The File allocation methods in OS are different ways that are used for storing the file on the hard disk. There are 5 different ways in which we can store the files on the hard disk in such a manner that there is efficient utilization of disk space and the file can be accessed faster by the Operating System.

**What is File Allocation in OS?**

Whenever a hard disk is formatted, a system has many small areas called blocks or sectors that are used to store any kind of file. File allocation methods are different ways by which the operating system stores information in memory blocks, thus allowing the hard drive to be utilized effectively and the file to be accessed. Below are the types of file allocation methods in the Operating System.

**Types of File Allocation Methods in Operating System.**

* Contiguous File allocation
* Linked File Allocation
* Indexed File Allocation
* File Allocation Table (FAT)
* Inode

Let's have an in-detail explanation about each of them,

**Contiguous File Allocation.**

First, let's understand the meaning of contiguous, here contiguous means adjacent or touching. Now let's understand what is contiguous file allocation.

**What is Contiguous File allocation?**

In contiguous file allocation, the block is allocated in such a manner that all the allocated blocks in the hard disk are adjacent.

Assuming a file needs 'n' number of blocks in the disk and the file begins with a block at position'x', the next blocks to be assigned to it will be x+1,x+2,x+3,...,x+n-1 so that they are in a contiguous manner.

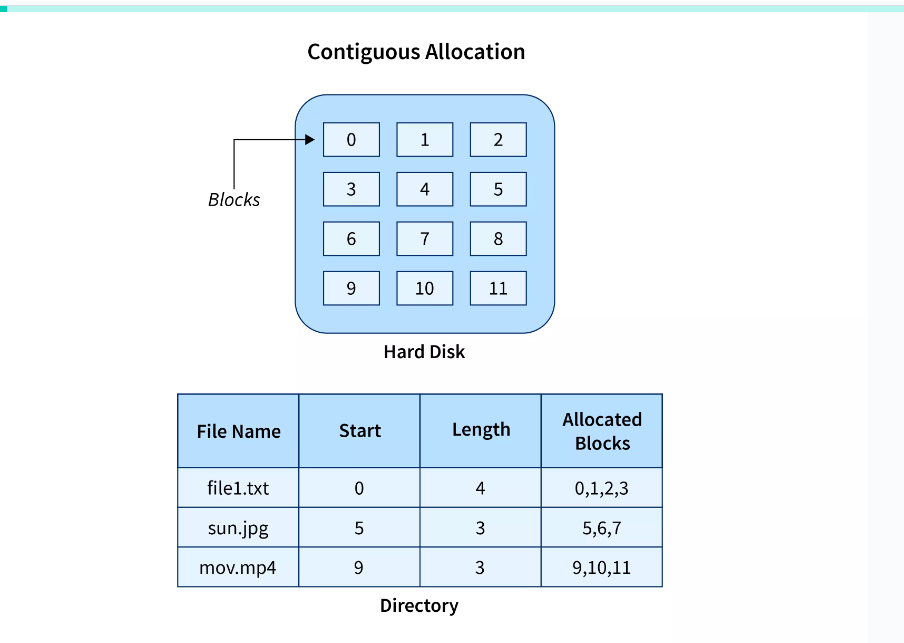
Let's understand this diagrammatically.

**Example**

We have three different types of files that are stored in a contiguous manner on the hard disk.

In the above image on the left side, we have a memory diagram where we can see the blocks of memory. At first, we have a text file named file1.txt which is allocated using contiguous memory allocation, it starts with the memory block 0 and has a length of 4 so it takes the 4 contiguous blocks 0,1,2,3. Similarly, we have an image file and video file named sun.jpg and mov.mp4 respectively, which you can see in the directory that they are stored in the contiguous blocks. 5,6,7 and 9,10,11 respectively.

Here the directory has the entry of each file where it stores the address of the starting block and the required space in terms of the block of memory.



**Advantages and Disadvantages**

**Advantages**

* It is very easy to implement.
* There is a minimum amount of seek time.
* The disk head movement is minimum.
* Memory access is faster.
* It supports sequential as well as direct access.

**Disadvantages**

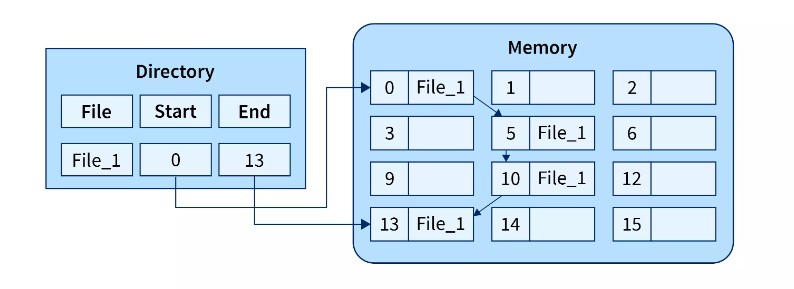
* At the time of creation, the file size must be initialized.
* As it is pre-initialized, the size cannot increase. As
* Due to its constrained allocation, it is possible that the disk would fragment internally or externally.

Quiz Pop

## Linked File Allocation.

### What is Linked File Allocation?

The Linked file allocation overcomes the drawback of contiguous file allocation. Here the file which we store on the hard disk is stored in a scattered manner according to the space available on the hard disk. Now, you must be thinking about how the OS remembers that all the scattered blocks belong to the same file. So as the name linked File Allocation suggests, the pointers are used to point to the next block of the same file, therefore along with the entry of each file each block also stores the pointer to the next block.



In the above image on the right, we have a memory diagram where we can see memory blocks. On the left side, we have a directory where we have the information like the address of the first memory block and the last memory block.

In this allocation, the starting block given is 0 and the ending block is 15, therefore the OS searches the empty blocks between 0 and 15 and stores the files in available blocks, but along with that it also stores the pointer to the next block in the present block. Hence it requires some extra space to store that link.

**Advantages and Disadvantages**

**Advantages**

* There is no external fragmentation.
* The directory entry just needs the address of starting block.
* The memory is not needed in contiguous form, it is more flexible than contiguous file allocation.

Disadvantages

1. Random Access is not provided.
2. Pointers require some space in the disk blocks.
3. Any of the pointers in the linked list must not be broken otherwise the file will get corrupted.
4. Need to traverse each block.

## Indexed File Allocation.

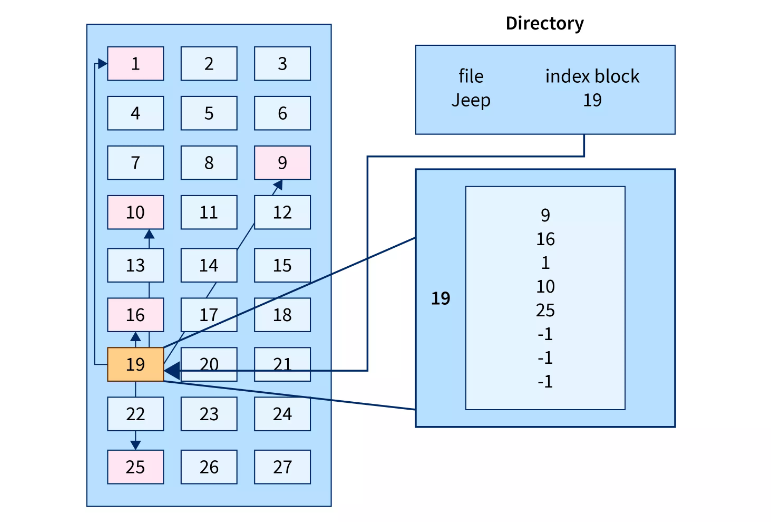
### What is Indexed File Allocation?

The indexed file allocation is somewhat similar to linked file allocation as indexed file allocation also uses pointers but the difference is here all the pointers are put together into one location which is called **index block**. That means we will get all the locations of blocks in one index file. The blocks and pointers were spread over the memory in the Linked Allocation method, where retrieval was accomplished by visiting each block sequentially. But here in indexed allocation, it becomes easier with the index block to retrieve.

Let's take an example to explain this better.

### Example

As shown in the diagram below block 19 is the index block which contains all the addresses of the file named **text1**. In order, the first storage block is 9, followed by 16, 1, then 10, and 25. The negative number -1 here denotes the empty index block list as the file text1 is still too small to fill more blocks.



**Advantages and Disadvantages**

**Advantages**

* It reduces the possibilities of external fragmentation.
* Rather than accessing sequentially it has direct access to the block.

**Disadvantages**

* Here more pointer overhead is there.
* If we lose the index block we cannot access the complete file.
* It becomes heavy for the small files.
* It is possible that a single index block cannot keep all the pointers for some large files.

To resolve this issue, we can use the following approaches:

1. Linked scheme
2. Multilevel Index
3. Combined Scheme

**1. Linked Scheme**

If the file is big then more blocks are required so one index block is insufficient to store all the pointers, therefore to store the pointers two or more index blocks are used where these index boxes are connected using linked file allocation that is each index block stores the pointer to the next index block.

**2. Multilevel Index**

In this method, the multiple indexes blocks along with the levels of these blocks. Here, the level 1 block is used for pointing to the level 2 block which points to the blocks occupied by the file. These index blocks can be extended to three or more levels according to the size of the file.

**3. Combined Scheme**

In Combined Scheme, a special block is used to store all the information related to the file like name, authority, size, etc. The special block is called **inode**(information-node). Some space of this special block is used to store the information related to the field as mentioned above and the remaining space is used to store the addresses of blocks that contain the actual file.

## File Allocation Table (FAT).

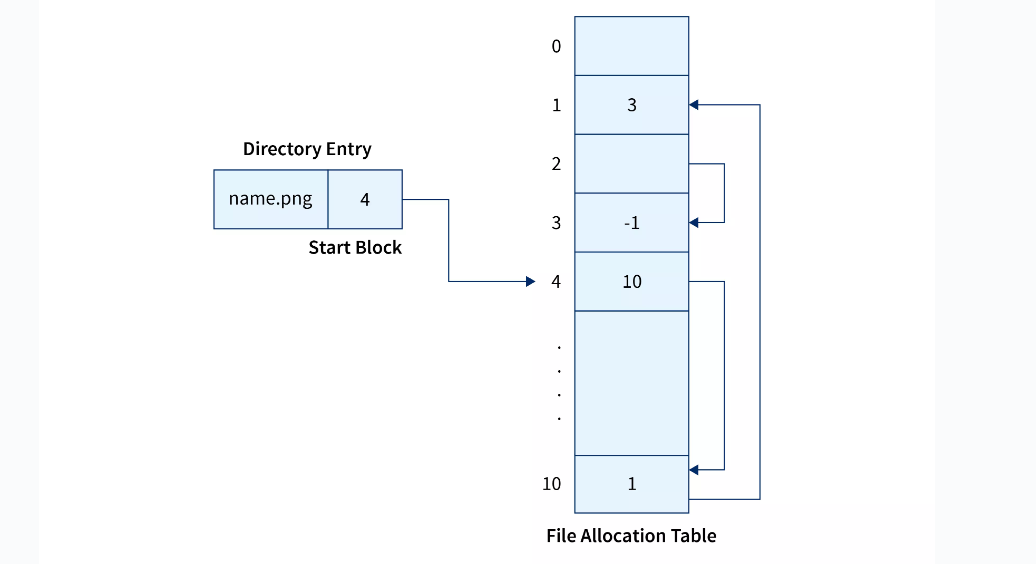
The File Allocation Table (FAT) overcomes the drawback of Linked File allocation. The random access of a particular block is not possible in the linked file allocation. To access a particular block it is necessary to access all its previous blocks. Let's see how File Allocation Table works.

### Explanation

In the file allocation table, all disk block links are collected and maintained. Here the number of head seeks is reduced by caching the file allocation table so that the head does not need to go through all the disk blocks to access one particular block.

The whole process of randomly accessing any block using FAT is completed by reading the desired entry of a block from the file allocation table and accessing that particular block.

The diagrammatic representation of FAT is given below -



**Advantages and Disadvantages**

**Advantages**

* Random Access to the block is possible in FAT.
* One bad/corrupted disk block cannot corrupt all the other blocks.
* It uses all the disk blocks for data as in linked file allocation it needs extra space for pointers.

**Disadvantages**

* If entries increase so the FAT size also increases.
* Each entry requires the FAT entry.
* If Entries increase the FAT size increases which increases the size of a block so there are chances of internal fragmentation.

# Structures of Directory in Operating System

A **directory** is a container that is used to contain folders and files. It organizes files and folders in a hierarchical manner.



Following are the logical structures of a directory, each providing a solution to the problem faced in previous type of directory structure.

### ****1) Single-level directory:****

The single-level directory is the **simplest directory structure**. In it, all files are contained in the same directory which makes it easy to support and understand.

A single level directory has a significant limitation, however, when the number of files increases or when the system has more than one user. Since all the files are in the same directory, they must have a **unique name**. If two users call their dataset test, then the unique name rule violated.



**Advantages:**

* Since it is a single directory, so its implementation is very easy.
* If the files are smaller in size, searching will become faster.
* The operations like file creation, searching, deletion, updating are very easy in such a directory structure.
* **Logical Organization**: Directory structures help to logically organize files and directories in a hierarchical structure. This provides an easy way to navigate and manage files, making it easier for users to access the data they need.
* **Increased Efficiency:** Directory structures can increase the efficiency of the file system by reducing the time required to search for files. This is because directory structures are optimized for fast file access, allowing users to quickly locate the file they need.
* **Improved Security**: Directory structures can provide better security for files by allowing access to be restricted at the directory level. This helps to prevent unauthorized access to sensitive data and ensures that important files are protected.
* **Facilitates Backup and Recovery**: Directory structures make it easier to backup and recover files in the event of a system failure or data loss. By storing related files in the same directory, it is easier to locate and backup all the files that need to be protected.
* **Scalability:** Directory structures are scalable, making it easy to add new directories and files as needed. This helps to accommodate growth in the system and makes it easier to manage large amounts of data.

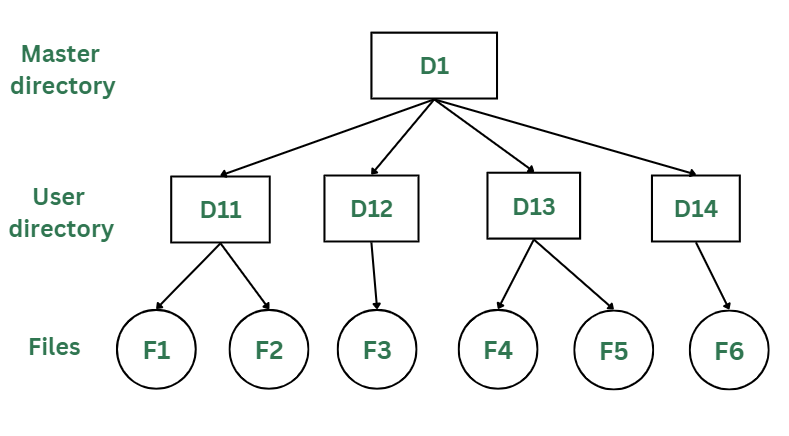
**Disadvantages:**

* There may chance of name collision because two files can have the same name.
* Searching will become time taking if the directory is large.
* This can not group the same type of files together.

### ****2) Two-level directory:****

As we have seen, a single level directory often leads to confusion of files names among different users. The solution to this problem is to create a **separate directory for each user**.

In the two-level directory structure, each user has their own **user files directory (UFD).** The UFDs have similar structures, but each lists only the files of a single user. System’s **master file directory (MFD*)*** is searched whenever a new user id is created.



*Two-Levels Directory Structure*

#### Advantages:

* The main advantage is there can be more than two files with same name, and would be very helpful if there are multiple users.
* A security would be there which would prevent user to access other user’s files.
* Searching of the files becomes very easy in this directory structure.

#### Disadvantages:

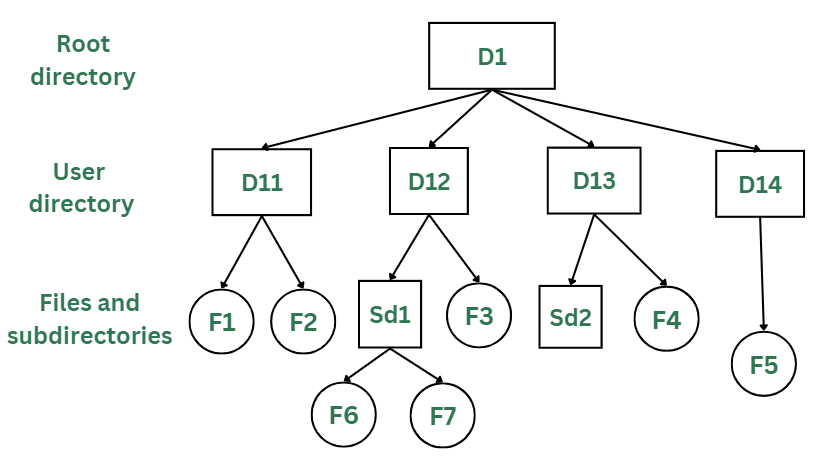
* As there is advantage of security, there is also disadvantage that the user cannot share the file with the other users.
* Unlike the advantage users can create their own files, users don’t have the ability to create subdirectories.
* Scalability is not possible because one use can’t group the same types of files together.

### 3) Tree Structure/ Hierarchical Structure:

Tree directory structure of operating system is most commonly used in our **personal computers**. User can create files and subdirectories too, which was a disadvantage in the previous directory structures.

This directory structure resembles a real tree upside down, where the **root directory** is at the peak. This root contains all the directories for each user. The users can create subdirectories and even store files in their directory.

A user do not have access to the root directory data and cannot modify it. And, even in this directory the user do not have access to other user’s directories.  The structure of tree directory is given below which shows how there are files and subdirectories in each user’s directory.



*Tree/Hierarchical Directory Structure*

#### Advantages:

* This directory structure allows subdirectories inside a directory.
* The searching is easier.
* File sorting of important and unimportant becomes easier.
* This directory is more scalable than the other two directory structures explained.

#### Disadvantages:

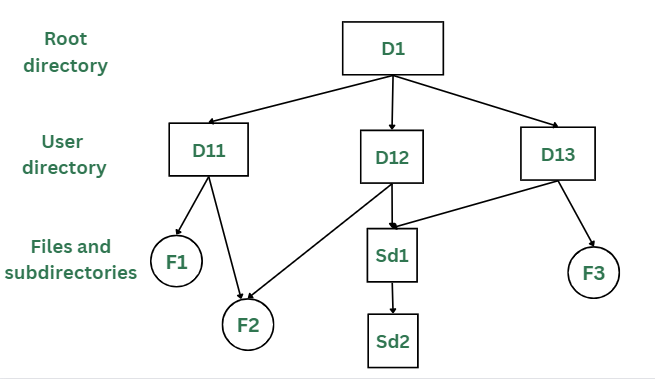
* As the user isn’t allowed to access other user’s directory, this prevents the file sharing among users.
* As the user has the capability to make subdirectories, if the number of subdirectories increase the searching may become complicated.
* Users cannot modify the root directory data.
* If files do not fit in one, they might have to be fit into other directories.

### 4) Acyclic Graph Structure:

As we have seen the above three directory structures, where none of them have the capability to access one file from multiple directories. The file or the subdirectory could be accessed through the directory it was present in, but not from the other directory.

This problem is solved in acyclic graph directory structure, where a file in one directory can be accessed from multiple directories. In this way, the files could be shared in between the users. It is designed in a way that multiple directories point to a particular directory or file with the help of links.

In the below figure, this explanation can be nicely observed, where a file is shared between multiple users. If any user makes a change, it would be reflected to both the users.



*Acyclic Graph Structure*

#### Advantages:

* Sharing of files and directories is allowed between multiple users.
* Searching becomes too easy.
* Flexibility is increased as file sharing and editing access is there for multiple users.

#### Disadvantages:

* Because of the complex structure it has, it is difficult to implement this directory structure.
* The user must be very cautious to edit or even deletion of file as the file is accessed by multiple users.
* If we need to delete the file, then we need to delete all the references of the file inorder to delete it permanently.