Mark Rizko
Prof. Ahmed Amer
MWF 8:00am
Lab R 5:15pm

Operating Systems Final Project

Objective

The program I chose for my final project was a simulation and comparison of three different disk scheduling algorithms that we learned in class. The three algorithms I compared were Shortest Seek Time First (SSTF), Circular-Look (C-LOOK), and my simple SWEEP algorithm. The grounds on which I compare the algorithms will be on seek distance.

The Sweep Algorithm

When I decided to do this project, I wanted to think of my own original algorithm to compete with the other two most efficient algorithms discussed in class. I eventually came up with two, one called the "Zig-Zag" algorithm (which was not efficient) and the other more simple and VERY consistent, the SWEEP algorithm. This algorithm starts and ends the head on the ends (alternating between starting position 0 and N*) and essentially "sweeps" the surface, covering every track as it goes and completing every job on the way. This algorithm stood out to me because the seek time for one round was always consistent, N.

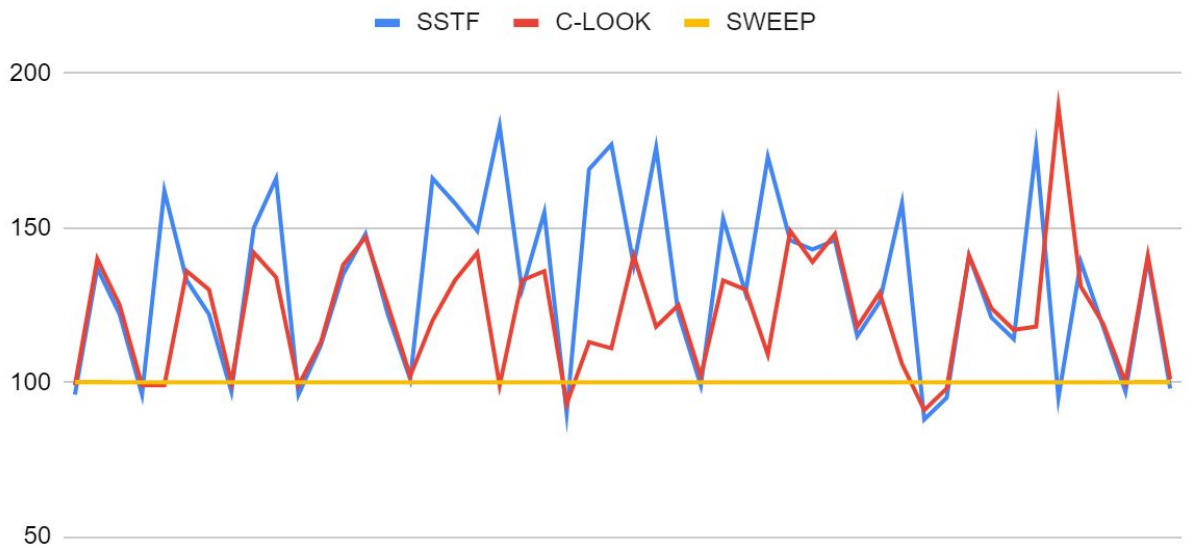
*N denotes number of tracks on surface

Implementation

The program simulates a disk surface with 100 tracks, where 25-75 tracks will require a job. The way I implemented the array was an array with all zeros that, when a job was required, filled with a one (like the reference bits in the second chance algorithm). The algorithms would then iterate through the list using their desired form, and change the ones to zeros. A variable inside each of the algorithms' functions tracks the seek distance and they are displayed at the end of the program.

Analysis

For my analysis, I simulated each algorithm on the same random load fifty times and graphed the seek times.



As you can see, the SWEEP algorithm was the most consistent, and if you were to add up all of the seek distances for all 50 tests ran on each of the algorithms, SWEEP has the lowest total seek distance. However, it is important to note that both the SSTF and C-LOOK algorithm reached a seek time less than 100 for about 20% of the tests. The lowest seek distance recorded for an individual test for C-LOOK was 93 and for SSTF was 88. While SSTF had the lowest seek distance for an individual test, it is clear from the chart that the C-LOOK algorithm was faster in most of the tests.

Conclusion

These tests results surprised me because they all have advantages in different areas. SWEEP has a consistency that cannot be argued with, SSTF had the fastest individual speed, and C-LOOK had a faster average time than SSTF while occasionally reaching speeds faster than SWEEP. While I expected both SSTF and C-LOOK to be faster than the consistent SWEEP, I was not expecting this behavior only 20% of the time. The SWEEP algorithm surprised me because it was faster than SSTF and C-LOOK 80% of the time, showing how even a simple algorithm can be better in some cases. While my tests produced these specific results, I cannot conclude that any of these algorithms are better than the other given the fact that my tests were over one controlled, fixed situation and the behavior of each algorithm **depends** on the situation that is being handled by the OS.