MODULE 5

File, I/O, Device Management & Security

Comprehensive Study Material

Course Code: DCSE250042 | B.Sc. Computer Science (Semester 3)

Table of Contents

Section 5.1: File Management Fundamentals

5.1.1 What is a File?

A file is a named collection of related information recorded on secondary storage. It is the logical unit of storage used by operating systems.

A file represents two things:

- Data (bytes): Text, Programs, Images, Audio, Video
- Metadata: Name, Size, Permissions, Timestamps, Owner

5.1.2 File Attributes

Attribute	Description	Example
Name	Human-readable identifier	report.docx
Identifier	Unique numeric tag (inode number in UNIX)	12345
Type	Indicates file format	.txt, .exe, .jpg
Location	Pointer to device and blocks	Sector 1024-2048
Size	Current file size in bytes	2,456,789 bytes
Protection	Access permissions	rw-r--r--
Time/Date	Creation, last access, last modification	2024-01-15 10:30:00
Owner	User who owns the file	student
Links	Number of hard links	1

5.1.3 File Operations

Operation	System Call (UNIX)	Description
Create	creat()	Create new file
Open	open()	Open existing file
Read	read()	Read data from file
Write	write()	Write data to file
Seek	lseek()	Move file pointer
Close	close()	Close file
Delete	unlink()	Delete file
Truncate	truncate()	Shorten file
Rename	rename()	Change file name

5.1.4 File Types

Regular Files:

- Text Files: .txt, .c
- Binary Files: .exe, .obj
- Executable Programs

Directory Files:

- Contain list of other files and directories (e.g., /home, /usr, C:\Windows)

Special Files:

- Block Device Files: /dev/sda
- Character Device Files: /dev/tty
- Sockets/Pipes: Inter-process communication

5.1.5 Open File Table Information

Field	Description
File pointer	Current read/write position (per process)
File open count	Number of processes using this file
Disk location	Where file is stored on disk
Access permissions	How file was opened (read/write)

Section 5.2: File Access Methods

5.2.1 Sequential Access

Simplest method — read/write data in order, from beginning to end (like a tape or audio cassette).

Operations:

- read_next() — read next block, advance pointer
- write_next() — write at end, advance pointer
- reset() — move pointer to beginning

5.2.2 Direct (Random) Access

Access any block directly — like a CD or hard disk. No pointer is maintained.

Operations:

- read(block_number) — read specific block
- write(block_number) — write to specific block

5.2.3 Indexed Access

Uses an index block containing pointers to actual data blocks.

- Advantage: Fast random access
- Disadvantage: Overhead of index block

5.2.4 Access Method Comparison

Feature	Sequential	Direct	Indexed
Access speed (start)	Fast	Fast	Fast
Access speed (middle)	Slow (traverse)	Fast (calculate)	Fast (index lookup)
Access speed (end)	Very slow	Fast	Fast
Storage overhead	None	None	Index block(s)
Best for	Batch processing, logs	Databases, direct lookup	Large files, DBMS
Example	Magnetic tape	Hard disk	Database index

5.2.5 Other Access Methods

Content-Based Access: Access by searching for specific content (e.g., database query SELECT * FROM users WHERE name='John').

Key-Based Access: Access via hashed key value (e.g., hash table, dictionary).

```
Key 'John' → Hash function → Hash value 42 → Block 42 → Record
```

Section 5.3: Directory Structures

5.3.1 What is a Directory?

A directory is a special file that contains information about other files and directories.

Directory entry fields: Filename, Inode #, Type, Size, Permissions, Owner, Group, Modified date.

5.3.2 Single-Level Directory

All files in one directory — simplest structure.

> Problems: Name conflicts (two users cannot have same filename), difficult to organize many files, slow searching.

5.3.3 Two-Level Directory

Separate directory per user — each user has their own directory.

Advantages:

- No name conflicts between users
- User isolation
- Simple structure

Disadvantages:

- Cannot group files within user's space
- User's files still in single directory

5.3.4 Tree-Structured Directory

Hierarchical structure — most common in modern OS (UNIX, Windows).

```
Absolute path:  /home/alice/Documents/report.txt Relative path:  Documents/report.txt
(if current dir is /home/alice)
```

5.3.5 Acyclic-Graph Directory

Allows sharing — directories/files can have multiple parents (links).

Types of Links:

- Hard link: Direct pointer to inode
- Soft/Symbolic link: Pathname pointer

5.3.6 General Graph Directory

Allows cycles — can cause infinite traversal problems.

Solutions:

- Garbage collection to detect cycles
- Restrict to acyclic graphs only

5.3.7 Directory Operations

Operation	Description	Example
Create	Create new directory	mkdir newdir
Delete	Remove empty directory	rmdir olddir
Open	Open directory to read contents	opendir('/home')
Read	Read directory entries	readdir()
Rename	Change directory name	mv old new
Link	Create link to file/directory	ln file1 file2
Unlink	Remove link (delete file when count=0)	rm file

Section 5.4: File System Implementation

5.4.1 File System Architecture

The file system is organized in layers:

- Logical File System — file operations, directory management, permission checking
- File Organization Module — file allocation, block mapping, access methods
- Basic File System — buffer cache management, I/O on physical blocks
- I/O Control — device drivers, interrupt handlers
- Hardware

5.4.2 File Allocation Methods

Method 1: Contiguous Allocation

File occupies consecutive blocks on disk.

```
Address = Start + Offset Block 10 = Start + 0 Block 12 = Start + 2
```

Pros	Cons
Simple, fast access	External fragmentation
Few seeks (sequential)	File size cannot grow easily

Method 2: Linked Allocation

Each block contains a pointer to the next block.

```
Chain: 10 → 15 → 20 → 25 (end) Sequential access: O(n) seeks Random access: Very slow
```

Pros	Cons
No external fragmentation	Slow random access
Easy file growth	Extra space for pointers
	Reliability (broken pointer)

Method 3: Indexed Allocation

A single index block contains all block pointers. For large files, multi-level indexing is used.

```
UNIX inode scheme:   direct[0..11]      → Data blocks (small files)   single indirect
→ Index block → Data blocks   double indirect    → Index block → Index block → Data
blocks   triple indirect    → Index block → Index block → Index block → Data blocks
```

5.4.3 Free Space Management

Method	Description	Example
Bit Vector	One bit per block (1=free, 0=allocated). Scan for first 1 to find free block.	16-block bit vector: 1001110010110010
Linked List	Free blocks linked together. LIFO or FIFO allocation.	Head→Block5→Block12→Block20 →NULL
Grouping	Store addresses of free blocks in other	Block 100 contains [101, 105, 200,

Method	Description	Example
	free blocks.	300...]
Counting	Store (starting block, count) for contiguous free blocks.	(50, 30), (120, 80) — ranges of free blocks

5.4.4 Directory Implementation

Method	Description	Pros	Cons
Linear List	Array of filenames + pointers	Simple	Slow search O(n)
Hash Table	Hash of filename to pointer	Fast search O(1)	Collision handling, fixed size

Section 5.5: I/O Management

5.5.1 I/O Communication Methods

Method 1: Programmed I/O (Polling)

CPU continuously checks device status.

```
while (device_busy) {     check_status_again(); } transfer_data();
```

PROS: Simple, no special hardware CONS: Wastes CPU cycles (busy waiting)

Method 2: Interrupt-Driven I/O

Device interrupts CPU when ready.

- CPU issues I/O command
- CPU continues other work (not waiting)
- Device performs operation
- Device sends interrupt to CPU
- CPU saves state, executes interrupt handler, transfers data, resumes

PROS: CPU efficient (no waiting) CONS: Overhead of interrupt handling

Method 3: Direct Memory Access (DMA)

Device transfers data directly to memory without CPU involvement.

- CPU sets up DMA: memory address, byte count, device, direction
- CPU continues other work while DMA transfers data block by block
- DMA controller interrupts CPU when transfer is complete

DMA Controller Registers: Source address, Destination address, Count register, Control register

5.5.2 I/O Methods Comparison

Method	CPU Involvement	Data Transfer Path	Best For
Programmed I/O	High (busy waiting)	CPU → Device	Simple devices
Interrupt-driven	Medium (on interrupt)	CPU → Device	Keyboards, mice
DMA	Low (setup only)	Device ↔ Memory	Disks, network

5.5.3 I/O Buffering

Buffers are temporary storage areas in memory. Double buffering allows user processing to continue while the next buffer is being filled, improving throughput.

Section 5.6: Disk Scheduling Algorithms

5.6.1 Disk Structure

Component	Description
Platter	Circular disk where data is stored
Track	Concentric circle on platter
Sector	Smallest addressable unit (typically 512 bytes or 4KB)
Cylinder	Same track on all platters
Seek time	Time to move head to correct track
Rotational latency	Time for correct sector to rotate under head

5.6.2 Disk Access Time Components

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

```
5400 RPM: Rotation = 60/5400 = 11.1 ms  →  Avg Latency = 5.55 ms 7200 RPM: Rotation = 60/7200 = 8.33 ms  →  Avg Latency = 4.16 ms SSD:      Seek ≈ 0.1 ms, Latency ≈ 0 (no moving parts)
```

5.6.3 Disk Scheduling Algorithms

Algorithm 1: FCFS (First-Come, First-Served)

Handle requests in arrival order.

Example: Queue: 98,183,37,122,14,124,65,67 | Head at 53 Sequence: 53 → 98 → 183 → 37 → 122 → 14 → 124 → 65 → 67 Total: 640 cylinders

Algorithm 2: SSTF (Shortest Seek Time First)

Service the request closest to current head position.

Example: Same queue | Head at 53 Sequence: 53 → 65 → 67 → 37 → 14 → 98 → 122 → 124 → 183 Total: 236 cylinders Problem: STARVATION — far requests may never be serviced!

Algorithm 3: SCAN (Elevator Algorithm)

Move back and forth across disk, servicing requests in path.

Example: Head at 53, moving toward 199 Sequence: 53 → 65 → 67 → 98 → 122 → 124 → 183 → (199) → 37 → 14 Total: 331 cylinders

Algorithm 4: C-SCAN (Circular SCAN)

One-way only — jumps back to beginning after reaching end.

Example: Head at 53, moving toward 199 Sequence: 53 → 65 → 67 → 98 → 122 → 124 → 183 → 199 → 0 → 14 → 37 Total: 382 cylinders Advantage: More uniform waiting time than SCAN

Algorithm 5: LOOK

Like SCAN but only goes as far as the last request in each direction (not to disk end).

> Example: Head at 53, direction → Sequence: 53→65→67→98→122→124→183→37→14 Total: 299 cylinders

Algorithm 6: C-LOOK

Like C-SCAN but jumps back to first waiting request, not disk start.

> Example: Head at 53, direction → Sequence: 53→65→67→98→122→124→183→14→37 Total: 322 cylinders

5.6.4 Algorithm Comparison Summary

Algorithm	Total Movement	Pros	Cons
FCFS	640 cylinders	Fair, simple	Poor performance
SSTF	236 cylinders	Good performance	Starvation possible
SCAN	331 cylinders	No starvation	Long wait at ends
C-SCAN	382 cylinders	Uniform waiting	Extra movement
LOOK	299 cylinders	Efficient (no extra travel)	More complex
C-LOOK	322 cylinders	Efficient + uniform	More complex

5.6.5 Selecting a Disk Scheduling Algorithm

Scenario	Recommended Algorithm
Light load	FCFS or SSTF
Heavy load	SCAN or C-SCAN
Real-time systems	Deadline-based or C-SCAN
Database servers	LOOK or C-LOOK
General purpose OS	LOOK or C-LOOK

Section 5.7: Device Management

5.7.1 Device Types

Classification	Type	Examples	Characteristics
By data rate	Low speed	Keyboard, mouse	Bytes/second
	Medium speed	Printer, scanner	KB/second
	High speed	Disk, network	MB/second
By direction	Input only	Keyboard, mouse	Read-only
	Output only	Printer, monitor	Write-only
	Input/Output	Disk, network	Read/Write
By access	Block device	Disk, tape	Random access, block size
	Character device	Keyboard, printer	Stream of characters

5.7.2 Device Controller Registers

Register	Purpose
Status Register	Indicates ready/busy/error state
Control Register	Start/stop/reset commands
Data Register	Data to/from device

5.7.3 Device Management Functions

Function	Description	Example
Device detection	Identify connected devices	Plug-and-play
Driver loading	Load appropriate driver	modprobe in Linux
Device allocation	Assign devices to processes	Printer spooling
Buffering	Temporary data storage	Print queue
Error handling	Detect and recover from errors	Disk retry
Power management	Manage device power states	Sleep, wake

5.7.4 Spooling

SPOOL = Simultaneous Peripheral Operation On-Line. Users don't wait for printer — their output is spooled to disk and printed when the device is available.

```
Process A ┐
Process B ─┤ → [Print Queue: Doc1, Doc2, Doc3] → Spooler Daemon → Printer
Process C ┘
```

Section 5.8: Protection and Security

5.8.1 Protection vs Security

Term	Definition	Focus
Protection	Mechanisms to control access to resources	Internal to system
Security	Overall defense against threats	External attacks

5.8.2 Authentication Methods

Method	Example	Strength
Something you know	Password, PIN	Weak
Something you have	Smart card, token	Medium
Something you are	Fingerprint, retina	Strong
Something you do	Signature, typing pattern	Medium
Somewhere you are	GPS location	Location-dependent

> Multi-Factor Authentication (MFA): Combine two or more methods. Example: Online Banking = Password + SMS code. Much more secure!

5.8.3 Access Control Models

Model 1: Discretionary Access Control (DAC)

Owner decides permissions.

```
UNIX permissions example: $ ls -l file.txt -rw-r--r-- 1 alice staff 1024 Jan 15 10:30
file.txt  Permission bits (octal):   4 = read (r)   2 = write (w)   1 = execute (x)
chmod 755 file.txt  →  rwxr-xr-x chmod 644 file.txt  →  rw-r--r-- chmod 777 file.txt
→  rwxrwxrwx  (insecure!)
```

Model 2: Mandatory Access Control (MAC)

System-wide policy enforced by OS.

```
Military Security Levels:   Top Secret (TS) > Secret (S) > Confidential (C) >
Unclassified (U)  Rules:   Read:  Subject level >= Object level  (no read up)   Write:
Subject level <= Object level  (no write down)  User with Secret clearance can:
READ:  Unclassified, Confidential, Secret  (NOT Top Secret)   WRITE: Unclassified,
Confidential         (NOT Secret or Top Secret)
```

Model 3: Role-Based Access Control (RBAC)

Access based on user role. User → Role(s) → Permissions.

Role	Permissions
Admin	All system resources
Manager	Sales reports, HR data

Role	Permissions
Employee	Personal records only
Guest	Public information only

5.8.4 Access Control Matrix

```
           F1    F2    F3    F4    D1 User A  [  R   | RW   |      |      |  X  ] User
B  [      |  R   |  RW  |      |      ] User C  [  RW  |      |  R   |  RW  |  R  ]
Process [      |      |      |  R   |      ] Admin   [  RW  |  RW  |  RW  |  RW  |  RW
]  R = Read, W = Write, X = Execute
```

5.8.5 Security Threats and Attacks

Threat	Description	Example
Virus	Self-replicating code attached to programs	File infector
Worm	Self-replicating network spread	Morris worm
Trojan	Malicious code disguised as legitimate	Fake antivirus
Ransomware	Encrypts files, demands payment	WannaCry
DoS/DDoS	Overwhelms system with requests	SYN flood
Phishing	Trick user into revealing info	Fake bank email
Man-in-Middle	Intercept communications	Evil twin Wi-Fi
SQL Injection	Inject malicious SQL queries	' OR '1'='1

5.8.6 Security Measures

Measure	Purpose	Implementation
Encryption	Confidentiality	AES, RSA
Firewall	Network filtering	iptables, pfSense
IDS/IPS	Threat detection	Snort, Suricata
Antivirus	Malware detection	ClamAV, commercial
Audit logs	Tracking	syslog, Windows Event Log
Backup	Recovery	Incremental, full

5.8.7 UNIX/Linux Security Commands

Command	Purpose
chmod	Change file permissions

Command	Purpose
chown	Change file owner
chgrp	Change group
passwd	Change password
umask	Set default permissions
sudo	Execute as root
ssh	Secure remote access
iptables	Firewall rules

Section 5.9: Numerical Problems with Solutions

Problem 1: Disk Scheduling (FCFS)

Question: Disk request queue: 55, 58, 39, 18, 90, 160, 150, 38, 184. Initial head at 50. Calculate total seek movement using FCFS.

Solution:

```
Sequence: 50→55→58→39→18→90→160→150→38→184  |50-55|  =   5 |55-58|  =   3 |58-39|  =
19 |39-18|  =  21 |18-90|  =  72 |90-160| =  70 |160-150|=  10 |150-38| = 112 |38-184|
= 146  Total = 5+3+19+21+72+70+10+112+146 = 458 cylinders
```

Answer: 458 cylinders

Problem 2: Disk Scheduling (SSTF)

Question: Same queue as above. Initial head at 50. Calculate SSTF.

Solution:

```
At each step, service the closest request: 50→55 (d=5)→58 (d=3)→39 (d=19)→38 (d=1)→18
(d=20)→90 (d=72)→150 (d=60)→160 (d=10)→184 (d=24)  Total = 5+3+19+1+20+72+60+10+24 =
214 cylinders
```

Answer: 214 cylinders

Problem 3: Disk Scheduling (SCAN)

Question: Same queue. Head at 50, moving toward 199. Calculate SCAN.

Solution:

```
Service on way to 199: 55,58,90,150,160,184,→199 Then reverse: 39,38,18  Sequence:
50→55→58→90→150→160→184→199→39→38→18  |50-55|  =  5 |55-58|  =  3 |58-90|  = 32 |90-
150| = 60 |150-160|= 10 |160-184|= 24 |184-199|= 15 |199-39| =160 |39-38|  =  1 |38-
18|  = 20  Total = 5+3+32+60+10+24+15+160+1+20 = 330 cylinders
```

Answer: 330 cylinders

Problem 4: Disk Scheduling (C-SCAN)

Question: Same queue. Head at 50, moving toward 199. Calculate C-SCAN.

Solution:

```
Service on way to 199: 55,58,90,150,160,184,→199 Jump to 0, then service: 18,38,39
Sequence: 50→55→58→90→150→160→184→199→0→18→38→39  |50-55|  =  5 |55-58|  =  3 |58-90|
= 32 |90-150| = 60 |150-160|= 10 |160-184|= 24 |184-199|= 15 |199-0|  =199 |0-18|   =
18 |18-38|  = 20 |38-39|  =  1  Total = 5+3+32+60+10+24+15+199+18+20+1 = 387 cylinders
```

Answer: 387 cylinders

Problem 5: Disk Access Time Calculation

Question: Disk: seek time 5ms, speed 7200 RPM, transfer rate 50MB/s. File size 100KB in 5 consecutive blocks. Calculate total access time.

Solution:

```
Seek time = 5 ms Rotational latency = 60 / (2 × 7200) = 4.17 ms Transfer time = 0.1 MB
/ 50 MB/s = 2 ms  Total = Seek + Latency + Transfer       = 5 + 4.17 + 2 = 11.17 ms
```

Answer: 11.17 ms

Problem 6: File System — Contiguous Allocation

Question: Block size 4KB. File needs 200KB. Free holes: blocks 50-80 (31 blocks), 120-200 (81 blocks), 300-400 (101 blocks). Best fit? Internal fragmentation?

Solution:

```
Blocks needed = 200 / 4 = 50 blocks  Hole 1: 31 blocks → too small! Hole 2: 81 blocks
→ fits (best fit, smallest that fits) Hole 3: 101 blocks → fits but larger  Best fit =
Hole 2 (81 blocks) Allocated = 81 × 4 = 324 KB Fragmentation = 324 - 200 = 124 KB
wasted
```

Answer: Best fit = Hole 2; Internal fragmentation = 124 KB

Problem 7: File System — Indexed Allocation (UNIX inode)

Question: Block size 4KB, 12 direct pointers, single indirect (1024 pointers), double indirect, triple indirect. Calculate maximum file size.

Solution:

```
Pointers per block = 4096 / 4 = 1024  Direct:          12 × 4KB           = 48 KB
Single indirect: 1024 × 4KB         = 4 MB Double indirect: 1024² × 4KB        = 4 GB
Triple indirect: 1024³ × 4KB        = 4 TB  Total ≈ 4 TB + 4 GB + 4 MB + 48 KB ≈ 4 TB
```

Answer: Approximately 4 TB

Problem 8: Sequential vs Direct File Access

Question: File with 1000 blocks. Seek 10ms, latency 5ms, transfer 1ms/block. Access block 500 sequentially vs directly.

Solution:

```
Sequential access to block 500:   = Seek + Latency + (500 × Transfer)   = 10 + 5 + 500
= 515 ms  Direct access to block 500:   = Seek + Latency + Transfer   = 10 + 5 + 1 =
16 ms
```

Answer: Direct access is ~32× faster (16 ms vs 515 ms)

Section 5.10: Quick Revision Notes

Key Formulas

```
Disk Access Time     = Seek Time + Rotational Latency + Transfer Time Rotational
Latency   = 60 / (2 × RPM)  seconds Transfer Time        = Block Size / Transfer Rate
Total Seek Movement  = Σ |current_position - next_position| Throughput           =
Number of I/O operations / Total time
```

Disk Scheduling — Quick Reference

Algorithm	Rule	Approach
FCFS	First come, first served	Service in arrival order
SSTF	Shortest seek time first	Closest request first
SCAN	Go to end, then reverse	Elevator algorithm
C-SCAN	Go to end, jump to start	Circular elevator
LOOK	Go to last request, reverse	Optimized SCAN
C-LOOK	Go to last request, jump	Optimized C-SCAN

File Allocation Methods — Quick Reference

Method	Pros	Cons
Contiguous	Fast, simple	Fragmentation
Linked	No fragmentation	Slow random access
Indexed	Fast random access	Index overhead

Directory Structures — Evolution

```
Single-level → Two-level → Tree → Acyclic Graph → General Graph (simplest)
(shared)        (cycles)
```

I/O Methods — Evolution

```
Programmed I/O (polling)  →  Interrupt-driven  →  DMA     (CPU busy)
(CPU efficient)     (no CPU for data)
```

UNIX Permission Bits

```
r = 4 (100)   w = 2 (010)   x = 1 (001)  Common permissions:   644  =  rw-r--r--
(files)   755  =  rwxr-xr-x   (directories/programs)   777  =  rwxrwxrwx
(dangerous!)
```

Key Points Summary

Topic	Key Takeaway
File	Named collection of data + metadata
Contiguous allocation	Fast but causes fragmentation
Linked allocation	No fragmentation but slow random access
Indexed allocation	Good balance, used in UNIX/Linux
FCFS	Simple but poor performance
SSTF	Good performance but starvation possible
SCAN	No starvation, elevator algorithm
C-SCAN	Uniform waiting time
DMA	Best I/O method for large transfers
DAC	Owner decides permissions
MAC	System enforces policy
RBAC	Access based on roles

Section 5.11: Practice Questions

Short Answer Questions (2-3 marks)

- 1. What is a file? List any five file attributes.
- 2. Differentiate between sequential and direct file access.
- 3. What is spooling? Why is it used for printers?
- 4. Explain the difference between hard links and soft links.
- 5. What is a directory? List different directory structures.
- 6. Define seek time, rotational latency, and transfer time.
- 7. What is DMA? How does it improve I/O efficiency?
- 8. Differentiate between DAC and MAC access control.
- 9. What is internal fragmentation in file systems?
- 10. Explain the purpose of a file descriptor table.

Medium Answer Questions (5-7 marks)

- 11. Compare and contrast contiguous, linked, and indexed file allocation with diagrams.
- 12. Explain SCAN and C-SCAN disk scheduling algorithms with examples.
- 13. Describe the UNIX inode structure and how it supports large files.
- 14. What are device drivers? Explain the layered architecture of I/O system.
- 15. Explain three methods of I/O communication (programmed, interrupt, DMA).

Long Answer Questions (10-15 marks)

- 16. Disk request queue: 45, 21, 67, 90, 4, 50, 89, 52, 61, 87. Initial head at 50, moving toward 0. Calculate total seek for FCFS, SSTF, SCAN, C-SCAN, LOOK, and C-LOOK.
- 17. Explain the complete file system architecture from user view to hardware. Include all layers and their functions.
- 18. Describe the role-based access control model. Compare it with DAC and MAC.
- 19. A disk has 8 platters, 1024 cylinders, 256 sectors per track, 512 bytes per sector. Calculate disk capacity. If seek=8ms, speed=5400 RPM, rate=40MB/s, find time to read a 256KB file.
- 20. Explain free space management techniques: bit vector, linked list, grouping, and counting. Give advantages and disadvantages of each.

Multiple Choice Questions

1. Which file allocation method suffers from external fragmentation?

- a) Linked
- b) Contiguous ✓
- c) Indexed
- d) All of the above

2. The time to move disk head to the correct track is called:

- a) Latency
- b) Seek time ✓
- c) Transfer time

- d) Access time

3. Which disk scheduling algorithm can cause starvation?

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

4. In UNIX, permission 755 represents:

- a) rwxr-xr-x ✓
- b) rw-r--r--
- c) rwxrwxrwx
- d) rwx------

5. Which I/O method requires CPU to continuously check device status?

- a) Interrupt-driven
- b) DMA
- c) Programmed I/O ✓
- d) None of the above

6. A file's inode contains:

- a) File name
- b) File data
- c) Metadata and block pointers ✓
- d) Directory entries

7. Which directory structure allows cycles?

- a) Tree
- b) Two-level
- c) Acyclic graph
- d) General graph ✓

8. C-SCAN differs from SCAN because:

- a) It jumps back to beginning
- b) It doesn't reverse direction
- c) Both a and b ✓
- d) It only goes one way

MCQ Answers: 1-b, 2-b, 3-c, 4-a, 5-c, 6-c, 7-d, 8-c

References:

- Silberschatz, Galvin & Gagne — Operating System Concepts (Chapters 11-14)
- Previous year question papers
- Disk scheduling simulators online