Pouria Ghadimi
CSS430
Professor Dimpsey

Program 2 : Operating System Scheduler

Report

Part 2's algorithm and design:

In order to design the MFQS scheduler I used the existing scheduler and modified it. First I added three vectors called queue0 queue1& queue2. I also had to modify the functions inside the existing scheduler: addThread to add new threads to queue0, getMyTcb in order to search the correct queue for the desired TCB and three different constructors to initialize the three queues.

Then inside the loop in the run function I implemented the following algorithm:

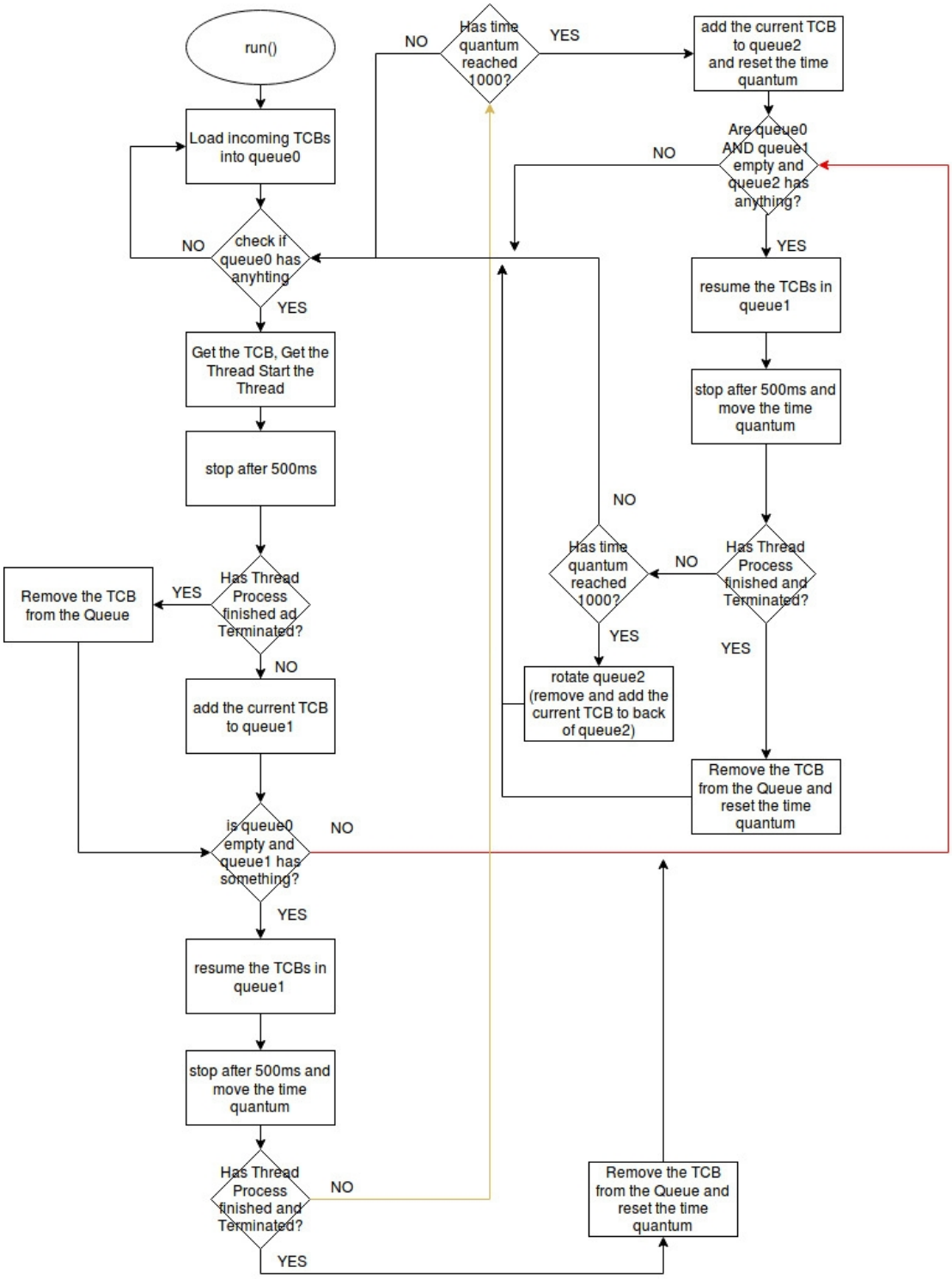
This will all happen inside a while loop *

1.check if there's anything in queue0
2. load the TCB from queue0
3. get the thread and execute
3. stop after 500ms
4. check if the thread is done and terminated
    if so remove it from queue0
    else suspend and add it to queue1
5. check if queue0 is empty and if there's anything in queue1
6. resume the Thread process
7. stop the process
8. move the time quantum by 500ms
9. check if the thread is done and terminated
    if so remove it from queue1 and reset the time quantum
    else if quantum has reached 1000ms suspend and add it to queue2
10. check if queue0 and queue1 are empty and if there's anything in queue2
11. resume the Thread process
12. stop after 500ms
13. move the time quantum by 500ms

14. check if the thread is done and terminated

if so remove it from queue2 and reset the time quantum

else if quantum has reached 2000ms suspend and add it to the end of queue2



My conclusion upon analyzing the results of both the Round Robin and MFQS scheduling algorithms, is that MFQS was the better algorithms in terms of the three metrics:

Response time: measures the amount of time that it took for the thread to begin its execution. Response time is a priority metric for I/O bound processes since a slow response time can greatly impact the experience that a user has with the operating system. By looking at the results it becomes evident that the response time of the MFQS algorithm is much shorter, and is thus more suitable for I/O bound processes than the Round Robin algorithm.

The reason why the response time was faster in the MFQS algorithm is because Queue 0 executes each thread for only a half timeSlice and therefore initially every thread gets to execute its first iteration faster than it would in the Round Robin where the time quantum was a full timeSlice.

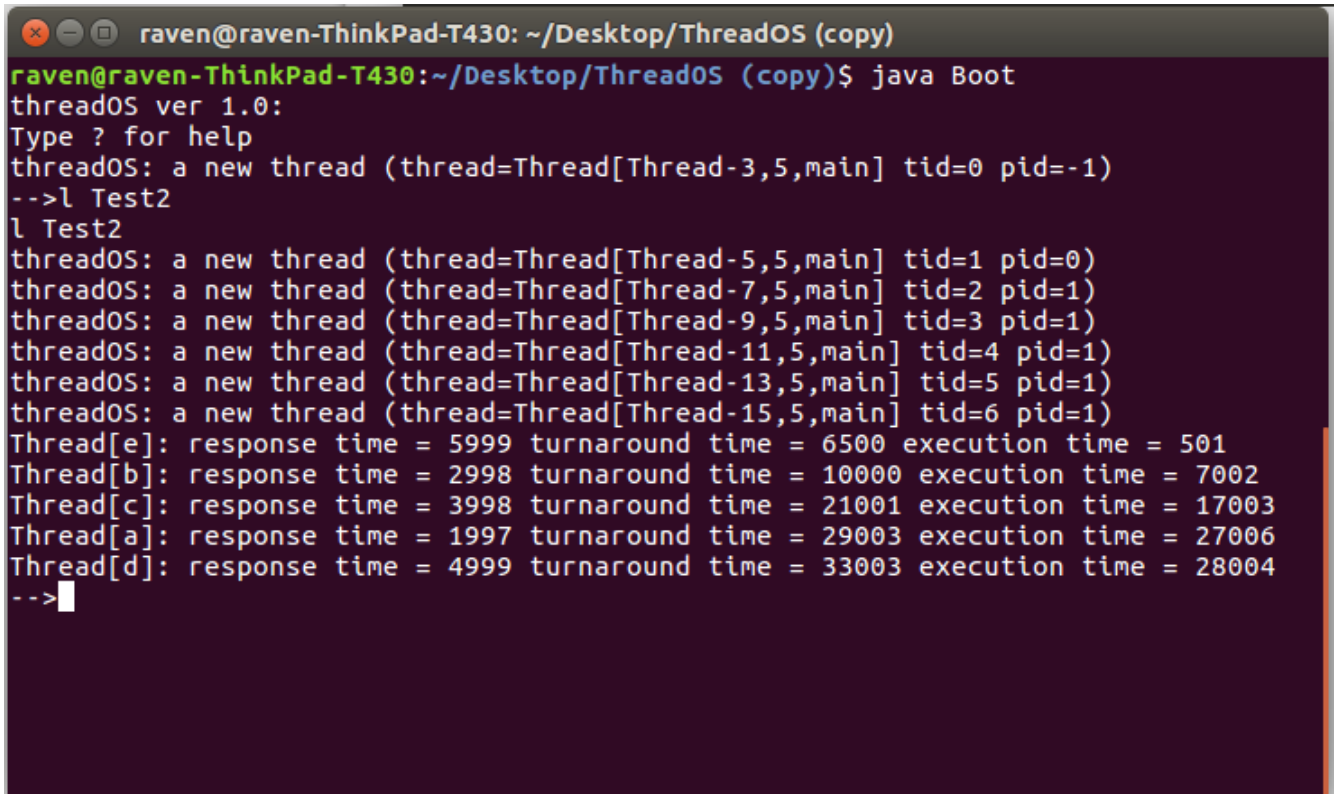
Turnaround time: a measurement of the total amount of time from the initial insertion of the thread in Queue 0, to its termination. Once again, the results in the figures indicate that the MFQS algorithm performed better. Turnaround time is an important metric because it shows the efficiency of the system as a whole. The reason why the turnaround time is maximized in the MFQS algorithm is because in the case where the thread is not yet terminated after a time quantum, it gets moved into a lower priority queue with a higher time quantum. It is said that turnaround time is maximized where more processes terminate in a single time quantum. That is precisely why the Multi-level design of the MFQS minimizes this metric in comparison to the Round Robin.

Execution time: the amount of time the process took from the moment it began execution to the moment it terminated. This metric is important because it can show the efficiency of preemptions and waits in an algorithm. MFQS had a shorter execution time on average due to the fact that the threads spend less time waiting on other threads. This occurred because of the multilevel queue structure within the MFQS as well as its varying time quantum’s.

FCFS BASED QUEUE2

If we have implemented part two in First Come First Served algorithm without preemption it would have take much longer than MFQS and Round robin (with suspend resume). Because nothing will be able to preempt the processes. If there's a thread that takes a long time to be executed and that comes first and smaller threads come in the end they have to wait a long time to get executed because every time that a thread is moved to the next level it is in the first priority and it will be always the first one to finish. Actually there's no point in implementing different levels/queues because in a way it will act similar to a process and not like a thread.

Part 1 Results TEST2B



Part 2 Results TEST2B