

OPERATING SYSTEM

Chapter 7

DISK MANAGEMENT

Detailed Study Notes for Engineering Students

Topics Covered

Disk Structure

Disk Scheduling (FCFS, SSTF, SCAN, C-SCAN)

Disk Reliability

Disk Formatting, Boot Block and Bad Blocks

15 MCQs with Answer Key | 10 Broad Questions

Contents

- 7.1 Introduction to Disk Management
- 7.2 Disk Structure
- 7.3 Disk Scheduling
- 7.4 Disk Formatting
- 7.5 Boot Block
- 7.6 Bad Blocks
- 7.7 Disk Reliability
- 7.8 Chapter Summary and Quick Revision
- 7.9 Multiple Choice Questions (15)
- 7.10 Answer Key
- 7.11 Broad Questions (10)

7.1 Introduction to Disk Management

Main memory is small and volatile, so a computer needs **secondary storage** to keep programs and data permanently. The most common secondary storage devices are **magnetic disks** (hard disk drives) and **solid-state drives** (SSDs). The operating system is responsible for using these devices efficiently. Because a disk is far slower than the CPU and memory, the way requests are ordered and the way the disk is organised have a large effect on overall system performance.

Responsibilities of the OS in Disk Management

- **Disk structure and addressing:** presenting the disk to software as a simple array of logical blocks.
- **Disk scheduling:** deciding the order in which pending I/O requests are served to reduce head movement.
- **Disk formatting:** preparing a raw disk for use (physical formatting, partitioning and logical formatting).
- **Boot management:** storing and loading the bootstrap program from the disk.
- **Bad-block management:** detecting and hiding defective sectors.
- **Reliability:** protecting data against disk failure using error-correcting codes, redundancy and RAID.

7.2 Disk Structure

A magnetic disk drive consists of one or more circular **platters** coated with magnetic material, mounted on a common **spindle** that rotates at a constant speed (commonly 5400, 7200 or 15000 revolutions per minute, RPM). Each platter has two **surfaces**. A **read-write head** floats just above each surface and all the heads are attached to a single **arm assembly** that moves them together across the platters.

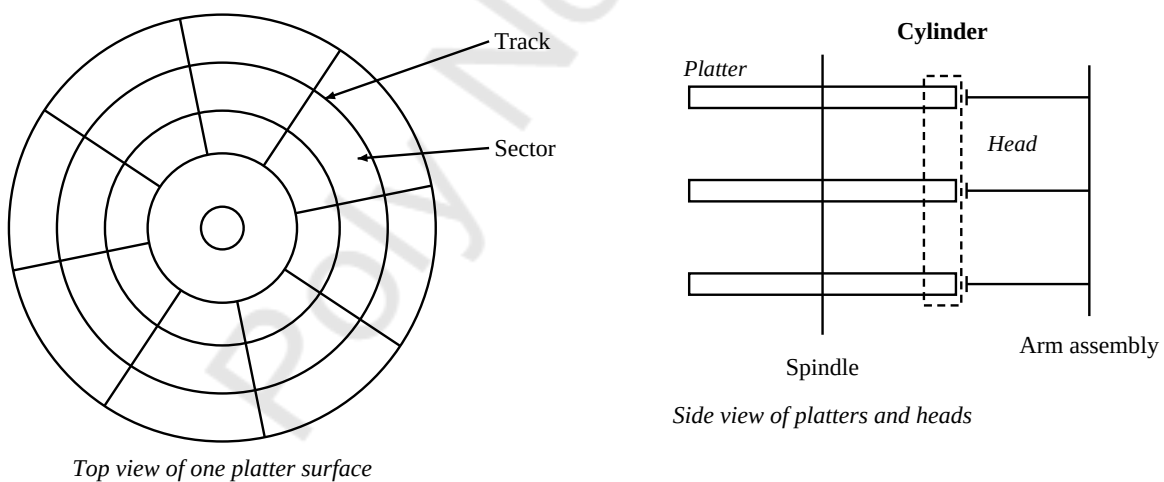


Fig. 7.1: Structure of a magnetic disk (top view and side view)

7.2.1 Physical Terms

Term	Meaning
Platter	Circular disk with magnetic coating; data is stored on both surfaces.
Track	A circular ring on a platter surface at a fixed distance from the centre.
Sector	Smallest unit of transfer; a track is divided into sectors (typically 512 bytes or 4 KB of data).
Cylinder	The set of tracks at the same arm position on all platters; the heads can read a whole cylinder without moving.
Read-write head	Device that reads or writes magnetic patterns on the surface (one head per surface).
Spindle and arm	Spindle rotates the platters; the arm moves the heads radially to select a track.
Disk controller	Electronics that control the drive and transfer data between the drive and the host bus.

7.2.2 Logical Addressing

Modern drives are addressed as a large one-dimensional array of **logical blocks**, the smallest unit of transfer. This array is mapped onto the sectors of the disk sequentially: block 0 is the first sector of the first track on the outermost cylinder, the mapping proceeds through that track, then the rest of the tracks in that cylinder, and then through the remaining cylinders from the outermost to the innermost. This is called **Logical Block Addressing (LBA)**. The older method used the triple (*cylinder, head, sector*), called CHS addressing.

- **Constant Linear Velocity (CLV):** the density of bits per track is uniform, so outer tracks hold more sectors; the rotation speed changes to keep the data rate constant (used in CD-ROM and DVD).
- **Constant Angular Velocity (CAV):** the rotation speed is constant, so the bit density decreases from the inner to the outer tracks to keep the data rate constant (used in hard disks).

7.2.3 Disk Access Time

The time to access data on a disk has three components:

Component	Meaning	Notes
Seek time	Time to move the arm to the required cylinder	Usually the largest component; average is a few milliseconds
Rotational latency	Time for the required sector to rotate under the head	Average = half of one rotation time
Transfer time	Time to actually read or write the data	= data size / transfer rate

Formula: Disk access time = Seek time + Rotational latency + Transfer time. For a 7200 RPM disk, one rotation takes $60/7200 \text{ s} = 8.33 \text{ ms}$, so the average rotational latency is $8.33/2 = 4.17 \text{ ms}$.

The **disk bandwidth** is the total number of bytes transferred divided by the time between the first request for service and the completion of the last transfer.

7.2.4 Magnetic Disk versus Solid-State Drive

Basis	Magnetic disk (HDD)	Solid-state drive (SSD)
Technology	Rotating platters, moving heads	Flash memory, no moving parts
Seek and rotational delay	Present (milliseconds)	None; very low latency
Speed	Slower, especially for random access	Much faster
Cost per GB	Low	Higher
Failure	Mechanical wear and head crashes	Limited write cycles per cell (wear levelling is used)
Disk scheduling	Important	Less important (FCFS often enough)

7.2.5 Disk Attachment

- **Host-attached storage:** accessed through local I/O ports (SATA, SAS, NVMe, USB).
- **Network-attached storage (NAS):** accessed remotely over a network using a file protocol such as NFS or CIFS.
- **Storage-area network (SAN):** a private network that connects servers and storage units, giving block-level access and flexibility.

7.3 Disk Scheduling

In a multiprogramming system many processes may be waiting for disk I/O at the same time, so a **queue of requests** builds up for each disk. Each request specifies the operation (read or write), the disk address, the memory address and the amount of data. If the drive and controller are free, the request is served immediately; otherwise it joins the queue. **Disk scheduling** is the method of choosing which pending request to serve next. Since seek time is the dominant cost, the main aim is to **minimise the total head movement** and so reduce the seek time.

Goals and Criteria

- Maximise **throughput** (requests served per unit time).
- Minimise average **response (waiting) time**.
- Minimise the **variance** of response time (predictability).
- Provide **fairness** and avoid **starvation** of any request.

Common example used below: A disk has 200 cylinders (0 to 199). The request queue, in arrival order, is 98, 183, 37, 122, 14, 124, 65, 67. The head is currently at cylinder 53.

7.3.1 FCFS (First-Come, First-Served) Scheduling

Requests are served strictly in the order of arrival. It is the simplest algorithm and is naturally fair, but it does not try to minimise head movement, so it often gives long swings of the head.

Head movement: 53 - 98 - 183 - 37 - 122 - 14 - 124 - 65 - 67

Move	Distance
53 to 98	45
98 to 183	85
183 to 37	146
37 to 122	85
122 to 14	108
14 to 124	110
124 to 65	59
65 to 67	2
Total head movement	640 cylinders

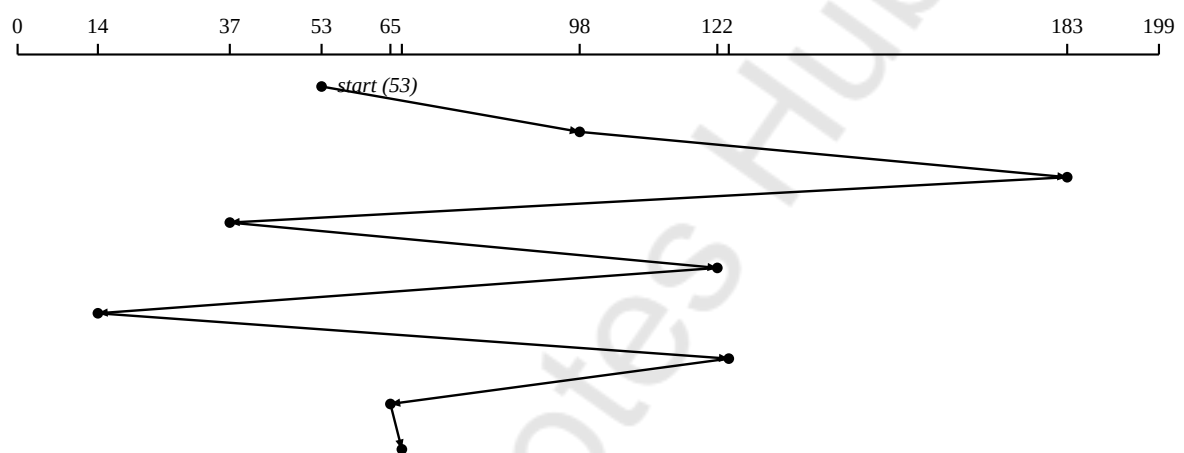


Fig. 7.2: FCFS head movement (total 640 cylinders)

- **Advantages:** simple, fair, no starvation.
- **Disadvantages:** high average seek time, poor throughput, wild swings (for example 122 to 14 and then to 124).

7.3.2 SSTF (Shortest Seek Time First) Scheduling

SSTF selects the pending request that is **closest to the current head position**, that is, with the minimum seek time. It is similar to Shortest-Job-First (SJF) scheduling of the CPU.

Head movement: 53 - 65 - 67 - 37 - 14 - 98 - 122 - 124 - 183

Move	Distance
53 to 65	12
65 to 67	2
67 to 37	30
37 to 14	23
14 to 98	84
98 to 122	24
122 to 124	2
124 to 183	59
Total head movement	236 cylinders

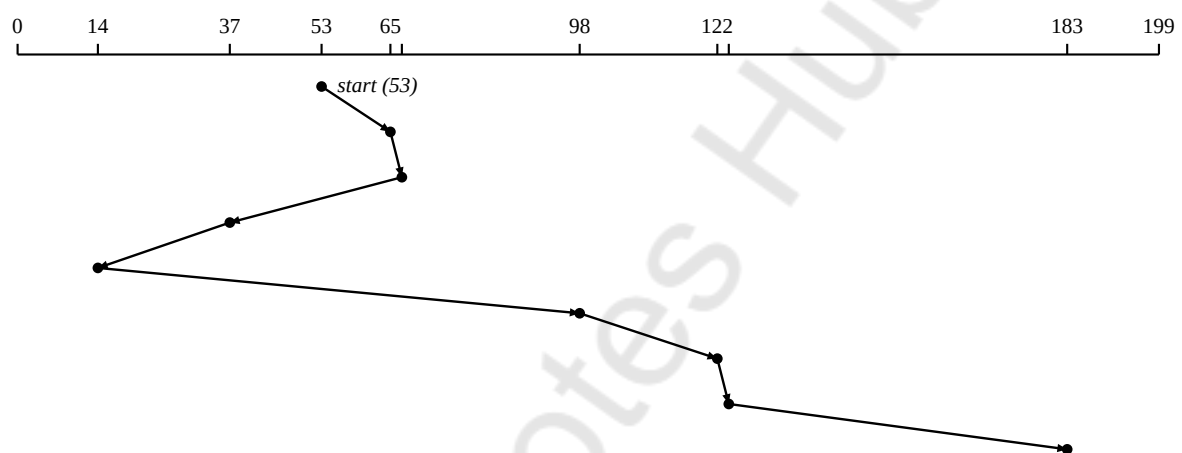


Fig. 7.3: SSTF head movement (total 236 cylinders)

- **Advantages:** much better throughput and lower average response time than FCFS.
- **Disadvantages:** may cause **starvation** because requests far from the head may be postponed indefinitely if new nearby requests keep arriving; it is not optimal; response time has high variance.

7.3.3 SCAN (Elevator) Scheduling

In SCAN, the arm starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder. At the other end the direction is **reversed** and servicing continues. It behaves like an elevator in a building, so it is also called the **elevator algorithm**. A request just in front of the head is served quickly, but one just behind the head must wait for the arm to go to the end and come back.

Assuming the head is moving toward cylinder 0: 53 - 37 - 14 - 0 - 65 - 67 - 98 - 122 - 124 - 183

Move	Distance
53 to 37	16
37 to 14	23
14 to 0	14
0 to 65	65
65 to 67	2
67 to 98	31
98 to 122	24
122 to 124	2
124 to 183	59
Total head movement	236 cylinders

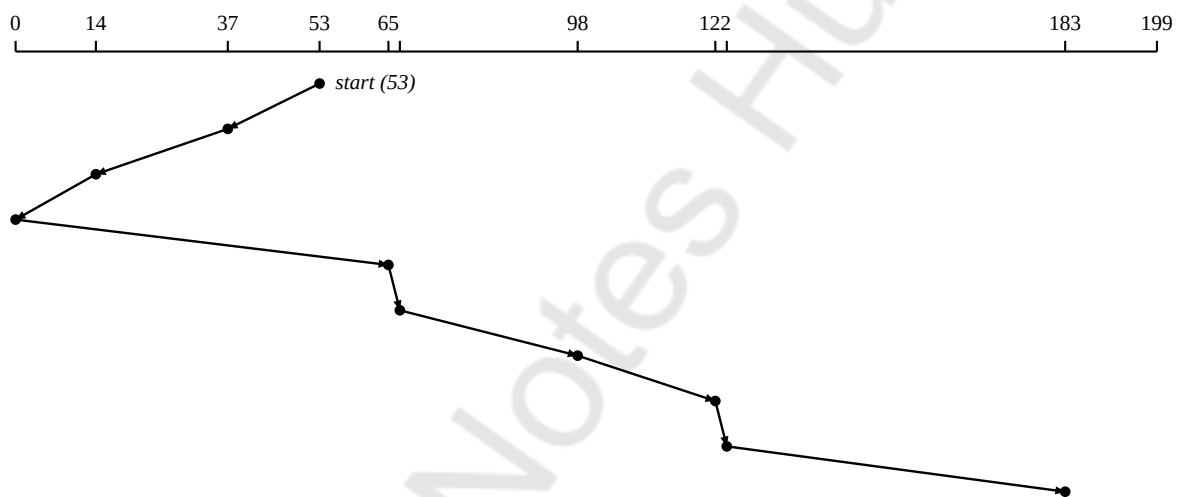


Fig. 7.4: SCAN head movement (total 236 cylinders)

- **Advantages:** no starvation, good throughput, lower variance than SSTF.
- **Disadvantages:** requests at the extreme ends are served less often; cylinders just visited by the head have to wait for a long time (uneven waiting time); the head goes to the end even if no request is there.

7.3.4 C-SCAN (Circular SCAN) Scheduling

C-SCAN is a variant of SCAN designed to give a more **uniform waiting time**. The head moves from one end of the disk to the other servicing requests along the way, but when it reaches the other end it **immediately returns to the beginning** without servicing any requests on the return trip, and then starts again. The cylinders are treated as a circular list that wraps from the last cylinder to the first.

Assuming the head moves toward cylinder 199: 53 - 65 - 67 - 98 - 122 - 124 - 183 - 199 - 0 - 14 - 37

Move	Distance
53 to 65	12
65 to 67	2
67 to 98	31
98 to 122	24
122 to 124	2
124 to 183	59
183 to 199	16
199 to 0 (return, no service)	199
0 to 14	14
14 to 37	23
Total head movement	382 cylinders (183 if the return trip is not counted)

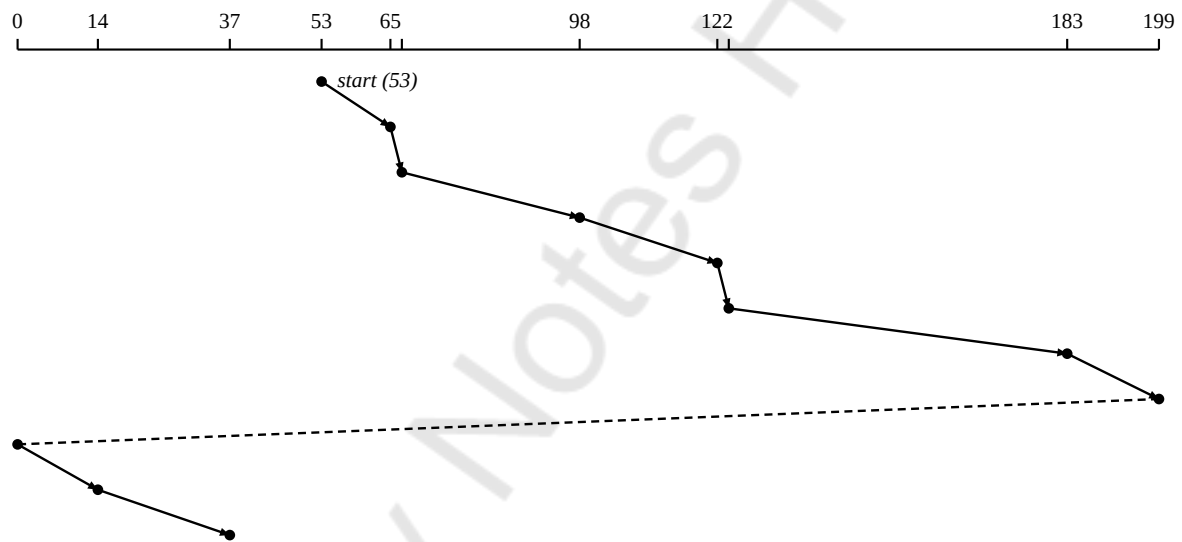


Fig. 7.5: C-SCAN head movement (dashed line = return without service)

- **Advantages:** more uniform waiting time than SCAN; treats all cylinders more fairly.
- **Disadvantages:** extra head movement for the return trip; head still travels to the ends.

7.3.5 LOOK and C-LOOK (Practical Versions)

In practice, SCAN and C-SCAN are implemented as **LOOK** and **C-LOOK**. The arm goes only as far as the **last request** in each direction and then reverses (LOOK) or jumps back to the first pending request (C-LOOK), instead of travelling to the physical end of the disk. For the same example: LOOK (moving toward 0 first) gives 208 cylinders and C-LOOK (moving toward 199 first, including the jump from 183 to 14) gives 322 cylinders.

7.3.6 Comparison of Disk Scheduling Algorithms

Basis	FCFS	SSTF	SCAN	C-SCAN
Selection rule	Order of arrival	Nearest request	Sweep to one end, then reverse	Sweep to one end, jump back, sweep again
Total movement (example)	640	236	236	382
Starvation	No	Possible	No	No
Throughput	Low	High	Good	Good
Waiting-time variance	Low (but long)	High	Moderate	Low (uniform)
Best suited for	Light load	Moderate load	Heavy load	Very heavy load

Choice of algorithm: SSTF is common and has a natural appeal because it increases performance over FCFS. SCAN and C-SCAN perform better for systems that place a heavy load on the disk, because they are less likely to cause starvation. Performance also depends on the number and type of requests, and on the file-allocation method (contiguous files cause less head movement than linked or indexed files). Because of this, the scheduling algorithm should be written as a separate module of the OS so that it can be replaced if needed. Note that the disk controller itself may also reorder requests (for example, to take rotational latency into account), and that the OS may impose its own priorities (such as favouring demand paging).

7.4 Disk Formatting

A new magnetic disk is just a blank slate: it is a platter of magnetic recording material. Before a disk can store data, it must be divided into sectors that the controller can read and write. This process is called **formatting** and has three stages.

7.4.1 Low-Level (Physical) Formatting

Low-level formatting fills the disk with a special data structure for each sector. The structure consists of a **header**, a **data area** (usually 512 bytes, now often 4 KB) and a **trailer**. The header and trailer contain information used by the disk controller, such as the sector number and an **error-correcting code (ECC)**. When the controller writes data, it updates the ECC 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; a mismatch shows that the data area has been corrupted, and ECC can also repair small errors (a *soft error*).

Header (sector no., sync)	Data area (512 bytes or 4 KB)	Trailer (ECC code)
------------------------------	----------------------------------	-----------------------

One sector after low-level (physical) formatting

Fig. 7.6: Layout of a sector after low-level formatting

Low-level formatting is normally done by the manufacturer. It lets the manufacturer test the disk and set up the mapping from logical block numbers to good sectors. Some drives allow the sector size to be chosen (for example 256, 512 or 1024 bytes); larger sectors mean fewer sectors per track and less header overhead, but more internal waste for small files.

7.4.2 Partitioning

To use the disk to hold files, the OS still needs to record its own data structures on the disk. The first step is to **partition** the disk into one or more groups of cylinders. The OS can treat each partition as though it were a separate disk (for example, one partition for the operating system and another for user files). The partition information is stored in a partition table at a fixed place on the disk.

7.4.3 Logical Formatting

The second step is **logical formatting**, or creation of a file system. In this step the OS stores the initial file-system data structures onto the disk. These structures may include maps of free and allocated space (a FAT or inodes) and an initial empty directory. To increase efficiency, most file systems group blocks together into larger chunks called **clusters**. Disk I/O is done in blocks, but file-system I/O is done in clusters.

Some programs, such as databases, prefer to use a partition as a large sequential array of logical blocks without any file-system structure. This array is called the **raw disk** and I/O to it is called **raw I/O**; it bypasses file-system services such as buffering, caching, file locking and space allocation.

Basis	Low-level formatting	Partitioning	Logical formatting
Also called	Physical formatting	Disk slicing	High-level formatting
Done by	Manufacturer (usually)	OS tool (fdisk, Disk Management)	OS (mkfs, format)
What it does	Creates sectors with header, data area and trailer	Divides disk into groups of cylinders	Creates file-system structures (free-space map, directory)
Result	Disk ready to be addressed by controller	Independent logical disks	Disk ready to store files

7.5 Boot Block

For a computer to start running, for instance when it is powered up or rebooted, it must have an initial program to run. This **bootstrap program** is simple. It initialises all aspects of the system, from CPU registers to device controllers and the contents of main memory, and then starts the operating system. To do its job, the bootstrap program finds the OS kernel on disk, loads that kernel into memory and jumps to an initial address to begin the operating-system execution.

Most computers store a tiny bootstrap loader in **ROM** (firmware such as BIOS or UEFI), because ROM needs no initialisation and is at a fixed location from which the processor can start executing on power-up. It is also read-only, so it cannot be infected by a computer virus. The problem is that changing this bootstrap code requires changing the ROM chips. For this reason most systems keep only a tiny loader in ROM, whose only job is to bring in a full bootstrap program from disk.

The full bootstrap program is stored in the **boot blocks** at a fixed location on the disk (usually the first sector, called the **boot sector** or **Master Boot Record, MBR**). 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 (no device drivers are loaded at this point) and then starts executing that code. The full bootstrap program is more sophisticated than the bootstrap loader in ROM: it is able to load the entire operating system from a non-fixed location on disk and start it running.

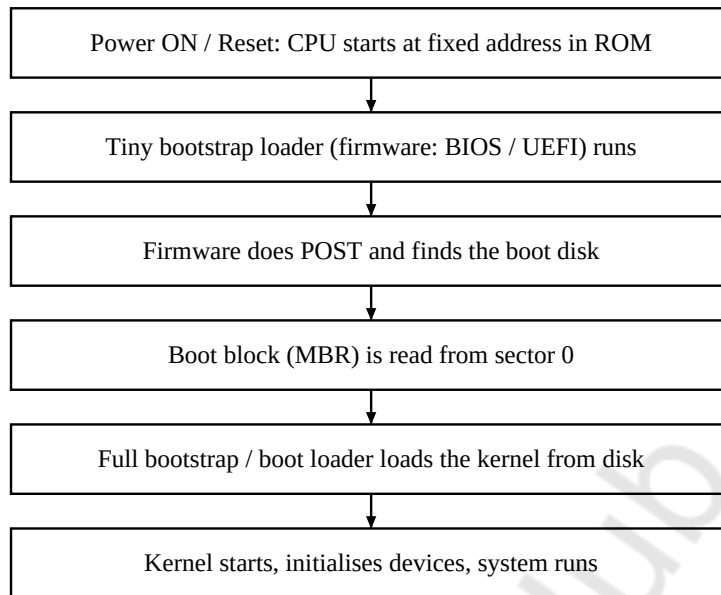


Fig. 7.7: Boot sequence of a computer

Example: Booting in Windows

Windows allows a hard disk to be divided into partitions, and one partition, identified as the **boot partition**, contains the operating system and device drivers. The system places its boot code in the first sector on the hard disk (the MBR). Booting begins by running code that is resident in the system's ROM. This code directs the system to read the boot code from the MBR. In addition to containing boot code, the MBR contains a table listing the partitions for the hard disk and a flag indicating which partition the system is to be booted from. Once the system identifies the boot partition, it reads the first sector from that partition (the **boot sector**), which directs it to the kernel, and then continues with the remainder of the boot process, loading the various subsystems and system services. Modern systems use **UEFI** with a **GPT** (GUID Partition Table), which supports very large disks.

Basis	Bootstrap loader (ROM)	Full bootstrap (boot block / MBR)
Location	ROM / firmware	First sector of boot disk
Size	Very small	Larger (one sector or a few blocks)
Can it be changed easily?	No (needs new ROM or firmware update)	Yes (rewritten on disk)
Job	Load the boot block into memory	Load the full operating system kernel

7.6 Bad Blocks

Because disks have moving parts and small tolerances (the heads float very close to the surface), they are prone to failure. Sometimes the failure is complete and the disk must be replaced; more often, one or more sectors become defective. Most disks even come from the factory with **bad blocks**. Depending on the disk and controller in use, these blocks are handled in a variety of ways.

7.6.1 Handling by Simple Disks (Manual)

On simple disks, such as some disks with IDE controllers, bad blocks are handled **manually**. For instance, the MS-DOS *format* command performs logical formatting and, as a part of the process, scans the disk to find bad blocks. If format finds a bad block, it writes a special value into the corresponding FAT entry to tell the allocation routines not to use that block. If blocks go bad during normal operation, a special program (such as *chkdsk*) must be run manually to search for the bad blocks and to lock them away. Data that resided on the bad blocks usually are lost.

7.6.2 Handling by Intelligent Controllers

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

Typical Bad-Sector Transaction

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

Such a redirection by the controller could invalidate any optimisation by the OS's disk-scheduling algorithm (because the spare may be far away). For this reason, most disks are formatted to provide a few spare sectors in each cylinder and a spare cylinder as well. When a bad block is remapped, the controller uses a spare sector from the **same cylinder**, if possible.

7.6.3 Sector Slipping

As an alternative to sector sparing, some controllers can be instructed to replace a bad block by **sector slipping**. Suppose that logical block 17 becomes defective and the first available spare sector 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 sector 17 can be mapped to it.

Basis	Sector sparing (forwarding)	Sector slipping
Method	Bad sector is replaced by a spare sector elsewhere	All sectors after the bad one are shifted by one position into the spare
Effect on sequential access	Head may have to jump to the spare sector	Sequential order is kept; no jump
Overhead	Small (one remap entry)	Higher (many sectors copied)
Spare used	Spare in same cylinder if possible	Next free spare sector

Soft and Hard Errors

A **soft error** is a recoverable error corrected by the ECC, and the controller may track it and warn of an approaching failure. A **hard error** is unrecoverable: the data in the sector is lost and the sector is replaced by sparing or slipping; the lost data must be restored from a backup. Note that replacing a bad block is not always completely automatic, since the data may be lost.

7.7 Disk Reliability

Reliability is the ability of a disk system to keep data safe and available even when parts fail. Since a disk is a mechanical device, failure is inevitable over time, and data loss can be very costly. The OS and storage hardware therefore use several techniques.

7.7.1 Causes of Disk Failure

- Head crash (the head touches the platter), motor or bearing wear, and power failures.
- Media defects, ageing, temperature, dust and mechanical shock.
- Controller or cable faults, and software errors that corrupt data.
- For SSDs: wear-out of flash cells after many write cycles.

7.7.2 Techniques for Improving Reliability

Technique	Description
Error-correcting code (ECC)	Extra check bits stored in each sector detect and correct small errors.
S.M.A.R.T.	Self-Monitoring, Analysis and Reporting Technology: the drive records error counts and temperature and warns of failure.
Bad-block management	Sparing and slipping hide defective sectors (see 7.6).
Backup	Periodic copies of data to another device, so lost data can be restored.
Stable storage	Data is written to two or more independent disks in a careful order so that a failure during a write never leaves both copies damaged.
RAID	Redundant Array of Independent Disks: uses many disks together for reliability and speed.

7.7.3 Redundant Array of Independent Disks (RAID)

The chance that some disk out of a set of N disks will fail is much greater than the chance that one specific disk will fail, so a large set of disks has a low mean time to failure. The solution is **redundancy**: store extra information that is not normally needed but can be used to rebuild lost data when a disk fails. RAID also improves performance by using **data striping**, that is, splitting the bits or blocks of a file across several disks so that they can be read in parallel.

- **Mirroring**: every disk is duplicated; every write goes to both disks. A read can use either, so reads are faster. Data is lost only if the second disk fails before the first is repaired. If the mean time to failure of a single disk is MTTF and the mean time to repair is MTTR, the mean time to data loss of a mirrored pair is about MTTF squared divided by (2 x MTTR).
- **Striping**: blocks are spread over several disks for higher transfer rate and throughput, but striping alone gives no redundancy.
- **Parity**: an extra disk (or distributed blocks) stores the XOR of the data blocks, which allows one lost disk to be rebuilt using far less space than mirroring.

RAID level	Technique	Minimum disks	Fault tolerance	Remarks
RAID 0	Block striping, no redundancy	2	None	Fastest, no protection
RAID 1	Mirroring	2	One disk per pair	50% usable space, fast reads
RAID 2	Bit-level striping with Hamming code	Many	One disk	Rarely used
RAID 3	Byte striping, dedicated parity disk	3	One disk	Parity disk is a bottleneck
RAID 4	Block striping, dedicated parity disk	3	One disk	Parity disk bottleneck on writes
RAID 5	Block striping, parity distributed over all disks	3	One disk	Popular; good balance of cost and speed
RAID 6	Block striping, two parity blocks (P+Q)	4	Two disks	Survives two failures
RAID 0+1 / 1+0	Striping combined with mirroring	4	One or more disks	High performance and reliability, higher cost

Remember: RAID is not a substitute for backup. It protects against disk failure, but not against accidental deletion, software bugs, viruses or a fire that destroys the whole array.

7.8 Chapter Summary and Quick Revision

- A disk has platters, tracks, sectors, cylinders, heads and an arm; the OS sees it as a linear array of logical blocks.
- Access time = seek time + rotational latency + transfer time; seek time dominates.
- **FCFS:** in order of arrival; fair but slow. **SSTF:** nearest request first; fast but may starve requests.
- **SCAN:** elevator sweep to both ends. **C-SCAN:** one-directional sweep with quick return; uniform wait. LOOK and C-LOOK stop at the last request.
- Disk formatting: low-level (sectors with header, data, ECC trailer), partitioning, logical formatting (file system).
- Boot: tiny loader in ROM reads the boot block (MBR) from disk, which loads the full OS kernel.
- Bad blocks: manual handling in simple disks; sector sparing and sector slipping in intelligent controllers.
- Reliability: ECC, S.M.A.R.T., backup, stable storage and RAID levels (0, 1, 5, 6, 10).

7.9 Multiple Choice Questions (15)

Choose the correct option for each question. The answer key is given in Section 7.10.

Q1. The main aim of disk scheduling is to reduce:

- (a) Rotational speed
- (b) Seek time
- (c) Transfer rate
- (d) Disk capacity

Q2. The set of tracks that are at the same arm position on all platters is called a:

- (a) Sector
- (b) Block
- (c) Cylinder
- (d) Cluster

Q3. The average rotational latency of a 7200 RPM disk is about:

- (a) 2.08 ms
- (b) 4.17 ms
- (c) 8.33 ms
- (d) 16.67 ms

Q4. Which disk-scheduling algorithm may cause starvation?

- (a) FCFS
- (b) SSTF
- (c) SCAN
- (d) C-SCAN

Q5. Which algorithm is also known as the elevator algorithm?

- (a) FCFS
- (b) SSTF
- (c) SCAN
- (d) LIFO

Q6. The total head movement of FCFS for queue 98, 183, 37, 122, 14, 124, 65, 67 with head at 53 is:

- (a) 236
- (b) 640
- (c) 382
- (d) 208

Q7. In C-SCAN, after reaching the last cylinder the head:

- (a) Reverses and serves requests
- (b) Stops
- (c) Returns to the first cylinder without serving requests
- (d) Serves the nearest request

Q8. The disk-scheduling algorithm that always selects the request closest to the current head position is:

- (a) FCFS
- (b) SCAN
- (c) SSTF
- (d) C-LOOK

Q9. Low-level formatting of a disk is also called:

- (a) Logical formatting
- (b) Physical formatting
- (c) Partitioning
- (d) Quick format

Q10. The trailer of a disk sector typically contains:

- (a) File name
- (b) Error-correcting code (ECC)
- (c) Boot program
- (d) Directory entries

Q11. The full bootstrap program is stored in the:

- (a) CPU cache
- (b) Boot block (MBR) of the disk
- (c) Main memory only
- (d) Printer buffer

Q12. Why is only a tiny bootstrap loader kept in ROM?

- (a) ROM is faster than RAM
- (b) Changing ROM code is difficult, so a full bootstrap is kept on disk
- (c) ROM cannot store programs
- (d) ROM is volatile

Q13. Replacing a bad sector with a spare sector by the controller is called:

- (a) Sector slipping
- (b) Sector sparing
- (c) Thrashing
- (d) Striping

Q14. Which RAID level uses mirroring?

- (a) RAID 0
- (b) RAID 1
- (c) RAID 5
- (d) RAID 6

Q15. Which RAID level uses block striping with distributed parity?

- (a) RAID 1
- (b) RAID 3
- (c) RAID 5
- (d) RAID 0

7.10 Answer Key

Q. No.	Answer	Short Reason
1	(b)	Seek time is the largest delay, so scheduling tries to reduce head movement.
2	(c)	That set of tracks is a cylinder.
3	(b)	One rotation = 8.33 ms; average latency is half of it.
4	(b)	SSTF can keep postponing far-away requests.
5	(c)	SCAN sweeps like an elevator.
6	(b)	$45+85+146+85+108+110+59+2 = 640$.
7	(c)	C-SCAN returns quickly to the start without service.
8	(c)	This is the definition of SSTF.
9	(b)	It creates the physical sector structure.
10	(b)	The trailer holds the ECC.
11	(b)	It is kept in the boot block at a fixed disk location.
12	(b)	ROM is hard to modify, so a small loader brings in the full bootstrap from disk.
13	(b)	Also known as forwarding.
14	(b)	RAID 1 duplicates every disk.
15	(c)	RAID 5 spreads parity over all disks.

7.11 Broad Questions (10)

1. Explain the physical structure of a magnetic disk with a neat diagram. Define platter, track, sector and cylinder.
2. What is disk access time? Explain seek time, rotational latency and transfer time. Calculate the average rotational latency of a 7200 RPM disk.
3. What is disk scheduling? Explain its goals and criteria for evaluating a scheduling algorithm.
4. Explain FCFS and SSTF disk scheduling with an example and head-movement diagram. State their advantages and disadvantages.
5. Explain SCAN and C-SCAN disk scheduling with an example. Why is C-SCAN said to give a more uniform waiting time?
6. A disk has 200 cylinders (0 to 199). The request queue is 98, 183, 37, 122, 14, 124, 65, 67 and the head is at 53. Find the total head movement for FCFS, SSTF, SCAN and C-SCAN and compare them.
7. Explain disk formatting. Describe low-level formatting, partitioning and logical formatting.
8. What is a boot block? Explain the boot process of a computer and the role of the bootstrap loader in ROM and the MBR.
9. What are bad blocks? Explain how they are handled using manual methods, sector sparing and sector slipping.
10. Discuss disk reliability. Explain RAID and describe the different RAID levels with their advantages and disadvantages.