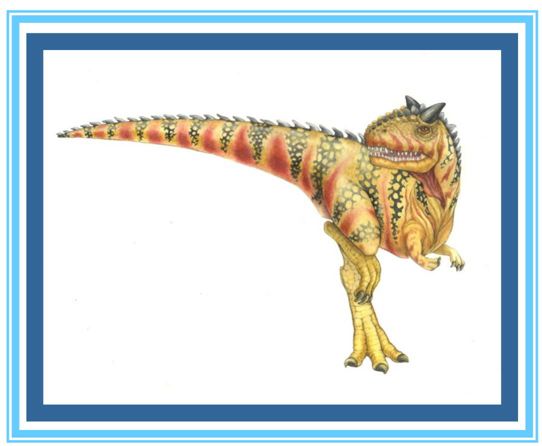
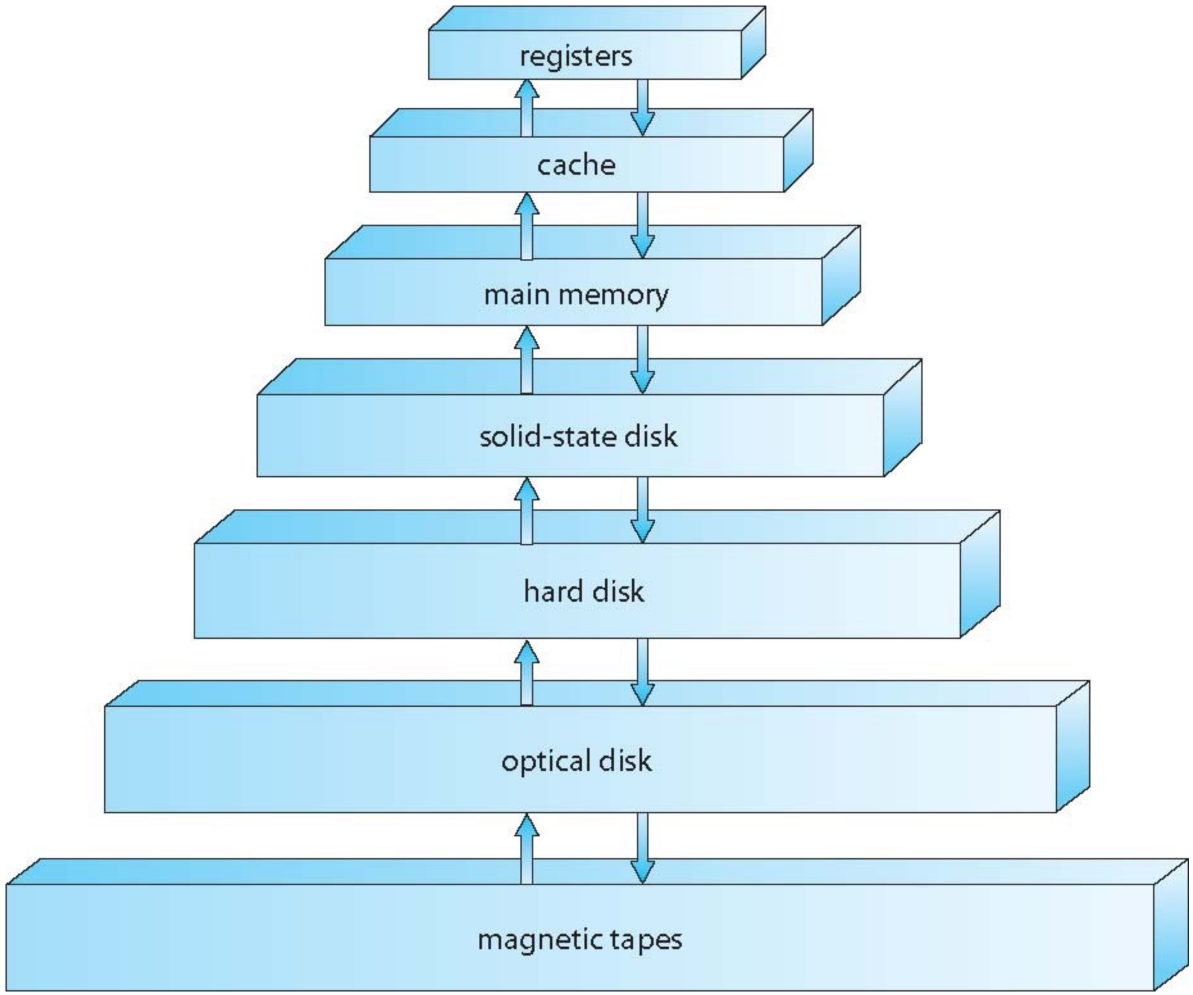
Chapter 11: Mass-Storage Systems



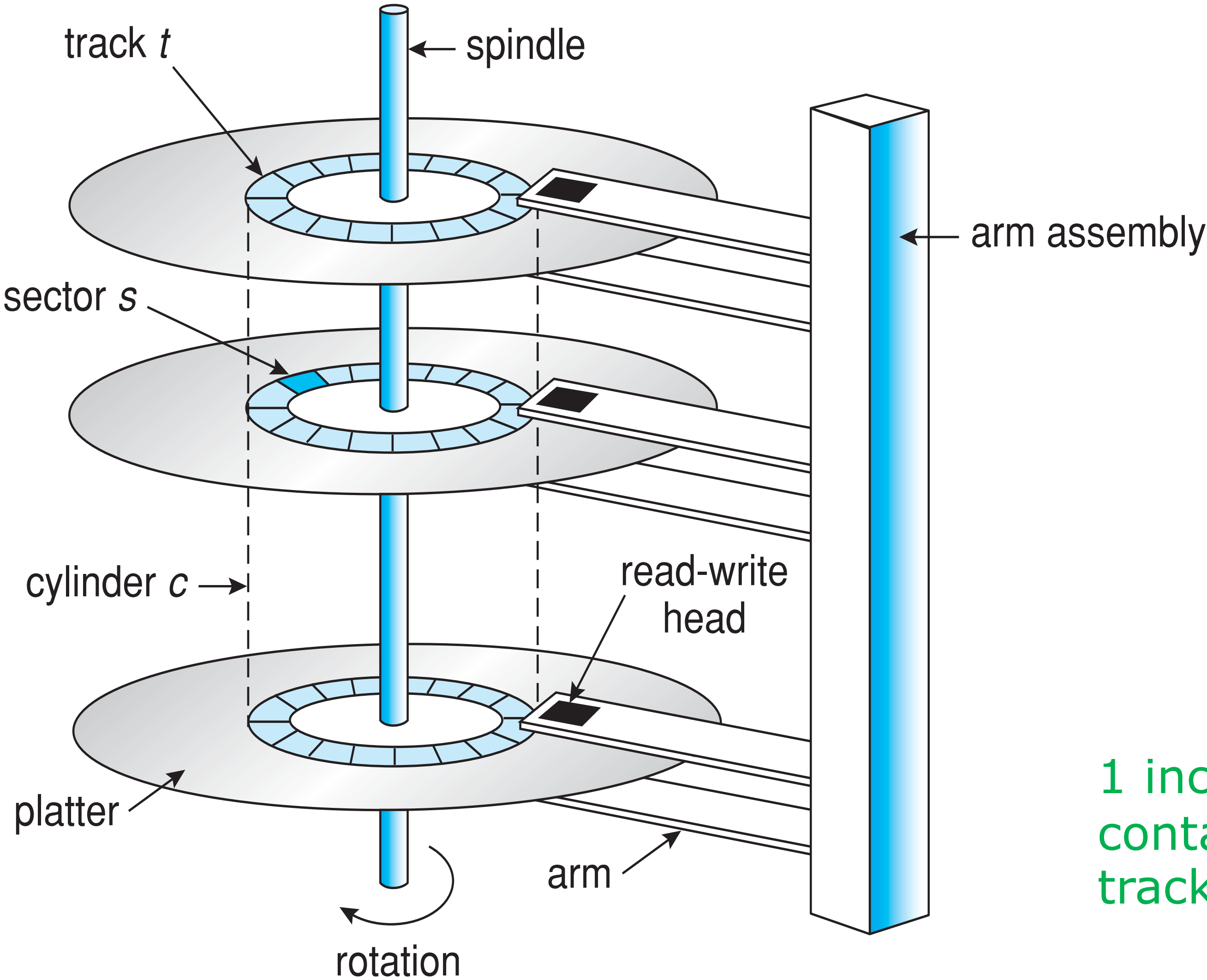
Objectives

- To describe the physical structure of secondary storage devices and its effects on the uses of the devices
- To explain the performance characteristics of mass-storage devices
- To evaluate disk scheduling algorithms

Storage-Device Hierarchy



Moving-head Disk Mechanism



1 inch of a disk platter contains up to 10K tracks

More information:
https://www.youtube.com/watch?v=wteUW2sL7bc
https://www.youtube.com/watch?v=NtPc0jI21i0&t=303s

Overview of Mass Storage Structure

- **Magnetic disks** provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 250 times per second
  - **Transfer rate** is rate at which data flow between drive and computer
  - **Positioning time** (**random-access time**) is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**)
  - **Head crash** results from disk head making contact with the disk surface -- That's bad
- Disks can be removable
- Drive attached to computer via **I/O bus**

Hard Disks

- Platters range from .85" to 14" (historically)
  - Commonly 3.5", 2.5", and 1.8"
- Range from 30GB to 3TB per drive
- Performance
  - Transfer Rate – theoretical – 6 Gb/sec
  - Effective Transfer Rate – real – 1Gb/sec
  - Seek time from 3ms to 12ms – 9ms common for desktop drives
  - Average seek time measured or calculated based on 1/3 of tracks
  - Latency based on spindle speed
    - 1 / (RPM / 60) = 60 / RPM (in seconds)
  - Average latency = ½ latency

Spindle [rpm]	Average latency [ms]
4200	7.14 (60/4200)/2
5400	5.56
7200	4.17
10000	3
15000	2

(From Wikipedia)

Hard Disk Performance

- **Access Latency** = **Average access time** = average seek time + average latency
  - For fastest disk 3ms + 2ms = 5ms
  - For slow disk 9ms + 5.56ms = 14.56ms
- Average I/O time = average access time + (amount to transfer / transfer rate) + controller overhead
- For example to transfer a 4KB block on a 7200 RPM disk with a 5ms average seek time, 1Gb/sec transfer rate with a .1ms controller overhead =
  - 5 ms + **60/7200 s** + 0.1 ms + transfer time
  - 5ms + **4.17ms** + 0.1ms + transfer time
  - transfer time = 4KB / 1Gb/s * 1024^2KB = 4 / (1024^2) = 0.0031 ms
  - Average I/O time for 4KB block = 9.27ms + .1031ms = 9.3731ms

The First Commercial Disk Drive

1956
IBM RAMDAC computer included the IBM Model 350 disk storage system

5M (7 bit) characters
50 x 24” platters
Access time = < 1 second

Solid-State Disks

- Nonvolatile memory (NVM) used like a hard drive
  - Many technology variations
  - Composed of a controller and **flash NAND die semiconductor chips**
  - NAND does not allow data overwritten
- Can be more reliable than HDDs
- More expensive per MB
- Maybe have shorter life span (measured in ***Drive Writes Per Day*** (***DWPD***).)
  - For example, a 1 TB NAND drive with a 5 DWPD rating is expected to have 5 TB per day written to it for the warranty period without failure.
- Less capacity
- But much faster
  - No moving parts, so no seek time or rotational latency

Disk Structure – Address Mapping

- Disk drives are addressed as large 1-dimensional arrays of **logical blocks**, where the logical block is the smallest unit of transfer
  - Low-level formatting creates **logical blocks** on physical media
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially
  - Sector 0 is the first sector of the first track on the outermost cylinder
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost
  - Logical to physical address should be easy
    - Except for bad sectors
    - Non-constant # of sectors per track via constant angular velocity

Disk Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth
- Minimize seek time
- Seek time ≈ seek distance
- 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

Disk Scheduling (Cont.)

- There are many sources of disk I/O request
  - OS
  - System processes
  - Users processes
- I/O request includes input or output mode, disk address, memory address, number of sectors to transfer
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, busy disk means work must queue
  - Optimization algorithms only make sense when a queue exists

Disk Scheduling (Cont.)

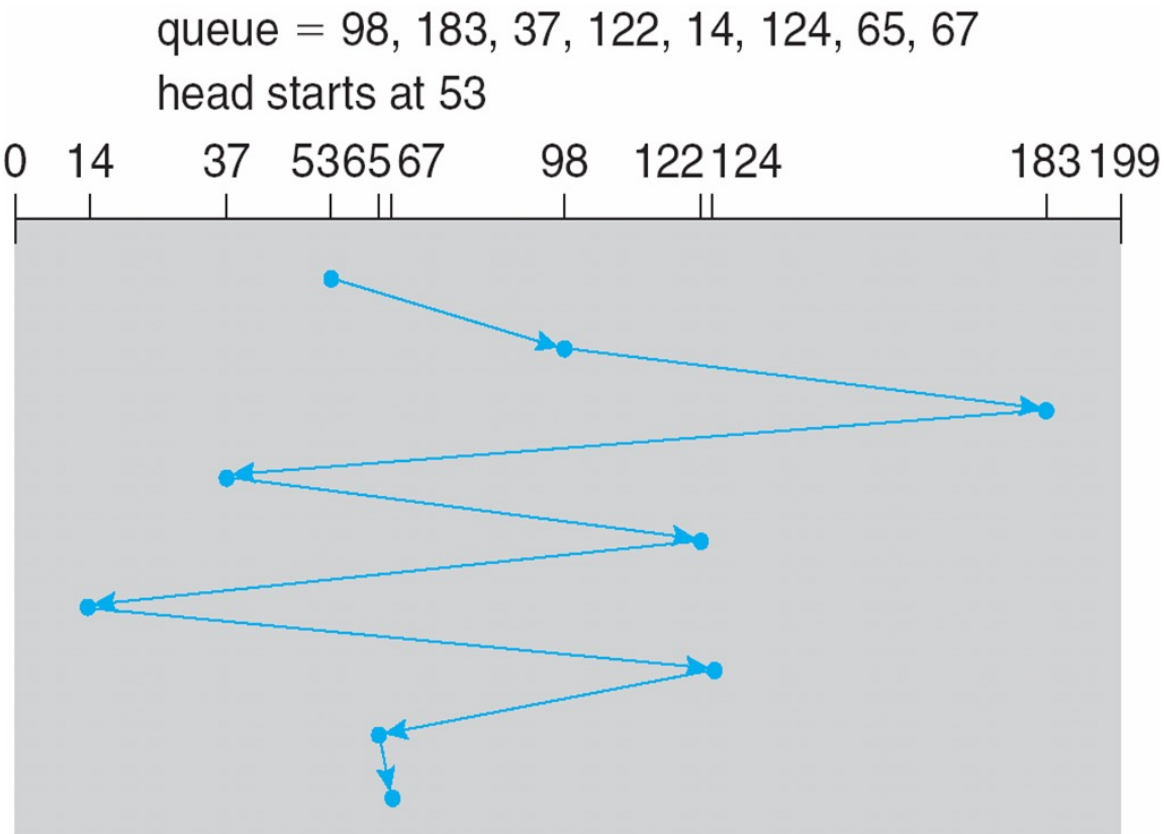
- Note that drive controllers have small buffers and can manage a queue of I/O requests (of varying "depth")
- Several algorithms exist to schedule the servicing of disk I/O requests
- The analysis is true for one or many platters
- We illustrate scheduling algorithms with a request queue (0-199)

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

Head pointer 53

FCFS

Illustration shows total head movement of 640 cylinders



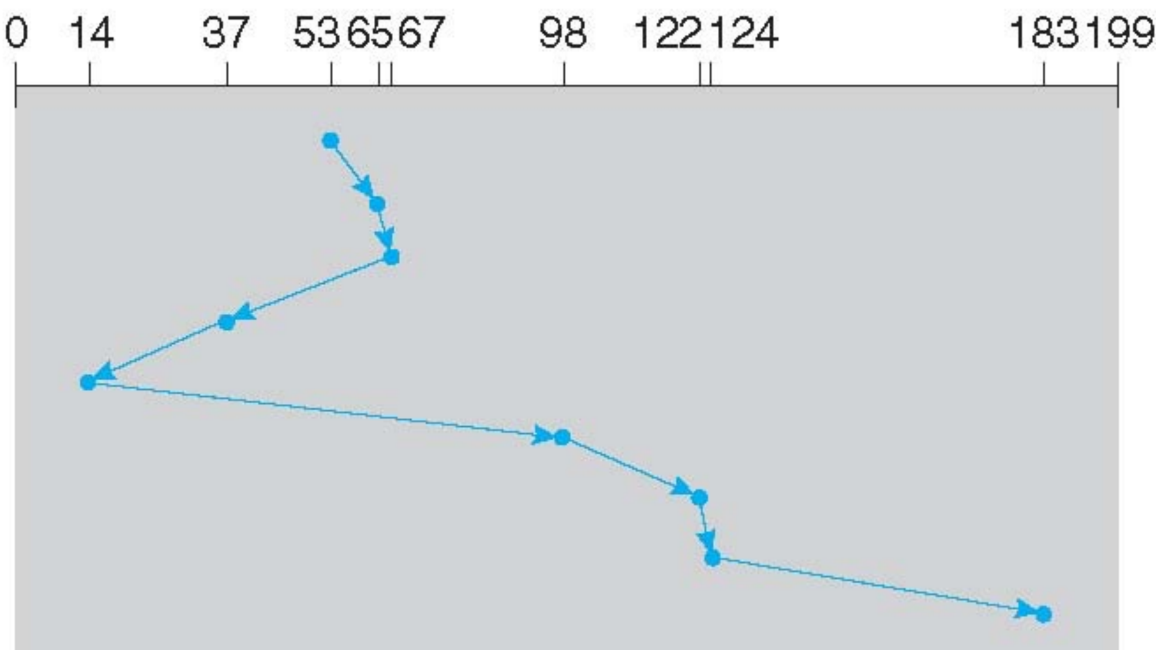
Start	End	# Movement
53	98	45
98	183	85
183	37	146
37	122	85
122	14	108
14	124	110
124	65	59
65	67	2
Total Movement		**640**

SSTF

- Shortest Seek Time First selects the request with the minimum seek time from the current head position
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests
- Illustration shows total head movement of **236 cylinders**

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53

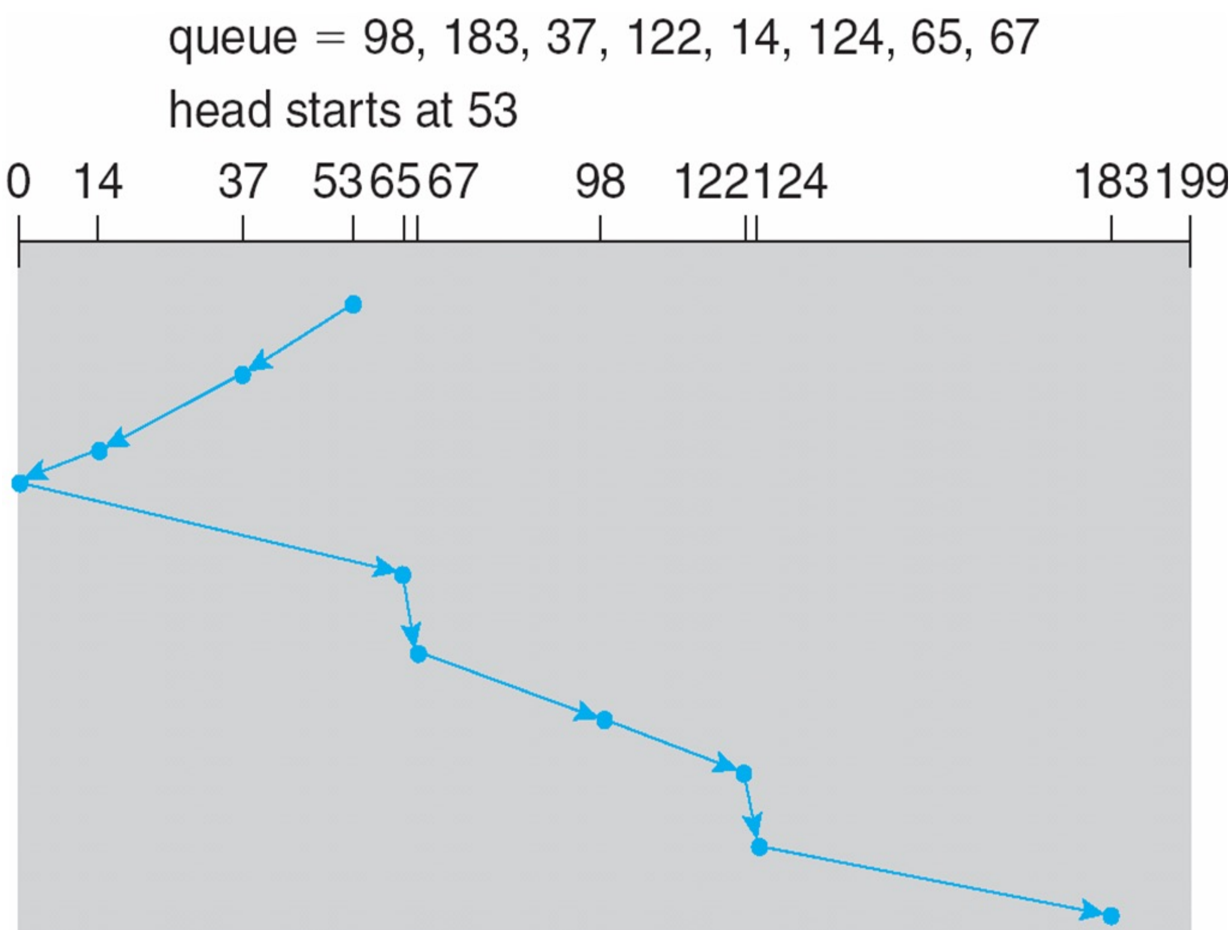


Start	End	# Movement
53	65	12
65	67	2
67	37	30
37	14	23
14	98	84
98	122	24
122	124	2
124	183	59
Total Movement		**236**

SCAN

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- **SCAN algorithm** Sometimes called the **elevator algorithm**
- Illustration shows total head movement of 236 cylinders
- But note that if requests are uniformly dense, largest density at other end of disk and those wait the longest

SCAN (Cont.)

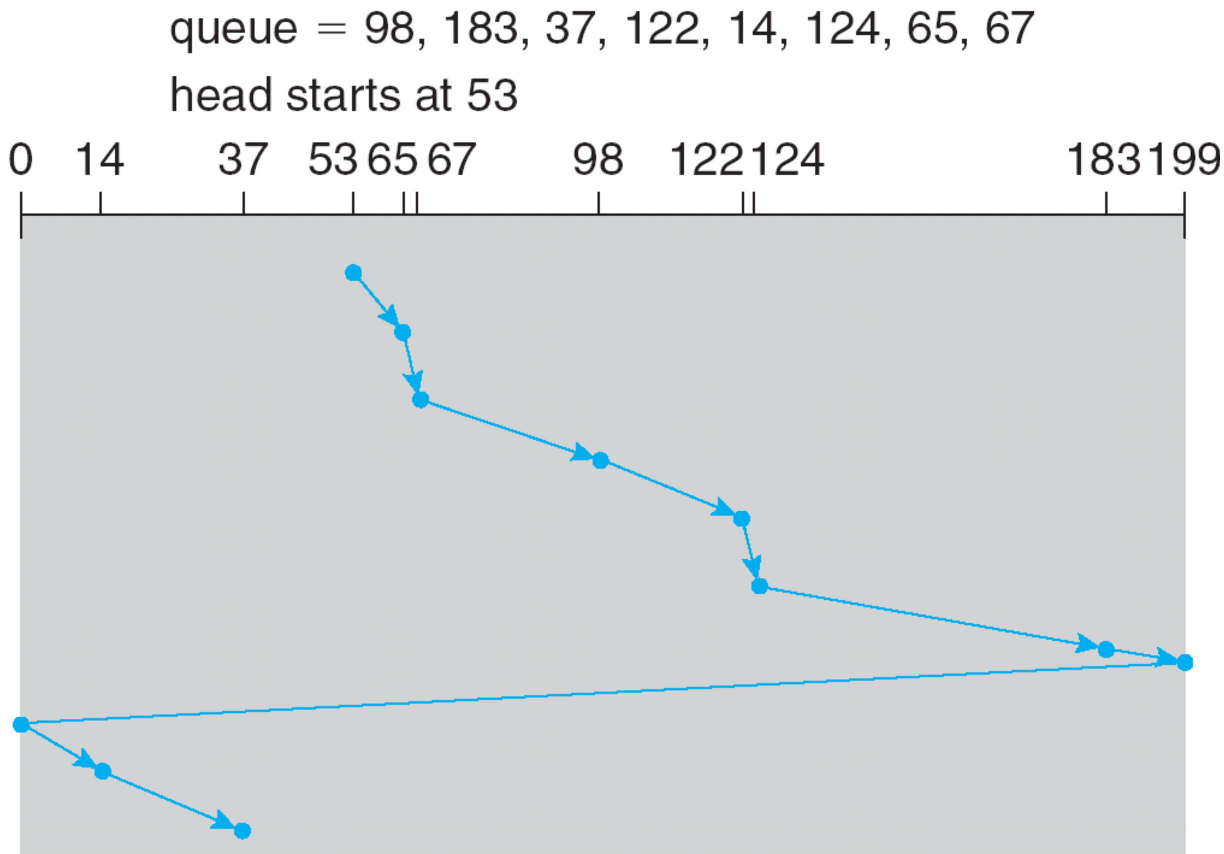


Start	End	# Movement
53	37	16
37	14	23
14	**0**	14
0	65	65
65	67	2
67	98	31
98	122	24
122	124	2
124	183	59
Total Movement		**236**

C-SCAN

- 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
- Total number of cylinders?

C-SCAN (Cont.)



Start	End	# Movement
53	65	12
65	67	2
67	98	31
98	122	24
122	124	2
124	183	59
183	199	16
199	**0**	****0**
0	14	14
14	37	23
Total Movement		**183**

**The head movement when its not servicing any requests is not counted.

Check your understanding (1)

1) Consider a disk queue holding requests to the following cylinders in the listed order: 116, 22, 3, 11, 75, 185, 100, 87. Using the FCFS scheduling algorithm, what is the order that the requests are serviced, assuming the disk head is at cylinder 88 and moving upward through the cylinders?

- ○ 116 - 22 - 3 - 11 - 75 - 185 - 100 - 87
- ○ 100 - 116 - 185 - 87 - 75 - 22 - 11 - 3
- ○ 87 - 75 - 100 - 116 - 185 - 22 - 11 - 3

What is the total movement?

Check your understanding (2)

2) Consider a disk queue holding requests to the following cylinders in the listed order: 116, 22, 3, 11, 75, 185, 100, 87. Using the SCAN scheduling algorithm, what is the order that the requests are serviced, assuming the disk head is at cylinder 88 and moving upward through the cylinders?

- ○ 116 - 22 - 3 - 11 - 75 - 185 - 100 - 87
- ○ 100 - 116 - 185 - 87 - 75 - 22 - 11 - 3
- ○ 100 - 116 - 185 - 3 - 11 - 22 - 75 - 87

What is the total movement?

Check your understanding (3)

3) Consider a disk queue holding requests to the following cylinders in the listed order: 116, 22, 3, 11, 75, 185, 100, 87. Using the C-SCAN scheduling algorithm, what is the order that the requests are serviced, assuming the disk head is at cylinder 88 and moving upward through the cylinders?

- ○ 116 - 22 - 3 - 11 - 75 - 185 - 100 - 87
- ○ 100 - 116 - 185 - 87 - 75 - 22 - 11 - 3
- ○ 100 - 116 - 185 - 3 - 11 - 22 - 75 - 87

What is the total movement?

Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk
  - Less starvation
- Performance depends on the number and types of requests
- Requests for disk service can be influenced by the file-allocation method
  - And metadata layout
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary

Selecting a Disk-Scheduling Algorithm

- To avoid starvation Linux implements deadline scheduler
  - Maintains separate read and write queues, gives read priority
  - Implements four queues: 2 x read and 2 x write
    - 1 read and 1 write queue sorted in Logic Block Addressing (LBA) order, essentially implementing C-SCAN
    - 1 read and 1 write queue sorted in FCFS order
    - All I/O requests sent in batch sorted in that queue's order
    - After each batch, checks if any requests in FCFS older than configured age (default 500ms)
    - If so, LBA queue containing that request is selected for next batch of I/O

LBA numbering starts with the first cylinder, first head, and track's first sector

14, 37, 59, 101, ….