Operating Systems
Course Code: **71203002004**

File Allocation Strategies

by -
Minal Rajwar

File Allocation in OS

- When a hard disk is formatted, it is divided into small parts called blocks (or sectors).
- File allocation is the method used by the operating system to decide how files are stored in these blocks.
- Different allocation methods help in using disk space efficiently and make file access easier.

Types of File Allocation Methods in OS

- **Contiguous Allocation** – Stores a file in consecutive blocks.
- **Linked Allocation** – Each file block points to the next block.
- **Indexed Allocation** – Uses an index table to keep track of all file blocks.
- **File Allocation Table (FAT)** – A table that maintains the mapping of files to their blocks.

The goal of these methods is to ensure efficient use of disk space and fast file access.

1. Contiguous Allocation

- In this method, a file is stored in adjacent (continuous) disk blocks.
- To access the file, we only need:
  1. Starting block address
  2. Length of the file (number of blocks)

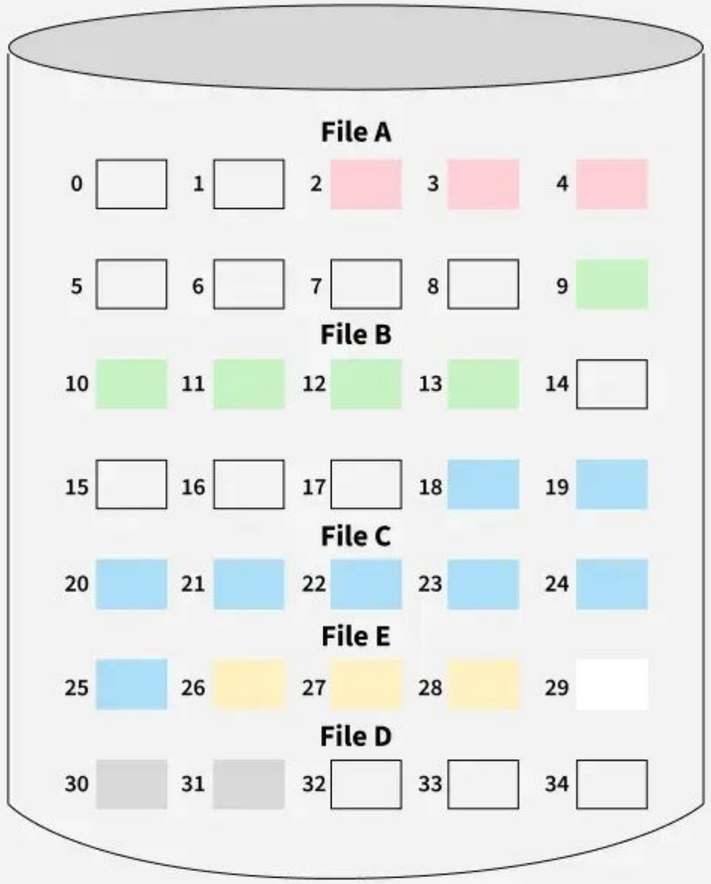
<u>Example:</u>

If a file starts at block x and needs n blocks, the file will occupy:

$x, x+1, x+2, \ldots, x+(n-1)$

All blocks are kept next to each other, making access fast and simple.

Continuous Allocation



File Name	Star block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

Contiguous Allocation: Advantages & Disadvantages

Advantages	**Disadvantages**
Easy to implement	File size must be fixed at creation
Very fast access (minimum seek time & head movement)	Difficult to increase file size later
Supports both sequential and direct access	Causes internal and external fragmentation
Efficient for small, fixed-size files	Not suitable for dynamic or growing files

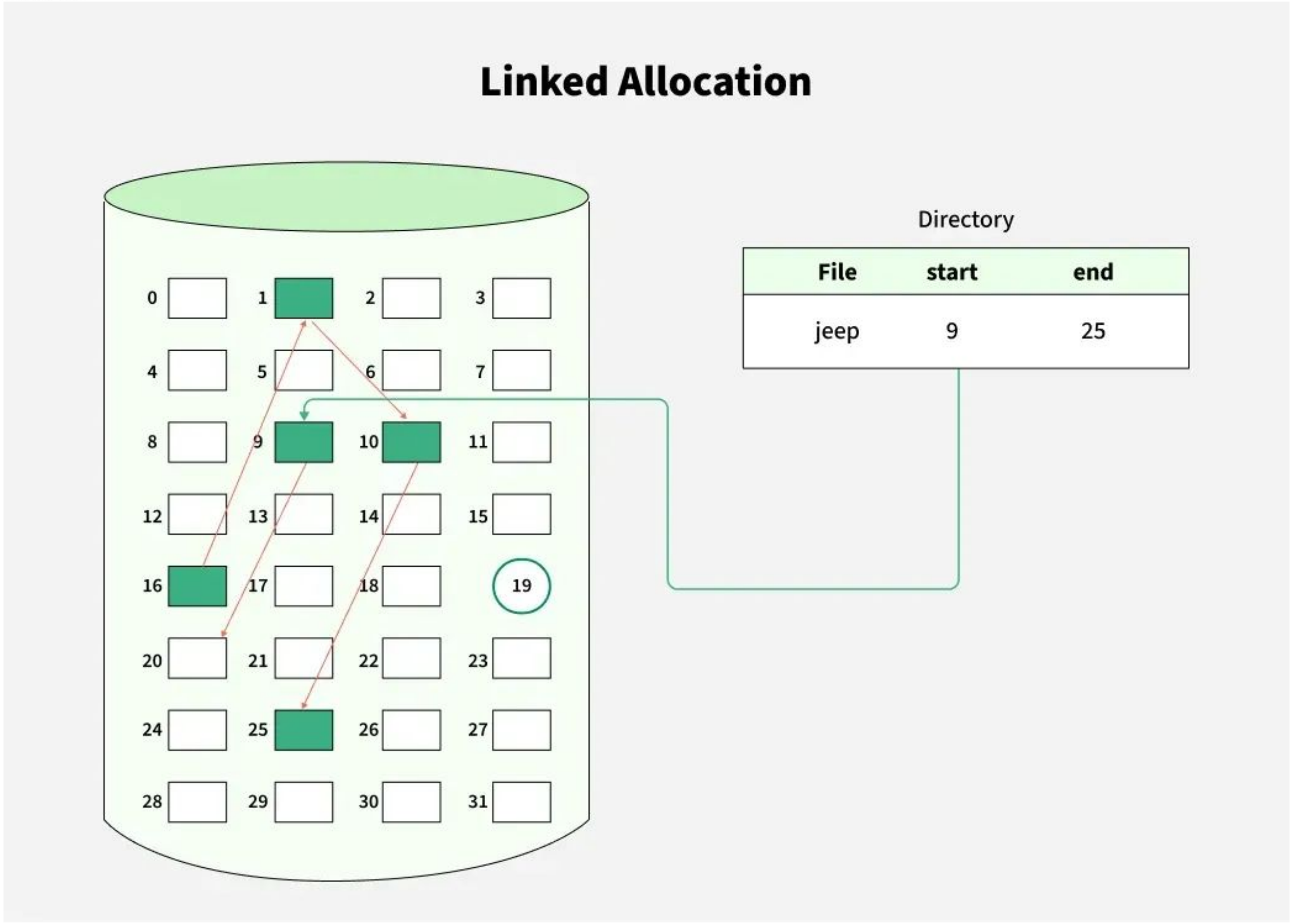
2. Linked Allocation

- A file is stored as a linked list of disk blocks, which can be scattered anywhere on the disk.
- Directory entry keeps:
  1. Pointer to the first block
  2. Pointer to the last block
- Each block has a pointer to the next block of the file.
- The last block contains a null pointer (-1) to show the end of the file.

This method solves the problem of contiguous allocation since blocks don't need to be adjacent.

<u>Example:</u>

If file jeep has blocks scattered randomly, each block stores the address of the next block, linking them together like a chain.



Linked Allocation

Directory

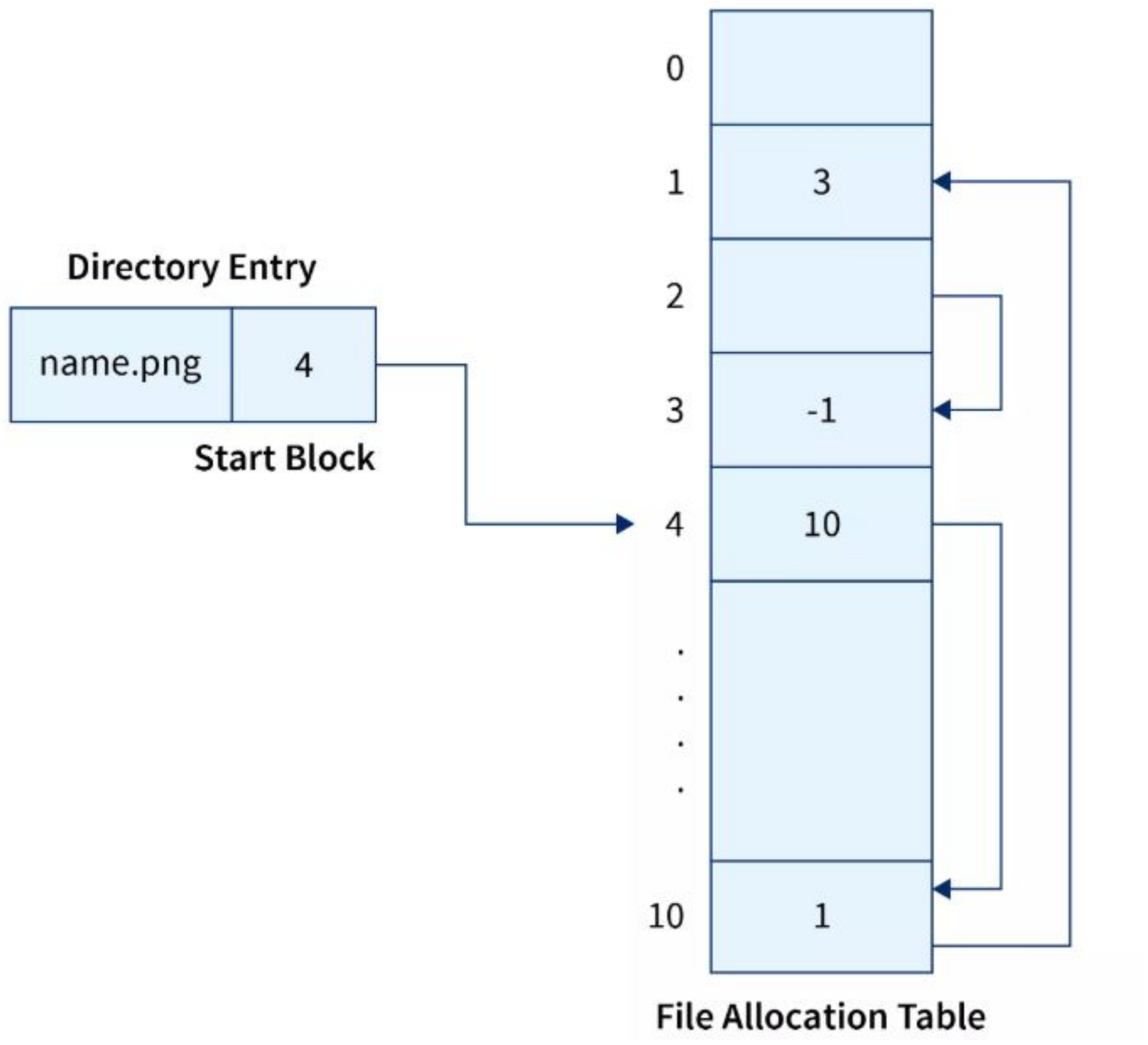
File	start	end
jeep	9	25

Linked Allocation: Advantages & Disadvantages

Advantages	**Disadvantages**
Very flexible – file size can grow easily	Slower access due to many seeks (blocks scattered)
No external fragmentation	Does not support direct/random access (only sequential)
Better disk space utilization	Extra space overhead for storing pointers

File Allocation Table (FAT)

- FAT improves on Linked Allocation, where random access is not possible.
- In FAT, all disk block links are stored in a table (File Allocation Table).
- To access a block, the OS simply looks it up in the table instead of traversing blocks sequentially.
- FAT is usually cached in memory → reduces head movements and improves performance.


Directory Entry
name.png
4
Start Block
0
1
3
2
3
-1
4
10
10
1
File Allocation Table

FAT (File Allocation Table): Advantages & Disadvantages

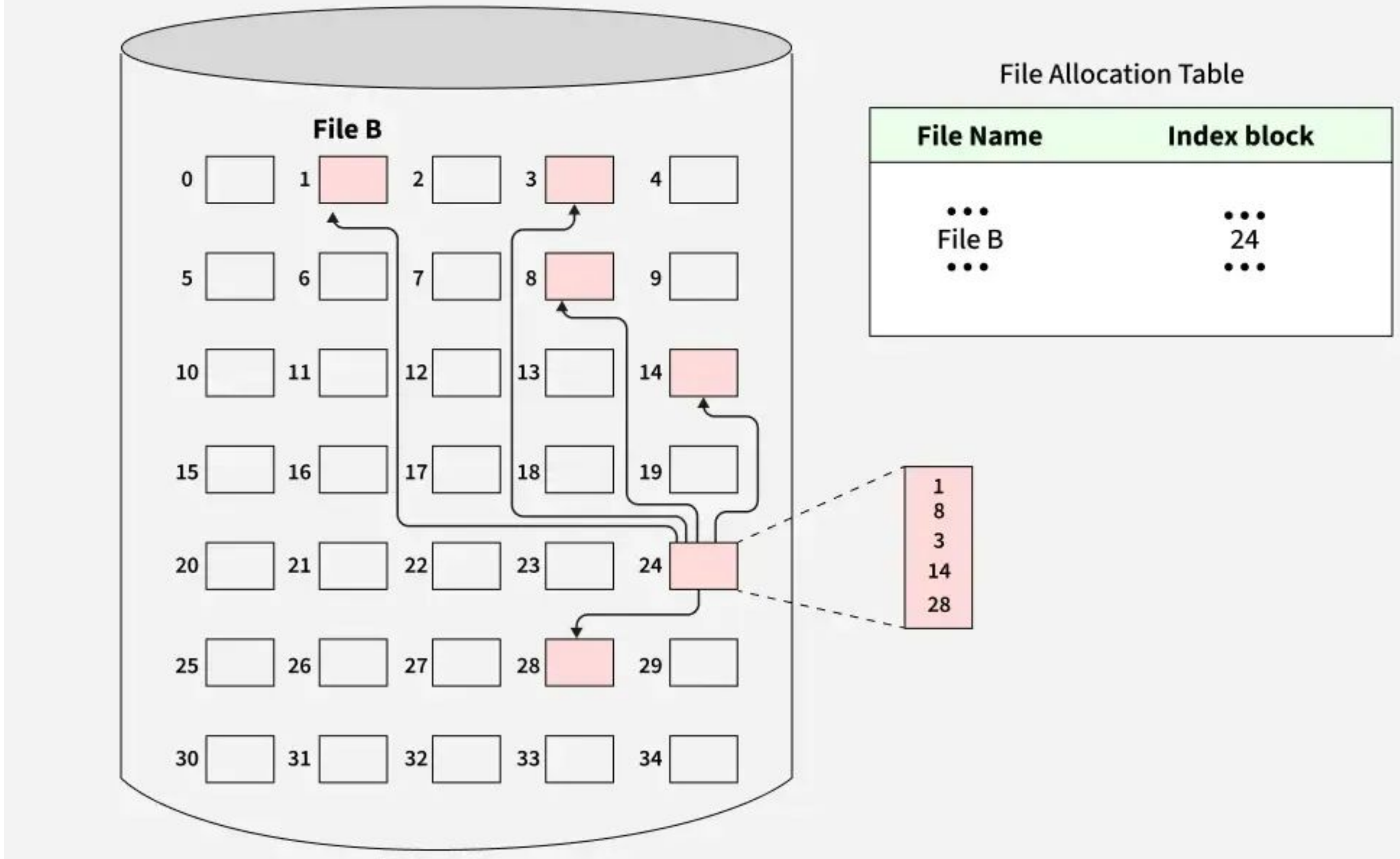
Advantages	Disadvantages
Supports random access to file blocks	FAT size grows as number of files/blocks increases
A bad/corrupted block does not affect entire file	Each block requires a FAT entry → memory overhead
No extra space needed inside blocks for pointers	Large FAT can lead to internal fragmentation

Indexed Allocation

- Each file has a special index block that contains pointers to all its disk blocks.
- The i-th entry in the index block gives the address of the i-th file block.
- The directory entry stores the address of this index block.
- Unlike Linked Allocation (where pointers are spread across blocks), here all pointers are stored in one place, making block retrieval easier.

This allows direct access to any block of the file.

Indexed Allocation



File Name	Index block
•••	•••
File B	24
•••	•••

Indexed Allocation: Advantages & Disadvantages

Advantages	**Disadvantages**
Supports direct access → fast block access	High pointer overhead (more than linked allocation)
No external fragmentation	Wastes space for small files (index block overhead)
Easier to retrieve blocks (all pointers in one place)	If index block is lost → entire file is lost
Suitable for random access of large files	A single index block may be too small for very large files

Indexing Schemes for Large Files

When a file is very large, one index block may not be enough. To solve this, different schemes are used:

1. **Linked Scheme**

- Multiple index blocks are linked together.
- Each index block has:
  - Some pointers to file blocks
  - A pointer to the next index block
- Works like a chain until all file blocks are covered.

Indexing Schemes for Large Files

2. Multilevel Index

- Uses a hierarchy of index blocks.
- Level 1 index block → points to Level 2 index blocks → which point to actual file blocks.
- Can extend to 3 or more levels for very large files.

3. Combined Scheme (Inode Structure)

- Uses a special block called Inode (Information Node).
- Inode stores:
  - File metadata (name, size, permissions, timestamps)
  - File block addresses (pointers)
- Some pointers in inode point directly to file blocks, while others may point to index blocks for large files.

Inode (Information Node) in OS

In UNIX-like systems, every file is represented by an Inode.

An inode is a special block created with the file system, which stores:

- Metadata (file name, size, permissions, timestamps, ownership, etc.)
- Pointers to file data blocks

Inode Structure

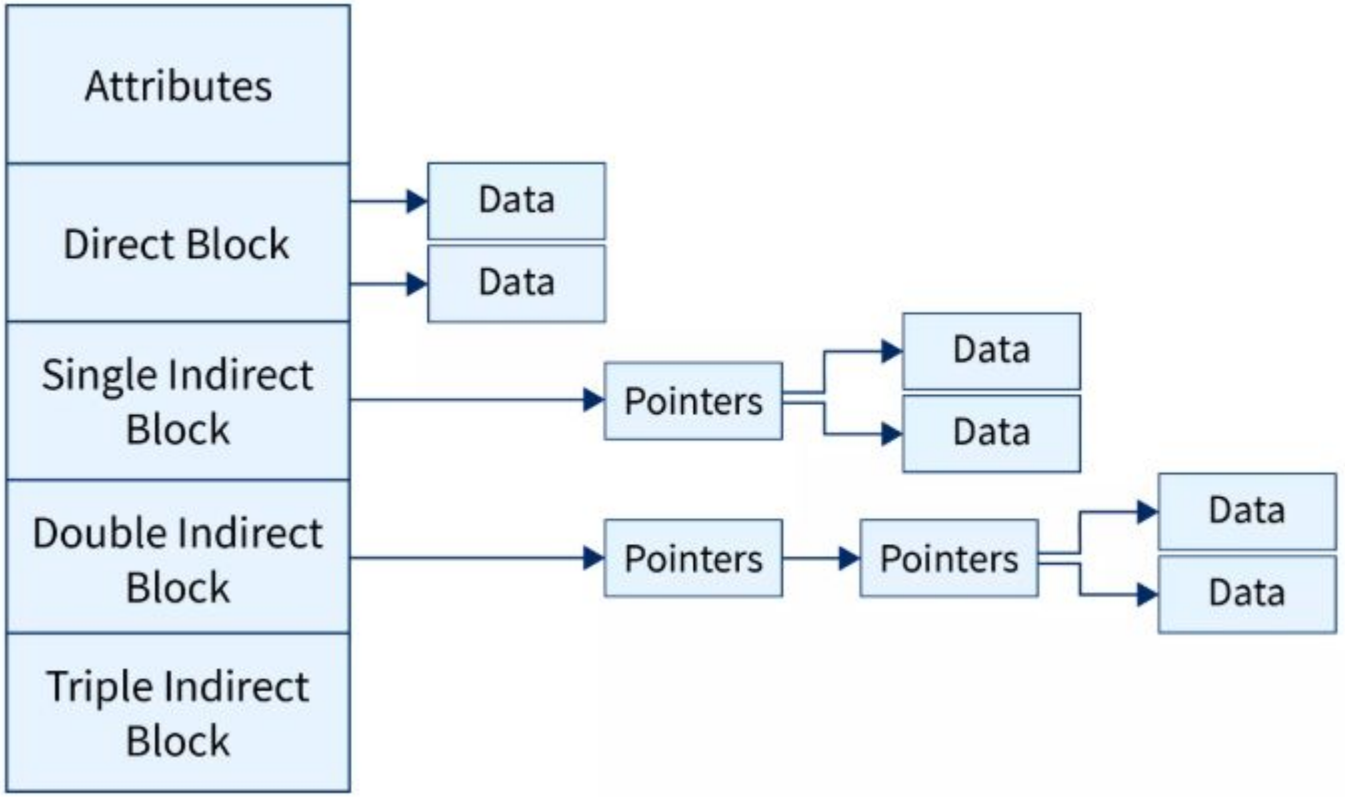
An inode keeps track of file details and block locations. It usually contains:

1. Direct Pointers → Point directly to data blocks (e.g., first 10).
2. Single Indirect Pointer → Points to a block that contains addresses of data blocks.
3. Double Indirect Pointer → Points to a block of addresses, which point to another block of addresses, then to data blocks.
4. Triple Indirect Pointer → Adds another layer of indirection for very large files.

This structure makes it possible to handle both small files (using direct pointers) and very large files (using multiple indirect levels) efficiently.

Inode Structure: Advantages & Disadvantages

Advantages	**Disadvantages**
Stores all file information (metadata + block addresses) in one place	Limited number of inodes → if exhausted, no new files/folders can be created
Supports both small and very large files using direct & indirect pointers	Even with free disk space, file creation fails if inodes are full
File accessibility is easy and efficient	Inode exhaustion may cause OS crashes, data loss, or application errors
File renaming does not affect inode (file identity remains intact)	Management of indirect pointers increases complexity for large files


Attributes
Direct Block
Single Indirect Block
Double Indirect Block
Triple Indirect Block
Data
Data
Pointers
Data
Data
Pointers
Pointers
Data
Data

COMPARISON

Method	How it Works	Advantages	Disadvantages
Contiguous Allocation	File stored in consecutive blocks	Fast access; easy implementation; supports sequential & direct access	File size must be known in advance; difficult to grow; causes fragmentation
Linked Allocation	File stored as linked list of scattered blocks; each block points to next	Flexible file size; no external fragmentation; good space utilization	Slower access; sequential access only; extra space for pointers

Method	How it Works	Advantages	Disadvantages
Indexed Allocation	File has an index block containing pointers to all blocks	Direct access; no external fragmentation; easy retrieval	Pointer overhead; wastes space for small files; loss of index block loses file
FAT (File Allocation Table)	Table stores mapping of all disk blocks; OS uses table for access	Random access possible; bad block does not affect entire file; no extra space in blocks for pointers	FAT size grows with number of blocks; memory overhead; possible internal fragmentation
Inode (UNIX)	Special block stores metadata + direct/indirect pointers	Easy file access; supports small & large files; renaming does not affect file; timestamps stored	Limited number of inodes; inode exhaustion prevents file creation; complexity for large files

DISCUSSION & REVISION

1. Which file allocation method stores files in consecutive blocks?
2. In Linked Allocation, each block contains a pointer to the ______ block.
3. Which allocation method uses an index block to store pointers to all file blocks?
4. What special structure in UNIX stores metadata and pointers for a file?
5. Which file allocation method allows random access using a table of block links?

REFERENCES

1. https://www.geeksforgeeks.org/operating-systems/file-allocation-methods/
2. https://www.scaler.com/topics/file-allocation-methods-in-os/