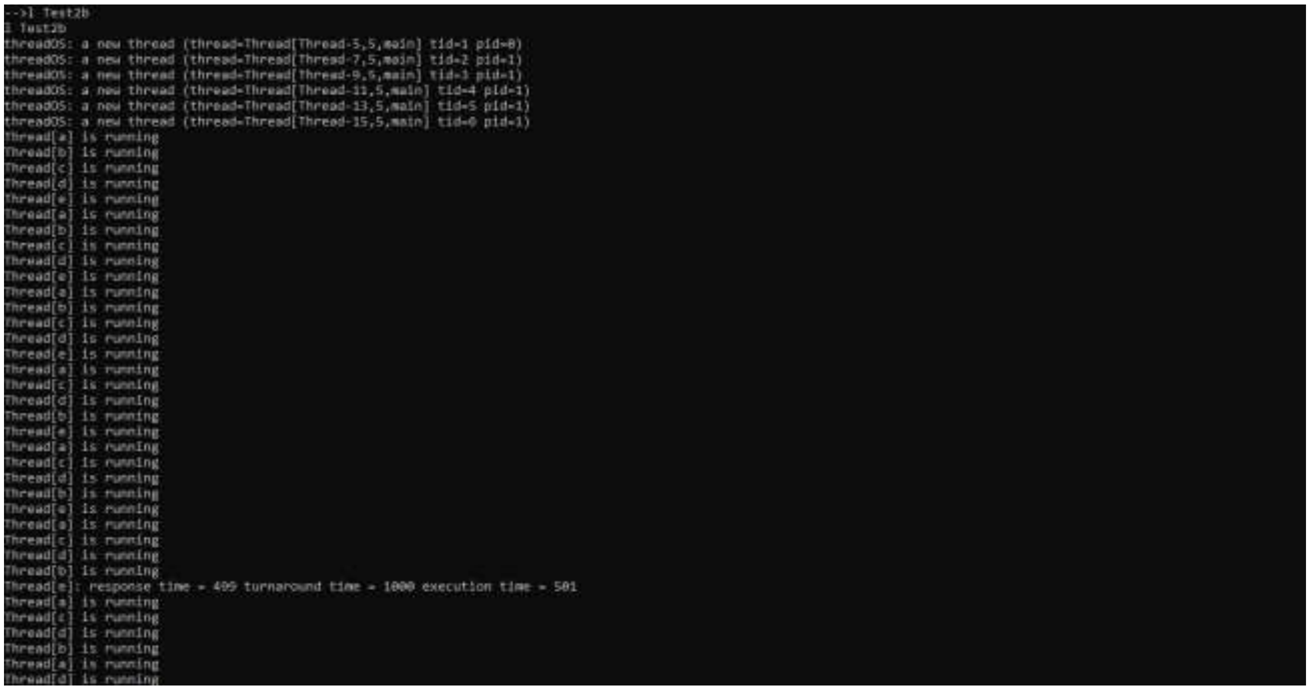


```
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[e]: response time = 6000 turnaround time = 6501 execution time = 501
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[b]: response time = 2999 turnaround time = 10001 execution time = 7002
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
```

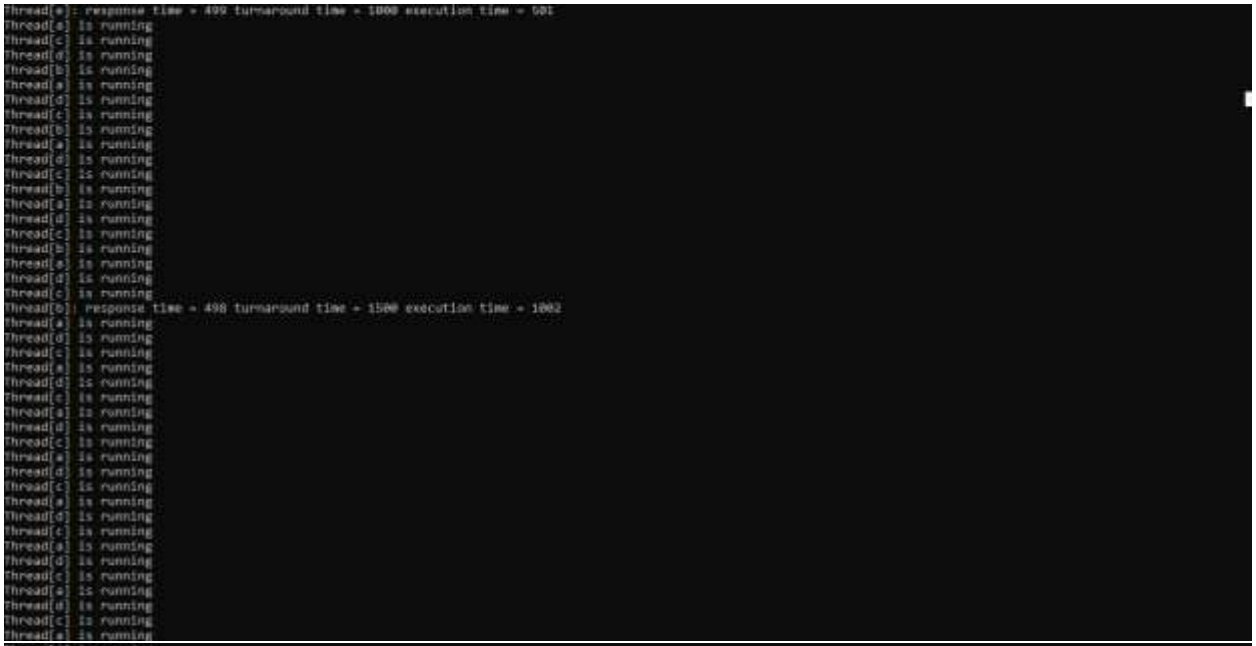


```
Thread[c]: response time = 3999 turnaround time = 21002 execution time = 17003
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[a]: response time = 1999 turnaround time = 29003 execution time = 27004
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d]: response time = 5000 turnaround time = 33004 execution time = 28004
Test2b finished
```

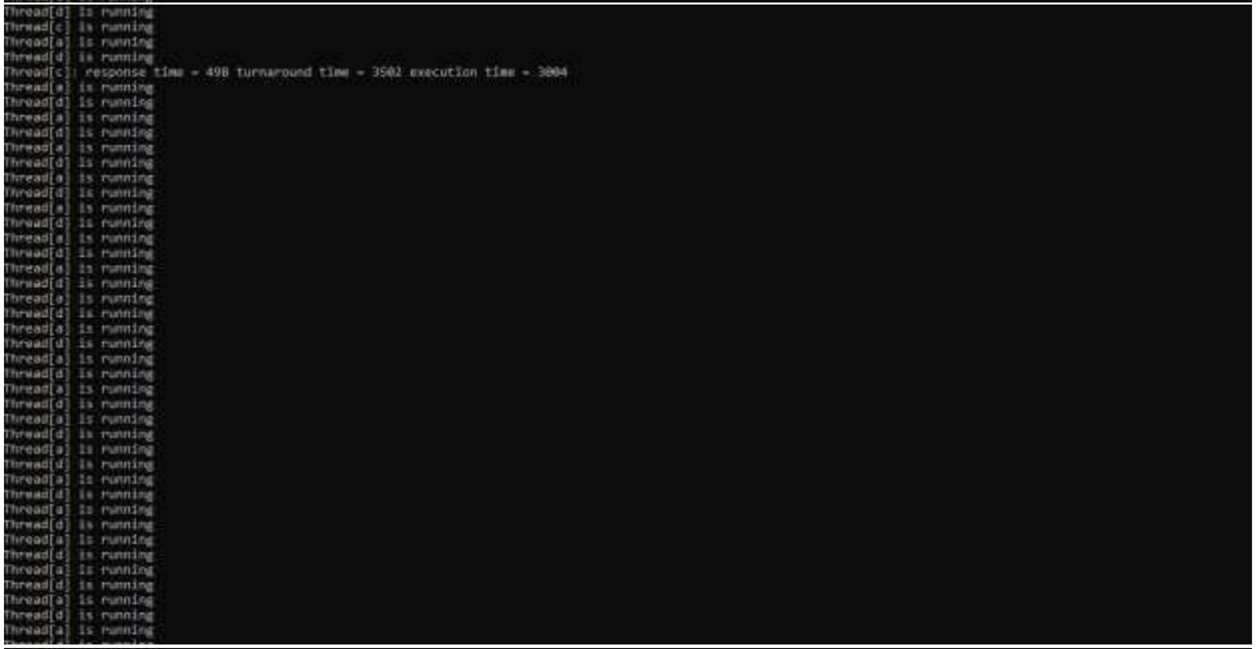
MFQS Output:



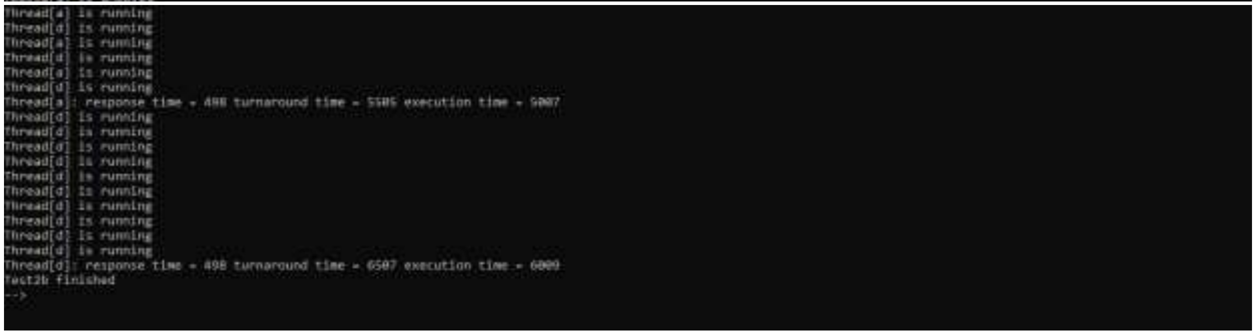
```
-->l Test2b
l Test2b
threadOS: a new thread (thread=Thread[Thread-5,5,main] tid=1 pid=0)
threadOS: a new thread (thread=Thread[Thread-7,5,main] tid=2 pid=1)
threadOS: a new thread (thread=Thread[Thread-9,5,main] tid=3 pid=1)
threadOS: a new thread (thread=Thread[Thread-11,5,main] tid=4 pid=1)
threadOS: a new thread (thread=Thread[Thread-13,5,main] tid=5 pid=1)
threadOS: a new thread (thread=Thread[Thread-15,5,main] tid=6 pid=1)
Thread[a] is running
Thread[b] is running
Thread[c] is running
Thread[d] is running
Thread[e] is running
Thread[a] is running
Thread[b] is running
Thread[c] is running
Thread[d] is running
Thread[e] is running
Thread[a] is running
Thread[b] is running
Thread[c] is running
Thread[d] is running
Thread[e] is running
Thread[a] is running
Thread[c] is running
Thread[d] is running
Thread[b] is running
Thread[e] is running
Thread[a] is running
Thread[c] is running
Thread[d] is running
Thread[b] is running
Thread[e] is running
Thread[a] is running
Thread[c] is running
Thread[d] is running
Thread[b] is running
Thread[e]: response time = 499 turnaround time = 1000 execution time = 501
Thread[a] is running
Thread[c] is running
Thread[d] is running
Thread[b] is running
Thread[a] is running
Thread[d] is running
```



```
Thread[e]: response time = 499 turnaround time = 1000 execution time = 501
Thread[a] is running
Thread[c] is running
Thread[d] is running
Thread[b] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[b] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[b] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[b] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[b]: response time = 498 turnaround time = 1500 execution time = 1002
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
```



```
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c]: response time = 498 turnaround time = 3502 execution time = 3004
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
```



```
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[a]: response time = 498 turnaround time = 5505 execution time = 5007
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d]: response time = 498 turnaround time = 6507 execution time = 6009
Test2b finished
-->
```

Table Comparison (using turnaround time)

Thread name	CPU burst (ms)	RR Scheduler	MFQS
Thread[a]	4900	29003	5505
Thread[b]	900	10001	1500
Thread[c]	2900	21002	3502
Thread[d]	5900	33004	6507
Thread[e]	400	6501	1000

The MFQS scheduler performed much better than the RoundRobin Scheduler did and I think the reason is that the time slice was too short and caused performance issues where a majority of the time was not spent on executing the TCBs but instead was spent trying to give and receive signals telling the other threads to start executing. The MFQS scheduler was able to easily finish shorter process quicker which in turn allows them to get to the longer processes faster without having to constantly switch between processes.

If we implemented MFQS on top of first come first serve I don't believe there would be too much of a difference, because longer processes will still end up being pushed to lower priority queues and the even if the shortest jobs are at the back end of the TCBs within a queue, anything too long will still be sent back to a lower priority queue if it cant finish within the allotted time slice given to it by the MFQS.