(Extra-credit) Assignment 5: Disk Scheduling Simulation

CSC 139 Operating System Principles - Spring 2019

Posted on April 18, due on May 17 (11:59 pm)

1 Objectives

In this assignment, you will implement three different algorithms for scheduling the disk head on a hard disk. That is, you will simulate the motion of the disk head by taking a set of requests for cylinders/tracks to read and then determining the order in which they will be read. You will then report the total "distance" the disk head had to travel.

2 Description

The disk scheduling algorithms to implement in this assignment are first-come-first-served (FCFS), shortest-seek-time-first (SSTF), and LOOK. The detailed algorithms are already described in class slides and textbook Chapter 11.

2.1 Part A

Write a C program called *fcfs.c* that implements the FCFS disk scheduling algorithm. Assume the disk arm starts at cylinder/track 50. The set of "requests" to the disk (cylinders/tracks to be read) should be provided as command-line arguments to your program (note: it is safe for your program to assume that the inputs are all well-formed integers between 0 and 100). The output of your program should show the order in which the cylinders/tracks are read, and the total distance traveled. For instance, if your program were executed like this:
`./fcfs 5 28 10 7 39 20 45 67 36 35`,
then the output should look like this:

```
Reading track 5
Reading track 28
Reading track 10
Reading track 7
Reading track 39
Reading track 20
Reading track 45
Reading track 67
Reading track 36
Reading track 35
Total distance:  219
```

2.2 Part B

Write a C program called *sstf.c* that implements the SSTF disk scheduling algorithm. As in Part A, the set of "requests" to the disk (cylinders/tracks to be read) should be provided as command-line arguments to your program, and the output of your program should show the order in which the cylinders/tracks are read, and the total distance traveled. Assume the disk arm starts at cylinder/track 50.

Note: the SSTF algorithm looks for the track closest to where the disk arm currently is positioned, not to where it started. In the case where two tracks are both equally as close to the current position, you can choose either one (the higher one or the lower one), as long as it's a "local" decision (i.e., you do not consider what you'll do after that).

2.3 Part C

Write a C program called *look.c* that implements the LOOK disk scheduling algorithm. As in Parts A and B, the set of "requests" to the disk (cylinders/tracks to be read) should be provided as command-line arguments to your program, and the output of your program should show the order in which the cylinders/tracks are read, and the total distance traveled. Assume the disk arm starts at cylinder/track 50.

Note: assume that the disk arm is moving down at the start of your program. And when you calculate the total distance, include the distance of the "giant leap."

3 Deliverables

Make sure your code can be compiled and work on Athena server correctly. Upload to Canvas the following:

- All source codes/files that you have added/modified and the demonstrative results.
- A `README.TXT` file that briefly describes each file, how to compile the file(s), and how to run the file.
- These files should be placed in a directory called `<ECS username>-asgmt5`.
- Use tar command to place all the files in a single file called `<ECS username>-asgmt5.tar`. Assuming you are in the directory `<ECS username>-asgmt5` do the following:
  - Goto the parent directory: `cd ..`
  - tar the files:
    `tar -cvf <ECS username>-asgmt5.tar ./<ECS username>-asgmt5`
  - Verify the files have been placed in a tar file:
    `tar -tvf <ECS username>-asgmt5.tar`
  - Compress the files using gzip: `gzip <ECS username>-asgmt5.tar`
  - Verify that the gzipped file exists: `ls <ECS username>-asgmt5.tar.gz`