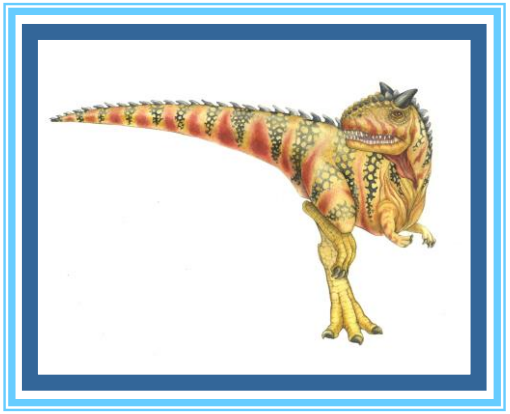
Chapter 10: Mass-Storage Systems



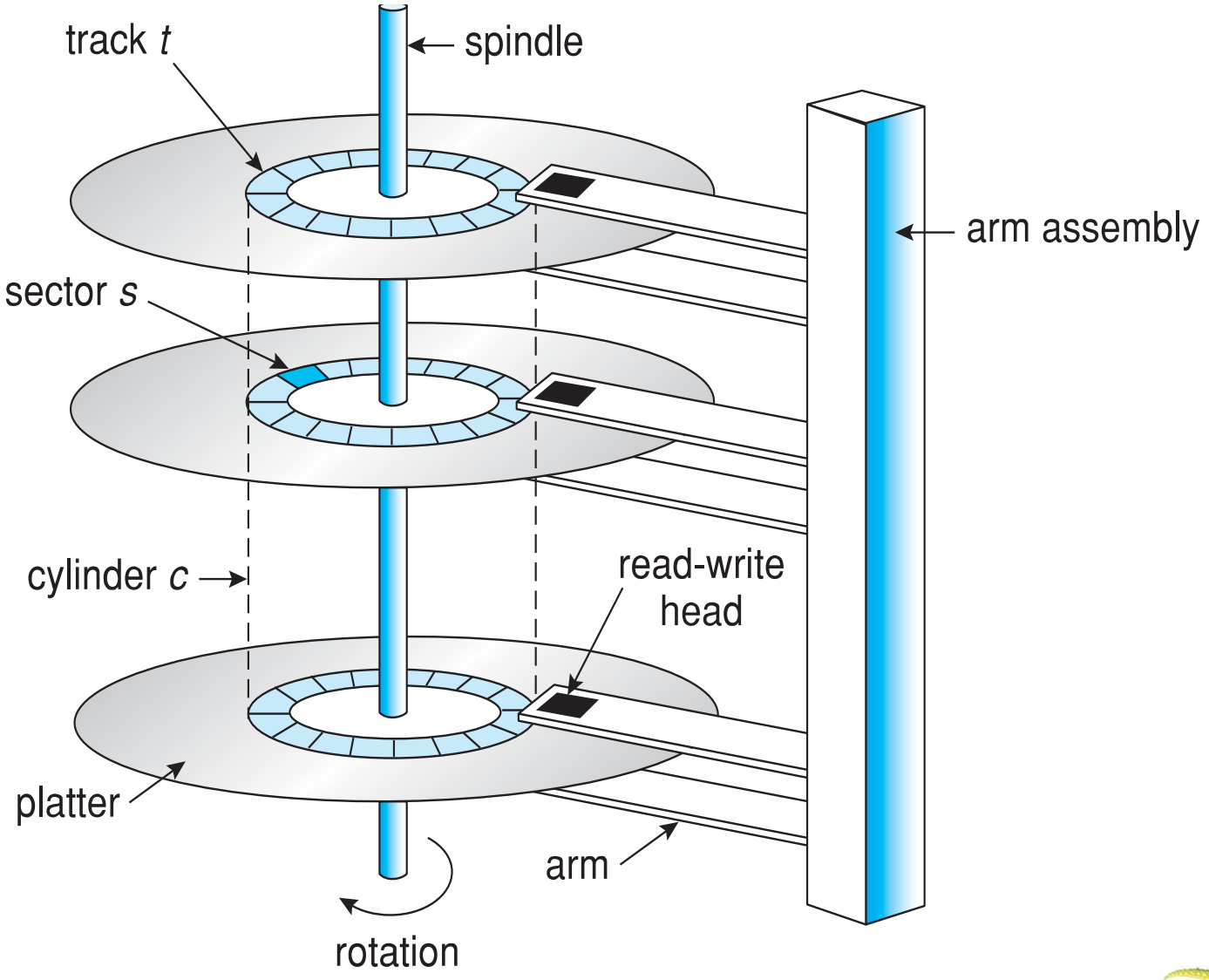
Objectives

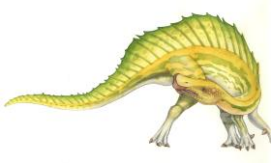
- To describe the physical structure of secondary storage devices and its effects on the uses of the devices
- To explain the performance characteristics of mass-storage devices
- To evaluate disk scheduling algorithms

Overview of Mass Storage Structure

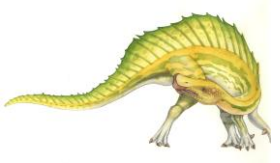
- **Magnetic disks** provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 250 times per second
  - Disk speed has two parts
    - **Transfer rate** is rate at which data flow between drive and computer
    - **Positioning time** (**random-access time**)
      - **seek time** : time to move disk arm to desired cylinder
      - **rotational latency :** time for desired sector to rotate under the disk head
  - **Head crash** results from disk head making contact with the disk surface -- That's bad

Moving-head Disk Mechanism



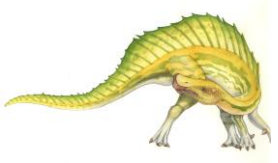


- The heads are attached to a **disk arm** that moves all the heads as a unit.
- The surface of a platter is logically divided into circular **tracks**, which are subdivided into **sectors**.
- The set of tracks that are at one arm position makes up a **cylinder**.
- There may be thousands of concentric cylinders in a disk drive, and each track may contain hundreds of sectors.
- The storage capacity of common disk drives is measured in gigabytes.

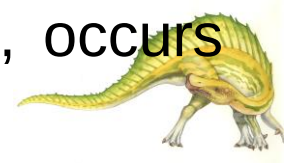


Overview of Mass Storage Structure

- Disks can be removable
  - Removable magnetic disks generally consist of one platter, held in a plastic case to prevent damage while not in the disk drive.
  - Other forms of removable disks include CDs, DVDs, Blu-ray discs, flash-memory devices known as **flash drives**
- Drive attached to computer via **I/O bus**
  - Buses vary, including **EIDE**, **ATA**, **SATA**, **USB**, **Fibre Channel**



- The data transfers on a bus are carried out by special electronic processors called **controllers**.
- The **host controller** is the controller at the computer end of the bus.
- A **disk controller** is built into each disk drive.
- To perform a disk I/O operation, the computer places a command into the HC
- The HC then sends the command via messages to the DC, and the DC operates the disk-drive hardware to carry out the command.
- DC usually have a built-in cache.
- Data transfer at the disk drive happens between the cache and the disk surface, and data transfer to the host, occurs between the cache and the host controller.



Hard Disks

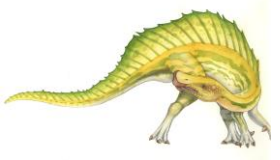
- Platters range from .85” to 14” (historically)
  - Commonly 3.5”, 2.5”, and 1.8”
- Range from 30GB to 3TB per drive
- Performance
  - Transfer Rate – theoretical – 6 Gb/sec
  - Effective Transfer Rate – real – 1Gb/sec
  - Seek time from 3ms to 12ms – 9ms common for desktop drives
  - Average seek time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

Spindle [rpm]	Average latency [ms]
4200	7.14
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 =
  - 5ms + 4.17ms + 0.1ms + transfer time =
  - Transfer time = 4KB / 1Gb/s * 8Gb / GB * 1GB / 1024^2KB = 32 / (1024^2) = 0.031 ms
  - Average I/O time for 4KB block = 9.27ms + .031ms = 9.301ms



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 used like a hard drive
  - Many technology variations
- Can be more reliable than HDDs
- More expensive per MB
- Maybe have shorter life span
- Less capacity
- But much faster
- Buses can be too slow -> connect directly to PCI for example
- No moving parts, so no seek time or rotational latency

Magnetic Tape

- Was early secondary-storage medium
  - Evolved from open spools to cartridges
- Relatively permanent and holds large quantities of data
- Access time slow
- Random access ~1000 times slower than disk
- Mainly used for backup, storage of infrequently-used data, transfer medium between systems
- Kept in spool and wound or rewound past read-write head
- Once data under head, transfer rates comparable to disk
  - 140MB/sec and greater
- 200GB to 1.5TB typical storage
- Common technologies are LTO-{3,4,5} and T10000

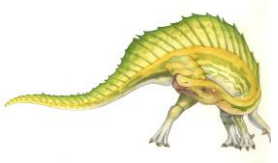
Disk Structure

- 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
    - The number of sectors per track is not a constant on some drives.

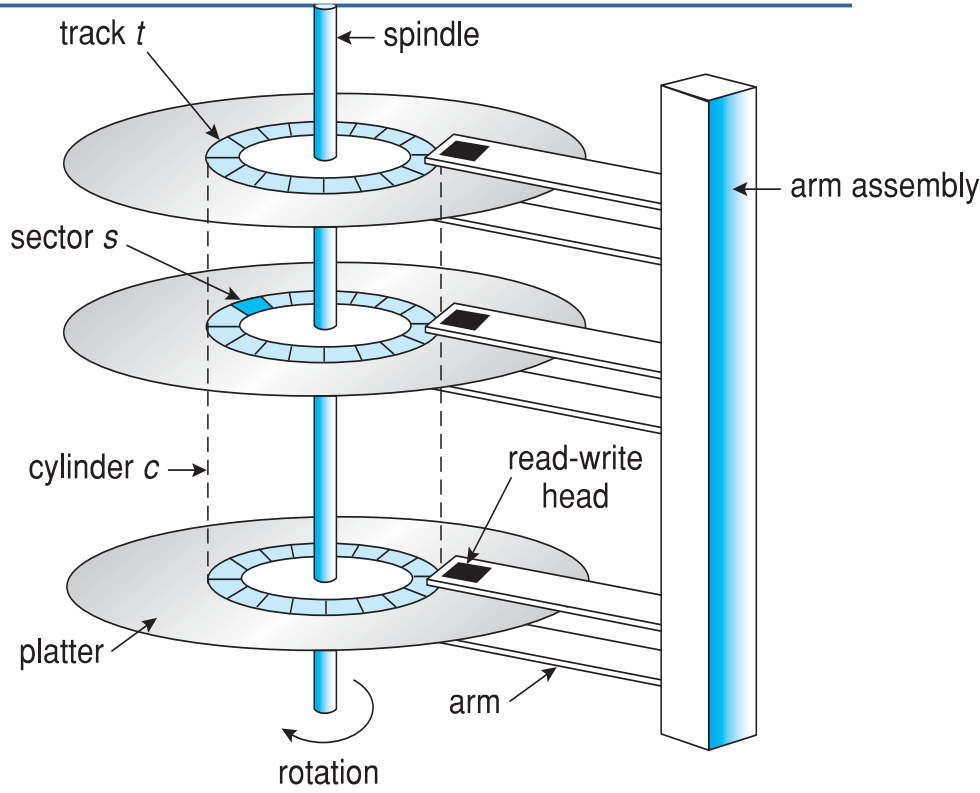
- On media that use **constant linear velocity (CLV)**, the density of bits per track is uniform.
  - The farther a track is from the center of the disk, the greater its length, so the more sectors it can hold.
  - As we move from outer zones to inner zones, the number of sectors per track decreases.
  - The drive increases its rotation speed as the head moves from the outer to the inner tracks to keep the same rate of data moving under the head. This method is used in CD-ROM and DVD-ROM drives.
- Alternatively, the disk rotation speed can stay constant
  - The density of bits decreases from inner tracks to outer tracks to keep the data rate constant.
  - This method is used in hard disks and is known as **constant angular velocity (CAV)**.

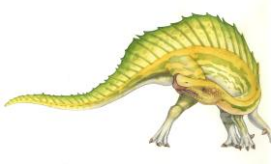
Disk Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth
- Access time has two major components
  - Seek time is the time for the disk arm to move the heads to the cylinder containing the desired sector.
  - Rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head
- 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



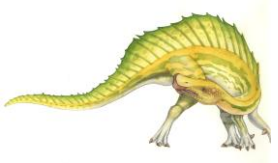
- Seek time : With rotating drives, the *seek time* measures the time it takes the head assembly on the actuator arm to travel to the track of the disk where the data will be read or written.
- Rotational latency is the delay waiting for the rotation of the disk to bring the required disk sector under the read-write head.
  - It depends on the rotational speed of a disk (or spindle motor),measured in revolutions per minute (RPM)





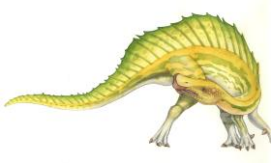
Disk Scheduling (Cont.)

- There are many sources of disk I/O request
  - OS
  - System processes
  - Users processes
- Whenever a process needs I/O to or from the disk, it issues a system call to the operating system.
- The request specifies several pieces of information:
  - Whether this operation is input or output
  - What the disk address for the transfer is
  - What the memory address for the transfer is
  - What the number of sectors to be transferred is



Disk Scheduling (Cont.)

- If the desired disk drive and controller are available, the request can be serviced immediately.
- If the drive or controller is busy, any new requests for service will be placed in the queue of pending requests for that drive.
- For a multiprogramming system with many processes, the disk queue may often have several pending requests.
- Thus, when one request is completed, the operating system chooses which pending request to service next.
- How does the operating system make this choice

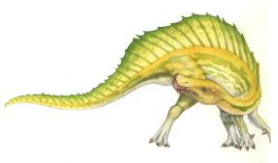


Disk Scheduling (Cont.)

- 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)

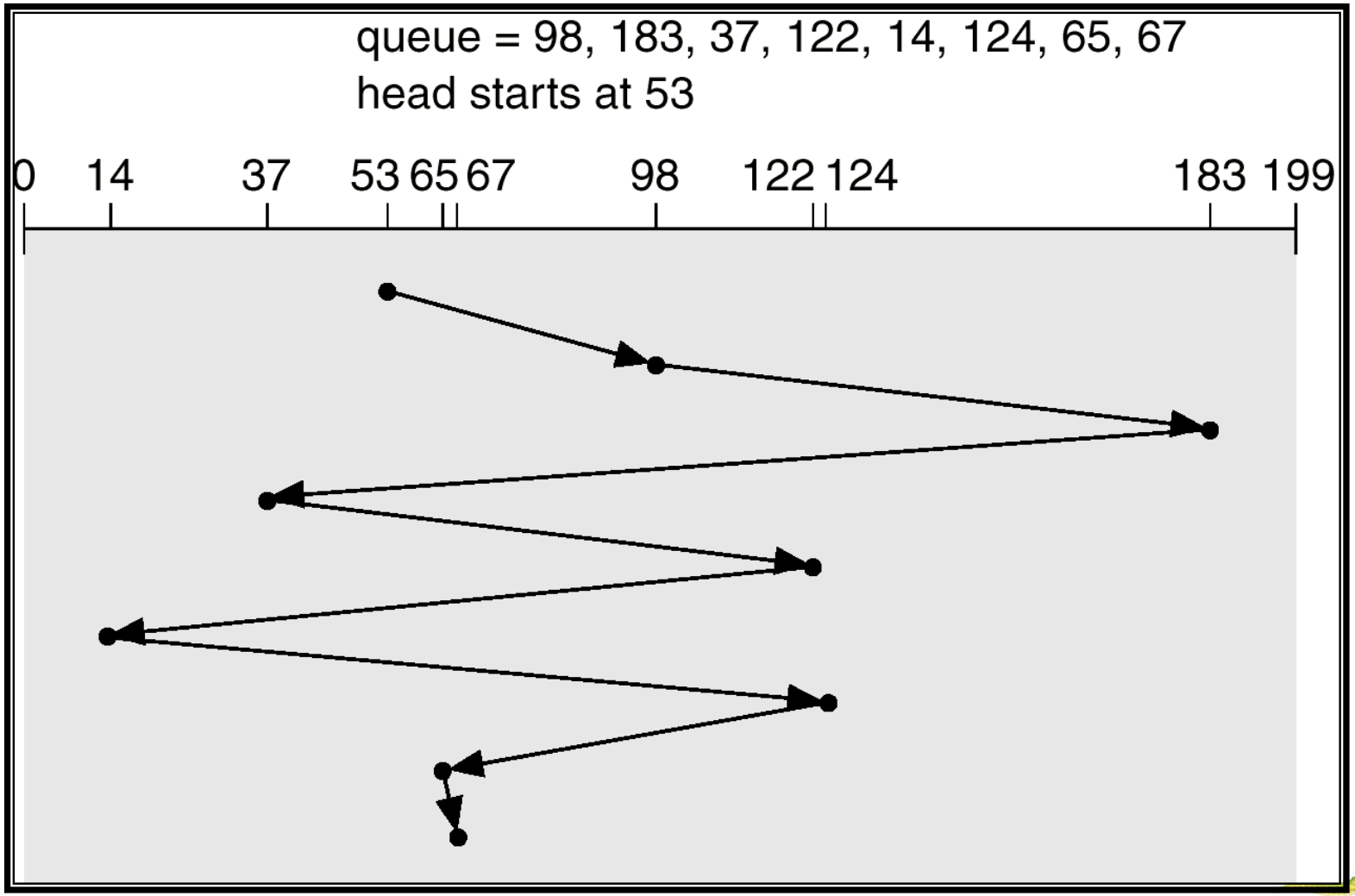
98, 183, 37, 122, 14, 124, 65, 67

Head pointer 53



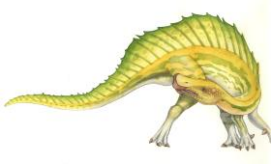
FCFS

Illustration shows total head movement of 640 cylinders



FCFS

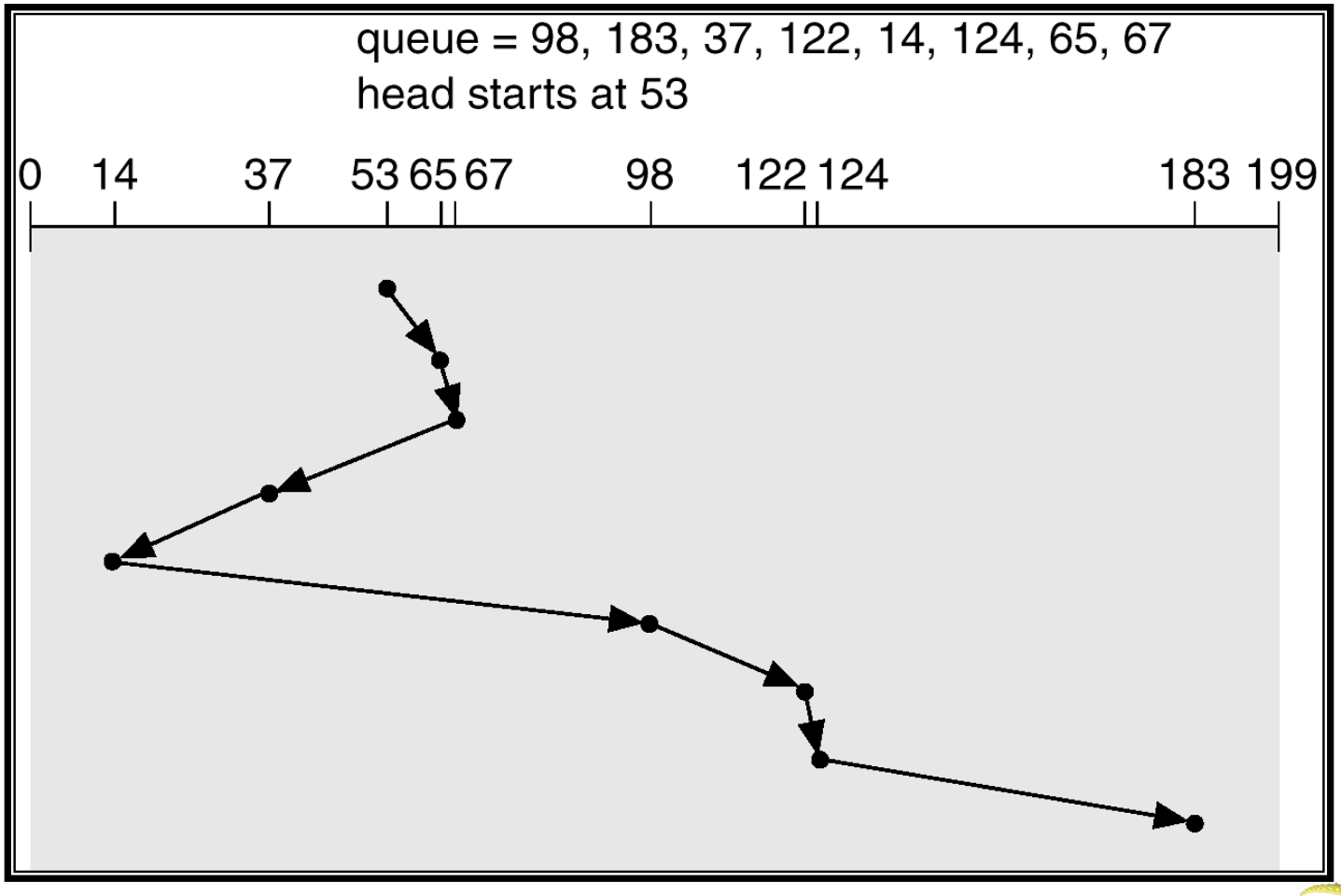
- Handle I/O requests sequentially.
- Fair to all processes.
- Approaches random scheduling in performance if there are many processes/requests.
- Suffers from global zigzag effect.



SSTF

- The SSTF algorithm selects the request with the least seek time from the current head position.
- In other words, SSTF chooses the pending request closest to the current head position.
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests
- Also called Shortest Seek Distance First (SSDF) – It's easier to compute distances.
- It's biased in favor of the middle cylinders requests.

- Illustration shows total head movement of 236 cylinders



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.
- It moves in both directions until both ends.
- Tends to stay more at the ends so more fair to the extreme cylinder requests.
- But note that if requests are uniformly dense, largest density at other end of disk and those wait the longest

SCAN (Cont.)

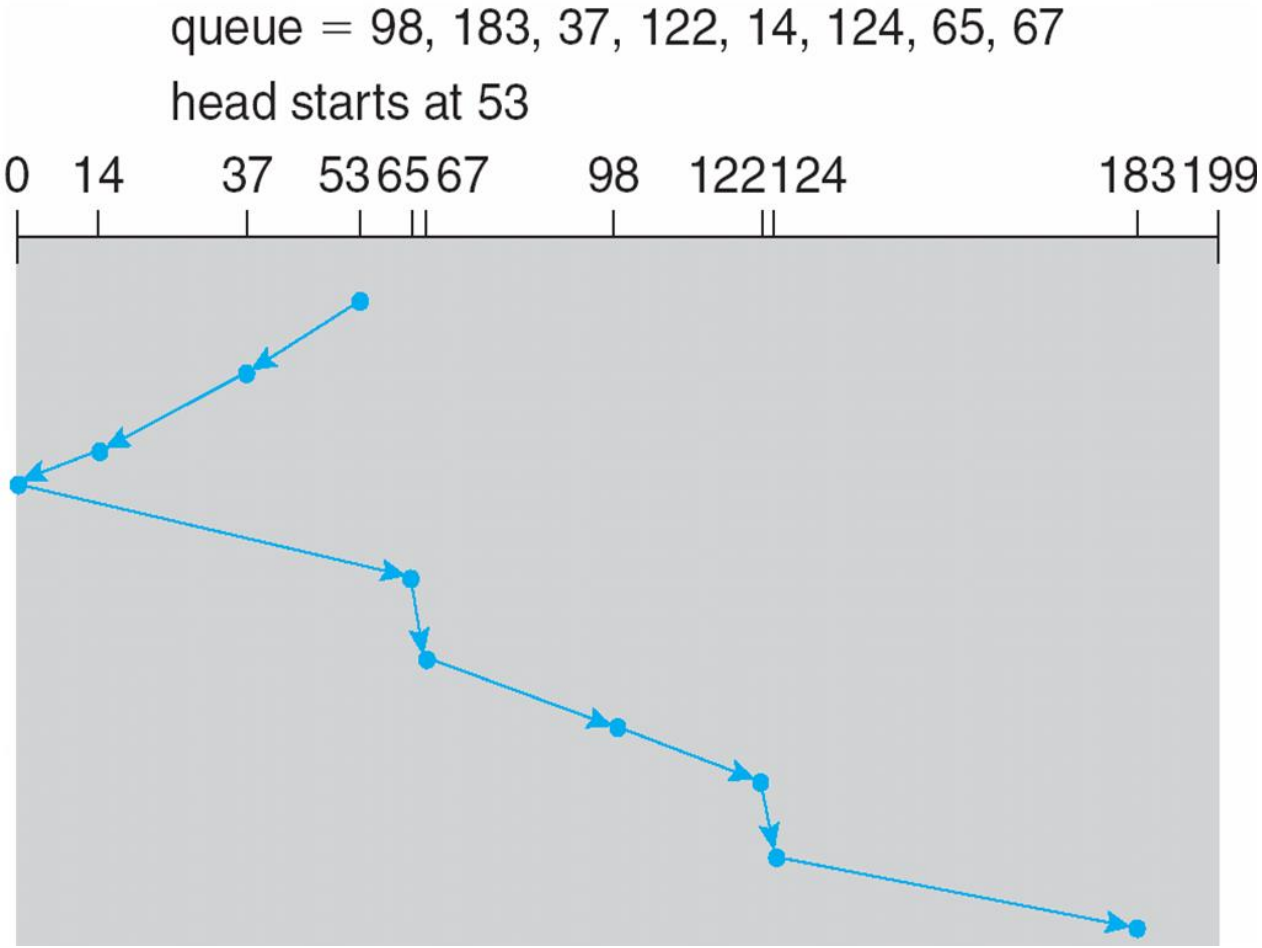
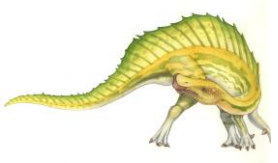


Illustration shows total head movement of 236 cylinders

Look

- The disk arm starts at the first I/O request on the disk, and moves toward the last I/O request on the other end, servicing requests until it gets to the other extreme I/O request on the disk, where the head movement is reversed and servicing continues.
- It moves in both directions until both last I/O requests; more inclined to serve the middle cylinder requests.

- Assuming a uniform distribution of requests for cylinders, consider the density of requests when the head reaches one end and reverses direction.
- At this point, relatively few requests are immediately in front of the head, since these cylinders have recently been serviced.
- The heaviest density of requests is at the other end of the disk.
- These requests have also waited the longest, so why not go there first?
- That is the idea of the next algorithm



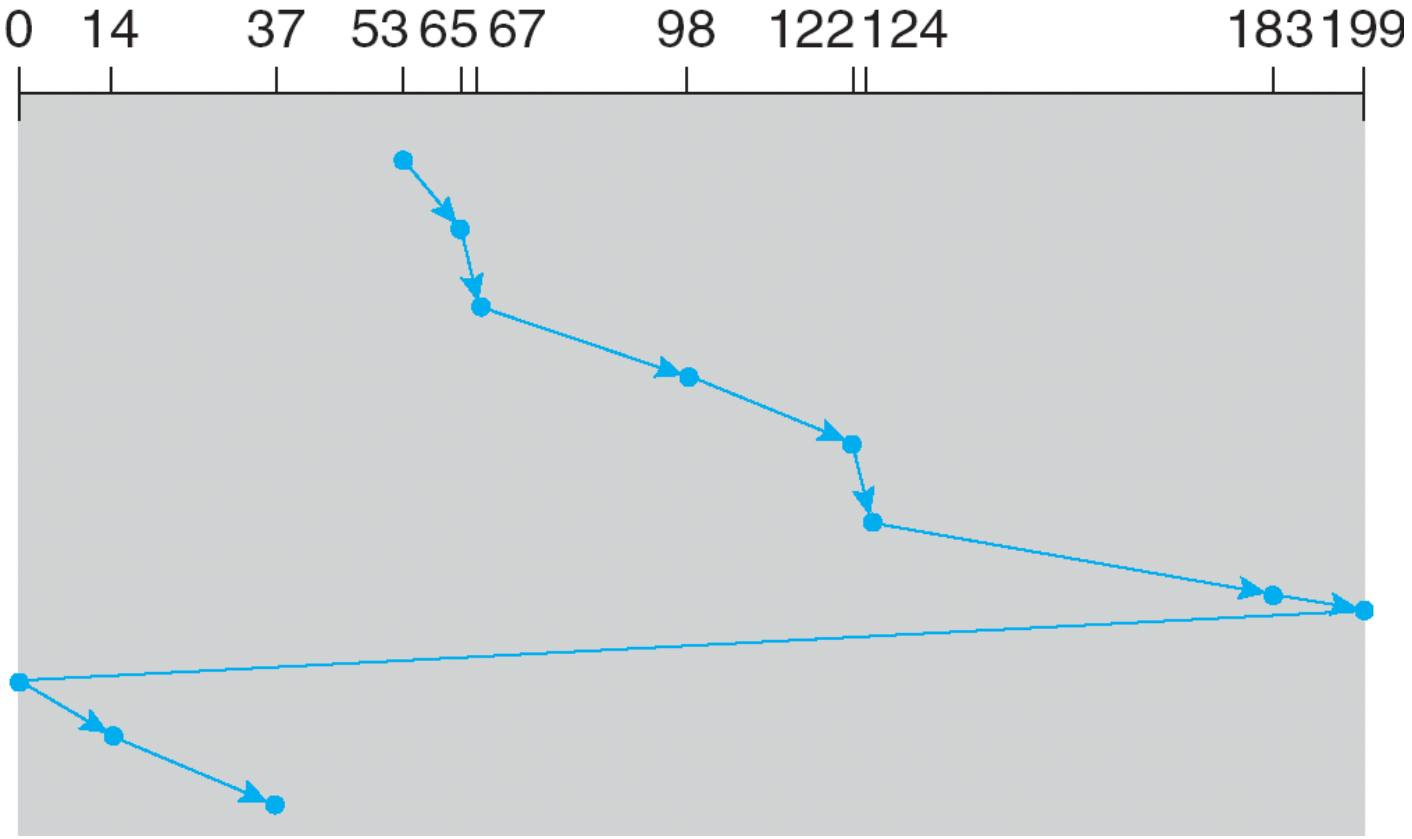
Circular-SCAN (C-SCAN)

- 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?
- Provides a more uniform wait time than SCAN; it treats all cylinders in the same manner.

C-SCAN (Cont.)

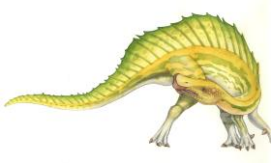
queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53



C-LOOK

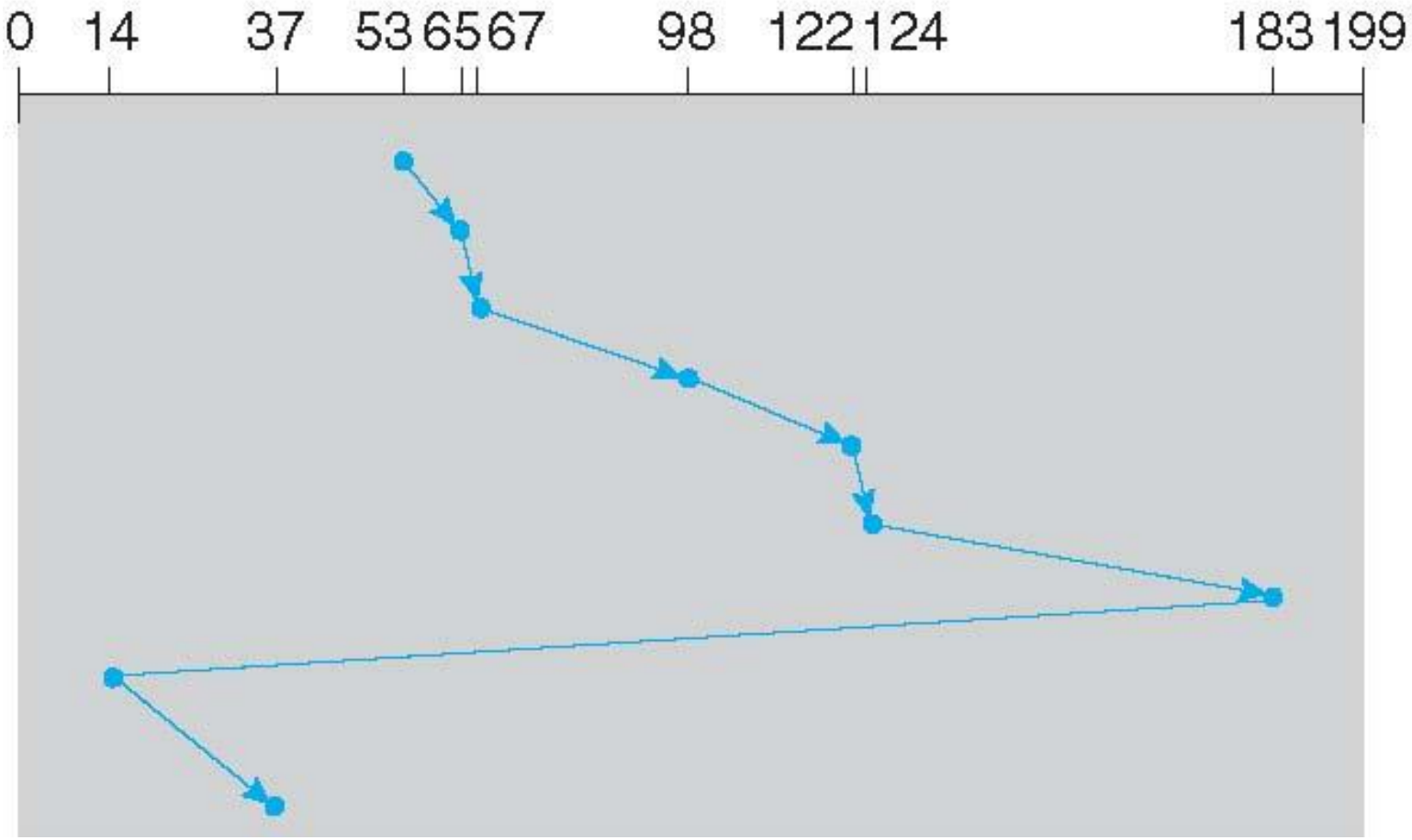
- LOOK a version of SCAN, C-LOOK a version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk
- Total number of cylinders?
- In general, Circular versions are more fair but pay with a larger total seek time.
- Scan versions have a larger total seek time than the corresponding Look versions.



C-LOOK (Cont.)

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

head starts at 53



Elevator Algorithms

- Algorithms based on the common elevator principle.
- Four combinations of Elevator algorithms:
  - Service in both directions or in only one direction.
  - Go until last cylinder or until last I/O request.

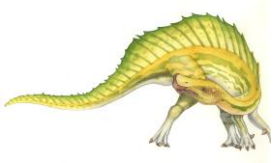
Direction \ Go until	Go until the last cylinder	Go until the last request
Service both directions	**Scan**	**Look**
Service in only one direction	**C-Scan**	**C-Look**

Suppose that a disk drive has 5,000 cylinders, numbered 0 through 4999. The disk is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests, in FIFO order, is 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130.

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all pending requests for the disk-scheduling algorithms SSTF, SCAN, LOOK, CSCAN, CLOOK

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 location of directories and index blocks is also important.
- Since every file must be opened to be used, and opening a file requires searching the directory structure, the directories will be accessed frequently.
- Suppose that a directory entry is on the first cylinder and a file's data are on the final cylinder.
  - In this case, the disk head has to move the entire width of the disk.
  - If the directory entry were on the middle cylinder, the head would have to move only one-half the width.
- Caching the directories and index blocks in main memory can also help to reduce disk-arm movement, particularly for read requests.

Selecting a Disk-Scheduling Algorithm

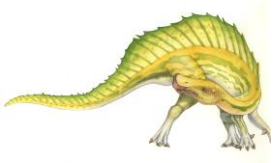
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary
- Either SSTF or LOOK is a reasonable choice for the default algorithm
- What about rotational latency?
  - Difficult for OS to calculate

Disk Management

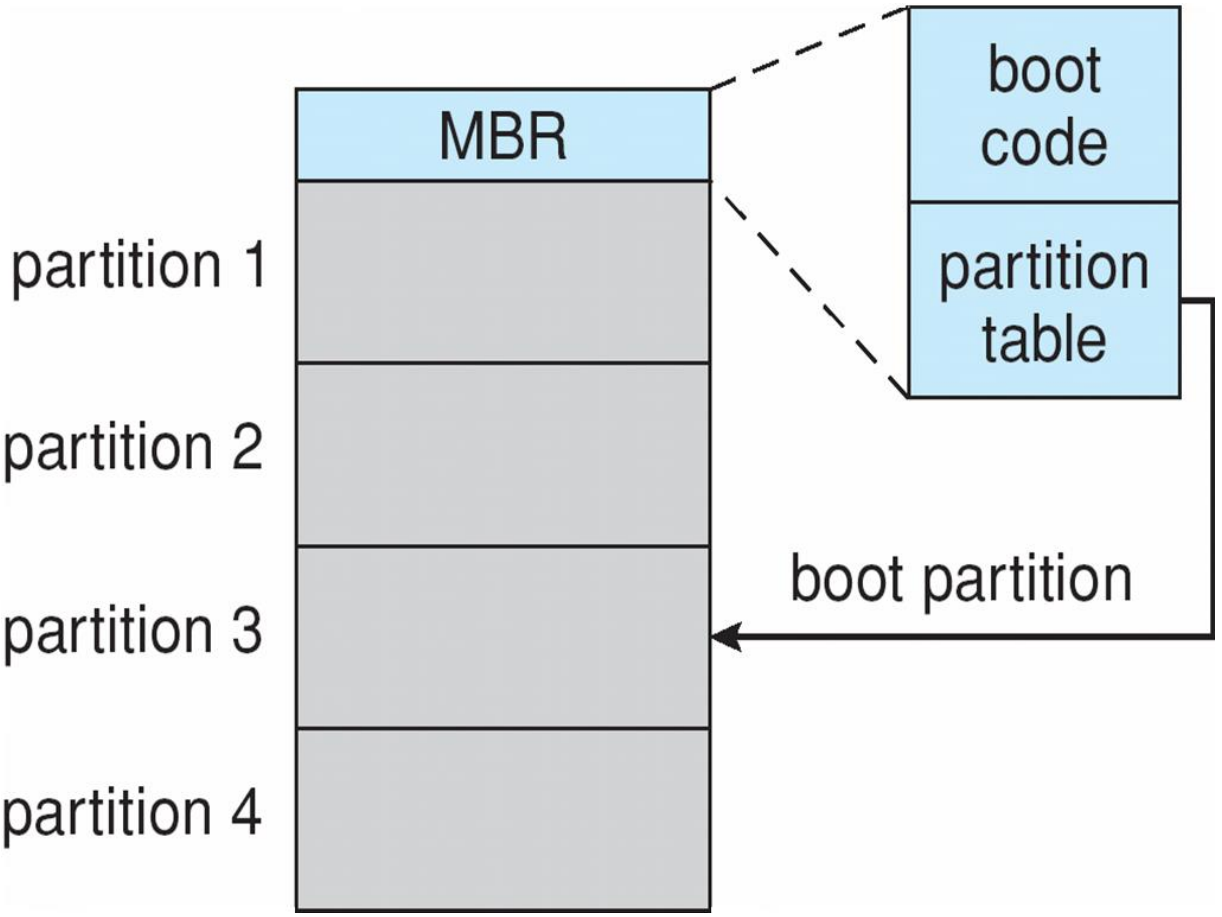
- **Low-level formatting**, or **physical formatting** — Dividing a disk into sectors that the disk controller can read and write
  - Each sector can hold header information, plus data, plus error correction code (**ECC**)
  - Usually 512 bytes of data but can be selectable
- To use a disk to hold files, the operating system still needs to record its own data structures on the disk
  - **Partition** the disk into one or more groups of cylinders, each treated as a logical disk
  - **Logical formatting** or "making a file system"
  - To increase efficiency most file systems group blocks into **clusters**
    - Disk I/O done in blocks
    - File I/O done in clusters

Disk Management (Cont.)

- Raw disk access for apps that want to do their own block management, keep OS out of the way (databases for example)
- Boot block initializes system
  - The bootstrap is stored in ROM
  - **Bootstrap loader** program stored in boot blocks of boot partition
- Methods such as **sector sparing** used to handle bad blocks



Booting from a Disk in Windows



Swap-Space Management

- Swap-space — Virtual memory uses disk space as an extension of main memory
  - Less common now due to memory capacity increases
- Swap-space can be carved out of the normal file system, or, more commonly, it can be in a separate disk partition (raw)
- Swap-space management
  - 4.3BSD allocates swap space when process starts; holds text segment (the program) and data segment
  - Kernel uses **swap maps** to track swap-space use
  - Solaris 2 allocates swap space only when a dirty page is forced out of physical memory, not when the virtual memory page is first created
    - File data written to swap space until write to file system requested
    - Other dirty pages go to swap space due to no other home
    - Text segment pages thrown out and reread from the file system as needed
- What if a system runs out of swap space?
- Some systems allow multiple swap spaces