Metropolitan Stat University, Saint Paul, Minnesota
ICS 462 Operating Systems
Assignment 6 Part 1

Goal: To learn to program using concurrent data structures and compare disk scheduling algorithms

Problem: In this assignment, you will create a set of Java classes that together compares several disk scheduling algorithms. In this process, you will employ parallel programming techniques. You will complete the assignment in stages.

The algorithms to be compared are:

1. First-Come First-Served
2. Shortest-Seek-Time-First
3. Elevator (SCAN)
4. C-SCAN

The project will be completed in 5 weeks (4 stages). You will have something to submit on July 9, 16, 23, and August 6.

I will post more detailed specs for stages 2, 3, and 4, on July 8, 15, 22, respectively.

Stage 1: Due on July 9.

Implement a class named Requests. It has synchronized methods to add and get requests for cylinders, which are integers. The class maintains a Vector (JDK) of cylinders that have generated access requests.

void add(Integer cylinder): adds a cylinder to the list. This method is called by a thread that generates requests for cylinders. There may be a thread waiting on such an addition. So after adding, notify all such threads.

Vector<Integer> get(boolean waitIfEmpty): returns all requests that have been added to the Vector object, but have not been retrieved yet. This is called by a thread that simulates one of the disk scheduling algorithms to get the next set of disk requests. The requests are returned in the chronological order of generation. If there are no requests, the invoking thread waits if the Boolean parameter waitIfEmpty is true. If there are requests in the Vector object or waitIfEmpty is false, the calling thread does not wait.

You may need to write additional code to test your implementation.

You need to submit the implementation by 11:59 PM on July 9. I will accept late submissions until 11:59 PM on July 10. There will be a 10% penalty for late submissions. I will post my implementation on July 10. You can use that implementation (if you wish) to work on Stage 2.

Stage 2: Due on July 16.
Write three classes.

1. Producer. A thread that generates 25 cylinder requests and puts them into a Requests object. Between each successive pair of requests, the thread sleeps for a random period of time up to an integer value stored in the final variable `DELAY_BETWEEN_REQUESTS` defined in the Driver class (see below). (Use the random number generator in Java.) The thread should display the requests it generated.
2. Consumer. A thread that calls the get() method of the same Requests object that the producer uses, to get all the 25 cylinder requests, possibly through multiple calls. If there are no requests, the consumer waits. Make sure that the thread sleeps between successive requests and that this sleep time is the integer value stored in `SLEEP_TIME`. defined in the Driver class (see below). The consumer should print the returned value from the get method calls.
3. Driver. Creates a Requests object and starts up a single Producer and a single Consumer thread to work on the Requests object.

You need to submit the implementation by 11:59 PM on July 16. I will accept late submissions until 11:59 PM on July 17. There will be a 10% penalty for late submissions. I will post my implementation on July 18.

Stage 3: Due on July 23.

Implement the Elevator Algorithm.

You need to submit the implementation by 11:59 PM on July 23. I will accept late submissions until 11:59 PM on July 24. There will be a 10% penalty for late submissions. I will post my implementation on July 25.

Stage 4: Due on August 6.

Implement a simulation of C-SCAN, FCFS, and SSTF and submit the code for your entire project. The structure of the program and the display of the results should be similar to my implementation of Elevator.

Execute the algorithms for different values of `DELAY_BETWEEN_REQUESTS` and compare the performance of the algorithms. This part of the assignment is open ended. You must seriously address this part and produce a report based on your findings. The grade for the assignment will to a good extent depend on that.

You need to submit the implementation and the report by 11:59 PM on August 6. I will accept late submissions until 11:59 PM on August 7. There will be a 10% penalty for late submissions.