File

A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the file's creator and user.

What is a File System?

A file system is a method an operating system uses to store, organize, and manage files and directories on a storage device. Some common types of file systems include:

1. **FAT (File Allocation Table):** An older file system used by older versions of Windows and other operating systems.

2. **NTFS (New Technology File System):** A modern file system used by Windows. It supports features such as file and folder permissions, compression, and encryption.

3. **ext (Extended File System):** A file system commonly used on Linux and Unix-based operating systems.

4. **HFS (Hierarchical File System):** A file system used by macOS.

5. **APFS (Apple File System):** A new file system introduced by Apple for their Macs and iOS devices.

A file is a collection of related information that is recorded on secondary storage. Or file is a collection of logically related entities. From the user's perspective, a file is the smallest allotment of logical secondary storage.

The name of the file is divided into two parts as shown below:

- name

- extension, separated by a period

File Type

File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc. Many operating systems support many types of files. Operating system like Windows, MS-DOS and UNIX have the following types of files.

Ordinary files

- These are the files that contain user information.
- These may have text, databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the entire file.

Directory files

- These files contain list of file names and other information related to these files.

Special files

- These files are also known as device files.
- These files represent physical device like disks, terminals, printers, networks, tape drive etc.

These files are of two types −

- **Character special files** − data is handled character by character as in case of terminals or printers.
- **Block special files** − data is handled in blocks as in the case of disks and tapes.

Directories

Directory can be defined as the listing of the related files on the disk. The directory may store some or the entire file attributes.

To get the benefit of different file systems on the different operating systems, A hard disk can be divided into the number of partitions of different sizes. The partitions are also called volumes or mini disks.

Each partition must have at least one directory in which, all the files of the partition can be listed. A directory entry is maintained for each file in the directory which stores all the information related to that file.

The Directory Structure

Below are information/attributes contained in a directory/folder.

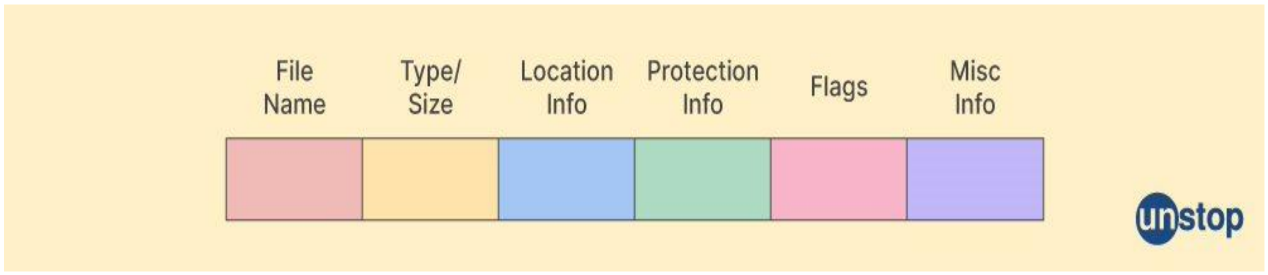
- Name
- Type
- Address
- Size
- Created by
- Date last accessed
- Date last updated
- Owner id
- Security information (if password protected)

The operation performed on the directory are:

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

Advantages of Maintaining Directories

- **Efficiency:** A file can be located more quickly.
- **Naming:** It becomes convenient for users as two users can have same name for different files or may have different name for same file.
- **Grouping:** Logical grouping of files can be done by properties e.g. all java programs, all games etc.



Types of Directory Structure

There are five types of directories:

- Single-Level Directory Structure
- Two-Level Directory Structure
- Tree-Structured Directory
- Acyclic-Graph Directory Structure
- General-Graph Directory Structure

Following are the details of the types of logical structures of a directory:

Single-Level Directory Structure

The most basic way is to keep a single large list of all files on a drive. However, when the number of files grows or the system has more than one user, a single-level directory structure becomes a severe constraint. No two files with the same name are allowed.

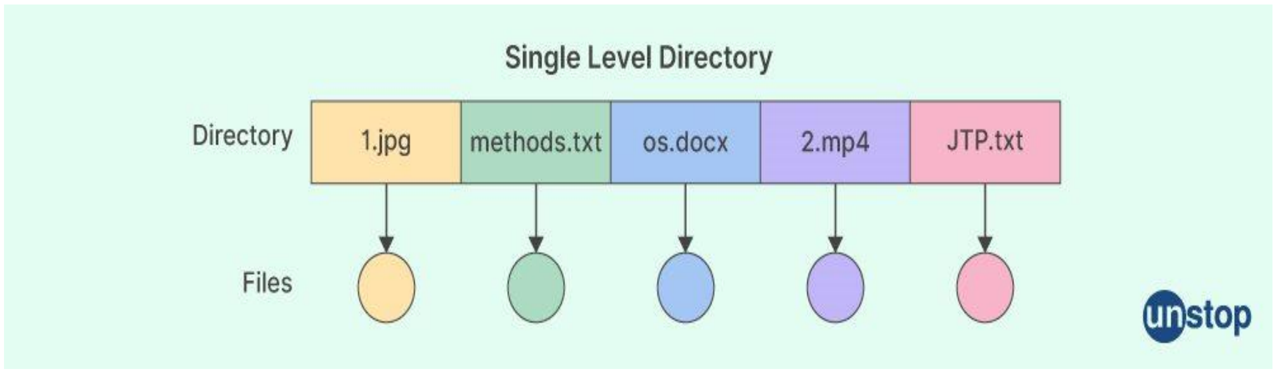
Multiple users/programs on a disc may necessitate this directory. There is no method to group files. There is only one long list. Searches must traverse the full directory.

Advantages:

1. Because it is just a single directory, therefore it is the simplest directory structure.
2. Searching will be faster if the files are smaller in size. Also, the groping capability of files increases.
3. In such a user file directory structure, actions like file creation, searching, file deletion, and updating are quite simple.

Disadvantages:

1. In a single-level directory structure, if the directory is vast, searching will be difficult.
2. We can't group the same type of files in a single-level directory structure.
3. Selecting a unique file name is a little more difficult than in other types.



Two-Level Directory Structure

Another sort of directory layout is the two-level directory structure. It is feasible to create a separate directory for each user in this way. This two-level directory structure enables the usage of the same file name across several user directories. There is a single master file directory that contains individual directories for each single user files directory. At the second level, there is a separate directory for each user, which contains a collection of users' files. The mechanism prevents a user from accessing another user's directory without their authorization.

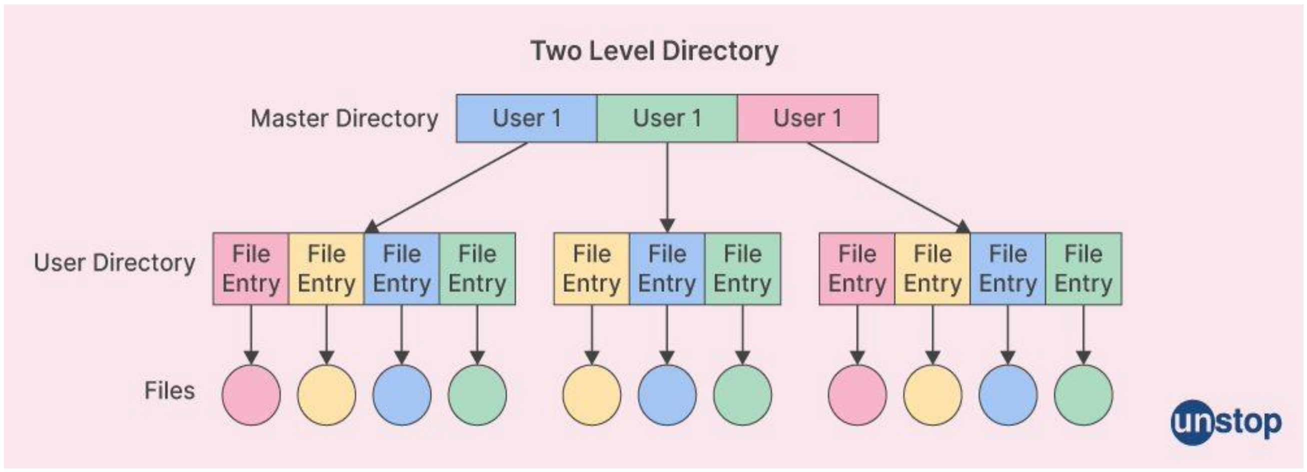
Files now have a path: */user1/directory-name*. Different users can have the same file name *(/user2/me and /user3/me)*. Because only one user's list needs to be searched, searching is more efficient. However, there is currently no way to group a user's files.

Advantages:

1. In a two-level directory, a full path like user-name/directory-name can be given.
2. Different users can have the same directory as well as the file name.
3. Searching files becomes easier.

Disadvantages:

1. One user cannot share a file with another user in a two-level directory without permission.
2. The two-level directory also has the problem of not being scalable.



Tree-Structured Directory Structure

In the tree directory structure, searching is more efficient, and the concept of a current working directory is used. Files can be arranged in logical groups. We can put files of the same type in the same directory.

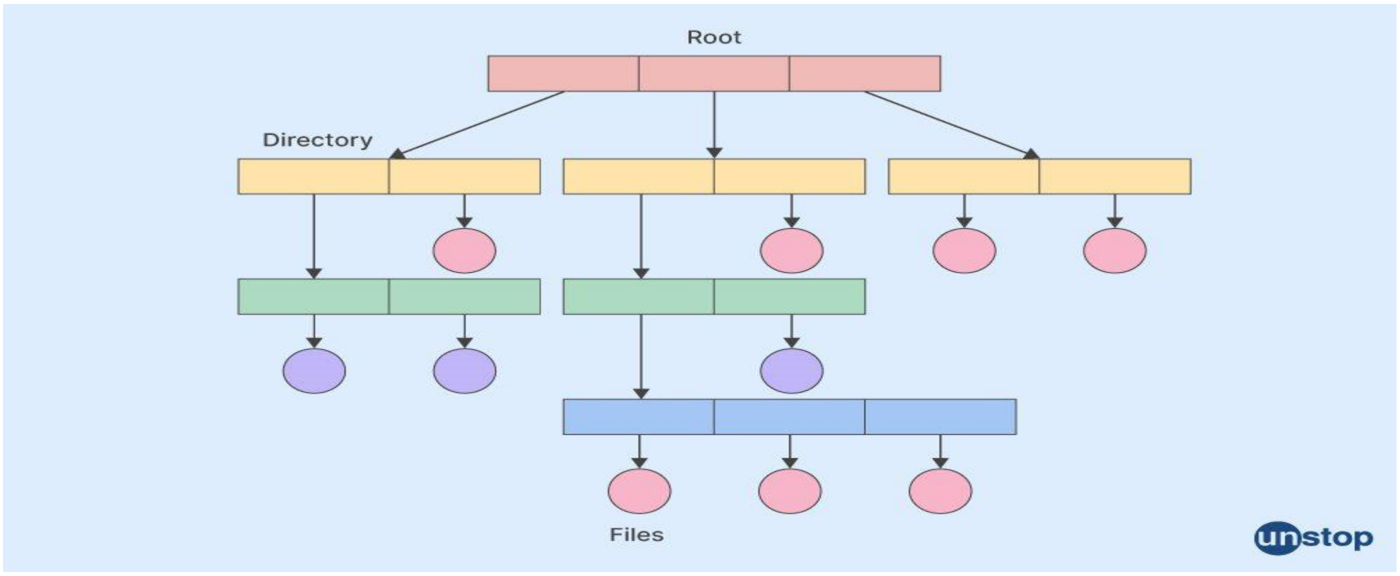
It works on files in the current directory by default, or you can even give a relative or absolute path. It also has the ability to create and delete directories. Every file in the system has a unique path. The tree has a root directory.

Advantages:

1. The directory, which is organized in a tree form, is extremely scalable.
2. Collisions are less likely in the tree-structured directory.
3. Searching in the tree-structure directory is relatively simple because we may utilize both absolute and relative paths.

Disadvantages:

1. Files may be saved in numerous directories if they do not fit into the hierarchical structure.
2. We are unable to share files in a tree-structured directory.
3. Because a file might be found in several folders in a tree-structured directory, it is inefficient.



Acyclic-Graph Directory Structure

The tree model forbids the existence of the same file in several directories. By making the directory an acyclic graph structure, we may achieve this. Two or more directory entries can lead to the same subdirectory or file, but we'll limit it, for now, to prevent any directory entries from pointing back up the directory structure.

Links or aliases can be used to create this type of directory graph. We have numerous names for the same file, as well as multiple paths to it. There are two types of links:

1. symbolic link or soft link (specify a file path: logical) and
2. hard link (actual link to the same file on the disc from multiple directories: physical).

If we delete the file, there may be more references to it. The file is simply erased via symbolic links, leaving a dangling pointer. A reference count is kept through hard links, and the actual file is only erased once all references to it have been removed.

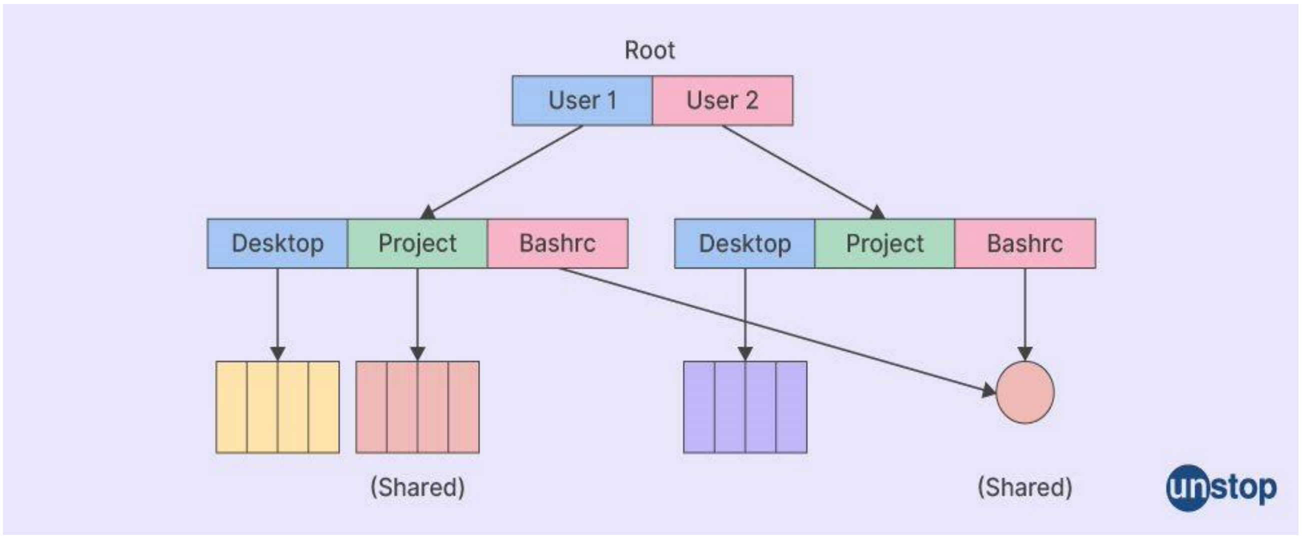
Advantages:

1. Acyclic-Graph directory structure provides the ability to share files.
2. Due to different-different paths, searching is simple in the Acyclic-Graph directory structure.

Disadvantages:

1. If the files are linked together, removing them may be difficult.

2. If we use softlink and if the file is removed, all that is left is a dangling(free) pointer.
3. If we use a hardlink when we delete a file, we must also erase all of the references that are associated with it.



General-Graph Directory Structure

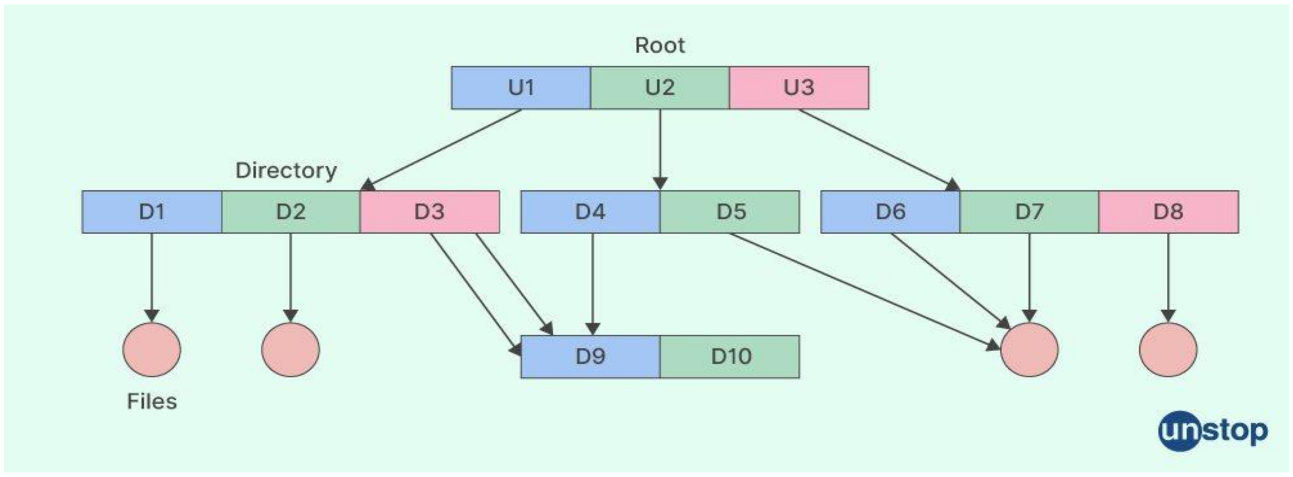
Cycles are allowed inside a directory structure where numerous directories can be derived from more than one parent directory in a general graph directory structure. When general graph directories are allowed, commands like, search a directory and its subdirectories, must be used with caution. If cycles are allowed, the search is infinite. The biggest issue with this type of directory layout is figuring out how much space the files and folders have used up.

Advantages:

1. The general-graph directory structure is more adaptable than the others.
2. In the general-graph directory, cycles are permitted.

Disadvantages:

1. It is more expensive than other options.
2. Garbage collection is required in the general-graph directory.

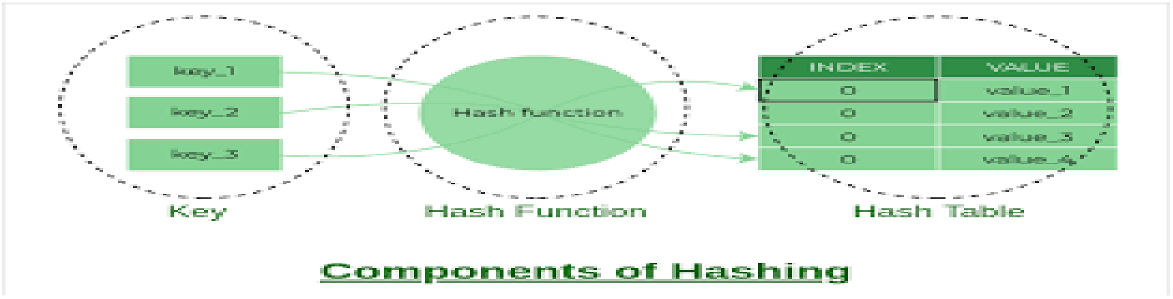


Directory Implementation

An individual subdirectory will typically contain a list of files. The choice of a suitable algorithm for directory implementation is critical since it has a direct impact on system performance. Based on the data structure, we may classify the directory implementation algorithm. It can be implemented with the following approaches:

1. **Linear List**: It contains a list of names, each of which has a pointer to the file's data blocks. It requires a costly search on large directories.
2. **Hash Table:** It is a hashed linear list that decreases search time, but is more complex to implement. We can identify the key and key points to the relevant file that is stored in a directory using the hash function on the respective file name.

A hash table is a type of data structure in which information is stored in an easy-to-retrieve and efficient manner. In the key-value method, keys are assigned random indexes where their values are stored in an array. The index is the information of where exactly in the array the value is stored.



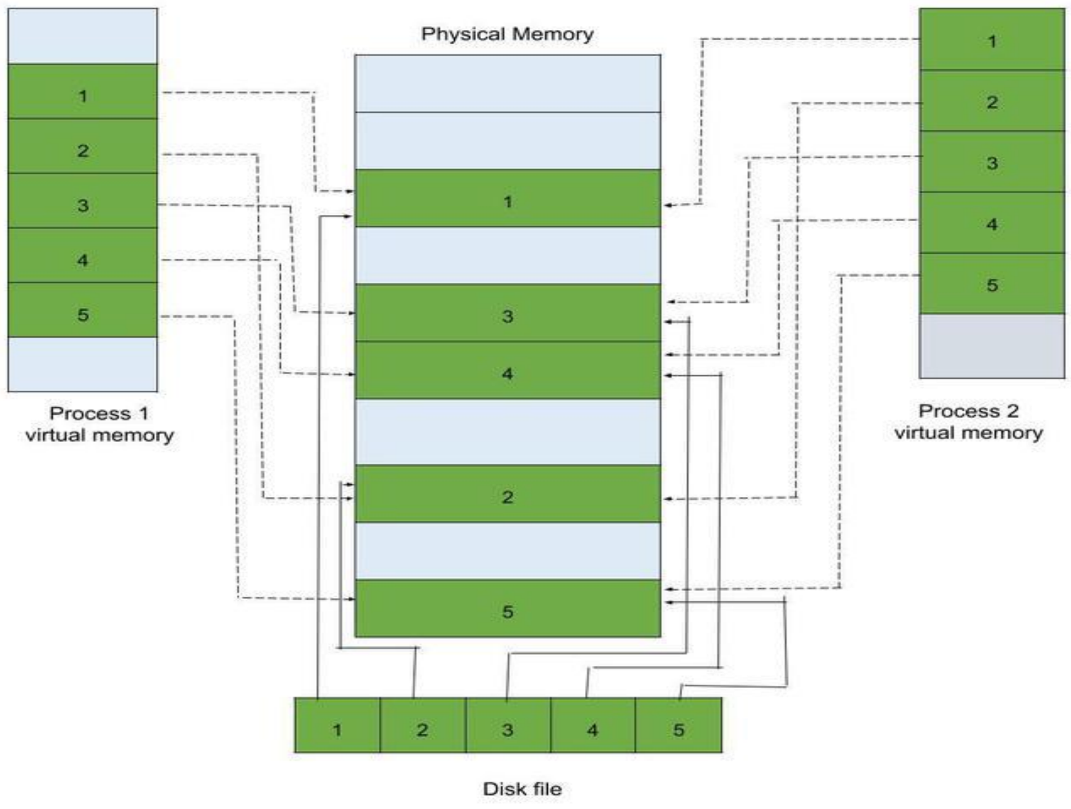
Memory Mapped Files in OS

We can use standard system calls like read(), seek(), open(), and so on to perform a sequential read of a file present on the disk. Thus, to access a file from the disk we need system calls and disk access. Memory mapping is a technique that allows a part of the virtual address space to be associated with a file logically. This technique of memory mapping leads to a significant increase in performance.

Basic Mechanism of Memory Mapping

- The OS uses **Virtual Memory** for memory mapping a file. It is performed by mapping a disk block to a page present in the physical memory. Initially, the file is accessed through **Demand Paging**. If a process references an address that does not exist in the physical memory, then page fault occurs and the Operating System takes charge of bringing the missing page into the physical memory.
- A page-sized portion of the file is read from the file system into a physical page.
- Manipulating the files through the use of memory rather than incurring the overhead of using the read() and write() system calls not only simplifies but also speeds up file access and usage.
- Multiple processes may be allowed to map a single file simultaneously to allow sharing of data.
- If any of the processes write data in the virtual memory, then the modified data will be visible to all the processes that map the same section of the file.
- The memory mapping system calls support copy-on-write functionality which allows processes to share a file in read-only mode but the processes can have their own copies of data that they have modified.

The sharing of memory is depicted with the help of a diagram shown below.



Memory Mapped Files

Types of Memory Mapped Files

Basically, there are two types of memory mapped files:

- **Persisted**: Persisted files are connected with a source file on a disk. After completing the final process, the data is saved to the source file on disc. Working with very big source files is appropriate with these types of memory-mapped files.
- **Non-persisted**: Non-persisted files are not connected to any disk-based files. The data is lost when the last process with the file completes its required task. The shared memory that these files enable for inter-process communications or IPC.

Advantages of Memory Mapped Files

- It increases the I/O performance especially when it is used on large files.
- Accessing memory mapped file is faster than using direct system calls like read() and write().
- Another advantage is lazy loading where small amount of RAM is used for a very large file.
- Shared memory is often implemented by memory mapping files. Thus, it supports data sharing.

Disadvantages of Memory Mapped Files

- In some cases, memory mapped file I/O may be substantially slower as compared to standard file I/O.
- Only hardware architecture that has MMU (Memory Management Unit) supports memory mapped files.
- In memory mapped files, expanding the size of a file is not easy.

Conclusion

Memory mapped files can be used as a process loader in several modern operating systems. Also, memory mapped file I/O is one of the most popular way to safely share memory. Thus, with the help of memory mapped file, applications can access files present on the disk much the same way they access dynamic memory with the help of pointers.

File Allocation Methods

The allocation methods define how the files are stored in the disk blocks. There are three main disk space or file allocation methods.

- Contiguous Allocation
- Linked Allocation
- Indexed Allocation

The main idea behind these methods is to provide:

- Efficient disk space utilization.
- Fast access to the file blocks.

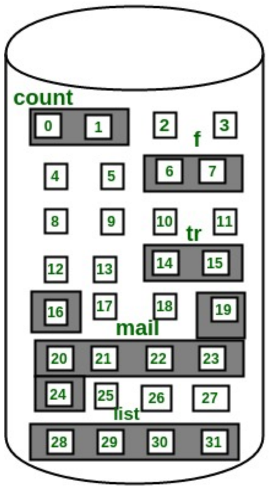
All the three methods have their own advantages and disadvantages as discussed below:

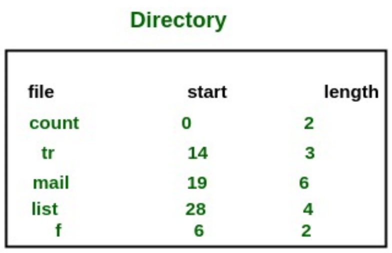
Contiguous Allocation

In this scheme, each file occupies a contiguous set of blocks on the disk. For example, if a file requires n blocks and is given a block b as the starting location, then the blocks assigned to the file will be: b, b+1, b+2,......b+n-1. This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file.
The directory entry for a file with contiguous allocation contains

- Address of starting block
- Length of the allocated portion.

The file '**mail**' in the following figure starts from the block 19 with length = 6 blocks. Therefore, it occupies 19, 20, 21, 22, 23, 24 blocks.




Directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Advantages:

- Both the Sequential and Direct Accesses are supported by this. For direct access, the address of the kth block of the file which starts at block b can easily be obtained as (b+k).
- This is extremely fast since the number of seeks are minimal because of contiguous allocation of file blocks.

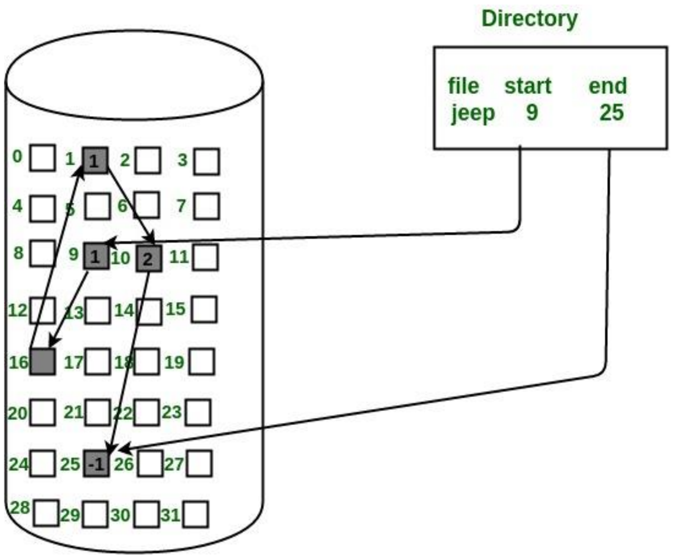
Disadvantages:

- This method suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization.
- Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

Linked List Allocation

In this scheme, each file is a linked list of disk blocks which need not be contiguous. The disk blocks can be scattered anywhere on the disk. The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.

The file 'jeep' in following image shows how the blocks are randomly distributed. The last block (25) contains -1 indicating a null pointer and does not point to any other block.



Advantages:

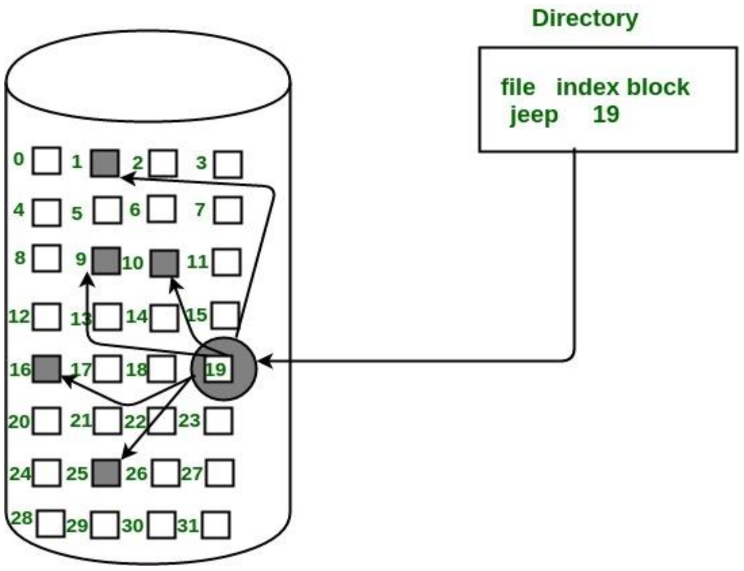
- This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
- This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.

Disadvantages:

- Because the file blocks are distributed randomly on the disk, a large number of seeks are needed to access every block individually. This makes linked allocation slower.
- It does not support random or direct access. We can not directly access the blocks of a file. A block k of a file can be accessed by traversing k blocks sequentially (sequential access) from the starting block of the file via block pointers.
- Pointers required in the linked allocation incur some extra overhead.

Indexed Allocation

In this scheme, a special block known as the Index block contains the pointers to all the blocks occupied by a file. Each file has its own index block. The n^{th} entry in the index block contains the disk address of the n^{th} file block. The directory entry contains the address of the index block as shown in the image:



Advantages:

- This supports direct access to the blocks occupied by the file and therefore provides fast access to the file blocks.
- It overcomes the problem of external fragmentation.

Disadvantages:

- The pointer overhead for indexed allocation is greater than linked allocation.
- For very small files, say files that expand only 2-3 blocks, the indexed allocation would keep one entire block (index block) for the pointers which is inefficient in terms of memory utilization. However, in linked allocation we lose the space of only 1 pointer per block.

For files that are very large, single index block may not be able to hold all the pointers. Following mechanisms can be used to resolve this:

1. **Linked scheme**: This scheme links two or more index blocks together for holding the pointers. Every index block would then contain a pointer or the address to the next index block.
2. **Multilevel index:** In this policy, a first level index block is used to point to the second level index blocks which in turn points to the disk blocks occupied by the file. This can be extended to 3 or more levels depending on the maximum file size.

File Access Methods in Operating System

When a file is used, information is read and accessed into computer memory and there are several ways to access this information of the file. Some systems provide only one access method for files. Other systems, such as those of IBM, support many access methods, and choosing the right one for a particular application is a major design problem.

There are three ways to access a file into a computer system: Sequential-Access, Direct Access, Index sequential Method.

Sequential Access:

It is the simplest access method. Information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, editor and compiler usually access the file in this fashion.

Read and write make up the bulk of the operation on a file. A read operation -*read next*- read the next position of the file and automatically advance a file pointer, which keeps track I/O location. Similarly, for the -write *next*- append to the end of the file and advance to the newly written material.

Key points:

- Data is accessed one record right after another record in an order.
- When we use read command, it move ahead pointer by one
- When we use write command, it will allocate memory and move the pointer to the end of the file
- Such a method is reasonable for tape.

Advantages of Sequential Access Method:

- It is simple to implement this file access mechanism.
- It uses lexicographic order to quickly access the next entry.
- It is suitable for applications that require access to all records in a file, in a specific order.

- It is less prone to data corruption as the data is written sequentially and not randomly.
- It is a more efficient method for reading large files, as it only reads the required data and does not waste time reading unnecessary data.
- It is a reliable method for backup and restore operations, as the data is stored sequentially and can be easily restored if required.

Disadvantages of Sequential Access Method:

- If the file record that needs to be accessed next is not present next to the current record, this type of file access method is slow.
- Moving a sizable chunk of the file may be necessary to insert a new record.
- It does not allow for quick access to specific records in the file. The entire file must be searched sequentially to find a specific record, which can be time-consuming.
- It is not well-suited for applications that require frequent updates or modifications to the file. Updating or inserting a record in the middle of a large file can be a slow and cumbersome process.
- Sequential access can also result in wasted storage space if records are of varying lengths. The space between records cannot be used by other records, which can result in inefficient use of storage.

Direct Access:
Another method is direct access method also known as relative access method. A fixed-length logical record that allows the program to read and write record rapidly. in no particular order. The direct access is based on the disk model of a file since disk allows random access to any file block. For direct access, the file is viewed as a numbered sequence of block or record. Thus, we may read block 14 then block 59, and then we can write block 17. There is no restriction on the order of reading and writing for a direct access file.
A block number provided by the user to the operating system is normally a relative block number, the first relative block of the file is 0 and then 1 and so on.

Advantages of Direct Access Method:

- The files can be immediately accessed decreasing the average access time.
- In the direct access method, in order to access a block, there is no need of traversing all the blocks present before it.

Index sequential method:
It is the other method of accessing a file that is built on the top of the sequential access method. These methods construct an index for the file. The index, like an index in the back of a book, contains the pointer to the various blocks. To find a record in the file, we first search the index, and then by the help of pointer we access the file directly.

Key points:

- It is built on top of Sequential access.
- It controls the pointer by using index.

OS File Systems

A File System is an integral part of an OS. A **File System** is a data structure that stores data and information on storage devices (hard drives, floppy disc, etc.), making them easily retrievable. Different OS's use different file systems, but all have similar features.

The most important part of the file system is the method used to index files on the hard drive. This index allows the OS to know at any given time where to find a specific file on the hard drive. This index is most typically based on file names. Depending on the operating system there are different file systems, and thus different indexing methods.

Windows

In the Windows environment, you will find one of three file systems, each with different indexing methods.

1. The first is known as the FAT (12, 16, or 32) file system. This system uses what is known as a **File Allocation Table** to index the files on the disc. This file allocation

table is very simple to implement and use, but can be somewhat slow. It divides hard disks into one or more partitions (parts) that become letters, like C:, D:, etc.

2. The second file system you may encounter when using Windows is known as NTFS (New Technology File System). *NTFS* uses binary trees that, while complex, allow for very fast access times. It builds on the features of FAT, adds new features, and changes a few others. It is a recoverable file system, which means that it keeps track of actions in the file system.

3. The third file system, exFAT, is a lightweight file system used primarily in flash storage applications and SD cards. It has large file size and partition size limits, which means you can store files over 4GB on a flash drive or SD card that is formatted with exFAT.

MAC OS

In the macOS environment you will find the HFS+ file system (pronounce it *HFS Plus*). As of June 2016, Apple has implemented their new file system "APFS". This has become the replacement for HFS+ on macOS High Sierra and onward, as well as iOS 10.3+.

Linux

Depending on which type of Linux environment you are running, you may run into several different file systems. Some of them are ext2(second extended filesystem), ext3, and ext4. XFS, JFS, and a few others are also used.