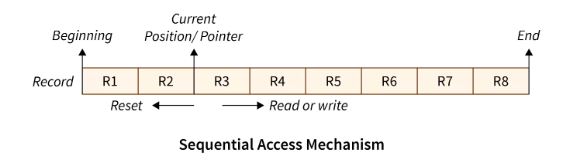
**File Access Methods in OS**

File access methods in OS are nothing but techniques to read data from the system's memory. There are various ways in which we can access the files from the memory like:

* Sequential Access
* Direct/Relative Access, and
* Indexed Sequential Access.

### 1. Sequential Access

The operating system reads the file word by word in a sequential access method of file accessing. A pointer is made, which first links to the file's base address. If the user wishes to read the first word of the file, the pointer gives it to them and raises its value to the next word. This procedure continues till the file is finished. It is the most basic way of file access. The data in the file is evaluated in the order that it appears in the file and that is why it is easy and simple to access a file's data using a sequential access mechanism. For example, editors and compilers frequently use this method to check the validity of the code.



**Advantages of Sequential Access:**

* The sequential access mechanism is very easy to implement.
* It uses lexicographic order to enable quick access to the next entry.

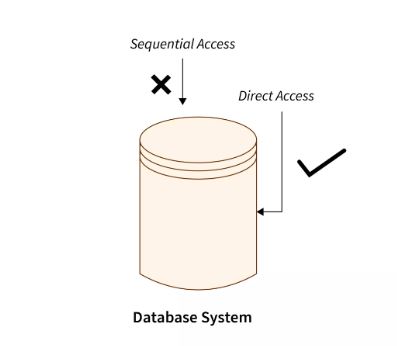
**Disadvantages of Sequential Access:**

* Sequential access will become slow if the next file record to be retrieved is not present next to the currently pointed record.
* Adding a new record may need relocating a significant number of records of the file.

### 2. Direct (or Relative) Access

A Direct/Relative file access mechanism is mostly required with the database systems. In the majority of the circumstances, we require filtered/specific data from the database, and in such circumstances, sequential access might be highly inefficient. Assume that each block of storage holds four records and that the record we want to access is stored in the tenth block. In such a situation, sequential access will not be used since it will have to traverse all of the blocks to get to the required record, while direct access will allow us to access the required record instantly.

The direct access mechanism requires the OS to perform some additional tasks but eventually leads to much faster retrieval of records as compared to sequential access.



**Advantages of Direct/Relative Access:**

* The files can be retrieved right away with a direct access mechanism, reducing the average access time of a file.
* There is no need to traverse all of the blocks that come before the required block to access the record.

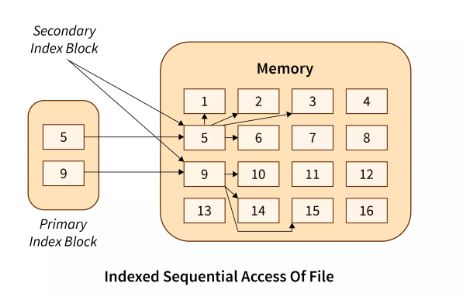
**Disadvantages of Direct/Relative Access:**

* The direct access mechanism is typically difficult to implement due to its complexity.
* Organizations can face security issues as a result of direct access as the users may access/modify the sensitive information. As a result, additional security processes must be put in place.

### 3. Indexed Sequential Access

It's the other approach to accessing a file that's constructed on top of the sequential access mechanism. This method is practically similar to the pointer-to-pointer concept in which we store the address of a pointer variable containing the address of some other variable/record in another pointer variable. The indexes, similar to a book's index (pointers), contain a link to various blocks present in the memory. To locate a record in the file, we first search the indexes and then use the pointer-to-pointer concept to navigate to the required file.

Primary index blocks contain the links of the secondary inner blocks which contain links to the data in the memory.



**Advantages of Indexed Sequential Access:**

* If the index table is appropriately arranged, it accesses the records very quickly.
* Records can be added at any position in the file quickly.

**Disadvantages of Indexed Sequential Access:**

* When compared to other file access methods, it is costly and less efficient.
* It needs additional storage space.

# 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

 Different structures of directories can be used to organize these files, making it easier to find and manage them.

Understanding these directory structures is important because it helps in efficiently organizing and accessing files on your computer.

## Different Types of Directory in OS

In an operating system, there are different types of directory structures that help organize and manage files efficiently.

* Single-Level Directory
* Two-Level Directory
* Tree Structure/ Hierarchical 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.

Single Level Directory

### ****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.

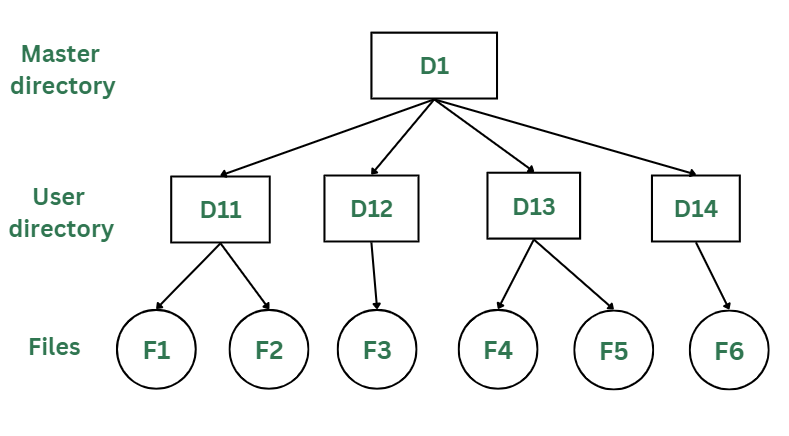
### ****Disadvantages****

* 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 create



### 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

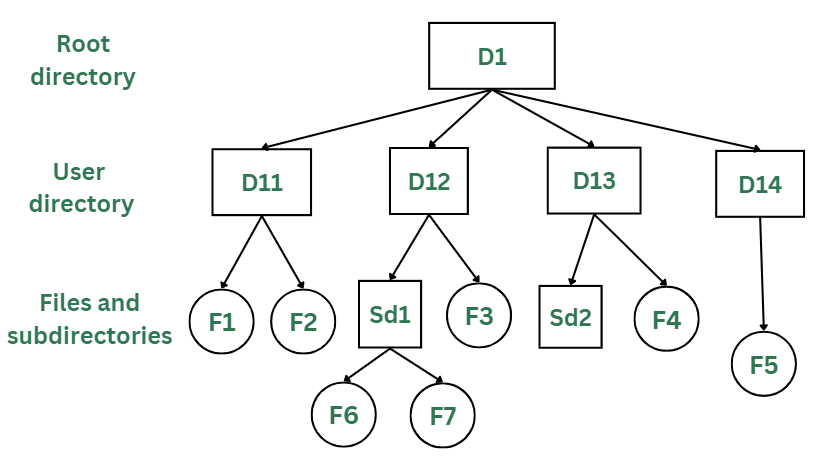
* As there is advantage of security, there is also disadvantage that the user cannot share the file with the other users.
* Unlike the advantage users can create their own files, users don’t have the ability to create subdirectories.
* Scalability is not possible because one user can’t group the same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

## 3) Tree Structure/ Hierarchical Structure

Tree directory structure of operating system is most commonly used in our [**personal computers**](https://www.geeksforgeeks.org/types-of-computers/). User can create files and subdirectories too,

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

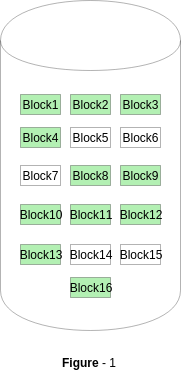
* This directory structure allows subdirectories inside a directory.
* The searching is easier.
* File sorting of important and unimportant becomes easier.
* This directory is more scalable than the other two directory structures explained.

### 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.

## ****1. Bitmap or Bit vector****

A [Bitmap](https://www.geeksforgeeks.org/bitmap-indexing-in-dbms/) or Bit Vector is series or collection of bits where each bit corresponds to a disk block. The bit can take two values: 0 and 1: 0 indicates that the block is free and 1 indicates an allocated​ block. The given instance of disk blocks on the can be represented by a bitmap of 16 bits as:**1111000111111001**.



**Advantages:**

* Simple to understand.
* Finding the first free block is efficient. It requires scanning the words (a group of 8 bits) in a bitmap for a non-zero word. (A 0-valued word has all bits 0). The first free block is then found by scanning for the first 1 bit in the non-zero word.

**Disadvantages:**

* For finding a free block, Operating System needs to iterate all the blocks which is time consuming.
* The efficiency of this method reduces as the disk size increases.

## Linked List****2. Linked List****

In this approach, the free disk blocks are linked together i.e. a free block contains a pointer to the next free block. The block number of the very first disk block is stored at a separate location on disk and is also cached in memory.

In Figure-2, the free space list head points to Block 5 which points to Block 6, the next free block and so on. The last free block would contain a null pointer indicating the end of free list. A drawback of this method is the I/O required for free space list traversal.

**Advantages:**

* The total available space is used efficiently using this method.
* Dynamic allocation in Linked List is easy, thus can add the space as per the requirement dynamically.

**Disadvantages:**

* When the size of Linked List increases, the headache of miniating pointers is also increases.
* This method is not efficient during iteration of each block of memory.

## ****3. Grouping****

This approach stores the address of the free blocks in the first free block. The first free block stores the address of some, say n free blocks. Out of these n blocks, the first n-1 blocks are actually free and the last block contains the address of next free n blocks. An **advantage** of this approach is that the addresses of a group of free disk blocks can be found easily.

**Advantage:**

* Finding free blocks in massive amount can be done easily using this method.

**Disadvantage:**

* The only disadvantage is, we need to alter the entire list, if any of the block of the list is occupied.

## ****4.Counting****

This approach stores the address of the first free disk block and a number n of free contiguous disk blocks that follow the first block. Every entry in the list would contain:

* Address of first free disk block.
* A number n.

**Advantages:**

* Using this method, a group of entire free blocks can take place easily and Fastly.
* The list formed in this method is especially smaller in size.

**Disadvantage:**

* The first free block in this method, keeps account of other free blocks. Thus, due to that one block the space requirement is more.

**Protection in File System**

# In computer systems, a lot of user’s information is stored, the objective of the operating system is to keep safe the data of the user from the improper access to the system.

**Access Control :**

There are different methods used by different users to access any file. The general way of protection is to associate *identity-dependent access* with all the files and directories an list called [access-control list (ACL)](https://www.geeksforgeeks.org/access-lists-acl/) which specify the names of the users and the types of access associate with each of the user. The main problem with the access list is their length. If we want to allow everyone to read a file, we must list all the users with the read access. This technique has two undesirable consequences:

Constructing such a list may be tedious and unrewarding task, especially if we do not know in advance the list of the users in the system.

Previously, the entry of the any directory is of the fixed size but now it changes to the variable size which results in the complicates space management. These problems can be resolved by use of a condensed version of the access list. To condense the length of the access-control list, many systems recognize three classification of users in connection with each file:

* **Owner –** Owner is the user who has created the file.
* **Group –** A group is a set of members who has similar needs and they are sharing the same file.
* **Universe –** In the system, all other users are under the category called universe.

The most common recent approach is to combine access-control lists with the normal general owner, group, and universe access control scheme. For example: Solaris uses the three categories of access by default but allows access-control lists to be added to specific files and directories when more fine-grained access control is desired.

**Other Protection Approaches:**

The access to any system is also controlled by the password. If the use of password is random and it is changed often, this may be result in limit the effective access to a file.

The use of passwords has a few disadvantages:

* The number of passwords are very large so it is difficult to remember the large passwords.
* If one password is used for all the files, then once it is discovered, all files are accessible; protection is on all-or-none basis.