JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY,NOIDA

OS PROJECT

ON

DISK SCHEDULING



***INTRODUCTION***

* ***In operating system disk scheduling is major concern to access the hard disk. Because of the using disk scheduling we can access or speed up the access hard disk.***
* ***Since all device requests are linked in queues, the seek time is increased causing the system to slow down. Disk Scheduling Algorithms are used to reduce the total seek time of any request.***
* *The purpose of this project is to provide one with help on disk scheduling algorithms.*

***DISK STRUCTURE***

* *It consists of platters that spin around the spindle.*
* *The two surfaces of a platter are covered with a magnetic material similar to that on a magnetic tape.*
* *Information is recorded magnetically on the disk.*
* *The surface of a platter is logically divided into circular tracks, which are subdivided into sectors.*
* *The arm assembly is moved in or out to position a head on a desired track.*
* *Only one disk head reads/writes at any time.*

***APPLICATION IN REAL WORLD***

***Disk scheduling is studied or use because of the it provide the better solution to minimize the access time required to read/write data to hard disk. If we not use disk scheduling then read/write time is more and overhead. But operating system is responsible for managing the hardware efficiently that’s why we use disk scheduling.***

***Total time required to access the hard disk is calculated as following***

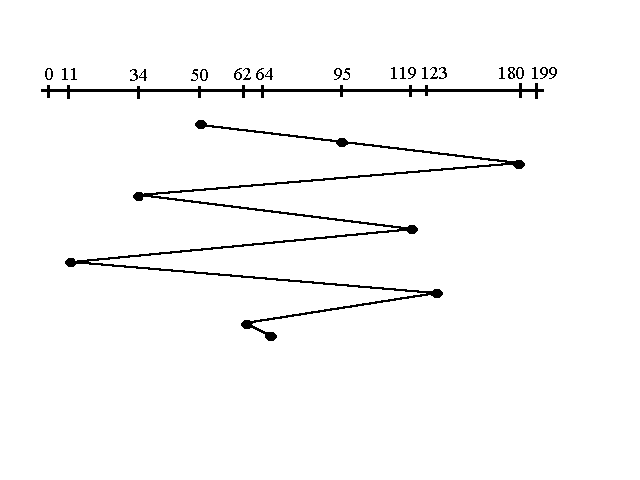
***Total Time required to access= Seek time + Rotation Latency + Transfer time.***

***HOW TO CHOOSE BETTER ALGORITHM?***

* *With low load on the disk, It’s FCFS anyway*
* *SSTF is common and has a natural appeal – good for medium disk load.*
* *SCAN and C-SCAN perform better for systems that place a heavy load on the disk.*
* *Either SSTF or LOOK is a reasonable choice for the default algorithm*

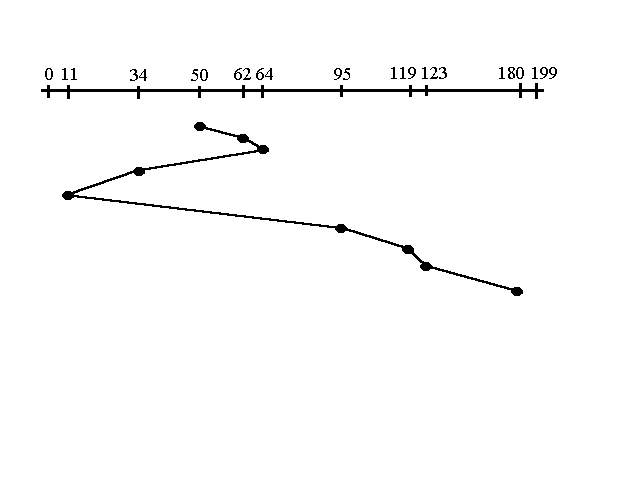
## ***First Come First Serve (FCFS)***

* *Jobs are executed on first come, first serve basis.*
* *Easy to understand and implement.*
* *Its implementation is based on FIFO queue.*
* *Poor in performance as average wait time is high.*
* *With low load on the disk, it’s FCFS always.*

**

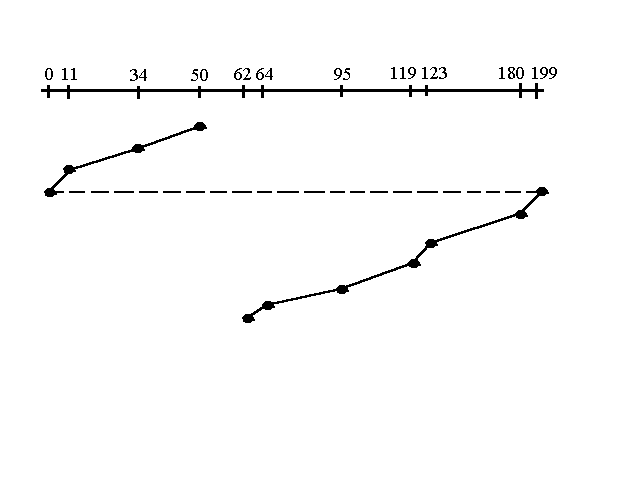
## ***Shortest Seek Time First (SSTF)***

* *This is also known as****shortest job first****, or SJF*
* *Best approach to minimize waiting time.*
* *Easy to implement in Batch systems where required CPU time is known in advance.*
* *Impossible to implement in interactive systems where required CPU time is not known.*
* *SSTF is common and has a natural appeal – good for medium disk load.*

******

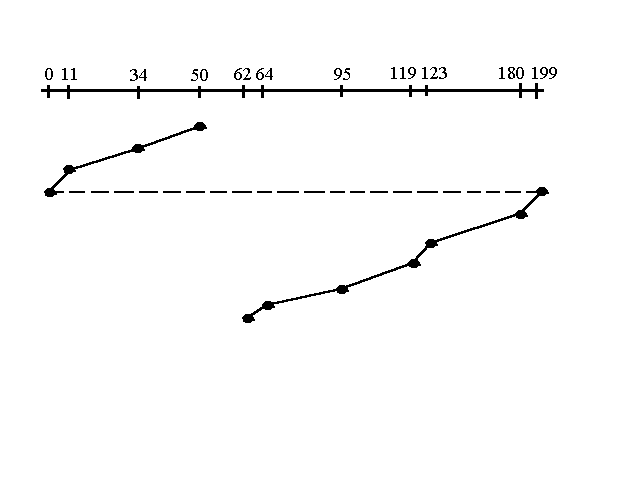
***SCAN***

* *The disk arm starts at one end of the disk, and moves toward that end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.*
* *If a request comes in after it has been scanned it will not be serviced until the process comes back down or moves back up.*
* *Sometimes it is also called the elevator algorithm.*



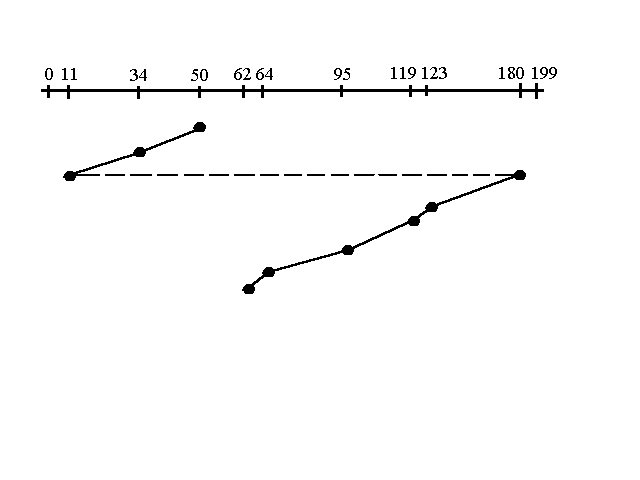
***C-SCAN***

* *Circular Scan Algorithm.*
* *Provides a more uniform wait time than SCAN.*
* *The head moves from one end of the disk to the other. servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.*
* *Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.*



***LOOK***

* *Version of SCAN.*
* This algorithm is similar to SCAN algorithm except for the end-to-end reach of each sweep.
* The R/W head is only tasked to go the farthest location in need of servicing.
* This is also a directional algorithm, as soon as it is done with the last request in one direction it then sweeps in the other direction.



***C-LOOK***

* *Circular Look Algorithm.*
* *Version of C-SCAN*
* *Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.*
* *Look for a request before continuing to move in a given direction.*

