README.txt

Gihwan Kwon (Kris) 1534798
CSS430 - Operating Systems
Programming 2 - Scheduling

<Part 1> Round-Robin Scheduling
Assumption
ThreadOS Scheduler which uses a Java-based round-robin scheduling algorithm is not working strictly in a RR fashion, because ThreadOS is a java application and applications running on the OS is subject to the JVM. Therefore, there is no guarantee that a thread with a higher priority will immediately preempt the current thread.

Analysis of Original /ThreadOS/Scheduler.java
- The scheduler gets the highest priority (6), current thread in the queue has priority of 4, and the rest of the threads in the queue has prioority of 2.
- a Lower-priority thread may get an opportunity to run at the expense of a higher-priority thread because the specification for JVM does not say that a scheduling policy must be preemptive. Therefore, it is possible that a thread with a lower priority may continue to run even as a higher-priority thread becomes runnable. (see the Output of original (naive RR))

- We do not have to worry about JVM does not knowing time slice because it is taken care in the run() method.

- Output of Original(naive RR) /ThreadOS/Scheduler.java

```
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->l Test2b
l Test2b
threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
threadOS: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
threadOS: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=1)
threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=1)
threadOS: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=1)
threadOS: a new thread (thread=Thread[Thread-15,2,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[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[c] is running
Thread[b]: response time = 2997 turnaround time = 4001 execution time = 1004
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] 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] 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[e] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[e] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[e] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[e] is running
Thread[d] is running

Thread[c] is running
Thread[a] is running
Thread[e] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[e]: response time = 5998 turnaround time = 6500 execution time = 502
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 = 3998 turnaround time = 7009 execution time = 3011
Thread[a]: response time = 1996 turnaround time = 7013 execution time = 5017
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[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[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[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d]: response time = 4997 turnaround time = 11020 execution time = 6023
Test2b finished

Solution: Redesigned /ThreadOS/Scheduler.java (RR)
- using Thread.suspend() and Thread.reusme()

- Instead of set priority for each thread, and hope that JVM figure out the way to implement it, whenever the operating system calls the run() method, the method could either start or resume the first thread in queue. After running thread in amount of timeSlice, the run() method suspend the current thread and put it back in the back of the queue.

- Output of redesigned RR /ThreadOS/Scheduler.java
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,5,main] tid=0 pid=-1)
-->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[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[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[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[e] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[e]: response time = 5999 turnaround time = 6501 execution time = 502
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 = 2998 turnaround time = 10001 execution time = 7003
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[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[a] 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[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[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[a] is running
Thread[c]: response time = 3997 turnaround time = 21002 execution time = 17005
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[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[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 = 1997 turnaround time = 29005 execution time = 27008
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 = 4998 turnaround time = 33005 execution time = 28007
Test2b finished

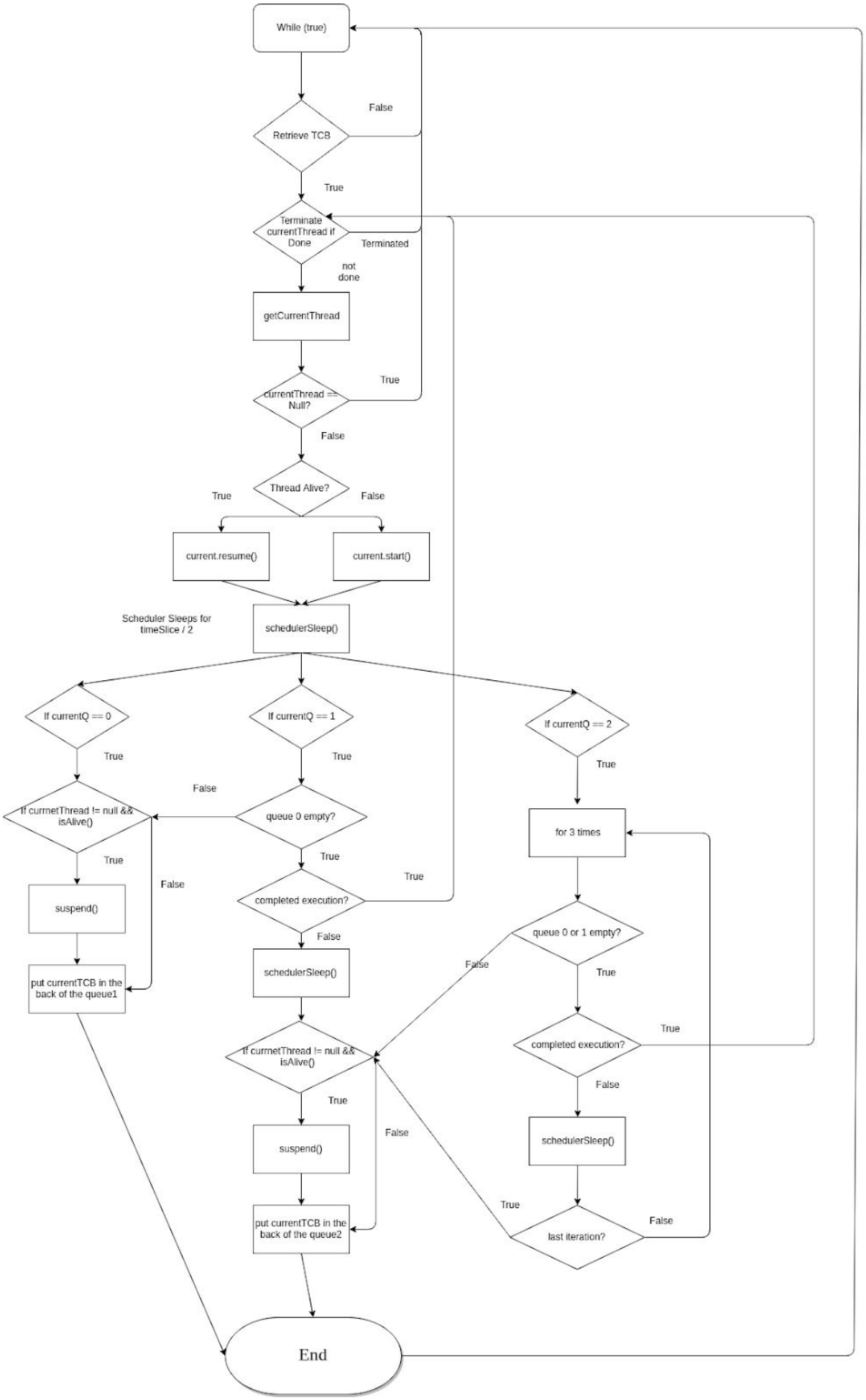
<Part 2> Multi-level Feedback Queue Scheduler (MFQS)
Assumption
- As mentioned in the Program 2 specification,
  - 3 queues, numberd from 0 to 2
  - new thread's TCB is always enqueued into queue 0
  - all threads in queue 0 have time quantum as (timeSlice / 2)
  - all threads in queue 1 have time quantum as timeSlice
  - all threads in queue 2 have time quantum as timeSlice * 2
  - threads in 0 preempt threads in 1 and threads in 1 preempt
    threads in 2. The check occurs every (timeSlice / 2) execution time.
  - if thread does not complete execution in time quantum of each queue,
    it is moved to next queue.
  - When threads in queue 2 do not complete execution in the time quantum,
    it goes back of the queue 2 (not FCFS as mentioned in the text book)
- In addition, the run() method still use Thread.resume() and suspend() to prevent
  lower priority threads to get a chance to run.

Redesign for MFQS
- Change single queue in to Vector[] queues = new Vector[3] and initialize it.
  In this way, we get three different queues 0, 1, and 2. (modify field variable
  and the constructors)

- Modify the getMyTcb() method to retrieve the current thread's TCB from the queue,
  because now we have 3 queues instead of 1.

- Modify addThread(Thread t) method to enforce scheduler to put new TCB in the back of the queue 0.

- Modify run() method
  1. When checking queues to get the next TCB and its thread, it needs to check the three queues starting from 0 to 2 in order. If there is no TCB in all the queues, the method ends this loop without doing anything.
  2. get current Thread from current TCB. flag from which queue the current thread was retrieved
  3. Either start or resume the thread for (timeSlice / 2) and sleep the scheduler while the thread is running
  4. if the current thread was retrieved from the queue 0,
     check if the execution was completed.
     if current thread is still alive, suspend and move the TCB
     of the current thread in the back of the queue 1.
  5. if the current thread was retrieved from the queue 1,
     check if queue 0 is empty.
     if a thread is available in queue 0, put current TCB back to queue 1.
     else, let current thread to execute for another (timeSlice / 2), then
     suspend() the thread and put current TCB in the back of the queue2
  6. if the current thread was retrieved from the queue 2,
     check if queue 0 and 1 is empty in order, if 0 or 1 has theads, suspend
     current thread and put current TCB back of the queue2
     else let scheduler to sleep for another round
  7. do modification 6. for 3 times if current thread was retrieved from the queue 2
- Flow Chart


While (true)
Retrieve TCB
False
True
Terminate currentThread if Done
Terminated
not done
getCurrentThread
currentThread == Null?
True
False
Thread Alive?
True
False
current.resume()
current.start()
Scheduler Sleeps for timeSlice / 2
schedulerSleep()
If currentQ == 0
If currentQ == 1
If currentQ == 2
True
True
True
False
If currnetThread != null && isAlive()
queue 0 empty?
for 3 times
True
True
False
suspend()
completed execution?
True
False
queue 0 or 1 empty?
put currentTCB in the back of the queue1
schedulerSleep()
False
True
If currnetThread != null && isAlive()
completed execution?
True
True
False
suspend()
schedulerSleep()
False
True
False
put currentTCB in the back of the queue2
last iteration?
End

Output of redesigned MFQS /ThreadOS/Scheduler.java

```
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,5,main] tid=0 pid=-1)
-->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[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[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[e] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[e] 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[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]: response time = 997 turnaround time = 5501 execution time = 4504
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[e]: response time = 2497 turnaround time = 7999 execution time = 5502
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[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[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[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]: response time = 1496 turnaround time = 16007 execution time = 14511
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[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[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]: response time = 496 turnaround time = 24013 execution time = 23517
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[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]: response time = 1997 turnaround time = 31013 execution time = 29016
Test2b finished
```

<Part 3> Report

- Assumption:

  The response time in Test2.java and Test2b.java only calculates time of first response time of thread - submission time. Therefore, we can't find out wait time within the scheduler. Therefore, execution time is CPU Burst + wait Time. Turn Around Time is the time between submission to the end.

- comparison between Round Robin and MFQS

Result from Round Robin

Thread[e]: response time = 5998 turnaround time = 6499 execution time = 501
Thread[b]: response time = 2997 turnaround time = 10000 execution time = 7003
Thread[c]: response time = 3997 turnaround time = 21003 execution time = 17006
Thread[a]: response time = 1997 turnaround time = 29006 execution time = 27009
Thread[d]: response time = 4997 turnaround time = 33007 execution time = 28010

mean reasponseTime = 3917.2; mean executionTime = 15908.8; mean TAT = 19903.0;

Result from MFQS
Thread[b]: response time = 997 turnaround time = 5501 execution time = 4504
Thread[e]: response time = 2498 turnaround time = 8001 execution time = 5503
Thread[c]: response time = 1497 turnaround time = 16510 execution time = 15013
Thread[a]: response time = 497 turnaround time = 24515 execution time = 24018
Thread[d]: response time = 1997 turnaround time = 31516 execution time = 29519

mean reasponseTime = 1497.2; mean executionTime = 15711.4; mean TAT = 17208.6;

Result Analysis

- Comparing mean responseTime between RR and MFQS, MFQS has shorter responseTime.
- Mean executionTime (CPU Burst + waitTime) is similar.
- Mean TAT (Turn Around Time) differs because of different responseTime between RR and MFQS.

Therefore, MFQS is better because it reduces the amount of responseTime and , therefore, turnAroundTime.

If MFQS was implemented with FCFS, the waitTime would have increased. Therefore, the executionTime and the TAT would have increased. In this respect, comparing TAT for MFQS and RR would have been similar.