Laboratory

11

Disk Scheduling

Objectives

- Watch how the operating system schedules the reading or writing of disk tracks.

References

Software needed:

1) A web browser (Internet Explorer or Netscape)

2) Applet from the CD-ROM:

   a) Disk Scheduling

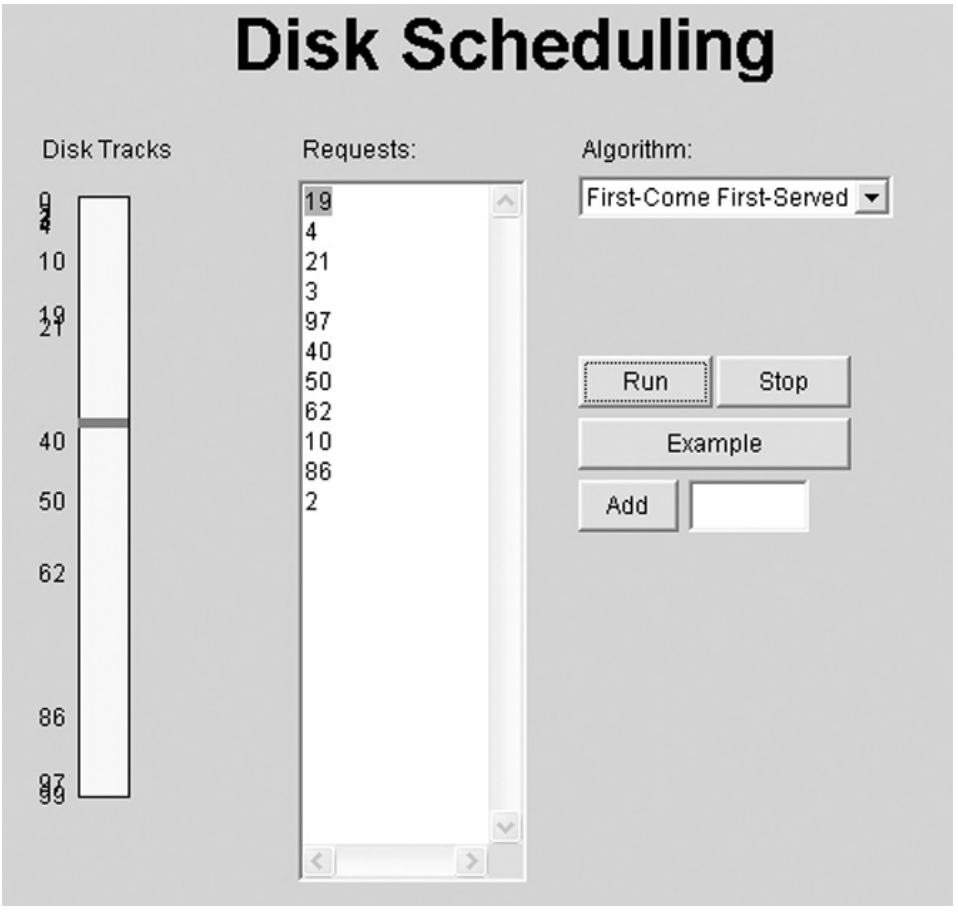
Textbook reference: Chapter 11, pp. 377–380

Background

Everything you need to learn is explained in Chapter 11, "File Systems and Directories."

Activity

Lab 10 simulated how the operating system places jobs into memory and schedules jobs. In this lab, we'll look at how the operating system handles scheduling the reading and writing of disk tracks. The "Disk Scheduling" applet presents an imaginary 100-track, single-surface disk drive. That is the tall yellow bar on the left side of the screen. Track 0 is at the top, and track 99 is at the bottom.



Requests arrive continuously and are kept in a list. Depending on the scheduling algorithm that the operating system uses, a given request may be processed when it is received or much later, depending on where the disk read/write currently is.

For this applet, you'll type in a list of requests. To simplify things, our requests have been stripped down to just which track should be visited next. (In the real world, a request would include information like whether this is a read or a write, which sector to read, what data is to be written, etc.)

Once you have entered some track numbers, click on the *Run* button. You can also preload a list of requested track numbers by pressing *Example*. To start the simulation, press *Run*.

As the simulator is running, you can add new requests by typing in new track numbers into the textfield next to the *Add* button, and then pressing the *Add* button or simply hitting RETURN. In the real world of operating systems, the disk driver is constantly receiving a stream of requests from user programs, which is what you are simulating when you add a new track request.

The read/write head's position is symbolized by a red bar that moves up and down inside the yellow area. As the operating system handles requests, the red bar stops briefly at the current track number, and then removes the request and its number when the request has been completed. Exactly where the red bar moves to next is the subject of the scheduling algorithm.

This applet allows you to experiment with the three scheduling algorithms. Select one of the three from the pull-down menu. The benefits and drawbacks of each algorithm are explained in the textbook.

Exercise 1

Name _____ Date _____

Section _____

1) Start the "Disk Scheduling" applet.
2) Click on the *Example* button, which fills the list with some requests.
3) Select *First-Come, First-Served* from the pull-down menu and click the *Run* button. Time the applet with your watch, or the computer's clock, and write down how many seconds it takes.
4) Select *Shortest Seek Time first* and run the applet again. Measure the time it takes and write it down. Also write down the order of the tracks that are visited. (This is different from the original request list.)
5) Repeat Step 4 after selecting *SCAN Disk (elevator)*.
6) Which algorithm took the least time?
7) Based on your observations of the three algorithms on the example request list, state what kind of request list would make First-Come First-Served (FCFS) take up a lot of time. In other words, what pattern would the numbers in the request list have to follow so that FCFS consumes a lot of time?

Exercise 2

Name _____ Date _____

Section _____

1) Some request lists might cause the disk scheduler to act the same when the three different algorithms are run. Create a request list of five track numbers that will cause all three algorithms to visit the same tracks in the same order.

2) If shortest seek time first starts with the disk head positioned at either 0 or 99, instead of at 50 (in the middle), which algorithm would it resemble: FCFS or SCAN?

   Why?

3) Review Laboratory 9A and Chapter 9, specifically refreshing your memory of stacks and queues. Is the request list of the disk scheduler a stack or a queue?

Exercise 3

Name _____ Date _____

Section _____

1) Start the "Disk Scheduling" applet and type the following numbers into the *Requests* text area:

   8
   20
   35
   80
   10
   90
   5
   87
   26
   94

   These numbers have been chosen so that there are two clusters, one at the lower end of the scale and the other at the upper end.

2) Select the *First-Come, First-Served* algorithm and start. When the disk head has reached 35, type 30 into the "Add" area and press RETURN. This will add a request to seek to track 30 to the list. What happens? Does the disk drive respond to this new request or not?

3) Stop the applet and remove 30 from the end of the list. Choose "Shortest Seek-time first" and rerun the applet. When it consumes 35, type 30 into the add area and press RETURN. Write down what happens. (Warning! You have to be fast because the applet might move into the upper cluster quickly. In that case, just retry.)

4) Stop the applet, remove 30 from the end of the list and choose "SCAN." Run it and when it consumes 35, type 30 again. Write down what happens. Again you must be fast!

5) Which algorithm is least responsive to new requests?

6) Stop the applet, remove 30 from the end of the list, and choose "Shortest seek-time first" and start it. Now try to "trap" the disk head into the lower cluster by typing in disk tracks that are in the lower half of the disk drive, pressing return after each one. You have to be quick, and you may have to try it several times.

7) Redo Step 6, but choose "SCAN" instead. Once again, try to "trap" the disk head into the lower cluster by typing in disk tracks that are in the lower half of the disk drive, pressing return after each one. Were you successful?

8) In real life, disk drives may see a cluster of track requests that could trap it in one section of the disk drive. What implications does this have for programs that requested tracks outside the busy area? (Hint: Computer Scientists have a gruesome term for this phenomenon. It is called starvation. Why do you think they chose this term?)

Deliverables

Turn in your hand-written answers.

Deeper Investigation

The next time you are standing in front of an elevator, waiting for it to arrive, think about the conflicts between fairness and efficiency that the elevator (and operating systems) have to resolve. Though you are undoubtedly the most important person waiting for the elevator, the controlling computer probably isn't aware of this and you won't get priority treatment. What would happen to these scheduling algorithms if some requests did have a higher priority? Do you suppose operating systems ever prioritize disk track requests? Under what circumstances would that be a good idea? What kinds of unforeseen effects might result?