Linux(Fedora) I/O Management and Disk Scheduling

Submitted by :

Rafid Faeq Ahmed st.no (163104081)

Email rafid.faeq@gmail.com

→ – Operating System Design Issues

– I/O Management (Buffering)

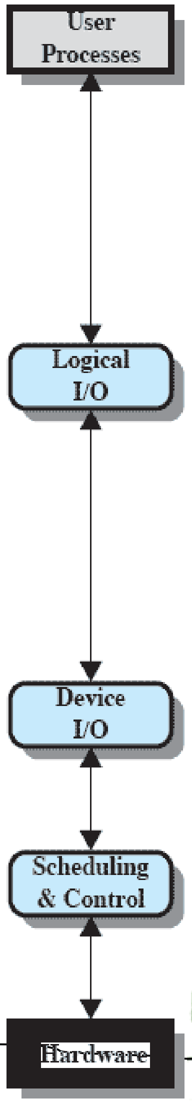
– Disk Scheduling

Goal: Generality

- For simplicity and freedom from error, it's better to handle all I/O devices in a uniform manner
- Due to the diversity of device characteristics, it is difficult in practice to achieve true generality
- Solution: use a hierarchical modular design of I/O functions
  - Hide details of device I/O in lower-level routines
  - User processes and upper levels of OS see devices in terms of general functions, such as read, write, open, close, lock, unlock

A Model of I/O Organization

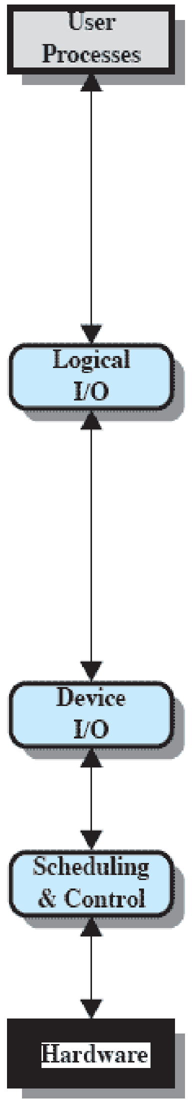
- Logical I/O:
  - Deals with the device as a logical resource and is not concerned with the details of actually controlling the device
  - Allows user processes to deal with the device in terms of a device identifier and simple commands such as open, close, read, write
- Device I/O:
  - Converts requested operations into sequence of I/O instructions
  - Uses buffering techniques to improve utilization



(a) Local peripheral device

A Model of I/O Organization

- Scheduling and Control:
  - Performs actual queuing / scheduling and control operations
  - Handles interrupts and collects and reports I/O status
  - Interacts with the I/O module and hence the device hardware



(a) Local peripheral device

Goal: Efficiency

- Most I/O devices are extremely slow compared to main memory

  → I/O operations often form a bottleneck in a computing system

- Multiprogramming allows some processes to be waiting on I/O while another process is executing

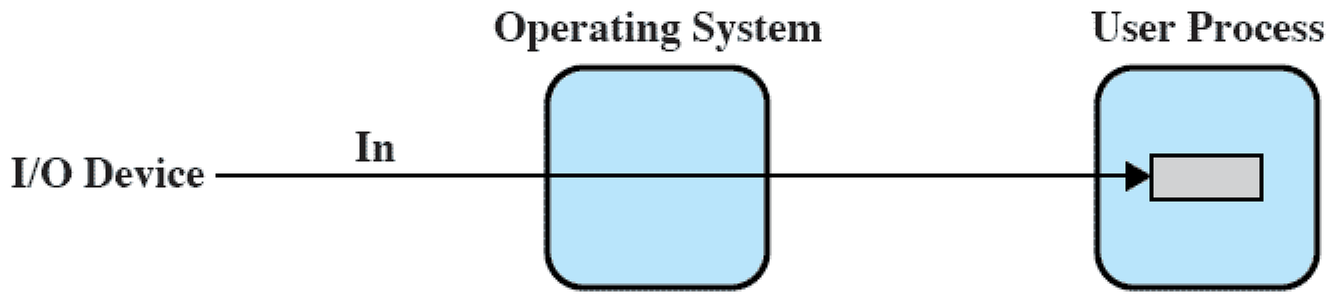
Goal: Efficiency

- Swapping brings in ready processes but this is an I/O operation itself
- A major effort in I/O design has been schemes for improving the efficiency of I/O
  - I/O buffering
  - Disk scheduling

I/O Management (Buffering)

No Buffering

- Without a buffer, OS directly accesses the device as and when it needs
- A data area within the address space of the user process is used for I/O



(a) No buffering

No Buffering

- Process must wait for I/O to complete before proceeding
  - busy waiting (like programmed I/O)
  - process suspension on an interrupt (like interrupt-driven I/O or DMA)
- 👎Problems
  - the program is hung up waiting for the relatively slow I/O to complete
  - interferes with swapping decisions by OS

I/O Buffering

- It may be more efficient to perform input transfers in advance of requests being made and to perform output transfers some time after the request is made.

Block-oriented Buffering

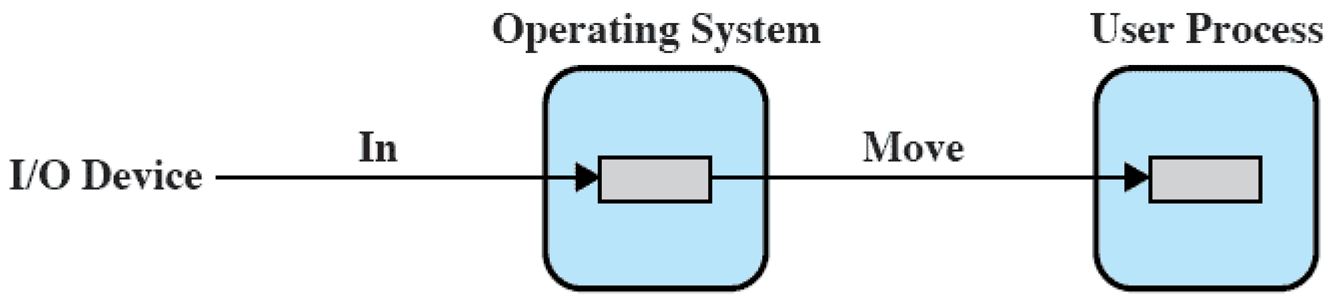
- For block-oriented I/O devices such as
  - disks and
  - USB drives
- Information is stored in fixed sized blocks
- Transfers are made a block at a time
- Can reference data by block number

Stream-Oriented Buffering

- For stream-oriented I/O devices such as
  - terminals
  - printers
  - communication ports
  - mouse and other pointing devices, and
  - most other devices that are not secondary storage
- Transfer information as a stream of bytes

Single Buffer

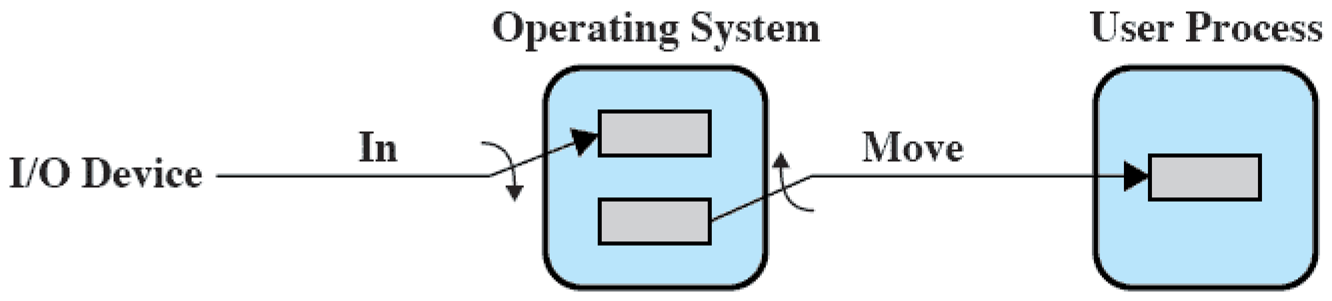
- OS assigns a buffer in the system portion of main memory for an I/O request



(b) Single buffering

Double Buffer

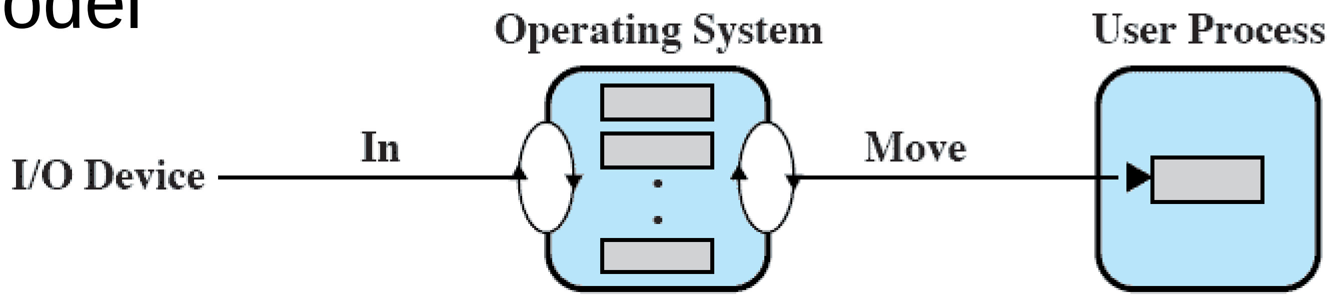
- Use two system buffers instead of one
- A process can transfer data to or from one buffer while OS empties or fills the other buffer



(c) Double buffering

Circular Buffer

- More than two buffers are used
- Each individual buffer is one unit in a circular buffer
- Used when I/O operation must keep up with process
- Follows the bounded-buffer producer/consumer model



(d) Circular buffering

– Disk Scheduling

Disk Performance Parameters

- Currently, disks are at least four orders of magnitude slower than main memory
  → performance of disk storage subsystem is of vital concern
- A general timing diagram of disk I/O transfer is shown here.

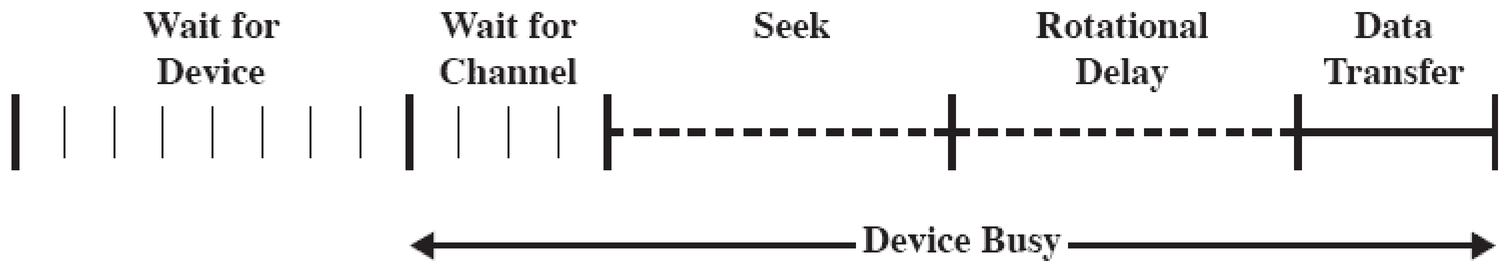


Figure 11.6 Timing of a Disk I/O Transfer

Disk Performance Parameters

- ***Access Time*** is the sum of:
  - ***Seek time:*** The time it takes to position the head at the desired track
  - ***Rotational delay*** or ***rotational latency:*** The time it takes for the beginning of the sector to reach the head
- ***Transfer Time*** is the time taken to transfer the data (as the sector moves under the head)

Disk Performance Parameters

- Total average access time T_a

$$T_a = T_s + 1 / (2r) + b / (rN)$$

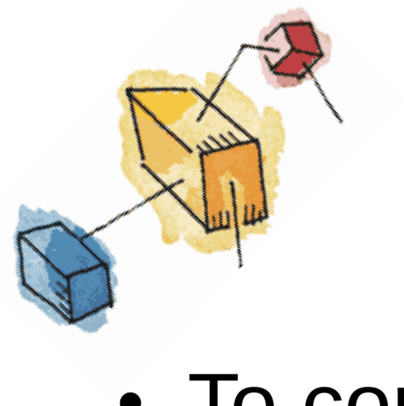
where T_s = average seek time

b = no. of bytes to be transferred

N = no. of bytes on a track

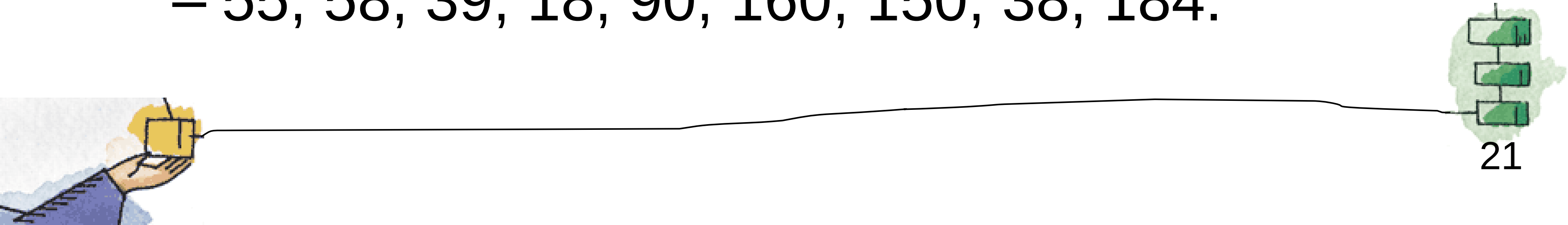
r = rotation speed, in revolutions / sec.

- Due to the seek time, the order in which sectors are read from disk has a tremendous effect on I/O performance



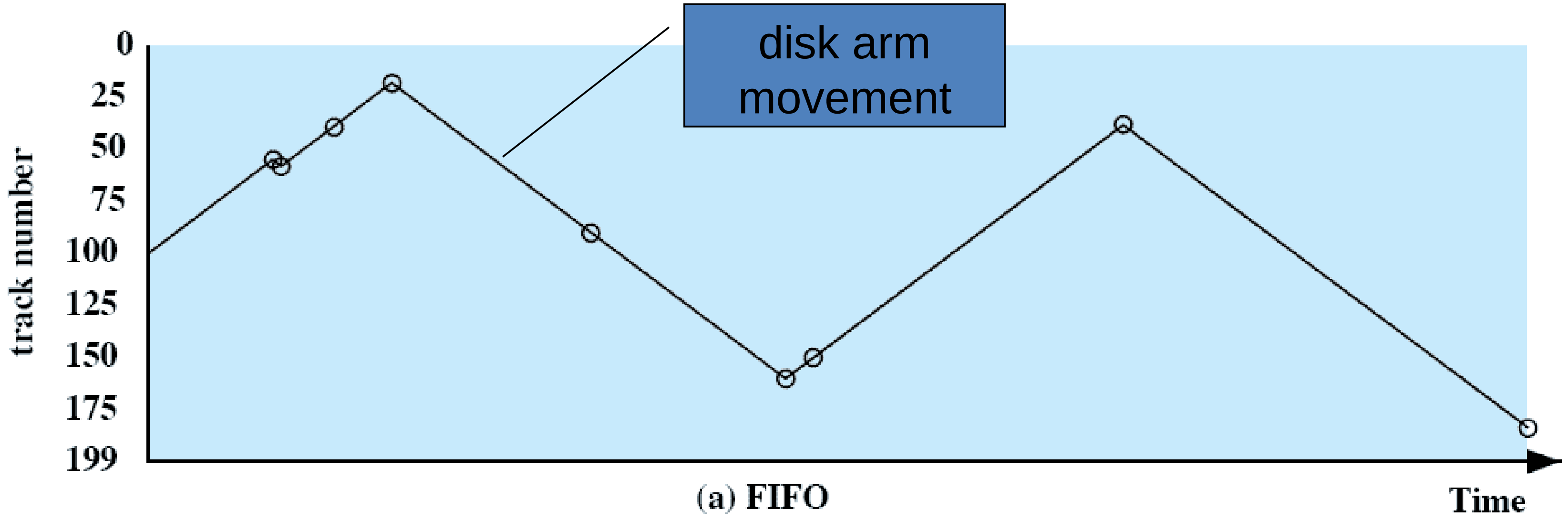
Disk Scheduling Policies

- To compare various schemes, consider a disk head is initially located at track 100.
  - assume a disk with 200 tracks and that the disk request queue has random requests in it.
- The requested tracks, in the order received by the disk scheduler, are
  - 55, 58, 39, 18, 90, 160, 150, 38, 184.



First-in, first-out (FIFO)

- Process requests sequentially
- Fair to all processes
- May have good performance if most requests are to clustered file sectors
- Approaches random scheduling in performance if there are many processes



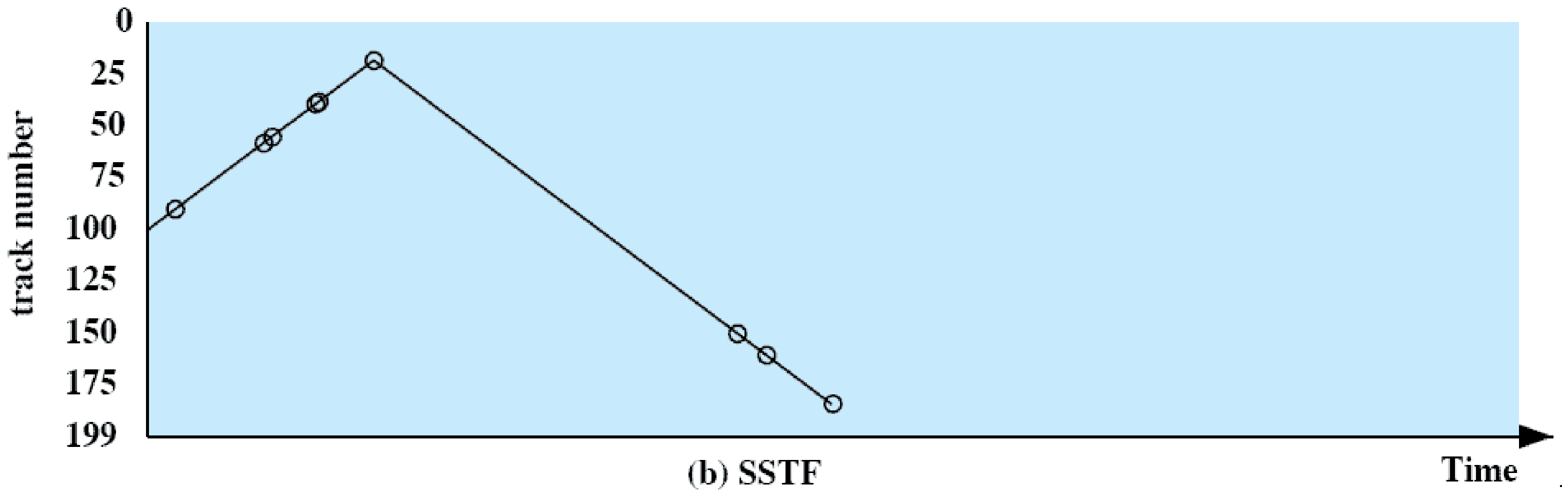
(a) FIFO

Last-in, first-out

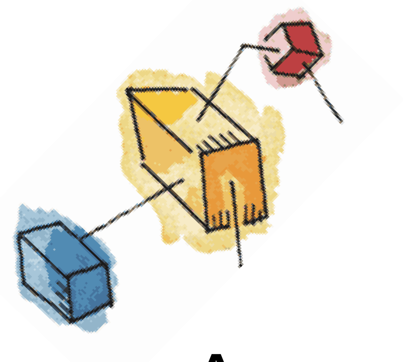
- Good for transaction processing systems
  - The device is given to the most recent user so there should be little arm movement for moving through a sequential file
- Possibility of starvation since a job may never regain the head of the line

Shortest Service Time First

- Select the disk I/O request that requires the least movement of the disk arm from its current position
- Always choose the minimum seek time

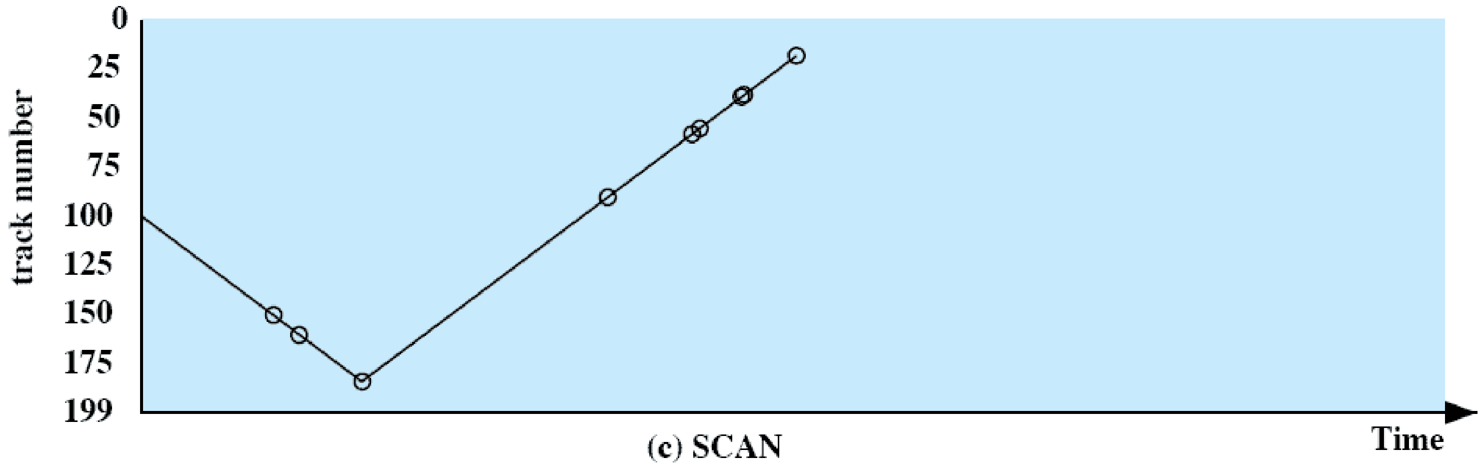


(b) SSTF



SCAN

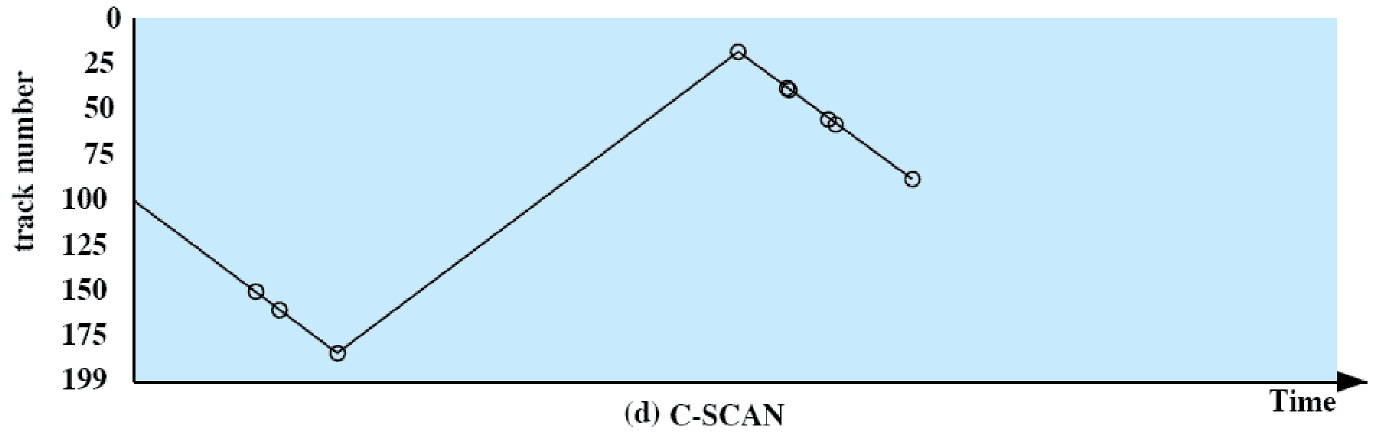
- Arm moves in one direction only, satisfying all outstanding requests until it reaches the last track in that direction then the direction is reversed
- LOOK policy: reverse direction when there are no more requests in a direction



(c) SCAN

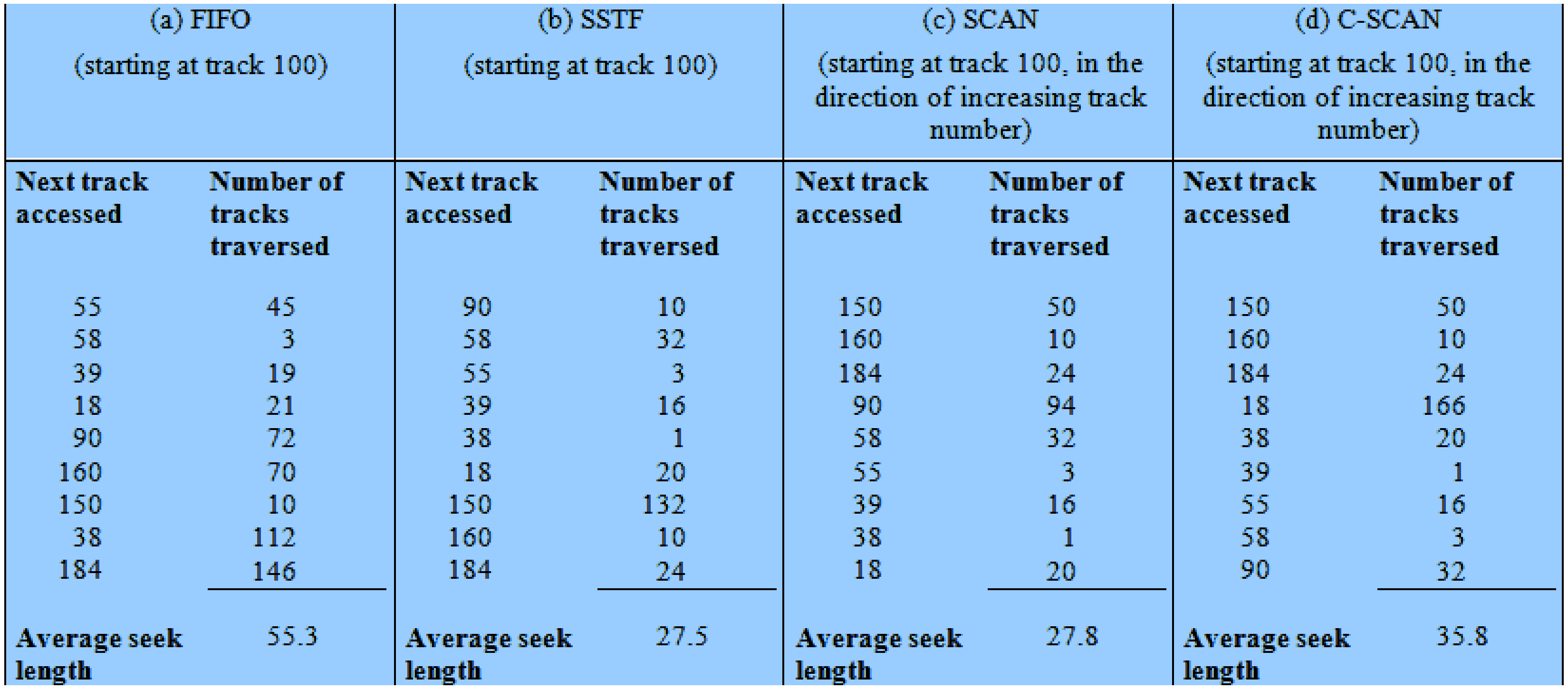
C-SCAN (Circular SCAN)

- Restricts scanning to one direction only
- When the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again
- Reduces the maximum delay experienced by new requests



Performance Compared

Comparison of Disk Scheduling Algorithms



(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	**Average seek length**	27.5	**Average seek length**	27.8	**Average seek length**	35.8

Thank you