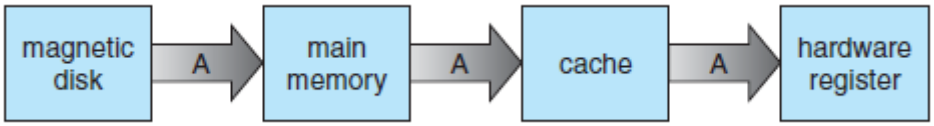
## MASS STORAGE STRUCTURE- OVERVIEW

- Main memory is usually too small to store all needed programs and data permanently.
- Main memory is a **volatile** storage device that loses its contents when power is turned off or otherwise lost.
- Thus, most computer systems provide **secondary storage** as an extension of main memory. The main requirement for secondary storage is that it be able to hold large quantities of data permanently.
- The most common secondary-storage device is a **magnetic disk**, which provides storage for both programs and data.
- Most of the secondary storage devices are internal to the computer such as the hard disk drive, the tape disk drive and even the compact disk drive and floppy disk drive.




**1) Magnetic Disks**

- Magnetic disks provide the bulk of secondary storage for modern computer systems.
- Each disk platter has a flat circular shape, like a CD. Common platter diameters range from 1.8 to 3.5 inches.
- The two surfaces of a platter are covered with a magnetic material. We store information by recording it magnetically on the platters.
- A read–write head "flies" just above each surface of every platter. 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**.
- **CYLINDER:** The set of tracks that are at one arm position makes up a **cylinder**.

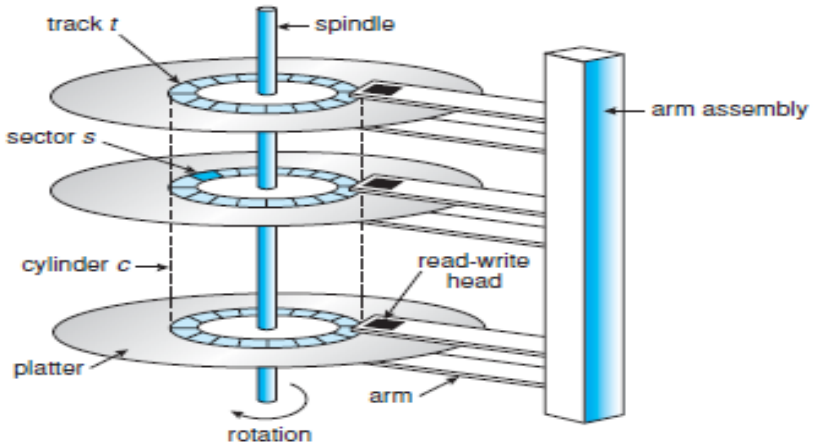



**Figure 10.1** Moving-head disk mechanism.

- The storage capacity of common disk drives is measured in gigabytes. When the disk is in use, a drive motor spins it at high speed. Most drives rotate 60 to 250 times per second, specified in terms of rotations per minute.
- Disk speed has two parts.
  - The **transfer rate** is the rate at which data flow between the drive and the computer.
  - The **positioning time**, or **random-access time**
- It consists of two parts:
  - **SEEK TIME**: The time necessary to move the disk arm to the desired cylinder, is called the **seek time**.

- **ROTATIONAL LATENCY**: The time necessary for the desired sector to rotate to the disk head, called the **rotational latency**.

- **HEAD CRASH**: The disk read write head has a danger that the head will make contact with the disk surface. Although the disk platters are coated with a thin protective layer, the head will sometimes damage the magnetic surface. This accident is called a **head crash**.
- A disk drive is attached to a computer by a set of wires called an **I/O bus**. Several kinds of buses are available, including **advanced technology attachment (ATA)**, **serial ATA (SATA)**, **eSATA universal serial bus (USB)**, and **fibre channel (FC)**.
- 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 host controller, typically using memory-mapped I/O ports

**2) Magnetic Tapes:**

- **Magnetic tape** was used as an early secondary-storage medium.
- It is relatively permanent and can hold large quantities of data.
- Its access time is slow compared with that of main memory and magnetic disk.
- In addition, random access to magnetic tape is about a thousand times slower than random access to magnetic disk, so tapes are not very useful for secondary storage.
- Tapes are used mainly for backup, for storage of infrequently used information, and as a medium for transferring information from one system to another.

**Disk Structure**

- Magnetic disk drives are addressed as large one-dimensional arrays of **logical blocks**, where the logical block is the smallest unit of transfer.
- The size of a logical block is usually 512 bytes, although some disks can be **low-level formatted** to have a different logical block size, such as 1,024 bytes.
- The one-dimensional array of logical blocks is mapped onto the sectors of the disk sequentially. Sector 0 is the first sector of the first track on the outermost cylinder.
- The number of sectors per track is not constant on some drives.
- For the disks that use **constant linear velocity (CLV)**, the density of bits per track is uniform.
- In **constant angular velocity (CAV)** the density of bits decreases from inner tracks to outer tracks to keep the data rate constant.

**Disk Attachment**

- Computers access disk storage in the following ways
    - Host attached Storage
    - Network Attached Storage
    - Storage Area Network.
- **HOST ATTACHED STORAGE:** Host-attached storage is storage accessed through local I/O ports. The typical desktop PC uses an I/O bus architecture called IDE or ATA.
- **NETWORK ATTACHED STORAGE**: A network-attached storage (NAS) device is a special-purpose storage system that is accessed remotely over a data network .Clients access network-attached storage via a remote-procedure-call interface such as NFS for UNIX systems or CIFS for Windows machines.
- **STORAGE AREA NETWORK**: A storage-area network (SAN) is a private network connecting servers and storage units.

**DISK SCHEDULING:**

- 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**: 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. This is called as Disk Scheduling.
- **Disk Components**:
- The two major components of the hard disk are Seek time and Rotational Latency.
- **Seek time**: The **seek time** is the time for the disk arm to move the heads to the cylinder containing the desired sector.
- **Rotational latency**: The **rotational latency** is the additional time for the disk to rotate the desired sector to the disk head.
- **Disk bandwidth**: The 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 Algorithms**:
    - First Come First Serve
    - Shortest Seek Time First
    - Scan Algorithm
    - Circular Scan Algorithm
    - Look Algorithm
    - Circular Look Algorithm

**1) FCFS Scheduling:**

- The simplest form of disk scheduling is, of course, the first-come, first-served (FCFS) algorithm.
- This algorithm is easy to implement but it generally does not provide the fastest service.
- **Example:** Consider, for example, Given a disk with 200 cylinders and a disk queue with requests 98, 183, 37, 122, 14, 124, 65, 67, for I/O to blocks on cylinders. Disk head is initially at 53.

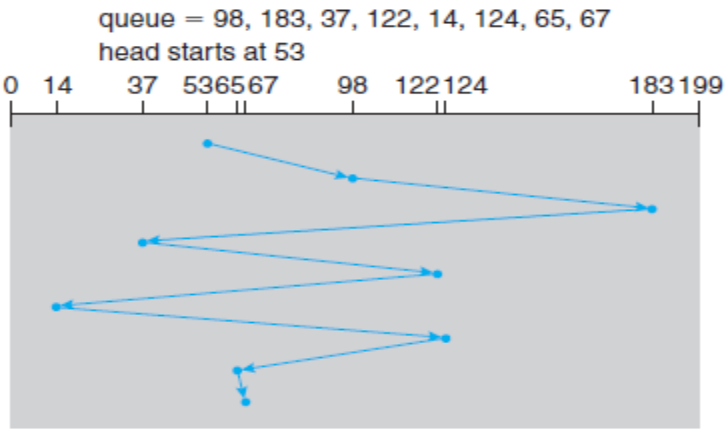
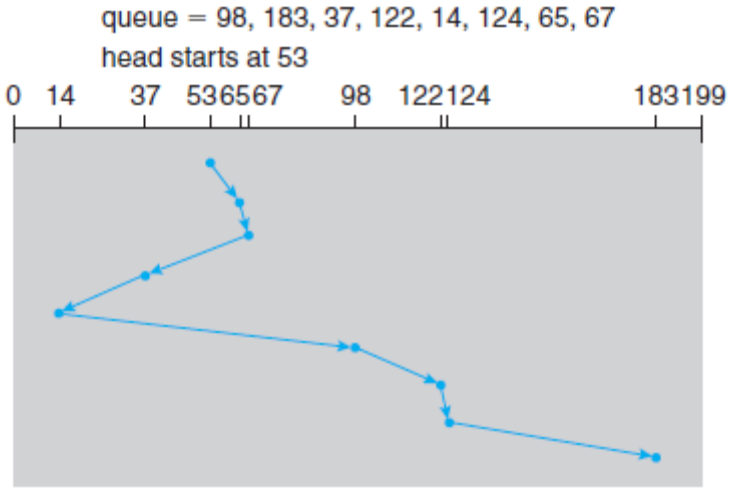



**Figure 10.4** FCFS disk scheduling.

- If the disk head is initially at cylinder 53, it will first move from 53 to 98, then to 183, 37, 122, 14, 24, 65, and finally to 67.
- The total head movements is
  Head Movements = (53-98)+(98-183)+((183-37)+(122-14)+(14-124)+(124-65)+(65-67)
  = 640 Head Movements.
- **Disadvantage**: The request from 122 to 14 and then back to 124 increases the total head movements.
- If the requests for cylinders 37 and 14 could be serviced together, before or after the requests for 122 and 124, the total head movement could be decreased and performance could be thereby improved.

**2) SSTF Scheduling:**

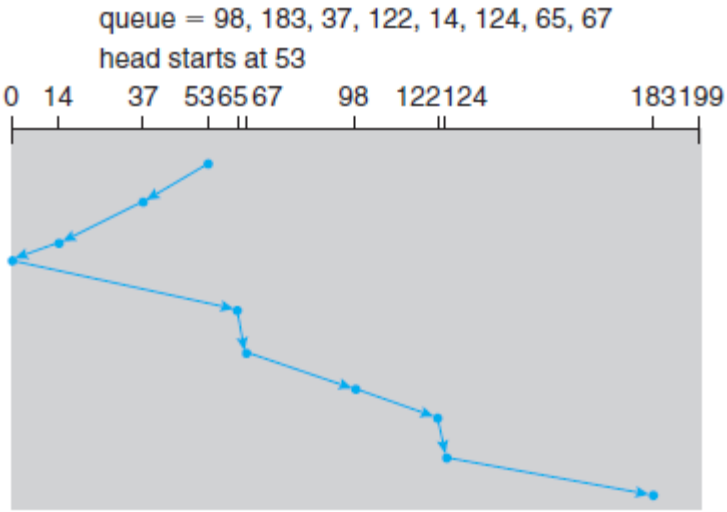
- The **shortest-seek-time-first (SSTF) algorithm** selects the request with the least seek time from the current head position. It chooses the pending request closest to the current head position.
- **Example:** Consider, for example, Given a disk with 200 cylinders and a disk queue with requests 98, 183, 37, 122, 14, 124, 65, 67, for I/O to blocks on cylinders. Disk head is initially at 53.
- The closest request to the initial head position (53) is at cylinder 65.
- Once we are at cylinder 65, the next closest request is at cylinder 67.
- From there, the request at cylinder 37 is closer than the one at 98, so 37 is served next.
- Continuing, we service the request at cylinder 14, then 98, 122, 124, and finally 183.
- This scheduling method results in a total head movement of only 236 cylinders
- SSTF algorithm gives a substantial improvement in performance.




- **Disadvantage**: SSTF may cause starvation of some requests.
- **STARVATION**: Suppose that we have two requests in the queue, for cylinders 14 and 186, and while the request from 14 is being serviced, a new request near 14 arrives. This new request will be serviced
  next, making the request at 186 wait. While this request is being serviced, another request close to 14 could arrive. In theory, a continual stream of requests near one another could cause the request for cylinder 186 to wait indefinitely.

**3) SCAN Scheduling:**

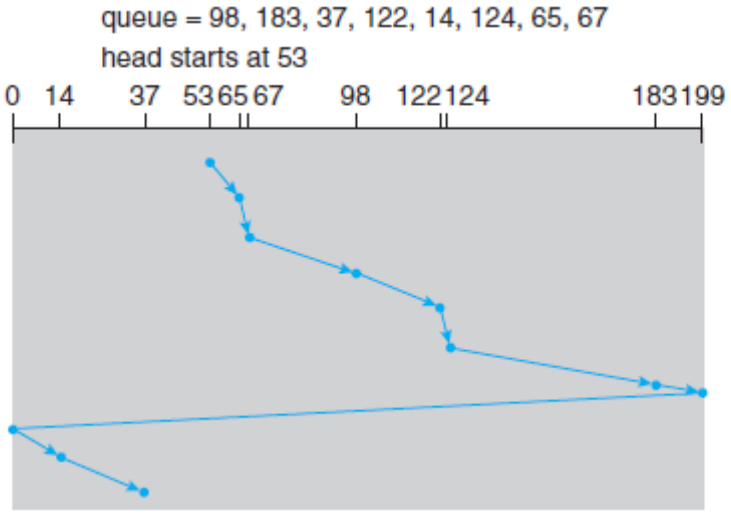
- In the **SCAN algorithm**, the disk arm starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk
- At the other end, the direction of head movement is reversed, and servicing continues.
- The head continuously scans back and forth across the disk.
- The SCAN algorithm is sometimes called the **elevator algorithm.**
- **Example:** Consider, for example, Given a disk with 200 cylinders and a disk queue with requests 98, 183, 37, 122, 14, 124, 65, 67, for I/O to blocks on cylinders. Disk head is initially at 53.
- Assuming that the disk arm is moving toward 0 the head will next service 37 and then 14.




- At cylinder 0, the arm will reverse and will move toward the other end of the disk, servicing the requests at 65, 67, 98, 122, 124, and 183.
- If a request arrives 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, reverses direction, and comes back.

**4) Circular SCAN Algorithm:**

- **Circular SCAN (C-SCAN) scheduling** is a variant of SCAN designed to provide a more uniform wait time.
- 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.
- The C-SCAN scheduling algorithm essentially treats the cylinders as a circular list that wraps around from the final cylinder to the first one.
- **Example:** Consider, for example, Given a disk with 200 cylinders and a disk queue with requests 98, 183, 37, 122, 14, 124, 65, 67, for I/O to blocks on cylinders. Disk head is initially at 53.




**5) LOOK scheduling:**

- The **LOOK** algorithm is the same as the **SCAN** algorithm in that it also services the requests on both directions of the disk head, but it “Looks" ahead to see if there are any requests pending in the direction of head movement.
- If no requests are pending in the direction of head movement, then the disk head traversal will be reversed to the opposite direction and requests on the other direction can be served.

- In LOOK scheduling, the arm goes only as far as final requests in each direction and then reverses direction without going all the way to the end.
- Consider an example, given a disk with 200 cylinders (0-199), suppose we have 8 pending requests: 98, 183, 37, 122, 14, 124, 65, 67 and that the read/write head is currently at cylinder 53. In order to complete these requests, the arm will move in the increasing order first and then will move in decreasing order after reaching the end. So, the order in which it will execute is 65, 67, 98, 122, 124, 183, 37, and 14.

  **Note :( Draw the Diagram and calculate the head movements for the previous example)**

**6) C-LOOK Scheduling:**

- This is just an enhanced version of C-SCAN.
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without servicing all the way to the end of the disk and then turns the next direction to provide the service.

  **Note :( Draw the Diagram and calculate the head movements for the previous example)**

## DISK MANAGEMENT:

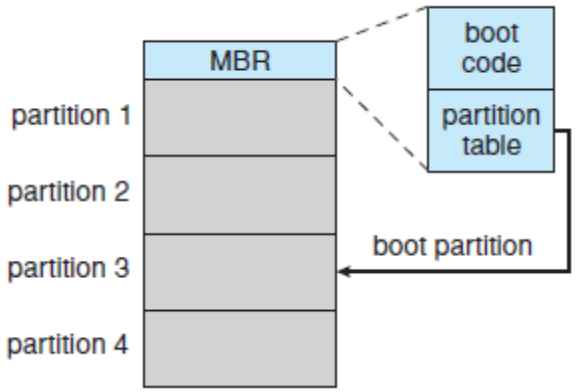
- The operating system is responsible for disk management.
- The Major Responsibility includes
  - Disk Formatting,
  - Booting from disk
  - Bad-block recovery.

**1. Disk Formatting:**

- The Disk can be formatted in two ways,
  - Physical or Low level Formatting,
  - Logical Or High Level Formatting
- **Physical or Low level Formatting:**
- Before a disk can store data, it must be divided into sectors that the disk controller can read and write. This process is called **low-level formatting**, or **physical formatting**.
- Low-level formatting fills the disk with a special data structure for each sector.
- The data structure for a sector typically consists of a header, a data area (usually 512 bytes in size), and a trailer.
- The Header contains the Sector Number and the Trailer contains the Error correction code.
- 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. It then reports a recoverable **soft error**.
- **Logical Formatting Or High Level Formatting:**
- The operating record its own data structures on the disk during Logical formatting.
- It does so in two steps.
- The first step is to **partition** the disk into one or more groups of cylinders.
- The operating system can treat each partition as though it were a separate disk.
- For instance, one partition can hold a copy of the operating system's executable code, while another holds user files.
- The second step is **logical formatting**, or creation of a file system. In this step, the operating system stores the initial file-system data structures onto the disk.
- These data structures may include maps of free and allocated space and an initial empty directory.
- **CLUSTERS:** To increase efficiency, most file systems group blocks together into larger chunks, frequently called **clusters**.
- **RAW DISK:** Some operating systems give special programs the ability to use a disk partition as a large sequential array of logical blocks, without any file-system data structures. This array is sometimes called the **raw disk,** and I/O to this array is termed **raw I/O.**

**2. Boot Block**

- The Process of Starting a computer System by loading the Operating system in the main memory is called as System Booting.
- This is done by a initial program called as Bootstrap program initializes all aspects of the system, from CPU registers to device controllers and the contents of main memory, and then starts the operating system.
- The bootstrap is stored in **read-only memory (ROM)**.The Problem here is that changing this bootstrap code requires changing the ROM hardware chips.
- To overcome this most systems store a tiny bootstrap loader program in the boot ROM whose only job is to bring in a full bootstrap program from disk.
- The full bootstrap program can be changed easily: a new version is simply written onto the disk.
- **BOOT DISK**: The full bootstrap program is stored in the "boot blocks" at a fixed location on the disk. A disk that has a boot partition is called a **boot disk** or **system disk**.
- The code in the boot ROM instructs the disk controller to read the boot blocks into memory and then starts executing that code which in turn loads the entire Operating System.
- **EXAMPLE: Boot Process in Windows.**
- **BOOT PARTITION**: Windows allows a hard disk to be divided into partitions, and one partition called as the **boot partition** contains the operating system and device drivers.
- The Windows system places its boot code in the first sector on the hard disk, which it terms the **master boot record**, or **MBR**.
- Booting begins by running code that is resident in the system's ROM memory. This code directs the system to read the boot code from the MBR.
- Once the system identifies the boot partition, it reads the first sector from that partition and continues with the remainder of the boot process, which includes loading the various subsystems and system services.




3. **Bad Blocks**

- A bad block is a damaged area of magnetic storage media that cannot reliably be used to store and retrieve data. Depending on the disk and controller in use, these blocks are handled in a variety of ways.
- One strategy is to scan the disk to find bad blocks while the disk is being formatted. Any bad blocks that are discovered are flagged as unusable so that the file system does not allocate them.
- In Some systems the controller maintains a list of bad blocks on the disk. This can be handled in two ways
    - Sector Sparing
    - Sector Slipping
- **SECTOR SPARING**: 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**.
- **Example**:
    - The operating system tries to read logical block 87.

- The controller calculates the ECC and finds that the sector is bad. It reports this finding to the operating system.
- The next time the system is rebooted, a special command is run to tell the controller to replace the bad sector with a spare.
- After that, whenever the system requests logical block 87, the request is translated into the replacement sector's address by the controller.

❖ **SECTOR SLIPPING**: The Process of moving all the sectors down one position from the bad sector is called as sector slipping.

❖ **Example:** If the logical block 17 becomes defective and the first available spare follows sector 202. Sector slipping then remaps all the sectors from 17 to 202, moving them all down one spot. That is, sector 202 is copied into the spare, then sector 201 into 202, then 200 into 201, and so on, until sector 18 is copied into sector 19. Slipping the sectors in this way frees up the space of sector 18 so that sector 17 can be mapped to it.