Module 4 - File Management

Nirmala Baloorkar

Assistant Professor

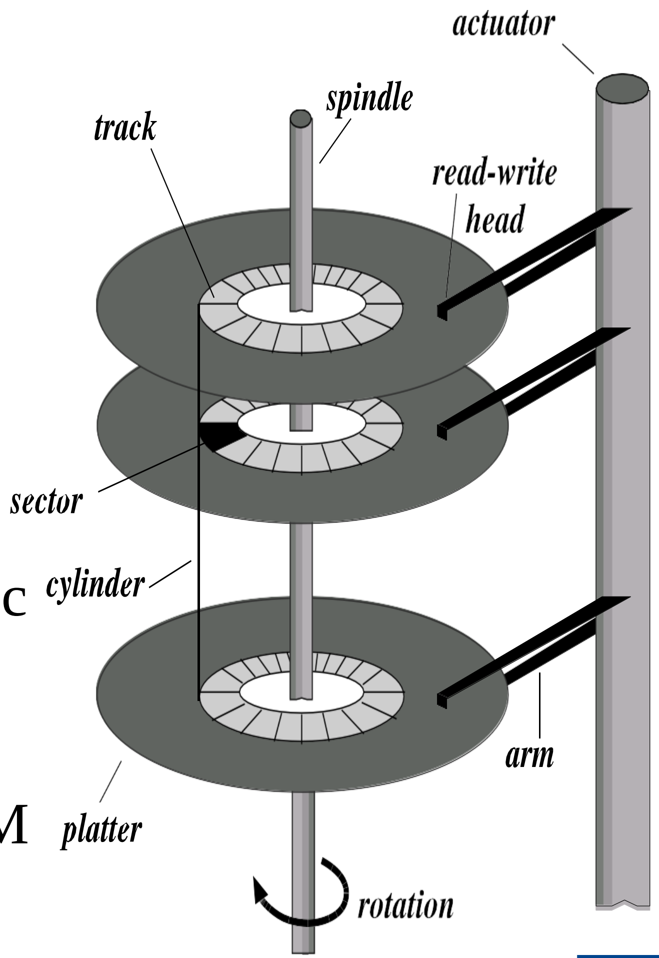
Faculty of Engineering and Technology

Objective

- Mass storage
- Disk scheduling
- Disk management
- Files
- Directories

Mass storage:Hard Disks (HDs)

- Stack of platters
  - Historically 0.85” to 14”
  - Commonly 3.5”, 2.5”, 1.8”
  - Capacity continually increases but perhaps 30GB – 3TB
- Performance
  - Transfer Rate (theoretical) = 6 Gb/sec
  - Effective Transfer Rate (real) = 1Gb/sec
  - Seek time 3–12ms with around 9ms common
  - Rotation typically 7200 or 15,000 RPM



Massstorage: Solid state disks (SSDs)

- Non-volatile memory used like a hard drive; many variations
- Pros
  - Can be more reliable than HDDs
  - No moving parts, so no seek time or rotational latency
  - Much faster
- Cons
  - Reads/writes wear out cells leading to unreliability and potentially shorter
  - More expensive per MB
  - Lower capacity

Some important terms:

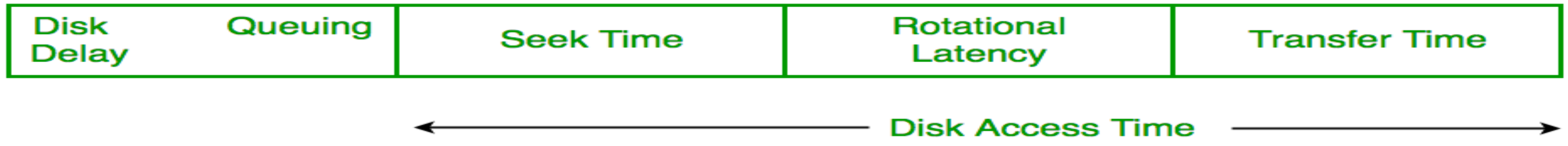
- **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.

- **Disk Access Time:** Disk Access Time is:

Disk Access Time = Seek Time + Rotational Latency + Transfer Time

Some important terms:

- **<u>Disk Response Time:</u>**
- Response Time is the average of time spent by a request waiting to perform its I/O operation.
- *Average Response time* is the response time of the all requests.
- *Variance Response Time* is measure of how individual request are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.



Disk scheduling

- The disk controller receives a sequence of read/write requests from the OS that it must schedule
  - How best to order reads and writes to achieve policy aim?
  - Analogous to CPU scheduling but with very different mechanisms, constraints and policy aims
  - Many algorithms exist

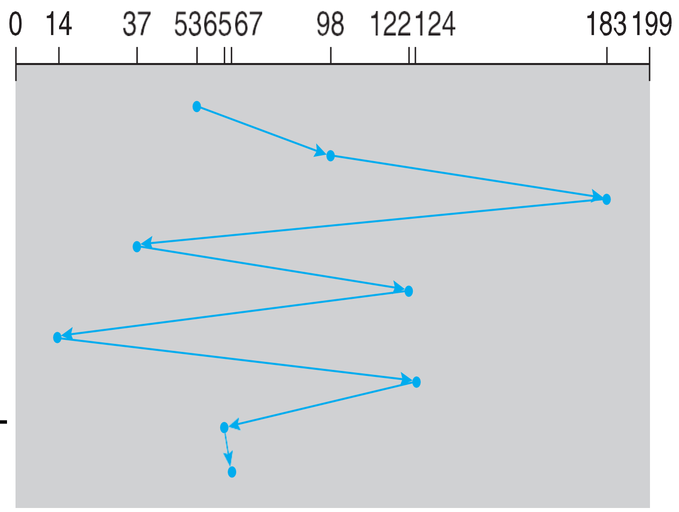
Disk Scheduling Algorithms

There are many Disk Scheduling Algorithms-

- FCFS
- SSTF
- SCAN
- C-SCAN
- LOOK
- C-LOOK

Disk scheduling- FCFS

- Simplest: First-come First-served (FCFS)
  - Here the request are addressed in the order they arrive in the disk queue.
- E.g., requests for blocks on cylinders are 98, 183, 37, 122, 14, 124, 65, 67
- Head starts 53.



Total Head Movement =
(98-53) + (183-98) +
(183-37) + (122-37) +
(122-14) + (124-14) +
(124 – 65) + (67-65)

=45 + 85 + 146 + 85 + 108 +
110 + 59 + 2
= 640

Disk scheduling- FCFS

- Maintain Queue of Requests
- Implemented as FIFO Queue

Advantages:

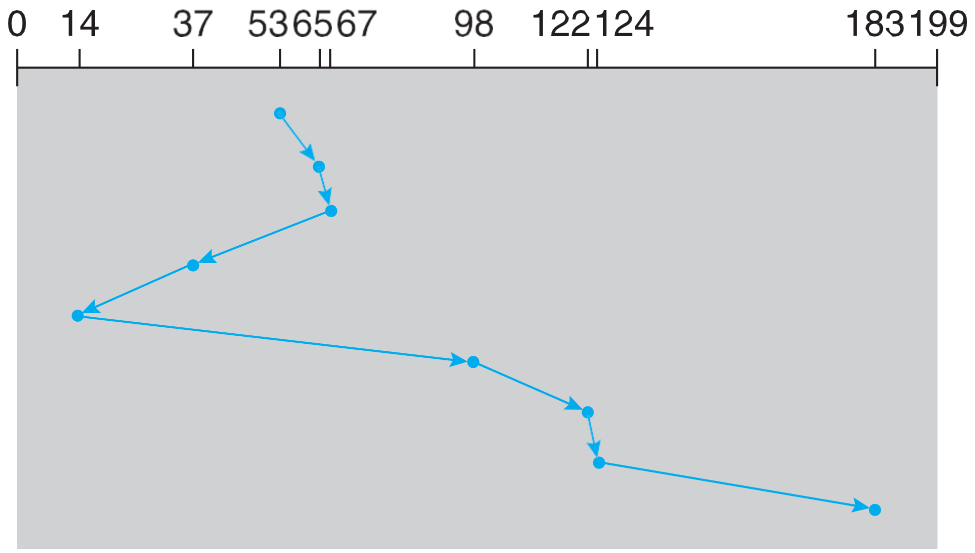
- **Every request gets a fair chance**
- No indefinite postponement

Disadvantages:

- **Does not try to optimize seek time**
- May not provide the best possible service
- **Unacceptably High Response time**

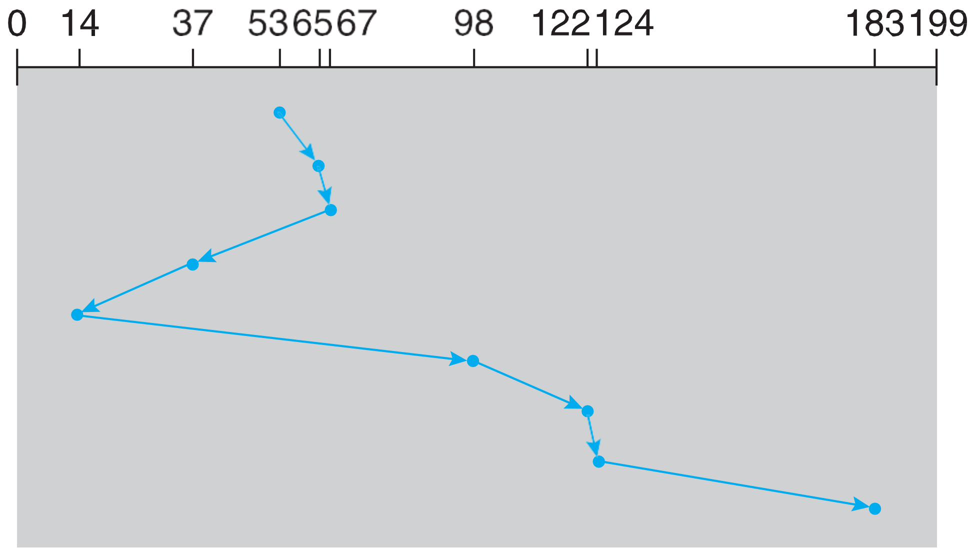
Disk scheduling- SSTF

- Service requests based on distance to current head position
  - Next request in queue is that with the shortest seek time
- E.g., requests for blocks on cylinders are 98, 183, 37, 122, 14, 124, 65, 67
- Head starts 53.



Disk scheduling- SSTF

- E.g., requests for blocks on cylinders are 98, 183, 37, 122, 14, 124, 65, 67
- Head starts 53.



Total Head Movement =
(65-53) + (67-65) +
(67-37) + (37 - 14) +
(98-14) + (122 - 98) +
(124 – 122) + (183 - 124)

=12 + 2 + 30 + 23 + 84 + 24
+ 2 + 59
= 236

Disk scheduling- SSTF

Advantages:

- This algorithm gives a substantial improvement in performance as compared to FCFS.

Disadvantages:

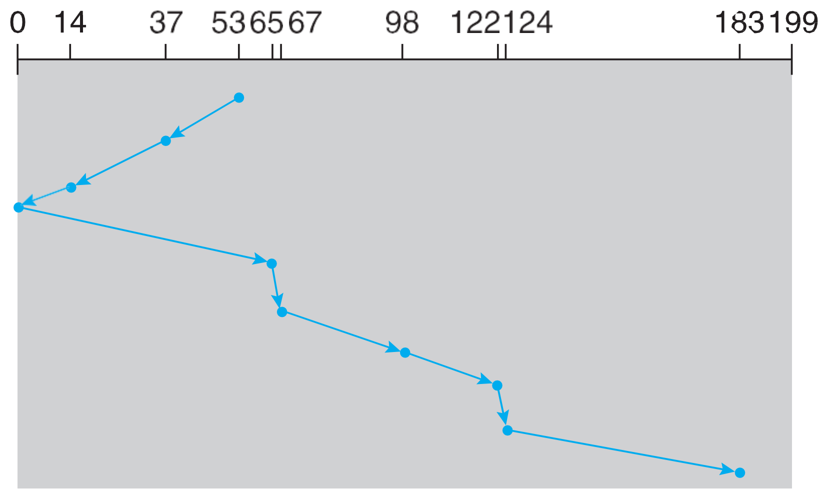
- Can cause starvation.
- Though it is better than FCFS, it is still not optimal.

Disk scheduling- SCAN

- In the SCAN algorithm, the disk arm starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk.
- At the other end, the direction of head movement is reversed, and servicing continues.
- The head continuously scans back and forth across the disk.
- Also called as Elevator.

Disk scheduling- SCAN

- E.g., requests for blocks on cylinders are 98, 183, 37, 122, 14, 124, 65, 67
- Head starts 53 and direction of head movement is towards 0.



Total Head Movement =
(53-37) + (37-14) +
(14-0) + (65-0) +
(67-65) + (98-67) +
(122 – 98) + (124-122)

=16 + 23 + 14 + 65 + 2 + 31
+ 24 + 2 +58
= 235

Disk scheduling- SCAN

Advantages:

- This algorithm is better than FCFS.
- No starvation.

Disadvantages:

- If a request arrive in the queue just in front of the head, it will be serviced almost immediately; a request arriving just behind the head will have to wait until the arm moves to the end of the disk, reverse direction, and comes back.

Disk scheduling- C-SCAN(Circular SCAN)

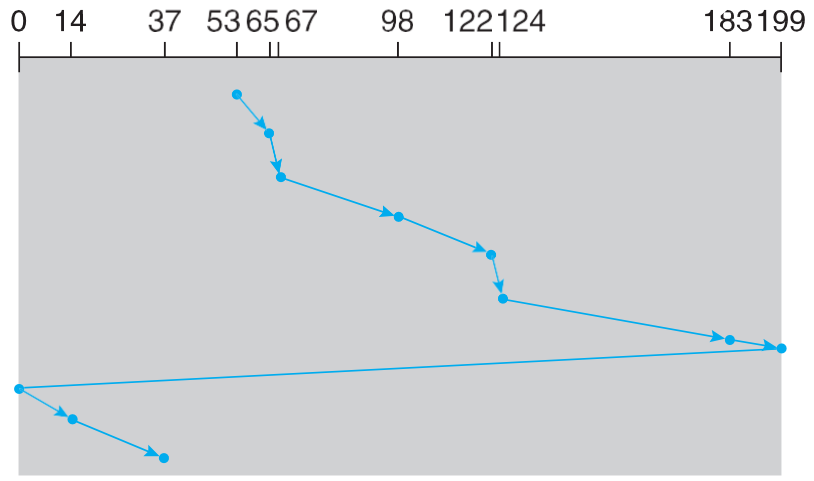
- In the SCAN algorithm, when the head reaches one end and reverse direction, the density of request immediately in front of the head are relatively few as these cylinders have recently been serviced.
- The heaviest density of requests is at the other end of the disk. These requests have also waited the longest, so why not go there first?
- Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way.
- When the head reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.

Disk scheduling- C-SCAN

- E.g., requests for blocks on cylinders are 98, 183, 37, 122, 14, 124, 65, 67
- Head starts 53 and direction of head movement is towards 199.

Total Head Movement =
(65-53) + (67-65) +
(98-67) + (122-98) +
(124-122) + (183-124) +
(198-183) + (199-0) +
(14-0) + (37-14)

=12 + 2 + 31 + 24 + 2 + 59 +
16 + 199 + 14 + 23
= 382



Disk scheduling- SCAN

Advantages:

- This algorithm provides a more uniform wait time.

Disadvantages:

- Total head movement is more as compared to SCAN algorithm.

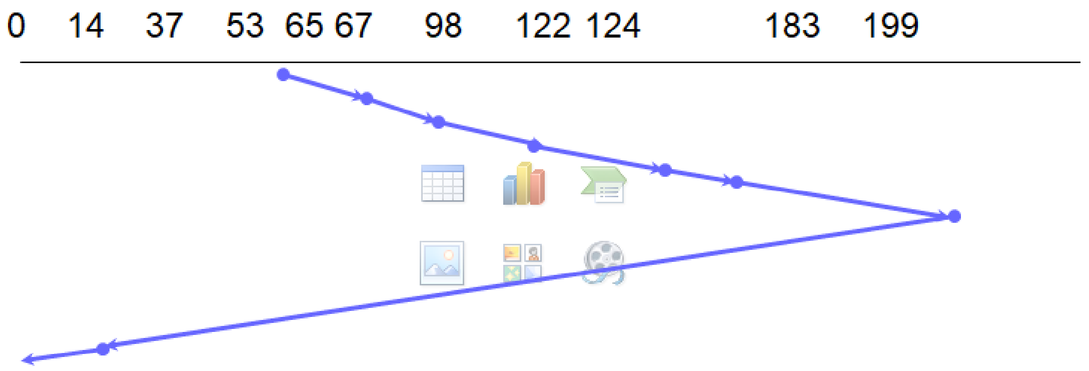
Disk scheduling- LOOK

- Both the SCAN and C-SCAN move the disk arm across the full width of the disk.
- But in LOOK algorithm, the arm goes only as fas as the final request in each direction.
- Then, it reverses direction immediately, without going all the way to the end of the disk.

Disk scheduling- LOOK

- E.g., requests for blocks on cylinders are 98, 183, 37, 122, 14, 124, 65, 67
- Head starts 53 and direction of head movement is towards 199.

Total Head Movement = (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (183-37) + (37-14)



=12 + 2 + 31 + 24 + 2 + 59 + 146 + 23
= 299

Disk scheduling- LOOK

Advantages:

- Starvation does not occur.
- Since the head does not go to the end of the disk, the time is not wasted here.

Disadvantage:

- The arm has to be conscious to find the last request.

Disk scheduling- C- LOOK

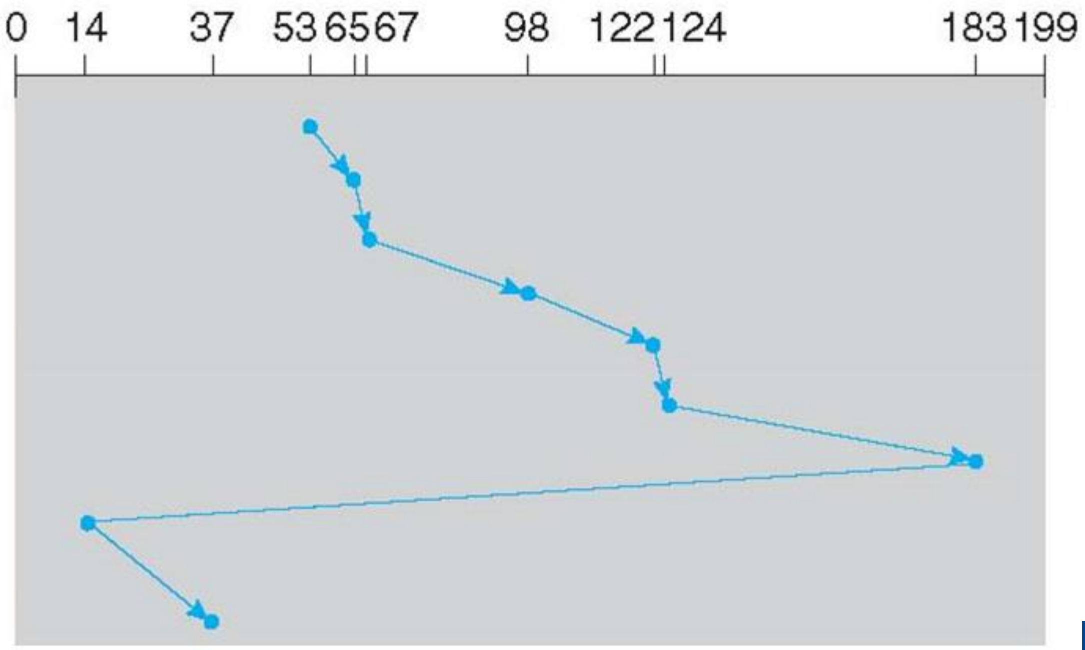
- C-LOOK algorithm moves the head in one direction servicing requests along the way.
- When the head reaches the last request, it immediately returns to the farthest request on the other end without servicing any requests on the return trip.
- Then from that position, it services the reaming requests in the same direction like before.
- Also called as Circular LOOK.

Disk scheduling- CLOOK

- E.g., requests for blocks on cylinders are 98, 183, 37, 122, 14, 124, 65, 67
- Head starts 53 and direction of head movement is towards 0.

Total Head Movement = (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (183-14) + (37-14)

=12 + 2 + 31 + 24 + 2 + 59 + 169 + 23
= 322



Disk scheduling- CLOOK

Advantages:

- The waiting time is decreased.
- If there are no requests till the end, it reverses the head direction immediately.
- Starvation does not occur.
- The time taken by the disk arm to find the desired spot is less.

Disadvantage:

- The arm has to be conscious about finding the last request.

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**

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
- How does disk-based queueing effect OS queue ordering efforts?

For I/O Operations

- **Unit of transfer:**
- Data may be transferred as a stream of bytes or characters(e.g., terminal I/O) or in larger blocks (e.g., disk I/O).

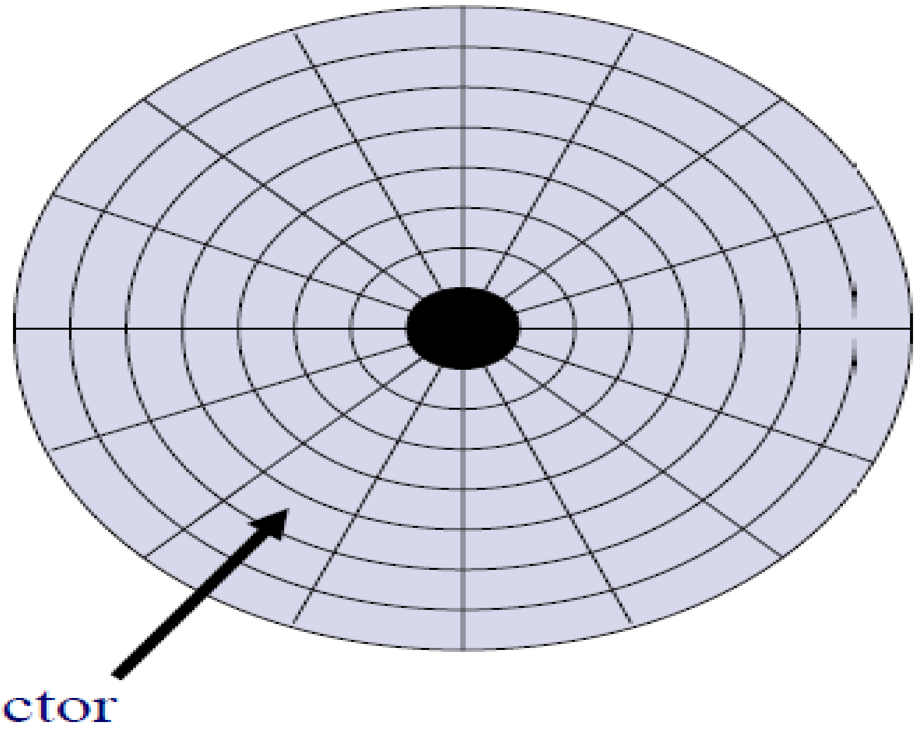
Disk Management

- The operating system is responsible for several other aspects of disk management like
  - disk initialization
  - disk formatting
  - booting from disk, and
  - bad-block recovery

Disk Management

Low-level formatting, or **physical formatting** —

- Dividing a disk into sectors that the disk controller can read and write



Disk Management

Low-level formatting, or **physical formatting** —

- Each sector can hold
- header information,
- plus data,
- plus error correction code (**ECC**)

Disk Management

Low-level formatting, or **physical formatting** —

- Disk controller handles how many bytes of data space to leave between the header and trailer of all sectors.
- It is usually possible to choose among a few sizes, such as 256,512, and 1,024 bytes.
- Usually 512 bytes of data

Disk Management

Low-level formatting, or **physical formatting** —

- Enables the manufacturer to test the disk

- To initialize the mapping from logical block numbers to defect-free sectors on the disk.

Disk Management

Partitioning

- 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

- The operating system can treat each partition as though it were a separate disk.
  - One partition can hold a copy of the operating system's executable code
  - Another holds user files

Disk Management

Logical formatting

- “making a file system”
- “creation of a file system“
- The OS stores the initial file-system data structures onto the disk.
- These data structures may include
  - maps of free and allocated space (a FAT or inodes)
  - an initial empty directory

Disk Management (Cont.)

Bad Blocks

- Methods such as **sector sparing** used to handle bad blocks

Disk Management (Cont.)

Bad Blocks

- Because disks have moving parts and small tolerances
- As the disk head flies just above the disk surface,
- They are prone to failure

1) If the failure is complete;
   - disk needs to be replaced and its contents restored from backup media to the new disk.
2) More frequently, one or more sectors become defective.

Disk Management (Cont.)

Bad Blocks

- On simple disks, such as some disks with IDE controllers, bad blocks are handled manually.

- More sophisticated disks, used in high-end PCs are smarter about bad-block recovery.
    - The controller maintains a list of bad blocks on the disk.
    - The list is initialized during the low-level formatting at the factory and is updated over the life of the disk.

Disk Management (Cont.)

Bad Blocks

- Low-level formatting also sets aside spare sectors not visible to the operating system.

- The controller can be told to replace each bad sector logically with one of the spare sectors. This scheme is known as sector sparing or forwarding