K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Exp 8 C2 110

TITLE: Disk Scheduling Algorithms

AIM: Implementation of Disk Scheduling Algorithm like FCFS, SSTF, SCAN, CSCAN, LOOK.

Expected Outcome of Experiment:

CO 4. To understand various Memory, I/O and File management techniques.

Books/ Journals/ Websites referred:

1. **Silberschatz A., Galvin P., Gagne G. “Operating Systems Principles”, Willey Eight edition.**
2. **Achyut S. Godbole , Atul Kahate “Operating Systems” McGraw Hill Third Edition.**
3. **William Stallings, “Operating System Internal & Design Principles”, Pearson.**
4. **Andrew S. Tanenbaum, “Modern Operating System”, Prentice Hall.**

Pre Lab/ Prior Concepts:

- Knowledge of disk scheduling algorithm.
- Calculation of seek time and transfer time etc.

Description of the application to be implemented:

First Come-First Serve (FCFS):

- FCFS is one of the simplest disk scheduling algorithms.
- It serves I/O requests in the order they arrive in the queue. The first request that arrives is the first to be serviced.

- While it is straightforward, FCFS can lead to poor performance because it doesn't consider the location of the requested data on the disk. This can result in long seek times if the requests are scattered across the disk.

Shortest Seek Time First (SSTF):

- SSTF is designed to reduce seek time by selecting the I/O request that is closest to the current position of the disk's read/write head.
- It constantly recalculates the distance between the current position and all pending requests and selects the one with the shortest seek time.
- SSTF is more efficient than FCFS in terms of minimizing seek time, but it can lead to starvation of some requests if there are always closer requests in the queue.

Elevator (SCAN):

- The SCAN algorithm, also known as the elevator algorithm, works like an elevator moving up and down the disk.
- It services requests in one direction (either inward or outward) until it reaches the end of the disk. Then, it reverses direction and continues servicing requests in the opposite direction.
- SCAN is efficient in preventing extreme cases of starvation and provides relatively good performance by servicing requests in a predictable manner.

Circular SCAN (C-SCAN):

- C-SCAN is an enhancement of the SCAN algorithm.
- Instead of reversing direction at the end of the disk, it "wraps around" to the beginning and starts servicing requests in the same direction again.
- This approach reduces the possibility of extreme seek times by ensuring that all requests get serviced regularly.

LOOK:

- LOOK is similar to SCAN, but it does not reach the end of the disk before changing direction.
- It reverses direction when there are no more pending requests in the current direction and starts servicing requests in the opposite direction.
- LOOK aims to reduce seek time while avoiding unnecessary movement to the ends of the disk, making it more efficient in certain situations compared to SCAN.

Implementation details:

```
num = 10

#num = int(input("please enter number of values
"))
tasks_input = [100,55,58,39,18,90,160,150,38,184]

#tasks_input = [int(input("please enter sectors
")) for i in range(num)]


def calculateSeekTime(tasks):

time = 0

for i in range(len(tasks)-1):

```

```
time += abs(tasks[i+1]-tasks[i])

#print(time)


time = time/ len(tasks)


print("average time = ",time)

return time


def FCFS(tasks):

#print(tasks)


calculateSeekTime(tasks)

return tasks



def SCAN(tasks):


answer = []



#run loop over to check new tasks
```

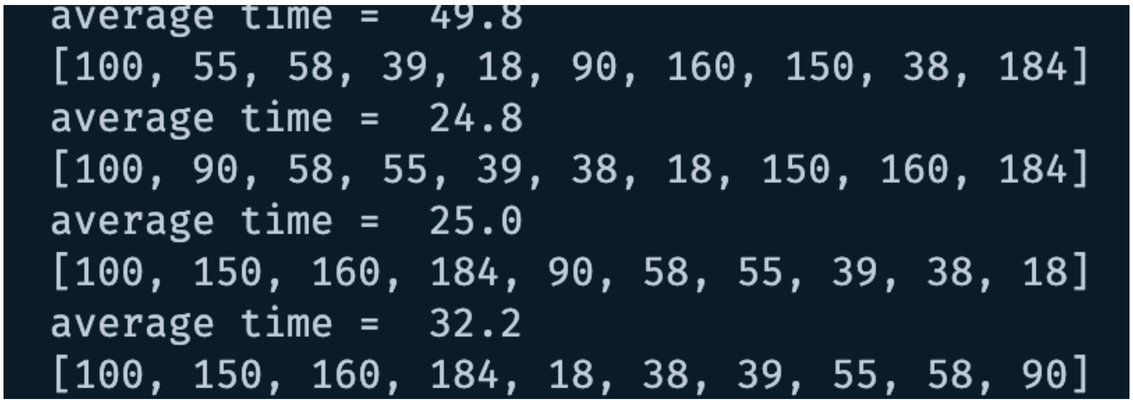
```
# check new tasks
initial = tasks[0] # last position of reader

less = [i for i in tasks if i >= initial]

less.sort()

answer = answer + less

for i in less:
tasks.remove(i)
######check more tasks arrived in future

# move down

tasks.sort()
answer = answer +tasks[::-1]
tasks = []

#loop end
#print(answer)
```

```
calculateSeekTime(answer)
return answer



def SCAN_C(tasks):

answer = []

initial = tasks[0] # last position of reader

less = [i for i in tasks if i >= initial]

less.sort()

answer = answer + less

for i in less:

tasks.remove(i)
```

```
##### check more tasks arrived in future

#run loop over to check new tasks

tasks.sort()

answer = answer + tasks
#print(answer)


calculateSeekTime(answer)
return answer


def SSTN(tasks):

initial = tasks[0] # last position of reader

answer = []
```

```
answer.append(initial)

while len(tasks):

tasks.remove(initial)

if(len(tasks) == 0):

break

minDifference = 10000

minTask = -1

for i in tasks:

if(abs(i - initial)<minDifference):

minDifference = abs(i - initial)

minTask = i
```

```
answer.append(minTask)
#print(answer)

initial = minTask
#print(initial)
#print(answer)

calculateSeekTime(answer)
return answer


print(FCFS(tasks_input.copy()))

print(SSTN(tasks_input.copy()))

print(SCAN(tasks_input.copy()))

print(SCAN_C(tasks_input.copy()))
```



```
average time =  49.8
[100, 55, 58, 39, 18, 90, 160, 150, 38, 184]
average time =  24.8
[100, 90, 58, 55, 39, 38, 18, 150, 160, 184]
average time =  25.0
[100, 150, 160, 184, 90, 58, 55, 39, 38, 18]
average time =  32.2
[100, 150, 160, 184, 18, 38, 39, 55, 58, 90]
```

Conclusion: Thus we have implemented various disc scheduling algorithms. These algorithms are used for the optimized scheduling of disc reads and writes. These algorithms reduce the time required for disc head movement in different sectors.

Post Lab Descriptive Questions

1. **A disk drive has 200 cylinders numbered from 0 to 199. The disk head is initially at cylinder 53. The queue of pending requests in FIFO order is :**

   98, 183, 37, 122, 14, 124, 65, 67.

   Starting from the current head position, what is the total distance travelled (in cylinders) by disk arm to satisfy the requests using CSCAN and Look. Illustrate with figures in each case.

   average time = 31.75

   [98, 122, 124, 183, 67.0, 65, 37, 14]

   average time = 38.375

   [98, 122, 124, 183, 14, 37, 65, 67.0]

Post Lab Objective Questions

1. In a hard disk, what rotates about a central spindle ______________
   a. Disk
   b. Platter

c. Sector
d. None of the above

Ans:d)

2. The time required to move the disk arm to the required track is known as ______________________________

a. Latency time
b. Access time
c. Seek time
d. None of the above

Ans:c)

Date: 30 oct 23 **Signature of faculty in-charge**