



MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

**Mini Project Report
of
Operating Systems LAB**

TITLE
DISK SCHEDULING ALGORITHMS

SUBMITTED BY

TEAM MEMBERS:

| NAME | Reg. No. | Roll No. | SECTION |
|-------------------|-----------------|-----------------|----------------|
| ANIKET P KILLEDAR | 200905170 | 27 | D |
| MANNE VARUN | 200905194 | 32 | D |
| ATLURI LUHIT | 200905164 | 25 | D |
| RITWIK RAJ | 200905186 | 29 | D |

ABSTRACT

- This Project is related to disk scheduling concept of operating systems.

Disk scheduling is a technique used by the operating system to schedule multiple requests for accessing the disk. Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling. One of the responsibilities of the operating system is to use the hardware efficiently. For the disk drives, meeting this responsibility entails having fast access time and large disk bandwidth.

The seek time is the time for the disk arm to move the heads to the cylinder containing the desired sector. The rotational latency is the additional time for the disk to rotate the desired sector to the disk head. The disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer. We can improve both the access time and the bandwidth by managing the order in which disk I/O requests are serviced. Whenever a process needs I/O to or from the disk, it issues a system call to the operating system.

If the desired disk drive and controller are available, the request can be serviced immediately. If the drive or controller is busy, any new requests for service will be placed in the queue of pending requests for that drive. For a multiprogramming system with many processes, the disk queue may often have several pending requests. Thus, when one request is completed, the operating system needs to choose which pending request to service next. The operating system makes this choice by implementing any one of the several disk-scheduling algorithms such as FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK, etc.

PROBLEM STATEMENT

To implement the disc scheduling algorithms in C language and find the optimal algorithm based on total head movement.

ALGORITHMS

The six algorithms implemented in this project are:

FCFS

FCFS (First Come-First Serve) is the easiest disk scheduling algorithm among all the scheduling algorithms. In the FCFS disk scheduling algorithm, each input/output request is served in the order in which the requests arrive. In this algorithm, starvation does not occur because FCFS address each request.

The advantages of FCFS disk scheduling algorithm are:

- In FCFS disk scheduling, there is no indefinite delay.
- There is no starvation in FCFS disk scheduling because each request gets a fair chance.

The disadvantages of FCFS disk scheduling algorithm are:

- FCFS scheduling is not offered as the best service.
- In FCFS, scheduling disk time is not optimized.

SSTF

In SSTF disk scheduling, the job which has less seek time will be executed first. So, in SSTF (shortest seek time first) scheduling, we must calculate the seek time first. and after calculating the seek time, each request will be served based on seek time. The request which is close to the disk arm will be first executed. There are some drawbacks in FCFS. To overcome the limitations that arise in the FCFS. SSTF scheduling is implemented.

The advantages of SSTF disk scheduling are:

- In SSTF disk scheduling, the average response time is decreased.
- Increased throughput.

The disadvantages of SSTF disk scheduling are:

- In SSTF, there may be a chance of starvation.
- SSTF is not an optimal algorithm.
- There are chances of overhead in SSTF disk scheduling because, in this algorithm, we must calculate the seek time in advanced.
- The speed of this algorithm can be decreased because direction could be switched frequently.

SCAN

In this algorithm, we move the disk arm into a specific direction (direction can be moved towards large value or the smallest value). Each request is addressed that comes in its path, and when it comes into the end of the disk, then the disk arm will move reverse, and all the requests are addressed that are arriving in its path. Scan disk scheduling algorithm is also called an elevator algorithm because its working is like an elevator.

The advantages of SCAN disk scheduling algorithm are:

- In SCAN disk scheduling, there is a low variance of response time.
- In this algorithm, throughput is high.
- Response time is average.
- In SCAN disk scheduling, there is no starvation.

The disadvantages of SCAN disk scheduling algorithm are:

- SCAN disk scheduling algorithm takes long waiting time for the cylinders, just visited by the head.
- In SCAN disk scheduling, we must move the disk head to the end of the disk even when we don't have any request to service.

C-SCAN

C-SCAN stands for Circular-SCAN. C-SCAN is an enhanced version of SCAN disk scheduling. In the C-SCAN disk scheduling algorithm, the disk head starts to move at one end of the disk and moves towards the other end and service the requests that come in its path and reach another end. After doing this, the direction of the head is reversed. The head reaches the first end without satisfying any request and then it goes back and services the requests which are remaining.

The advantages of the C-SCAN disk scheduling algorithm are:

- C-SCAN offers better uniform waiting time.
- It offers a better response time.

The disadvantages of the C-SCAN disk scheduling algorithm are:

- In C-SCAN disk scheduling, there are more seek movements as compared to SCAN disk scheduling.
- In C-SCAN disk scheduling, we must move the disk head to the end of the disk even when we don't have any request to service.

LOOK

LOOK scheduling is an enhanced version of SCAN disk scheduling. Look disk scheduling is the same as SCAN disk scheduling, but in this scheduling, instead of going till the last track, we go till the last request and then change the direction.

The advantages of LOOK disk scheduling are:

- In LOOK disk scheduling, there is no starvation.
- LOOK disk scheduling offers low variance in waiting time and response time.
- LOOK disk scheduling offers better performance as compared to the SCAN disk scheduling as there is no requirement of disk head to move till the end to the disk when we do not have any request to be serviced.

The disadvantages of LOOK disk scheduling are:

- In LOOK disk scheduling, there is more overhead to find the end request.
- LOOK disk scheduling is not used in case of more load.

C-LOOK

C-LOOK means Circular-LOOK. It takes the advantages of both the disk scheduling C-SCAN and LOOK disk scheduling. In C-LOOK scheduling, the disk arm moves and service each request till the head reaches its highest request, and after that, the disk arm jumps to the lowest cylinder without servicing any request, and the disk arm moves further and service those requests which are remaining.

The advantages of C-LOOK disk scheduling are:

- There is no starvation in C-LOOK disk scheduling.
- The performance of the C-LOOK scheduling is better than Look disk scheduling.
- C-LOOK disk scheduling offers low variance in waiting time and response time.

The disadvantages of C-LOOK disk scheduling are:

- In C-LOOK disk scheduling there may be more overhead to determine the end request.
- There is more overhead in calculations.

OUTCOME/CODE SNIPPET

GitHub Link

<https://gist.github.com/pun1sher729/7a3d3b4be3efb400466079dbcc3e04d9>

RESULT

Input format:

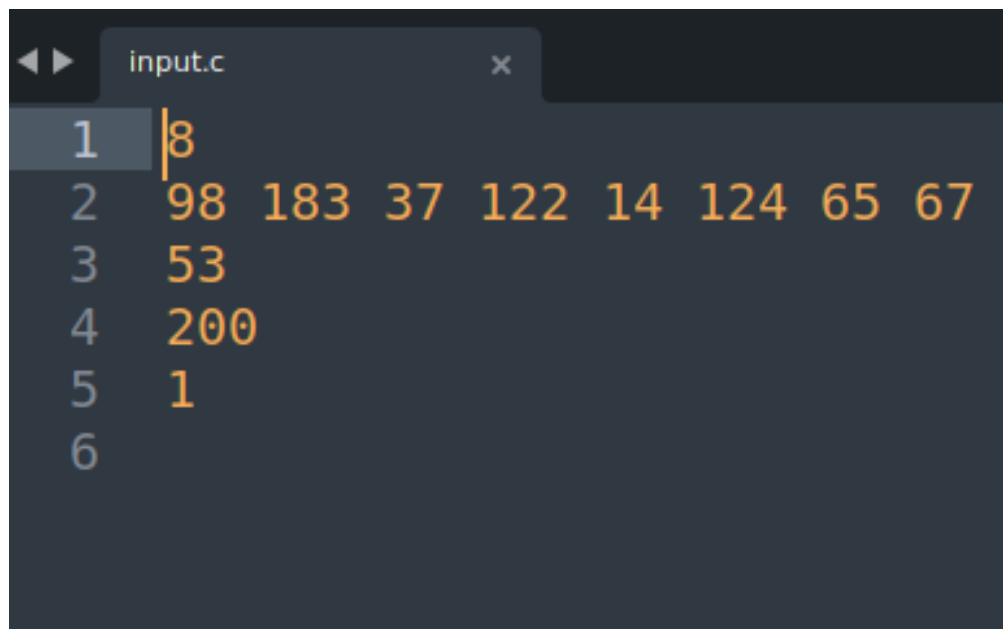
Line 1: Number of requests

Line 2: Request Queue

Line 3: Initial Head Position

Line 4: Disk Size or Length of cylinder

Line 5: Direction of Head Movement [0 – low, 1 – high]



The screenshot shows a code editor window titled 'input.c'. It contains six lines of input data for a disk scheduling problem. The first line is '8', the second line is '98 183 37 122 14 124 65 67', the third line is '53', the fourth line is '200', the fifth line is '1', and the sixth line is empty. The text is displayed in a monospaced font with a dark background and light-colored text.

```
1 8
2 98 183 37 122 14 124 65 67
3 53
4 200
5 1
6
```

Output:

```
aniket@aniket-virtual-machine:~/Desktop/OS Mini Project$ cc v4.c -lpthread && ./a.out
```

Menu for disk scheduling algorithm:

- 1: FCFS
- 2: SSTF
- 3: SCAN:
- 4: C_SCAN
- 5: LOOK
- 6: C_LOOK
- 7: Run all Algorithms
- 8: EXIT

Enter your choice:

7

Enter name of file to read: input.c

In FCFS:

Current Head Position

53 98 183 37 122 14 124 65 67

Total head movement = 640

In SSTF:

Current Head Position

53 65 67 37 14 98 122 124 183

Total head movement = 236

In SCAN:

Current Head Position

53 65 67 98 122 124 183 199 37 14

Total head movement = 331

In C-SCAN:

Current Head Position

53 65 67 98 122 124 183 199 0 14 37

Total head movement = 382

In LOOK:

Current Head Position

53 65 67 98 122 124 183 37 14

Total head movement = 299

In C-LOOK:

Current Head Position

53 65 67 98 122 124 183 14 37

Total head movement = 322

The Algorithms with least Head movements is/are: SSTF

CONCLUSION

SSTF is common and has a natural appeal because it increases performance over FCFS. SCAN and C-SCAN perform better for systems that place a heavy load on the disk, because they are less likely to cause a starvation problem. For any list of requests, we can define an optimal order of retrieval, but the computation needed to find an optimal schedule may not justify the savings over SSTF or SCAN.

With any scheduling algorithm, however, performance depends heavily on the number and types of requests. For instance, suppose that the queue usually has just one outstanding request. Then, all scheduling algorithms behave the same, because they have only one choice of where to move the disk head: they all behave like FCFS scheduling.

Requests for disk service can be greatly influenced by the file-allocation method. A program reading a contiguously allocated file will generate several requests that are close together on the disk, resulting in limited head movement. A linked or indexed file, in contrast, may include blocks that are widely scattered on the disk, resulting in greater head movement.

The location of directories and index blocks is also important. Since every file must be opened to be used, and opening a file requires searching the directory structure, the directories will be accessed frequently. Suppose that a directory entry is on the first cylinder and a file's data are on the final cylinder. In this case, the disk head must move the entire width of the disk. If the directory entry were on the middle cylinder, the head would have to move only one-half the width. Caching the directories and index blocks in main memory can also help to reduce disk-arm movement, particularly for read requests. Because of these complexities, the disk-scheduling algorithm should be written as a separate module of the operating system, so that it can be replaced with a different algorithm if necessary. Either SSTF or LOOK is a reasonable choice for the default algorithm.

REFERENCES

Website

- <https://www.tutorialandexample.com/scan-disk-scheduling-algorithm>

Textbook

- Operating System Concepts by Abraham Silberschatz, Greg Gagne, Peter B. Galvin 9th Edition