D - 1

**Batch: D - 1 Roll No.: 16010122096**

**Experiment No. 09**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE:** Implementation of Disk Scheduling Algorithm like FCFS, SSTF, SCAN, CSCAN, LOOK |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**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. **Sumitabha Das “ UNIX Concepts & Applications”, McGraw Hill Second**

**Edition.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

 **Disk Scheduling Algorithms**: Understanding various algorithms that manage how disk I/O requests are processed, affecting performance and efficiency.

 **Seek Time and Transfer Time**: Calculating seek time (the time it takes for the disk arm to move to the requested track) and transfer time (the time it takes to read/write data once the arm is in position).\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Description of the application to be implemented**:

#### First Come-First Serve (FCFS):

FCFS is the simplest disk scheduling algorithm. It processes requests in the order they arrive, ensuring fairness but often resulting in longer average seek times due to potential delays from far-apart requests.

#### LOOK:

LOOK is an optimized version of the SCAN algorithm. It moves the disk arm in one direction, servicing requests until it reaches the end of the requested track, then reverses direction. This reduces unnecessary movement and can lead to shorter average seek times compared to FCFS.

**Implementation details:**

#include <bits/stdc++.h>

using namespace std;

vector<int> lookSeq(const vector<int> &reqSeq, int head, string direction)

{

vector<int> seek;

vector<int> sortedSeq = reqSeq;

sort(sortedSeq.begin(), sortedSeq.end());

auto it = lower\_bound(sortedSeq.begin(), sortedSeq.end(), head);

int index = distance(sortedSeq.begin(), it);

if (direction == "right")

{

for (int i = index; i < sortedSeq.size(); ++i)

{

seek.push\_back(sortedSeq[i]);

}

for (int i = index - 1; i >= 0; --i)

{

seek.push\_back(sortedSeq[i]);

}

} else if (direction == "left")

{

for (int i = index - 1; i >= 0; --i)

{

seek.push\_back(sortedSeq[i]);

}

for (int i = index; i < sortedSeq.size(); ++i)

{

seek.push\_back(sortedSeq[i]);

}

}

return seek;

}

vector<int> fcfsSeq(const vector<int> &reqSeq, int head)

{

vector<int> seek;

for (int request : reqSeq)

{

seek.push\_back(request);

}

return seek;

}

int calculateSeekTime(const vector<int> &seek, int head)

{

int seekTime = abs(seek[0] - head);

for (int i = 1; i < seek.size(); ++i)

{

seekTime += abs(seek[i] - seek[i - 1]);

}

return seekTime;

}

void displaySeq(const vector<int> &a)

{

for (int i : a)

{

cout << i << " ";

}

cout << endl;

}

int main()

{

while (true)

{

cout << "Choose scheduling algorithm:\n";

cout << "1. LOOK\n";

cout << "2. FCFS\n";

cout << "3. Exit\n";

int choice;

cin >> choice;

if (choice == 3)

{

break;

}

cout << "Enter the number of blocks: ";

int n;

cin >> n;

cout << "Please enter the request sequence:\n";

vector<int> reqSeq(n);

for (auto &i : reqSeq) cin >> i;

cout << "Enter the head position: ";

int head;

cin >> head;

if (choice == 1)

{

cout << "Enter the direction in which the head should move initially (left/right): ";

string direction;

cin >> direction;

vector<int> seekOrder = lookSeq(reqSeq, head, direction);

cout << "Seek sequence: ";

displaySeq(seekOrder);

int totalSeekTime = calculateSeekTime(seekOrder, head);

cout << "Total seek time: " << totalSeekTime << endl;

}

else if (choice == 2)

{

vector<int> seekOrder = fcfsSeq(reqSeq, head);

cout << "Seek sequence: ";

displaySeq(seekOrder);

int totalSeekTime = calculateSeekTime(seekOrder, head);

cout << "Total seek time: " << totalSeekTime << endl;

}

else

{

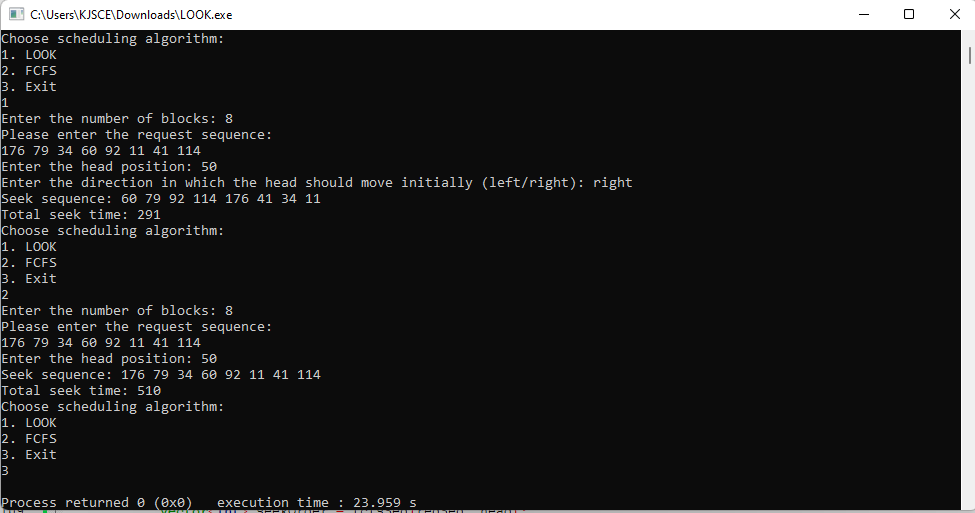
cout << "Invalid choice. Please try again.\n";

}

}

return 0;

}

**Output:**

**Conclusion:**

The implementation of FCFS and LOOK algorithms demonstrates efficient disk scheduling, optimizing seek time and improving overall system performance.

**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.

#### Given:

* **Cylinders**: 0 to 199
* **Initial Head Position**: 53
* **Request Queue (FIFO)**: 98, 183, 37, 122, 14, 124, 65, 67

#### 1. ****CSCAN (Circular SCAN)****:

In CSCAN, the disk arm moves in one direction (upward) and, upon reaching the last cylinder, jumps back to the first cylinder.

**Steps**:

1. **Current Position**: 53
2. **Service requests in ascending order**:
   * Move to 65 (12 cylinders)
   * Move to 67 (2 cylinders)
   * Move to 98 (31 cylinders)
   * Move to 122 (24 cylinders)
   * Move to 124 (2 cylinders)
   * Move to 183 (59 cylinders)
3. **Jump back to 0** (200 - 183 = 17 cylinders)
4. **Continue servicing remaining requests**:
   * Move to 14 (14 cylinders)
   * Move to 37 (23 cylinders)

**Total Distance**:

* From 53 to 65: 12
* From 65 to 67: 2
* From 67 to 98: 31
* From 98 to 122: 24
* From 122 to 124: 2
* From 124 to 183: 59
* Jump back to 0: 17
* From 0 to 14: 14
* From 14 to 37: 23

**Total Distance = 12 + 2 + 31 + 24 + 2 + 59 + 17 + 14 + 23 = 184 cylinders**

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.

In LOOK, the disk arm moves in one direction until the last request in that direction, then reverses direction.

**Steps**:

1. **Current Position**: 53
2. **Service requests in ascending order**:
   * Move to 65 (12 cylinders)
   * Move to 67 (2 cylinders)
   * Move to 98 (31 cylinders)
   * Move to 122 (24 cylinders)
   * Move to 124 (2 cylinders)
   * Move to 183 (59 cylinders)
3. **Reverse direction and service remaining requests**:
   * Move to 14 (169 cylinders back to 14 from 183)
   * Move to 37 (23 cylinders)

**Total Distance**:

* From 53 to 65: 12
* From 65 to 67: 2
* From 67 to 98: 31
* From 98 to 122: 24
* From 122 to 124: 2
* From 124 to 183: 59
* From 183 to 14: 169
* From 14 to 37: 23

**Total Distance = 12 + 2 + 31 + 24 + 2 + 59 + 169 + 23 = 322 cylinders**

**Post Lab Objective Questions**

1. In a hard disk, what rotates about a central spindle
   1. Disk
   2. Platter
   3. Sector
   4. None of the above

**Ans: a] Disk**

1. The time required to move the disk arm to the required track is known as
   1. Latency time
   2. Access time
   3. Seek time
   4. None of the above

**Ans: c] Seek Time**

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**