Multi-Level Queue Dispatcher

Pseudocode

1. Initialize the queues (normal job dispatch queue, real time job dispatch queue, level 0 queue, level 1 queue, level 2 queue), also initialise variables accumulated_arrival_times, accumulated_service_times, accumulated_finish_times, n_process, set initial value = 0.
2. Fill job dispatch queue from job dispatch file, with real-time jobs being loaded in the RT job dispatch queue, and normal jobs loaded in the Normal job dispatch queue
3. Ask the user to enter an integer value for time_quantum
4. While there is currently running processes or either queue is not empty
    I. Unload any arrived pending process from the job dispatch queue
        A. dequeue process from RT Job Dispatch Queue and enqueue on level 0 queue
        B. dequeue process from normal Job Dispatch Queue and enqueue on level 1 queue
    II. If there is a currently running process
        A. Decrement the process's remaining_cpu_time by quantum;
        B. If the process's allocated time has expired:
            1) Terminate the process: send SIGINT to the process
            2) Increasing accumuate_service_times and accumutaed_arrival_times based on current process information. Also Increasing accumulated_finish_times by current timer. Increase n_process by 1.
            3) Deallocate the PCB (process control block)'s memory
        C. Else if the current running process cannot finish within the time_quantum and either L1 queue or L2 queue is not empty
            1) Suspend the currently running process: send SIGSTP to the process
            2) Add this process to the L2 queue
                a. append job to the end of L2 queue, if priority is 1, and modify priority = 2
                b. append job to the front of L2 queue, if priority is 2
        D. Else if the current running process with priority = 1
            1) Suspend the currently running process: send SIGSTP to the process
            2) append job to the end of L2 queue, and modify priority = 2
        E. Set the current running process as null

    III. If there is no running process and there is a process ready to run(level 0 or level 1 or level 2 queue is not empty):
        A. If level 0 queue is not empty, dequeue the process at the head of the level 0 queue and set it to current_process
        B. Else if level 1 queue is not empty, dequeue the process at the head of the level 1 queue and set it to current_process

C. Else if level 2 queue is not empty, dequeue the process at the head of the level 2 queue and set it to current_process
D. If the process job is a suspended process, send SIGCONT signal to resume it
E. Else start it and set its status as running

IV. Calculate the time_quantum
A. If there is current running process:
1) For the real-time job with priority = 0, quantum is its remaining_cpu_time
2) for the normal job with priority = 1, quantum is the minimum value of time_quamtum and remaining_cpu_time
3) for the normal job with priority = 2, set the quantum = 1, and keep check every second
B. If either job dispatch queue is not empty, but currently no job arrive and all queues are empty
1) if either normal job dispatch queue or RT job dispatch queue is not empty and multi-level queues are all empty, set quantum = 1
2) Else set quantum = 0
V. Let the dispatcher sleep for quantum;
VI. Increment the dispatcher's timer;
VII. Go back to Step 4.

5. Calculate and print the average turnaround time and the average waiting time
6. Terminate the job dispatcher

Test case

General cases need to be tested:

1. **Test level 0 queue**: **real-time jobs** can run in a FCFS manner **without interruption** before completion. (all test cases)
2. **Test level 1 queue**: Test normal jobs can run quantum first, and if can finish within one time_quantum, then will not appear in level 2 queue. (test case 2, 3, 4)
3. **Test level 1 queue**: Test normal jobs can run quantum first, and if cannot finish within one time_quantum, then will appear in level 2 queue. (test case 1, 4)
4. **Test level 1 queue:** No jobs in the level 1 queue can run more than one time_quantum, and the level 1 queue run in a FCFS manner (all test cases)
5. **Test level1 queue:** the normal job with priority = 1, can run time_quantum **without interpretation** within the given time_quantum even if a real-time job arrives during the quantum period (test case 3)
6. **Test level1 queue:** The quantum of level 1 job is the minimum value between the remaining CPU time and time_quantium (test case 2, 5)

7. **Test level 2 queue**: the level 2 job can be scheduled only if both level 0 and level 1 queue are empty. (test case 1)
8. **Test level 2 queue:** Test when a new job arrives (RT job or normal job), the currently running level-2 job will be preempted be placed in the front of the level-2 queue. And when the level-2 job can be scheduled again, this job will re-start before other level-2 jobs. (test case 1)

Special cases considered:

1. **Test level1 queue**: if both level0 and level1 queue are empty, and the running job with priority 1 cannot finish within one time_quantum, then it will be pushed at the end of the level 2 queue. If no other level 2 jobs, this job will continue to run until the new job arrives. (test case 1)
2. **Test level1 queue**: if both level0 and level1 queue are empty, and the running job with priority 1 cannot finish within one time_quantum, then it will be pushed at the end of the level 2 queue. If there is some other level 2 jobs, then the level 2 job in front of the queue will start. (test case 4)
3. **Test time_quantum**: when three-level queues are empty, but still jobs in the job dispatcher queue, the time_quantum will keep counting, and ensure all jobs in the job list can finish before the program terminates. (test case 4, 5)
4. **Test level1 queue:** the normal job with priority = 1, can run time_quantum **without interpretation** within the given time_quantum, even if the new real-time job arrives during this period. (test case 3, 4)
5. The level2 queue is empty all the time (test case 3)
6. **Test empty system behavior**: when multiple level queues are empty, but there are still some unarrived jobs, the system will wait for the unarrived jobs, and scheduled them to run until they arrive. (all test cases)

(My program gives the the expected result based all test cases below)

Job Dispatch List 1

0, 12, 1 (job 1)
3, 7, 1 (job 2)
10, 6, 0 (job 3)
12, 5, 0 (job 4)
14, 3, 1 (job 5)
17, 6, 1 (job 6)
43, 3, 0 (job 7)

We design this situation:

- **Normal jobs with long CPU time come first**, followed by some real-time jobs and normal jobs. We design this case to test the FCFS behavior and time quantum of the level 1 queue. Also can test the execution pattern of level 2 queue(FCFS): the first coming normal job will finish first, even if it has the longest CPU time.

- **Test empty system behavior**: when multiple level queues are empty, but there are still some unarrived jobs, the system will wait for the unarrived jobs, and schedule them to run until they arrive. (The system will be empty few seconds before the last real-time job (arrived at timer = 43) is scheduled to run)

Test case 1: Job dispatch list 1 + time_quantum = 2

- Using time_quantum = 2, all given normal jobs cannot finish within one time_quantum, and thus all of the normal jobs appear in the level 2 queue. By doing so, we can test the FCFS behavior of level 1 queue and level 2 queue.
- The level 2 job can be scheduled only if both level 0 and level 1 queue are empty. And we can also test the FCFS behavior of the level 2 queue: the first coming normal job will finish first, even if it has the longest CPU time. The last-second coming normal jobs (arrival time = 17) will be the last second finished job.
- The last arrived real-time job (arrived at timer = 43) will be scheduled to run at the end
- The first job will run 3 sec (the first 2 secs is time_quantum, then it will be in level 2 queue, and after it runs 1 sec, a new job comes, so it is interrupted), and the second job will start at timer = 3.
- The first real-time job will start immediately when it arrives at timer = 10 (interrupt the running level 2 job)
- The last-second arrival job can only run 2 sec, and then suspend, even if no more level 0 or level 1 jobs, since the job with priority = 1 can only run the given quantum time, then push to the end of the level 2 queue.

Expected result for test case 1

The average turnaround time is: 17.285715

The average waiting time is: 11.285714

Timer	Job	Priority	Job arrive time	Remaining Cpu time	Quantum	Action at the end of quantum
0-2	1	1	0	12	2	Push to the end of level 2 queue
-3	1	2	0	10	1	Push to the front of level 2 queue
-5	2	1	3	7	2	Push to the end of level 2 queue
-10	2	2	0	9	5	Push to the front of level 2 queue
-16	3	0	10	6	6	Finish and terminate
-21	4	0	12	5	5	Finish and terminate
-23	5	1	14	3	2	Push to the end of level 2 queue
-25	6	1	17	6	2	Push to the end of level 2 queue
-29	1	2	0	4	4	Finish and terminate

-34	2	2	3	5	5	Finish and terminate
-35	5	2	14	1	1	Finish and terminate
-39	6	2	17	4	4	Finish and terminate
-43	-	-	-	-	-	Empty system
-46	7	0	43	3	3	Finish and terminate

Test case 2: Job dispatch list 1 + time_quantum = 6

- Using time_quantum = 6, only 2 normal jobs (arrival time = 0 and 3) will appear in the level 2 queue (other normal jobs can finish within one time_quantum). The first arrival job will restart until the last second arrived job terminates. And based on the FCFS rule, the level 2 job with arrival timer = 3 will be scheduled to run after the level2 job arrived at timer = 0 finish.
- The quantum of level 1 job is the minimum value between remaining CPU time and time_quantium (thus, the quantum of 4-th and 5-th arrive job is less than the time_quantum)
- The last real-time job (arrived at timer = 43) will be scheduled to run at the end.

Expected result for test case 2

The average turnaround time is: 17.571428

The average waiting time is: 11.571428

Timer	Job	Priority	Job arrive time	Remaining Cpu time	Quantum	Action at the end of quantum
0-6	1	1	0	12	6	Push to the end of level 2 queue
-12	2	1	3	7	6	Push to the end of level 2 queue
-18	3	0	10	6	6	Finish and terminate
-23	4	0	12	5	5	Finish and terminate
-26	5	1	14	3	3	Finish and terminate
-32	6	1	17	6	6	Finish and terminate
-38	1	2	0	6	6	Finish and terminate
-39	2	2	3	1	1	Finish and terminate
-43	-	-	-	-	-	Empty system
-46	7	0	43	3	3	Finish and terminate

Test case 3: Job dispatch list 1 + time_quantum = 12

- Using **time_quantum = 12**, all jobs can finish within one time_quantum, and thus the level 2 queue is empty all the time.
- The second normal job (arrive at timer = 3) will start until the first and second real-time job terminate (arrive at timer = 10 and 12), as the real-time job has higher priority.
- The real-time jobs run in FCFS manner. Normal jobs also run in FCFS manner.
- The first arrived normal job can run time_quantum (12) **without interpretation** within the given time_quantum even two real-time jobs arrive during the quantum period at (timer = 10 and 12)
- The last arrived real-time job (arrived at timer = 43) will be scheduled to run at the end

Expected result for test case 3

The average turnaround time is: 14.571428

The average waiting time is: 8.571428

Timer	Job	Priority	Job arrive time	Remaining Cpu time	Quantum	Action at the end of quantum
0-12	1	1	0	12	12	Finish and terminate
-18	3	0	10	6	6	Finish and terminate
-23	4	0	12	5	5	Finish and terminate
-30	2	1	3	7	7	Finish and terminate
-33	5	1	14	3	3	Finish and terminate
-39	6	1	17	6	6	Finish and terminate
-43	-	-	-	-	-	Empty system
-46	7	0	43	3	3	Finish and terminate

Job Dispatch List 2

2, 7, 0 (job 1)
3, 2, 1 (job 2)
5, 3, 0 (job 3)
7, 1, 0 (job 4)
20, 8, 1 (job 5)
21, 6, 1 (job 6)
30, 3, 0 (job 7)
33, 7, 1 (job 8)
35, 4, 1 (job 9)

We design this situation:

- The real-time job with relatively long CPU time comes first, and no job arrives before timer 2. By doing so, we can test the timing simulator behavior before the job arrives.
- The first 4 arrive jobs can finish until timer = 15, but the fifth job arrives until timer = 20. By doing so, we can test if the timer counts correctly in the middle of the whole process when no process is running.

Test case 4: Job dispatch list 2 + time_quantum = 2

- The first job can run without interruption until terminated since it is a real-time job.
- Using time_quantum = 2, one job can finish within time quantum. And another job will be pushed to the level 2 queue. And since the last job is a real-time job, the normal job (arrived at timer = 35) will be the last terminated job (based on FCFS rule, the normal jobs arrived earlier will finish earlier).
- The normal job (arrived at timer = 20) can run quantum = 2 without interruption before pushed to level 2 job, even if another real-time job arrived during the quantum time.
- The normal job arrived at timer = 35 can only run 2 sec, then blocked and push to the level 2 queue, as there are other level 2 jobs come early, and need to finish first.

Expected result for test case 4

The average turnaround time is: 10.111111

The average waiting time is: 5.555555

Timer	Job	Priority	Job arrive time	Remaining Cpu time	Quantum	Action at the end of quantum
0-2	-	-	-	-	-	Empty system
-9	1	0	2	7	7	Finish and terminate
-12	3	0	5	3	3	Finish and terminate
-13	4	0	7	1	1	Finish and terminate
-15	2	1	3	2	2	Finish and terminate
-20	-	-	-	-	-	Empty system
-22	5	1	20	8	2	Push to the end of level 2 queue
-24	6	1	21	6	2	Push to the end of level 2 queue
-30	5	2	20	6	6	Finish and terminate
-33	7	0	30	3	3	Finish and terminate

-35	8	1	33	7	2	Push to the end of level 2 queue
-37	9	1	35	4	2	Push to the end of level 2 queue
-41	6	2	21	4	4	Finish and terminate
-46	8	2	33	5	5	Finish and terminate
-48	9	2	35	2	2	Finish and terminate

Test case 5: Job dispatch list 2 + time_quantum = 4

- Using time_quantum = 3, the last arrival job (arrive at timer 35) can finish within one time_quantum. And thus the last second arrival job (arrive at timer 33)will be the last terminated job.
- The quantum of level 1 job is the minimum value between remaining CPU time and time_quantium (thus, the quantum of 2-ed arrive job is less than the time_quantum)
- At timer = 30, the running level2 job push back to the front of level 2 queue, as a new real-time job arrives at timer = 30.

Expected result for test case 5

The average turnaround time is: 11.444445

The average waiting time is: 6.888889

Timer	Job	Priority	Job arrive time	Remaining Cpu time	Quantum	Action at the end of quantum
0-2	-	-	-	-	-	Empty system
-9	1	0	2	7	7	Finish and terminate
-12	3	0	5	3	3	Finish and terminate
-13	4	0	7	1	1	Finish and terminate
-15	2	1	3	2	2	Finish and terminate
-20	-	-	-	-	-	Empty system
-24	5	1	20	8	4	Push to the end of level 2 queue
-28	6	1	21	6	4	Push to the end of level 2 queue
-30	5	2	20	4	2	Push to the front of level 2 queue
-33	7	0	30	3	3	Finish and terminate
-37	8	1	33	7	4	Push to the end of level 2 queue

-41	9	1	35	4	4	Push to the end of level 2 queue
-43	5	2	20	2	2	Finish and terminate
-45	6	2	21	2	2	Finish and terminate
-48	9	2	33	3	3	Finish and terminate