Unit 6: Device Management

6.1 I/O Devices and Hardware Interfaces

1. Classification of I/O Devices

Category	Characteristics	Examples
Block Devices	Fixed-size data transfers (blocks/sectors)	Hard disks, SSDs, USB drives
Character Devices	Stream-based byte transfers	Keyboards, mice, serial ports
Network Devices	Packet-based communication	Ethernet cards, Wi-Fi adapters
Special Devices	Non-standard interfaces	`/dev/null`, `/dev/random`

Key Concept:

- Block devices support random access; character devices are sequential.

---

2. Device Controllers

Function: Bridge between hardware and OS, translating high-level commands to device-specific operations.
Components:

- **Registers:** Store commands/data (e.g., status, control).
- **Local Buffer:** Temporary data storage.

Example (Disk Controller):

1. OS sends "read sector 42" command.
2. Controller moves disk arm, reads data into buffer.
3. Generates interrupt upon completion.

Advantages:

- Standardizes interface.
- Offloads low-level control from CPU.

---

3. Memory-Mapped I/O

It is a hardware/software architecture paradigm where:

- I/O devices are accessed by reading/writing to specific memory addresses
- The CPU uses standard memory access instructions (load/store) to interact with devices
- Eliminates the need for special I/O instructions (like x86's IN/OUT)

Pros:

- No special I/O instructions needed.

Cons:

- Memory space consumption.

---

4. Direct Memory Access (DMA)

Operation:

1. CPU sets up DMA controller (source, destination, length).
2. DMA transfers data directly between device and RAM.
3. Interrupt upon completion.

Example:

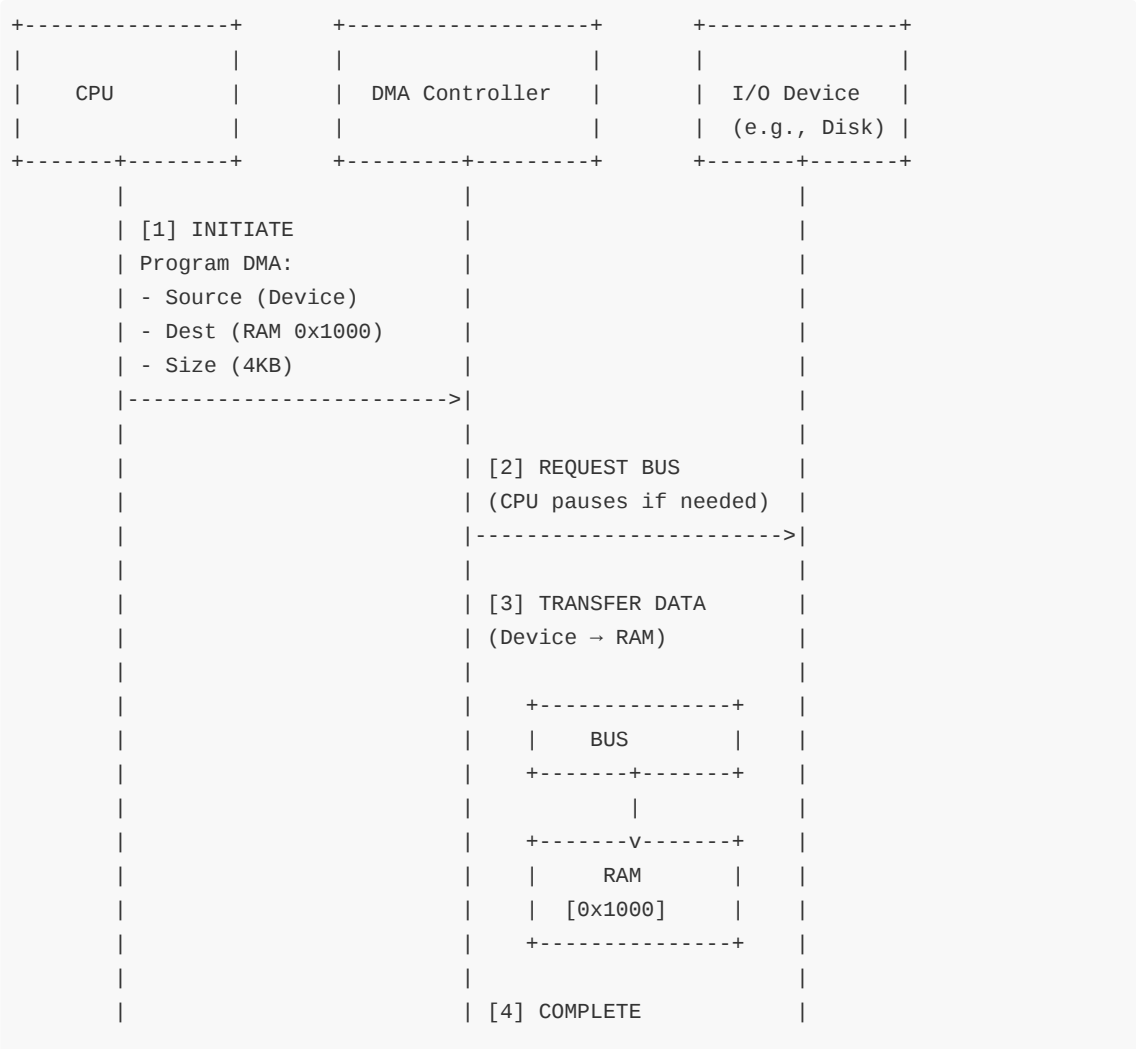
- Disk reads 4KB to RAM without CPU byte-by-byte involvement.

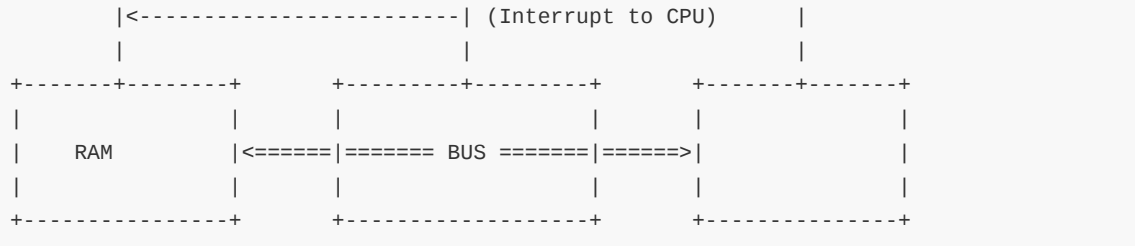
Advantages:

- Frees CPU for other tasks.
- Faster bulk transfers.

Disadvantages:

- Complexity (cache coherency issues).





DMA Data Transfer Process Steps:

1. **CPU Initiates DMA Transfer**
    - CPU programs the DMA controller with:
        - **Source:** The I/O device (e.g., disk)
        - **Destination:** RAM address (e.g., 0x1000)
        - **Size:** Amount of data to transfer (e.g., 4 KB)
    - CPU then signals the DMA controller to start the transfer.
2. **DMA Controller Requests Bus Control**
    - DMA controller requests control of the system bus to perform the transfer.
    - If CPU is currently using the bus, it **pauses** (bus arbitration) to allow DMA controller access.
3. **DMA Transfers Data**
    - DMA controller directly transfers data from the I/O device to RAM over the system bus without CPU intervention.
    - Data flows: **Device → DMA controller → RAM**
4. **DMA Transfer Complete - Interrupt CPU**
    - Once the transfer finishes, the DMA controller sends an **interrupt** signal to the CPU.
    - CPU resumes control and processes the data as needed.

---

5. Interrupts

Type	Trigger	Use Case
Hardware Interrupt	Device signals (e.g., keypress)	I/O completion
Software Interrupt	`int 0x80` (system calls)	Kernel service requests
Exception	CPU faults (e.g., divide by zero)	Error handling

Interrupt Handling Steps:

1. CPU finishes current instruction.
2. Saves context (registers, PC).
3. Jumps to Interrupt Service Routine (ISR).
4. Restores context and resumes.

6.2 I/O Software

Overview

I/O Software is a collection of programs and routines that control and manage input/output devices, acting as an intermediary between the hardware and user or application programs. It hides the complicated details of device hardware, allowing programs to perform I/O tasks easily and consistently.

For example, when a program wants to read data from a keyboard or print text on the screen, it uses I/O software like device drivers and operating system routines to communicate with the actual hardware without dealing directly with the device specifics.

Functions of I/O Software

1. **Device Independence**
    - Programs use generic I/O calls without needing to know hardware specifics.
    - I/O software translates these calls to device-specific commands.
2. **Buffering**
    - Temporarily stores data between the device and the program.
    - Helps match the speed differences between devices and CPU/memory.
3. **Error Handling**
    - Detects and recovers from device errors.
    - Reports errors to the system or user.
4. **Device Scheduling**
    - Manages multiple I/O requests.
    - Schedules device access efficiently, often optimizing throughput and response time.
5. **Device Initialization and Control**
    - Initializes device registers and configures device modes.
    - Controls device operations by sending commands.

Layers of I/O Software

Typically, I/O software is organized into layers that progressively abstract complexity:

1. **User-level I/O software (Application Programs)**
    - High-level interfaces and APIs for performing I/O.
    - Example: Reading a file, printing to screen.
2. **Device-independent I/O software**
    - Provides generic I/O operations regardless of device type.
    - Manages buffering, error handling, and device scheduling.
    - Examples: File system routines, buffering modules.

3. **Device Drivers (Device-dependent I/O software)**
    - Specific to each hardware device.
    - Translates generic requests into hardware-specific commands.
    - Manages device registers, interrupts, and data transfer.
4. **Interrupt Handlers and Device Controllers**
    - Lowest-level software interacting directly with hardware.
    - Handles interrupts and manages immediate hardware status.

Types of I/O Software Interfaces

- **System Calls**: OS provides system calls for I/O operations like `read()`, `write()`, `open()`, `close()`.
- **I/O Libraries**: Provide higher-level routines using system calls.
- **Device Drivers**: Offer device-specific routines to control hardware.

Example: Reading a File Using I/O Software

1. Application calls `read()` system call.
2. Device-independent I/O software checks buffers or schedules device access.
3. Device driver sends commands to hardware device.
4. Device performs data transfer using programmed I/O or DMA.
5. Interrupt signals completion; device driver and OS update status.
6. Data returns to application.

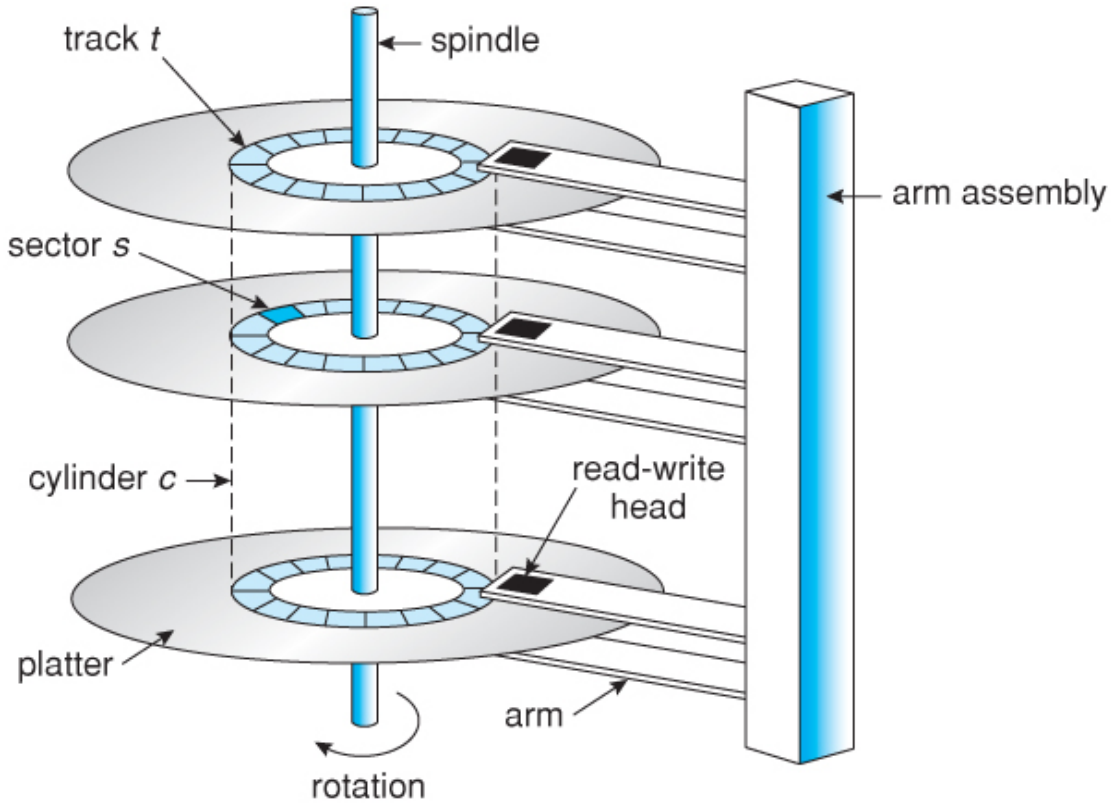
Summary Table

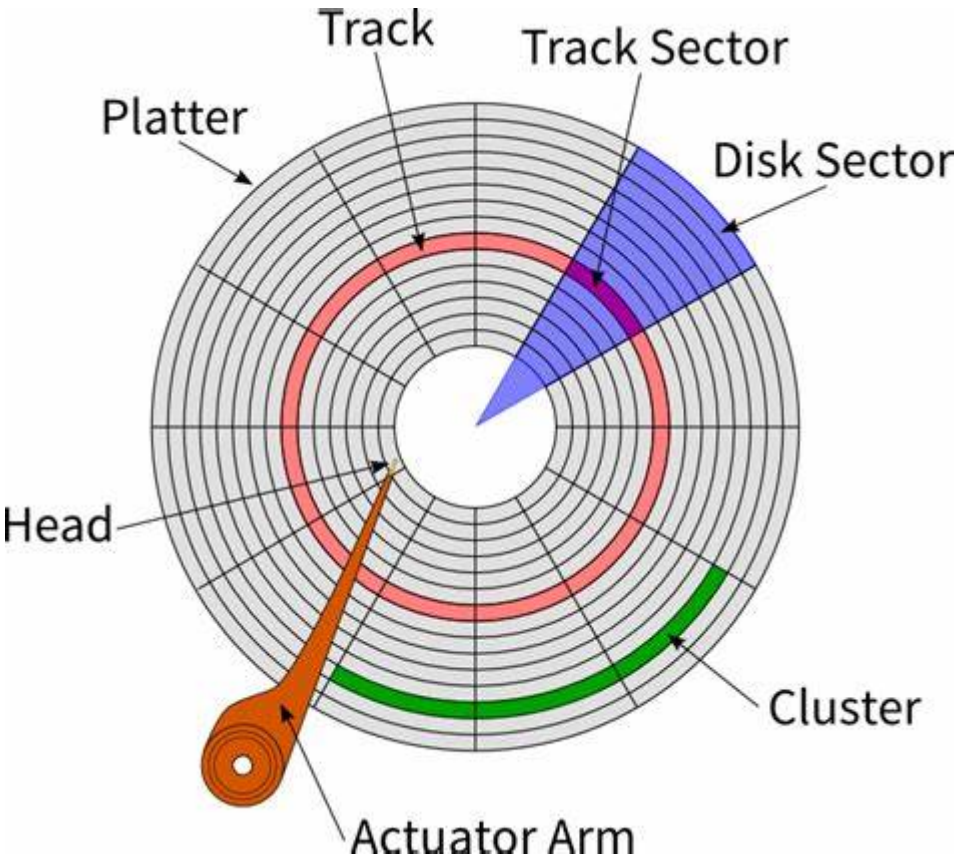
Function	Description
Device Independence	Hides device details from user programs
Buffering	Temporarily stores data to smooth transfers
Error Handling	Detects and manages device errors
Device Scheduling	Orders device access for efficiency
Device Control	Sends commands and configures devices

6.3 Disk Management

1. Disk Structure

- **Platters, Tracks, Sectors:**
    - 512B–4KB/sector (modern disks use 4K sectors).
- **Cylinder:** Same track across platters.


track t
spindle
arm assembly
sector s
cylinder c
read-write
head
platter
arm
rotation



2. Disk Scheduling Algorithms

Algorithm	Description
FCFS	First-Come-First-Served
SSTF	Shortest Seek Time First
SCAN	Elevator algorithm (full sweep)
LOOK	SCAN but reverses at last request
C-SCAN	Circular SCAN (resets to start)

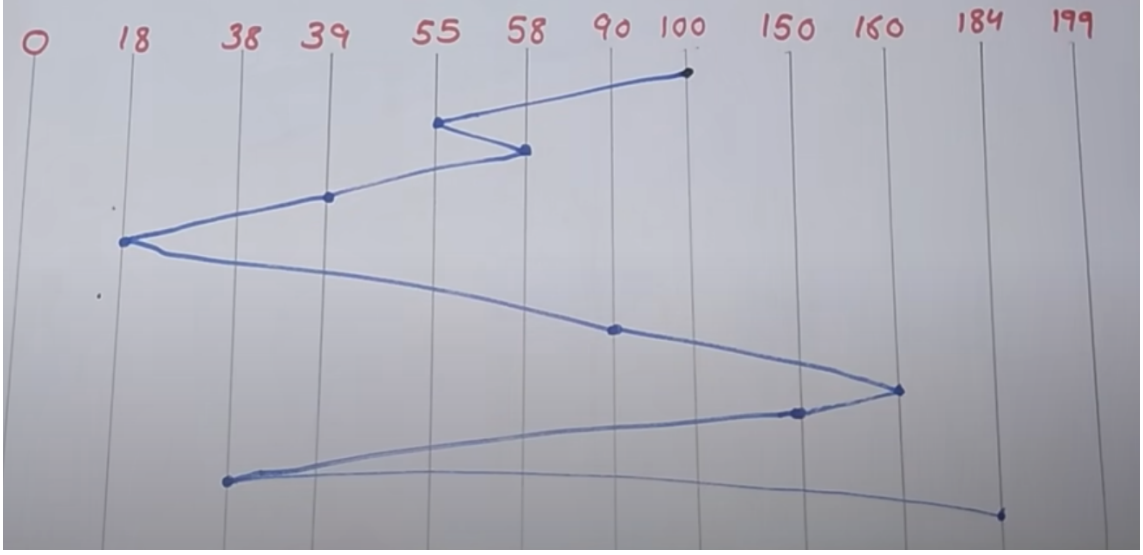
Numericals

Q) We assume a disk with 200 tracks and disk req^{t} queue has random request in it. The requested tracks in the order received by disk Schedular are 55, 58, 39, 18, 90, 160, 150, 38, 184 starting at track 100. Calculate the average seek length using FIFO, SSTF, SCAN, CSCAN, LOOK and CLOOK. Give which disk Scheduling is best for the scenario.

Seek time = 498 FCFS start (100)

Request (55, 58, 39, 18, 90, 160, 150, 38, 184)

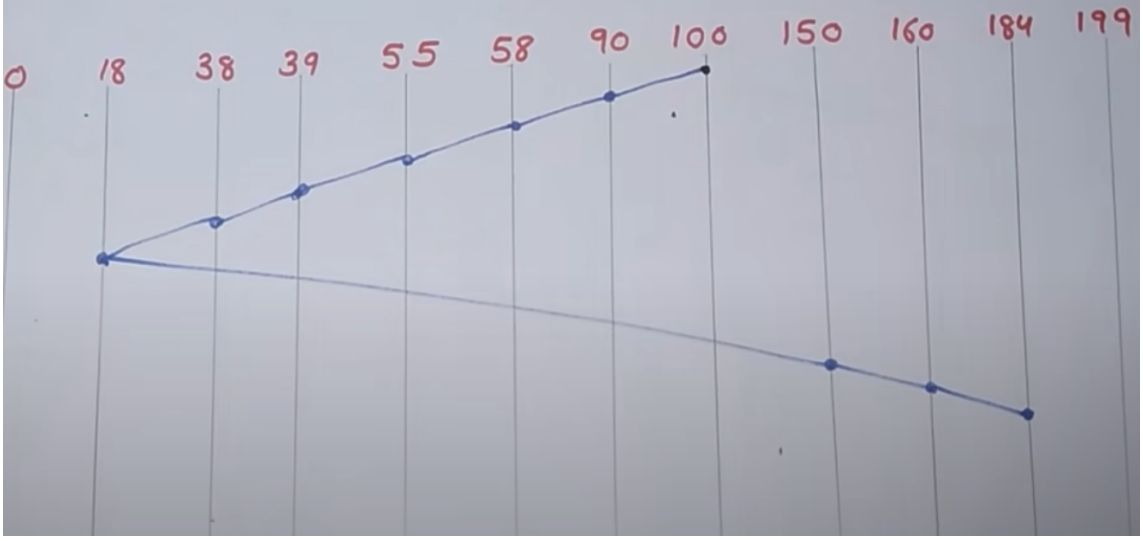
Queues (0, 55, 58, 39, 18, 90, 160, 150, 38, 184, 199)



Seek time: 248 SSTF start (100)

Request (55, 58, 39, 18, 90, 160, 150, 38, 184)

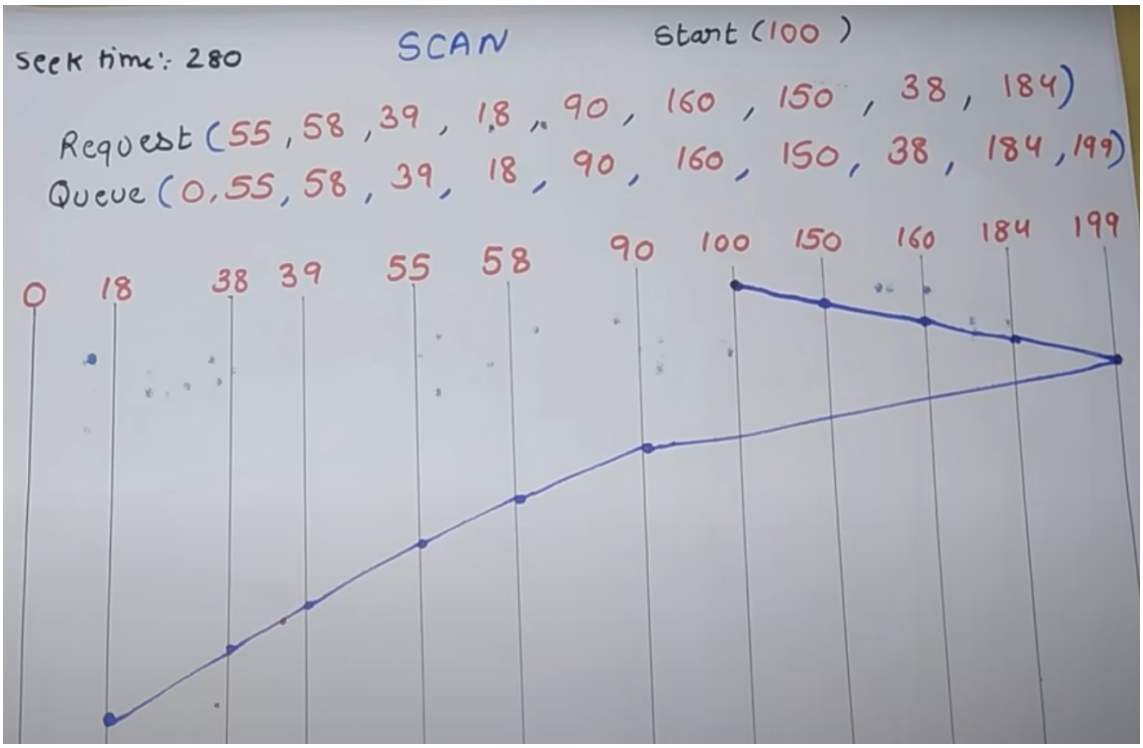
Queues (0, 55, 58, 39, 18, 90, 160, 150, 38, 184, 199)

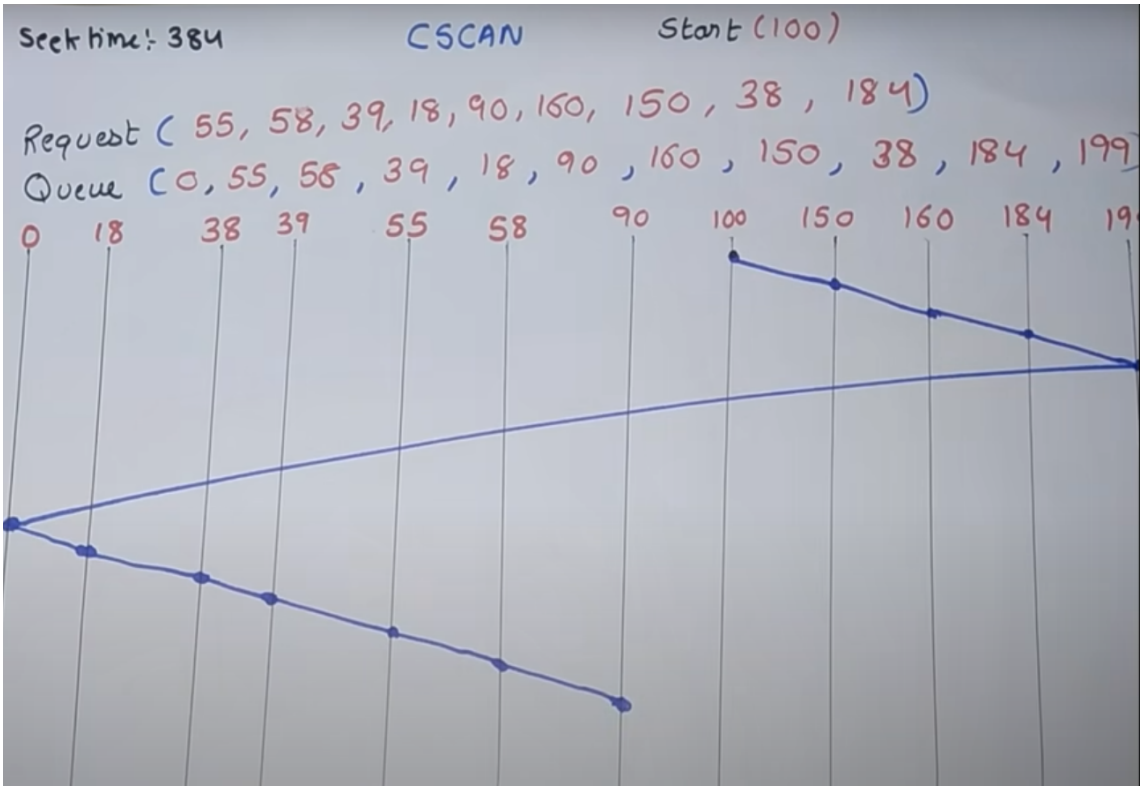


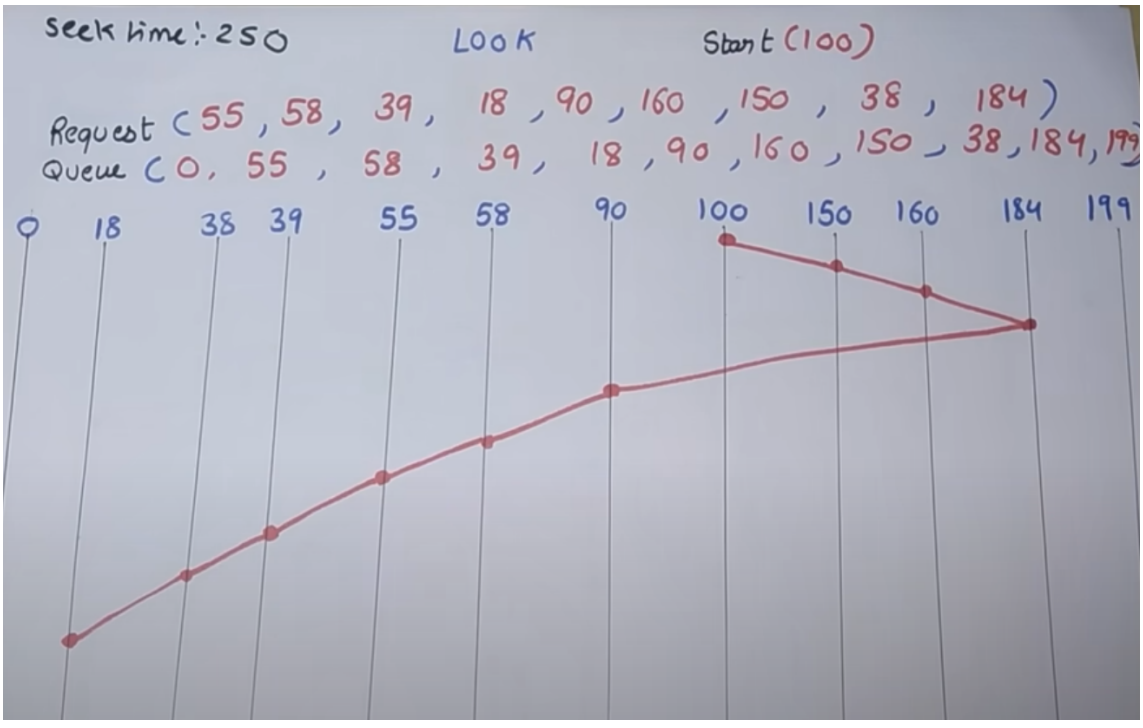
Seek time: 280 SCAN start (100)

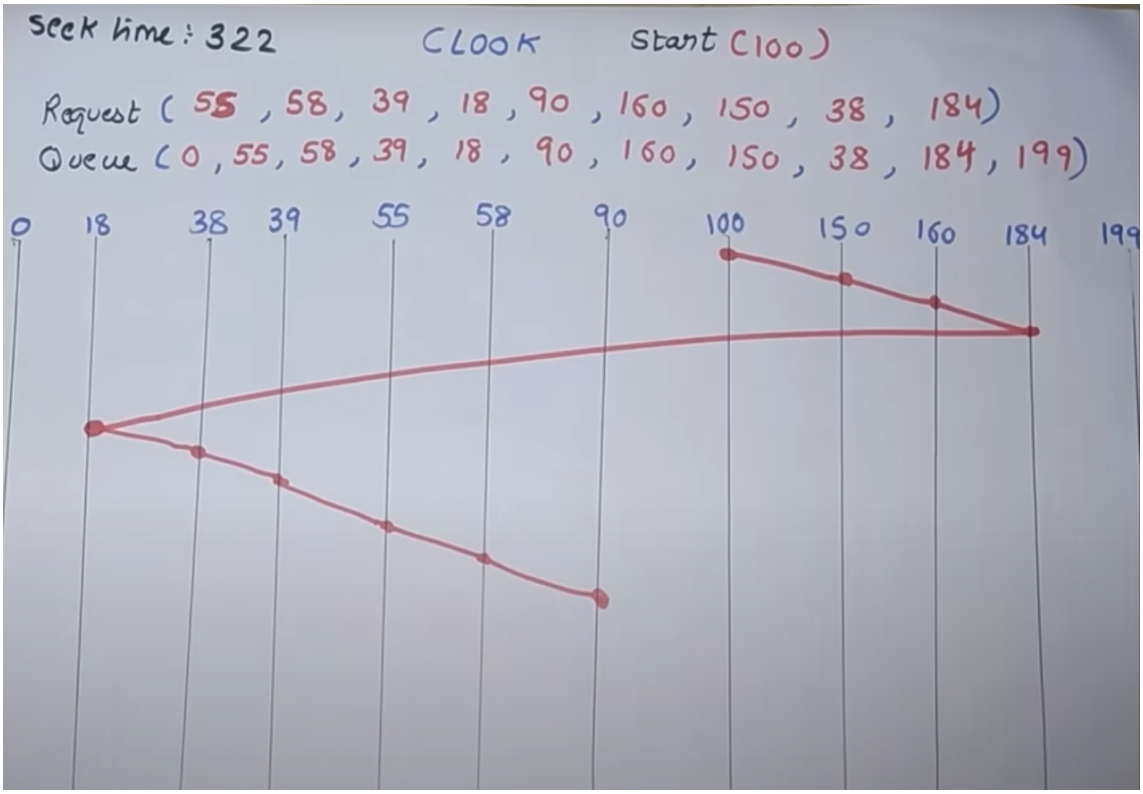
Request (55, 58, 39, 18, 90, 160, 150, 38, 184)

Queue (0, 55, 58, 39, 18, 90, 160, 150, 38, 184, 199)




Seek time: 384
CSCAN
Start (100)
Request (55, 58, 39, 18, 90, 160, 150, 38, 184)
Queue (0, 55, 58, 39, 18, 90, 160, 150, 38, 184, 199)
0 18 38 39 55 58 90 100 150 160 184 19


Seek time: 250
LOOK
Start (100)
Request (55, 58, 39, 18, 90, 160, 150, 38, 184)
Queue (0, 55, 58, 39, 18, 90, 160, 150, 38, 184, 199)
0 18 38 39 55 58 90 100 150 160 184 199



2. Find the seek time using SCAN, C-SCAN, Look and C-Look disk scheduling algorithms for processing the following request queue: 35, 70, 45, 15, 65, 20, 80, 90, 75, 130. Suppose the disk has tracks numbered from 0 to 150 and assume the disk arm to be at 30 and moving outward.
3. Consider a disk with 200 tracks and the queue has random requests from different processes in the order : 45, 48, 29, 17, 80, 150, 28 and 188. Find the seek time using FIFO, SSTF and SCAN. Assume the initial position of head as 100.
4. Suppose a disk has 201 cylinders, numbered from 0 to 200. At the same time the disk arm is at cylinder 10, and there is a queue of disk access requests for cylinders 30, 85, 90, 100, 105, 110, 135, and 145. Find the total seek time for the disk scheduling algorithm FCFS and SSTF. Assume the head is moving inward.
5. Suppose a disk has 201 cylinders, numbered from 0 to 200. At the same time the disk arm is at cylinder 95, and there is a queue of disk access requests for cylinders 82,170,43,140,24,16 and 190. Calculate the seek time for the disk scheduling algorithm FCFS,SSTF,SCAN and C-SCAN.

---

3. Disk Formatting Techniques

Disk formatting is the process of preparing a storage device like a hard disk or SSD for initial use by an operating system. It sets up the structure needed to store files and manage space.

Types of Disk Formatting

1. **Low-Level Formatting (Physical Formatting)**

- This is the process of dividing the disk into sectors and tracks.
- It defines the physical layout on the disk surface.
- Originally done at the factory to mark sector boundaries.
- Modern hard drives come pre-formatted at this level, so users rarely perform low-level formatting.
- It also checks for bad sectors on the disk.

2. **High-Level Formatting (Logical Formatting)**

   - Creates a file system structure on the disk.
   - Sets up file allocation tables or other metadata structures.
   - Prepares the disk so the OS can store files and directories.
   - Examples include formatting a disk as NTFS, FAT32, ext4, etc.
   - Usually done by the user when initializing a new disk or reformatting.

3. **Quick Format**

   - Only deletes the file system structures and marks the disk as empty.
   - Does **not** erase the actual data blocks.
   - Faster than a full format.
   - Used when you want to quickly erase and reuse the disk without scanning for bad sectors.

4. **Full Format**

   - Deletes file system structures **and** scans the entire disk surface for bad sectors.
   - Takes longer but ensures disk health.
   - Erases data by overwriting (depending on OS and options).

Summary Table

Formatting Type	Description	Performed by	Purpose
Low-Level Formatting	Divides disk into sectors and tracks	Usually factory or special tools	Physical layout setup
High-Level Formatting	Creates file system and metadata	Operating system / user	Prepare disk for files
Quick Format	Erases file system, no bad sector check	User / OS	Fast disk reset
Full Format	Erases file system + checks for bad sectors	User / OS	Disk health and data wipe

---

Example

When you buy a new hard drive, it usually has low-level formatting done by the manufacturer. To use it on your computer, you perform a **high-level format** by selecting a file system like NTFS or FAT32. If you want to erase everything quickly without checking the disk, you choose a **quick format**. To thoroughly check the disk for errors and erase all data, you perform a **full format**.

---

4. RAID Levels

RAID (Redundant Array of Independent/Inexpensive Disks) is a method of combining multiple physical hard drives (or SSDs) into one logical unit to improve performance, reliability, or both.

- It distributes and/or duplicates data across the drives.
- Managed by hardware (RAID controller) or software (OS-level RAID).
- Used in servers, data centers, and high-performance systems.

Level	Description	Redundancy?	Performance	Use Case
RAID 0	Striping	No	High	Video editing
RAID 1	Mirroring	Yes	Moderate	Critical databases
RAID 5	Striping + Parity	Yes	High read	File servers
RAID 6	Dual parity	Yes	High read	Archival storage
