NUB
NAHDA UNIVERSITY
IN BENI SUEF
جامعة النهضة . ببني سويف

Operating System 2- CS402

Lecture 2
Disk scheduling

2025

Learning Objectives

- Learn the process of disk initialization and formatting.

  (Explicitly includes formatting, which is a key part of initialization.)

- Understand disk structure and organization.
- Explore different techniques for disk scheduling.

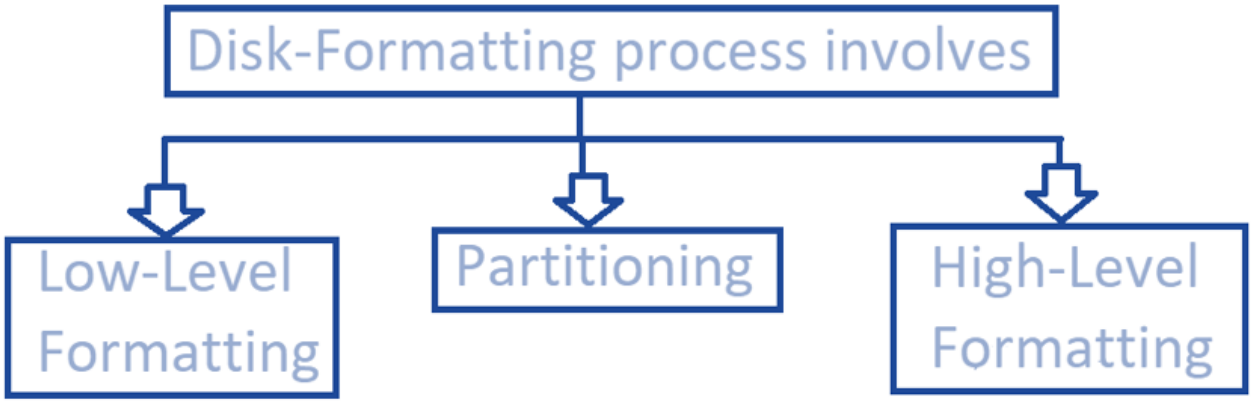
Disk Management

- Computers use disks as the principal on-line storage medium for both programs and data.
- Programs are stored on a disk until loaded into memory. They then use the disk as both the source and destination of their processing.
- The operating system is responsible for several aspects of disk management. Here we discuss:
  - Disk formatting.
  - Booting from disk.
  - Bad-block recovery.

Disk Formatting

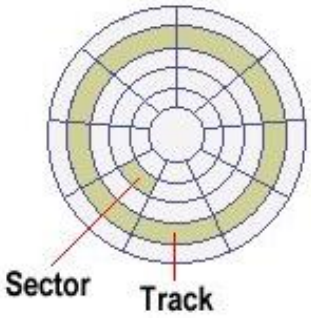
A new magnetic disk is a blank slate: it is just a platter of a magnetic recording material.

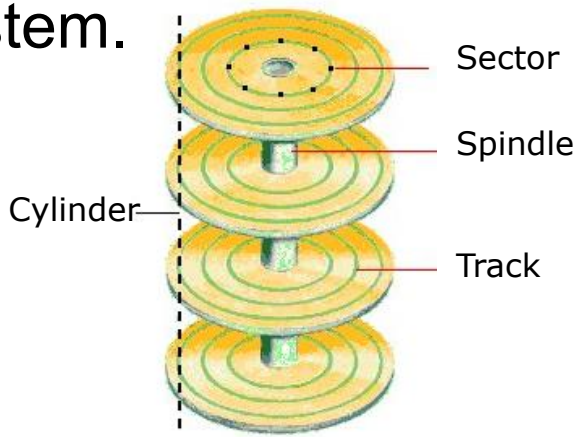
1. Physical Formatting (Low-Level Formatting)
2. Partitioning (Optional but Recommended)
3. Logical Formatting (High-Level Formatting)



Physical Formatting (Low-Level Formatting)

- Purpose:
  - Divides the disk into tracks and sectors.
  - Marks bad sectors to avoid data corruption.
- Performed By:
  - Manufacturers before shipping the disk.
  - Can be done manually using specialized low-level formatting.
- Result: The disk is structured but **not yet ready for data storage**. It still needs a file system.





Partitioning (Optional but Recommended)

- Purpose:

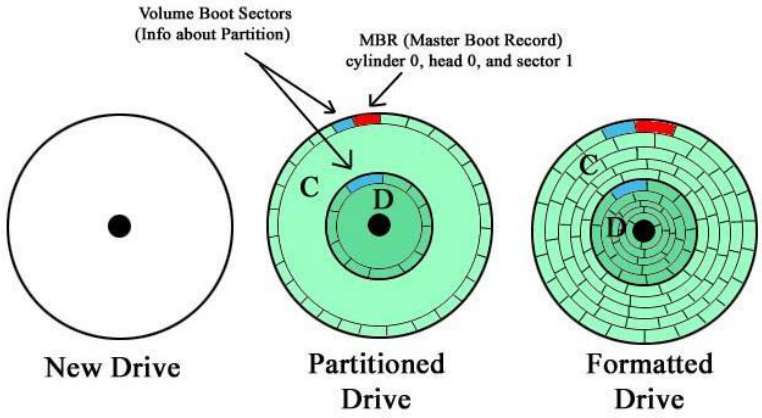
  Partitioning is the process of dividing the hard-disk into one or more regions (one or more groups of cylinders). The regions are called as partitions.

- Types of Partitions:
  - **Primary Partition** (Bootable, where OS is installed).
  - **Logical Partition** (Used for storing data, applications, etc.).
  - **Extended Partition** (cannot store data directly but contain multiple logical partitions).

- Tools to Create Partitions:
  - **Windows**: Disk Management (diskpart)
  - **Linux**: fdisk, gparted

Logical Formatting (High-Level Formatting)

- Purpose:
  - Creates a **file system** (e.g., NTFS, FAT32, ext4) on the partition to manage data storage.
  - Creates a **boot sector**, file allocation tables, and directory structures.
  - Makes the disk **usable for storing files**.
- Tools to Perform Logical Formatting:
  - **Windows**: Right-click disk → Format
  - **Linux**: mkfs -t ext4 /dev/sda1
  - **Mac**: Disk Utility



Bad-Block Recovery

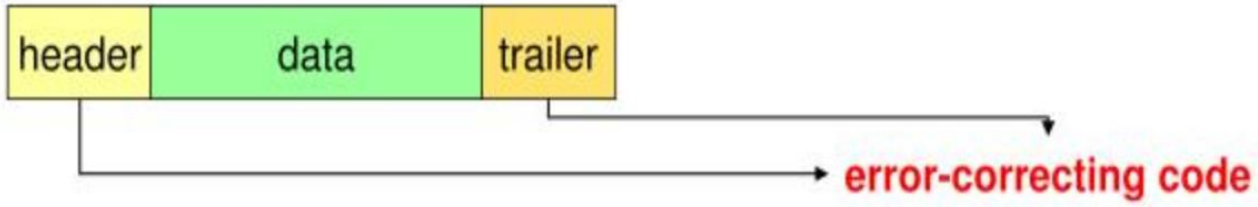
- The process of identifying and handling defective blocks (or sectors) on a storage device to prevent data loss and maintain system reliability.
- Bad blocks can develop due to:
  - Manufacturing defects.
  - Aging.
  - Physical damage.

Bad Blocks Detection Methods

- Error-Correcting Code (ECC): Storage devices use ECC to detect and correct minor errors in bad sectors (Immediate Error Correction).
- Surface Scanning: a low-level disk check that detects bad sectors (physical damage or corruption). Disk utilities (e.g., chkdsk on Windows, badblocks on Linux) scan for unreadable sectors.
- SMART Monitoring (Self-Monitoring, Analysis, and Reporting Technology): a built-in monitoring system in HDDs and SSDs that predicts hardware failures. It tracks disk health based on various performance indicators.

Sector Data Structure

- Low-level formatting fills the disk with a special data structure for each sector.
- The data structure for a sector typically consists of:
  - Header.
  - Data area (usually 512 bytes in size).
  - Trailer.
- The header and trailer contain information used by the disk controller, such as a sector number and an **error-correcting code (ECC)**.



Error-Correcting Code (ECC)

- When the controller writes a sector of data during normal I/O, the **ECC is updated** with a value calculated from all the bytes in the data area.
- When the sector is read the **ECC is recalculated** and compared with the stored value. If the stored and calculated numbers are different, this mismatch indicates that the **data area of the sector has become corrupted** and that the disk sector may be bad.
- The **controller automatically does the ECC processing** whenever a sector is read or written.

Error-Correcting Code (ECC) (Cont.)

- The ECC is an error-correcting code because It contains enough information, if only a few bits of data have been corrupted, to enable the controller to:
  - Identify which bits have changed.
  - Calculate what their correct values should be.
  - Reports a recoverable **soft error**.

Bootstrapping & System Initialization

Bootstrap Program: The initial program executed when a computer is powered on or rebooted.

- Functions of the Bootstrap Program:
  - Initializes CPU registers, device controllers, and memory.
  - Locates the operating system (OS) kernel on disk.
  - Loads the kernel into memory and starts OS execution.
- Storage of Bootstrap Program:
  - Stored in ROM – Non-volatile and immune to viruses.
  - **Limitations** – ROM cannot be easily modified.
- Solution: Boot loader in ROM
  - A small bootstrap loader in ROM loads a full bootstrap program from disk.
  - The full bootstrap is stored in boot blocks of a boot disk (system disk).
- Key Benefit: Allows easy updates to the bootstrap without hardware changes.

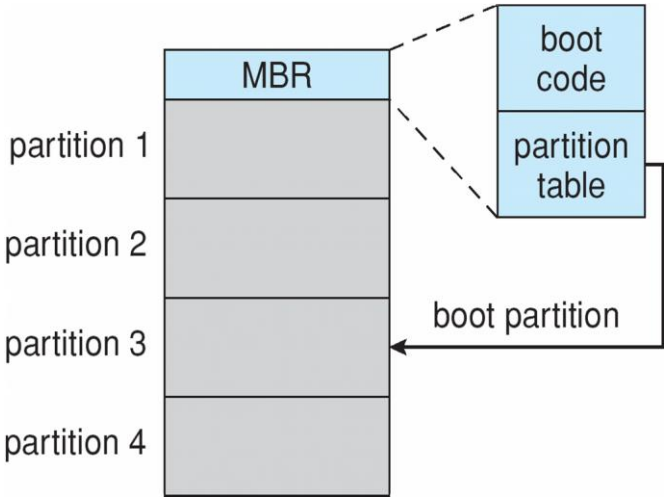
Boot process in Windows

- Windows divides the hard disk into partitions, with one designated as the **boot partition**, containing the OS and device drivers.
- Master Boot Record (MBR): The first sector of the hard disk, the MBR, **holds** boot code, partition details, and a flag for the active boot partition.
- Boot Process Steps:

1. **ROM Code Execution:** The system starts by running boot code from ROM.
2. **MBR Loading:** The ROM directs the system to load boot code from the MBR.

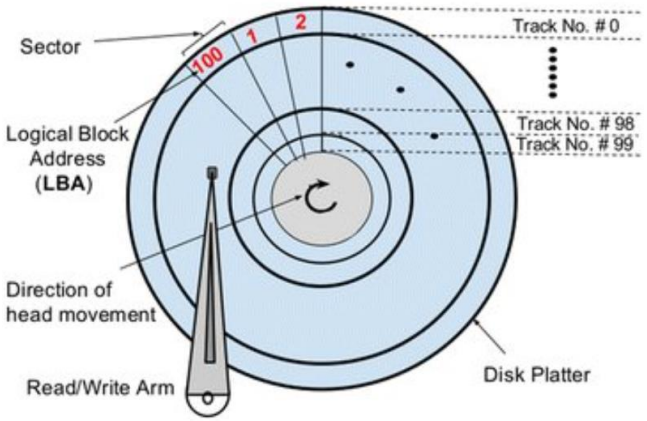
- Boot Sector Loading: The system identifies the boot partition, reads its boot sector (the first sector**)**, and continues loading OS components.

This process ensures Windows initializes essential subsystems and system services.



Disk Addressing

- For efficiently manage data access and retrieval addressing is done during Low-level formatting:
  - Disk addressing: determines how data is located on the disk by the CHS (Cylinder, Head, Sector) scheme that uses three numbers.
  - Logical Block Addressing (LBA): creates a 1D array of uniquely numbered logical blocks (smallest unit of transfer).
  - Logical Block Mapping: translates logical storage structures into physical locations on the disk.



Disk Addressing (Cont.)

- Disk addressing:
  - Sector 0 → First sector of the first track on the outermost cylinder.
  - Mapping follows track-by-track, cylinder-by-cylinder (outer to inner).
- Logical blocks are **sequentially** mapped to disk sectors.
- Logical to Physical Mapping is straightforward **except:**
  - Bad sectors (unusable areas).
  - **Non-constant** number of sectors per track (due to varying disk geometry).

HDD Scheduling

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

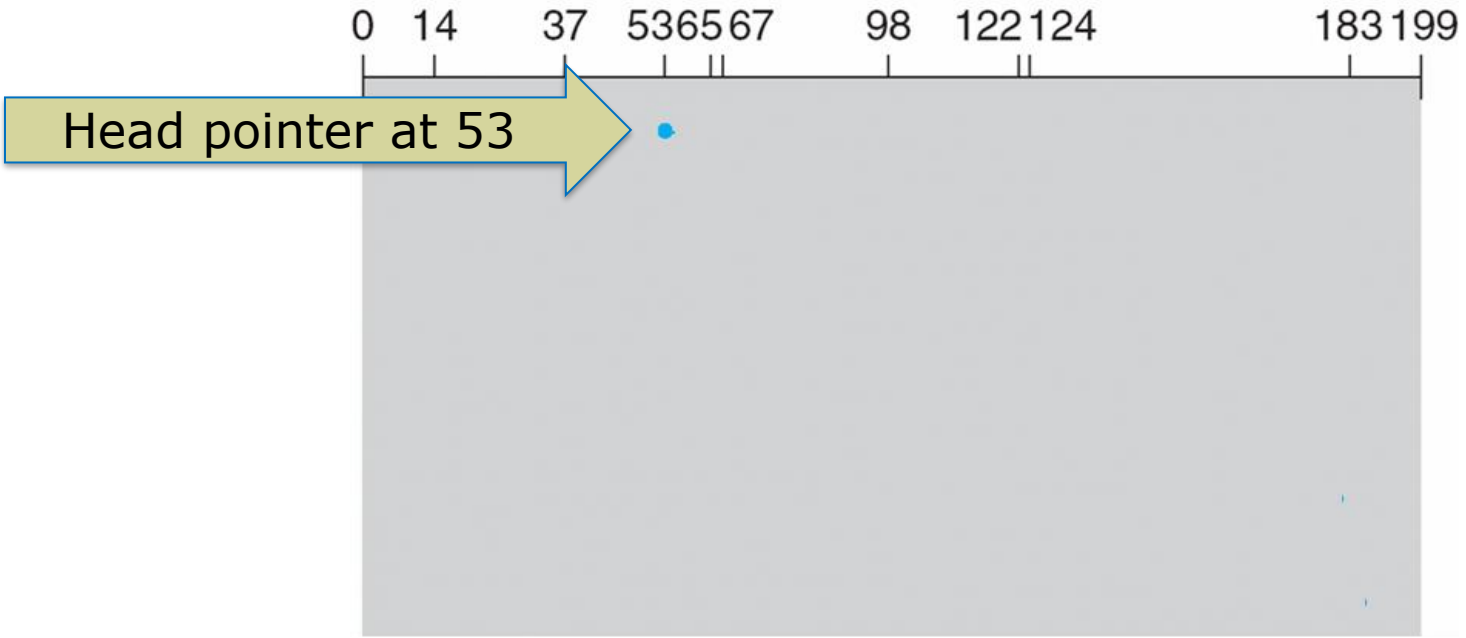
Disk Scheduling (Cont.)

- 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 is the disk address for the transfer.
  - What is the memory address for the transfer.
  - What is the number of sectors to be transferred.
- 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.
- When one request is completed, the operating system **chooses** which pending request to service next.

Disk Scheduling (Cont.)

- 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).

Queue = 98, 183, 37, 122, 14, 124, 65, 67
Head pointer at 53



First-Come, First-Served (FCFS)

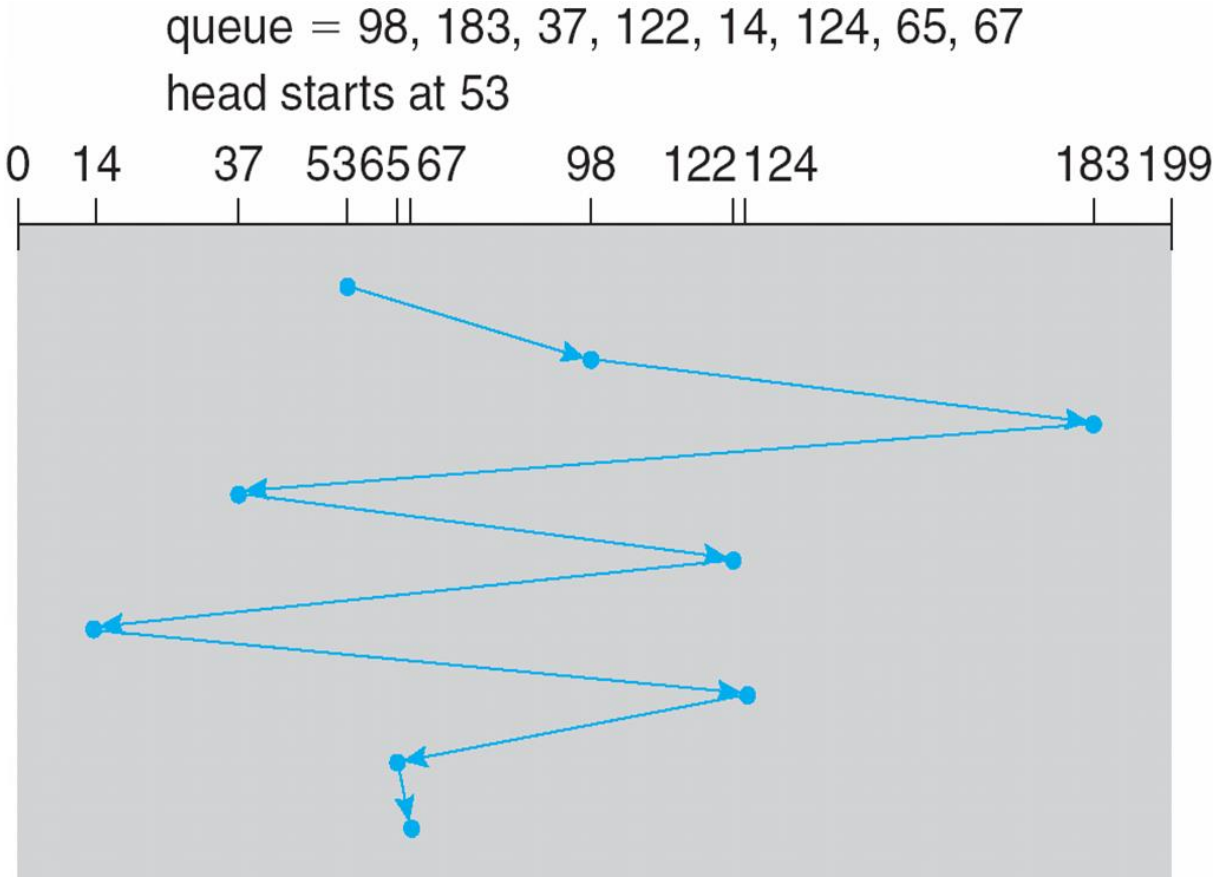
- The simplest form of disk scheduling.
- This algorithm is fair, but it generally does not provide the fastest service.
- Consider, for example, a disk queue with requests for I/O to blocks on cylinders

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

  head starts at 53
- In that order. If the disk head is initially at cylinder 53:
  - It will first move from 53 to 98, then to 183, 37, 122, 14, 124, 65, and finally to 67.
  - For a total head movement of 640 cylinders.

First-Come, First-Served (FCFS) (Cont.)

- The wild swing illustrates the problem with this schedule.
- The total head movement could be decreased substantially.
- Performance could be improved.



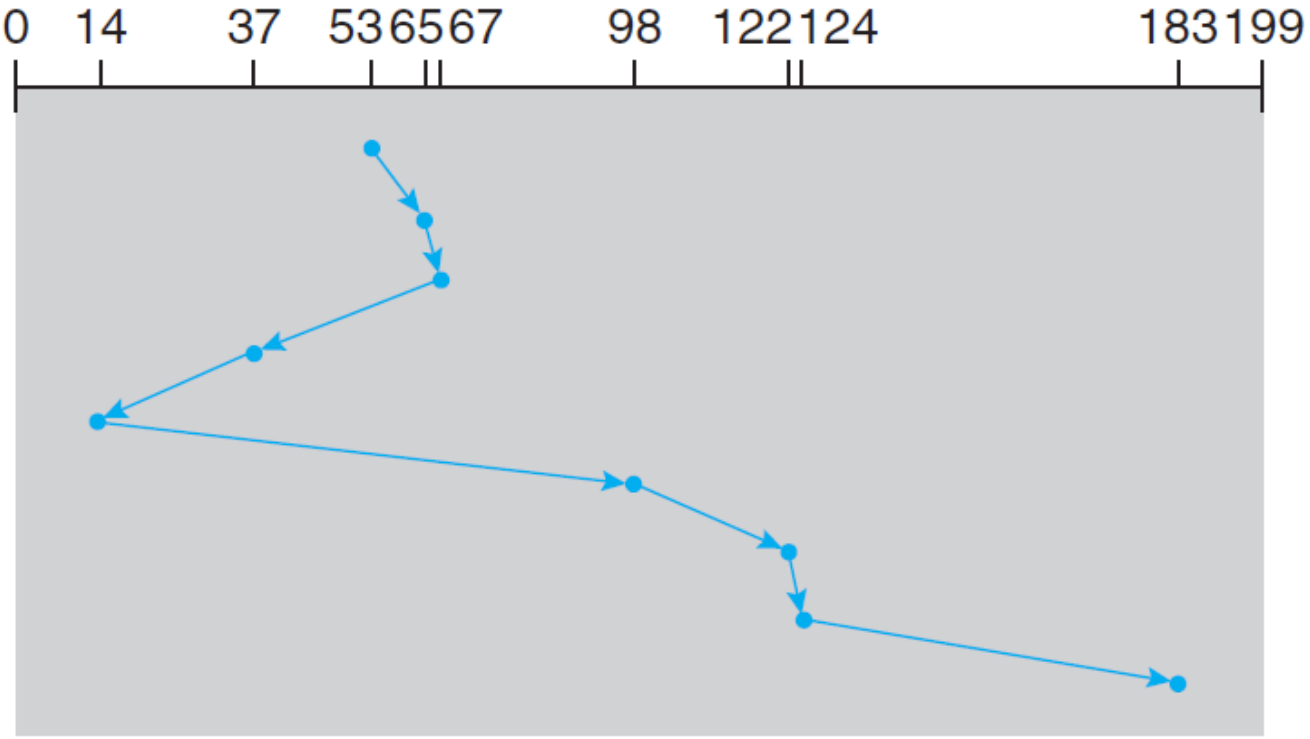
Shortest-Seek-Time-First (SSTF)

- The SSTF algorithm selects the request with the least seek time from the current head position.
- Consider, for example, a disk queue with requests for I/O to blocks on cylinders 98, 183, 37, 122, 14, 124, 65, 67 and head starts at 53
- For our example request queue, the closest request to the initial head position (53) is at cylinder 65.

53 ➡ 65 ➡ 67 ➡ 37 ➡ 14 ➡ 98 ➡ 122 ➡ 124

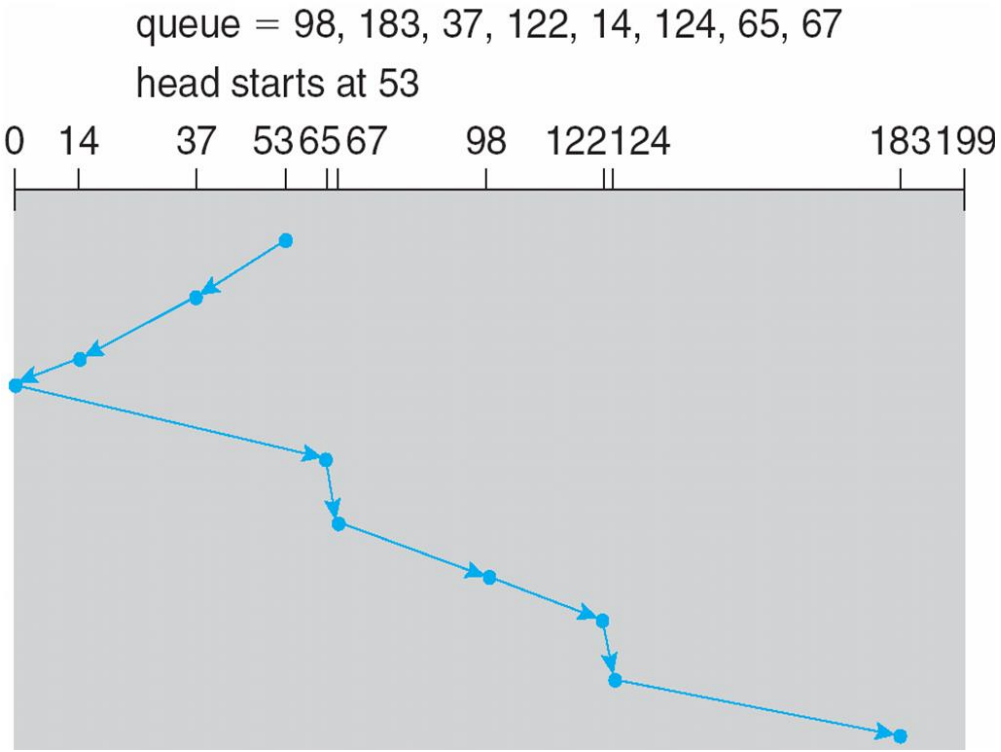
Shortest-Seek-Time-First (SSTF) (Cont.)

- This scheduling method results in a total head movement of only 236 cylinders
- Reduces seek time but may lead to starvation for distant requests.



SCAN Scheduling

- Sometimes called the elevator algorithm
- 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.
- Total head movement of 208 cylinders.



SCAN Scheduling (Cont.)

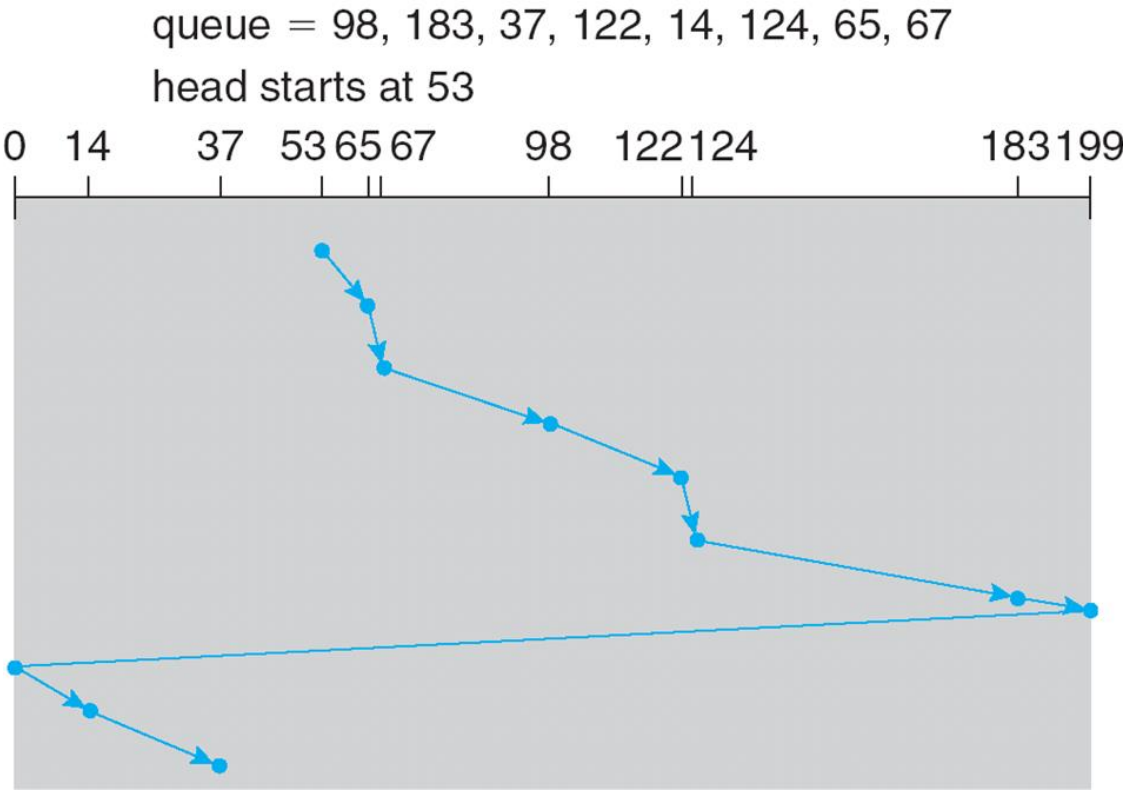
- Assuming a uniform distribution of requests for cylinders, consider the density of requests when the head reaches one end and reverses direction.
- At this point, relatively few requests are immediately in front of the head, since these cylinders have recently been serviced.
- The heaviest density 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.

C-SCAN Scheduling

- Circular SCAN (C-SCAN) scheduling is a variant of SCAN designed to provide a more uniform wait time.
- 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 (Total number of cylinders?)

Assignment.

Consider initial head position 53 and request sequence [98, 183, 37, 122, 14, 124, 65, 67] Compute the total distance traveled using C-SCAN Assume Disk size = 0 to 199.



Assignment. Solution

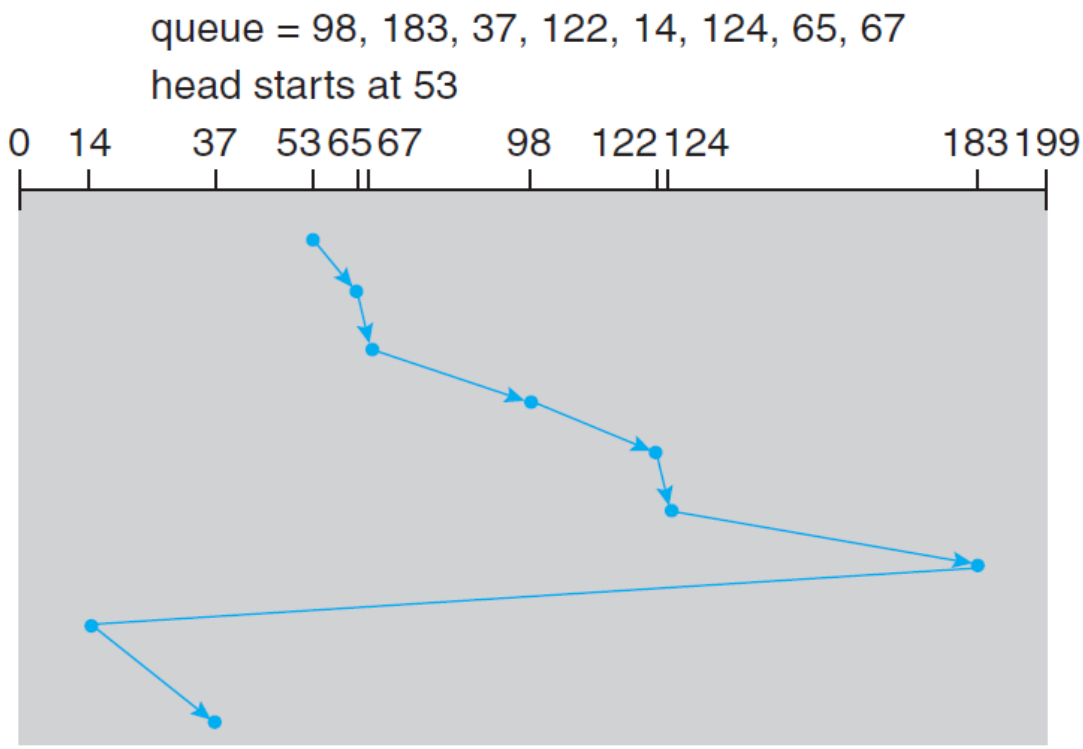
Consider initial head position 53 and request sequence [98, 183, 37, 122, 14, 124, 65, 67] Compute the total distance traveled using C-SCAN Assume Disk size = 0 to 199.

Smart Solution:

= (199-53) +199+ (37) = 382

LOOK Scheduling

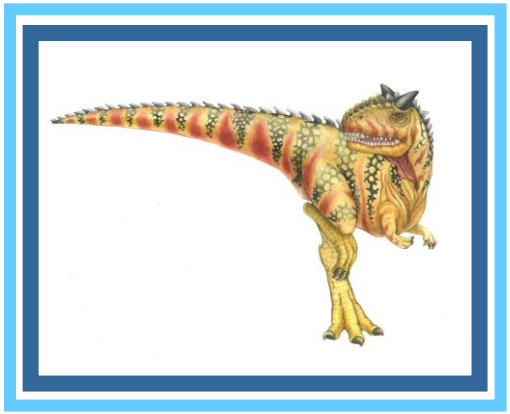
- LOOK and C-LOOK scheduling are versions of SCAN and C-SCAN that look for a request before continuing to move in a given direction.
- The arm goes only as far as the final request in each direction.
- Then, it reverses direction immediately, without going all the way to the end of the disk.



NVM Scheduling

- NVM Scheduling (Non-Volatile Memory Scheduling) for SSD is fundamentally different from traditional disk scheduling due to the lack of mechanical movement.
- Optimizing parallelism, wear leveling, and queue management is critical for modern SSDs and persistent memory.
- FCFS is about all that's needed for NVMs, with some variation, like merging adjacent writes.
- NVMs are much faster than HDDs at random reads and writes.
- HDDs can sometimes be as fast as NVMs for sequential I/O, especially sequential writing.

The End



Silberschatz, Galvin and Gagne ©2018

Disk Management

- Disk Partitioning OS allows the disk to be divided into multiple partitions (e.g., C: drive, D: drive).
- File System Management OS **formats** disks using file systems like NTFS, FAT32, ext4, etc.
- Disk Scheduling and Optimization OS schedules read/write operations efficiently.
- Disk Space Allocation OS decides how storage space is allocated to files and directories.
- Disk Protection and Security Implements access control and user permissions.
- Disk Error Handling & Backup OS detects bad sectors and recovers lost files.

Logical Formatting (High-Level Formatting)

- Boot Sector: the first sector of a storage device (hard drive, SSD, USB). contains the boot loader, which helps start the operating system. Stores information about the file system type, disk layout, and partition details.
- File Allocation Tables (FAT): a data structure that keeps track of where files are stored on the disk. Maps blocks to files.
- Directory Structures: organizes files and folders in a hierarchical format. Each directory (folder) contains metadata about its files (file name, size, location, (pointers to data blocks), permissions)