# OS – Module 5 – Input Output and File Management

## IO Devices

An input/output device, often known as an IO device, is any hardware that allows a human operator or other systems to interface with a computer. Input/output devices, as the name implies, are capable of delivering data (output) to and receiving data from a computer (input). An input/output (I/O) device is a piece of hardware that can take, output, or process data. It receives data as input and provides it to a computer, as well as sends computer data to storage media as a storage output.

## Input Devices

Input devices are the devices that are used to send signals to the computer for performing tasks. The receiver at the end is the CPU (Central Processing Unit), which works to send signals to the output devices. Some of the classifications of Input devices are:

- Keyboard Devices
- Pointing Devices
- Composite Devices
- Game Controller
- Visual Devices
- Audio Input Devices

Some of the input devices are described below.

## Keyboard

The keyboard is the most frequent and widely used input device for entering data into a computer. Although there are some additional keys for performing other operations, the keyboard layout is similar to that of a typical typewriter.
Generally, keyboards come in two sizes: 84 keys or 101/102 keys but currently keyboards with 104 keys or 108 keys are also available for Windows and the Internet.

## Types of Keys

- **Numeric Keys:** It is used to enter numeric data or move the cursor. It usually consists of a set of 17 keys.
- **Typing Keys:** The letter keys (A-Z) and number keys (09) are among these keys.
- **Control Keys:** These keys control the pointer and the screen. There are four directional arrow keys on it. Home, End, Insert, Alternate(Alt), Delete, Control(Ctrl), etc., and Escape are all control keys (Esc).
- **Special Keys:** Enter, Shift, Caps Lock, NumLk, Tab, etc., and Print Screen are among the special function keys on the keyboard.
- **Function Keys:** The 12 keys from F1 to F12 are on the topmost row of the keyboard.

## Mouse

The most common pointing device is the mouse. The mouse is used to move a little cursor across the screen while clicking and dragging. The cursor will stop if you let go of the mouse. The computer is dependent on you to move the mouse; it won't move by itself. As a result, it's an input device.
A mouse is an input device that lets you move the mouse on a flat surface to control the coordinates and movement of the on-screen cursor/pointer.
The left mouse button can be used to select or move items, while the right mouse button when clicked displays extra menus.

**Joystick**

A joystick is a pointing device that is used to move the cursor on a computer screen. A spherical ball is attached to both the bottom and top ends of the stick. In a socket, the lower spherical ball slides. You can move the joystick in all four directions.

The joystick's function is comparable to that of a mouse. It is primarily used in CAD (Computer-Aided Design) and playing video games on the computer.

**Track Ball**

Track Ball is an accessory for notebooks and laptops, which works on behalf of a mouse. It has a similar structure to a mouse. Its structure is like a half-inserted ball and we use fingers for cursor movement. Different shapes are used for this like balls, buttons, or squares.

**Light Pen**

A light pen is a type of pointing device that looks like a pen. It can be used to select a menu item or to draw on the monitor screen. A photocell and an optical system are enclosed in a tiny tube. When the tip of a light pen is moved across a monitor screen while the pen button is pushed, the photocell sensor element identifies the screen location and provides a signal to the CPU.

**Scanner**

A scanner is an input device that functions similarly to a photocopier. It's employed when there's information on paper that needs to be transferred to the computer's hard disc for subsequent manipulation. The scanner collects images from the source and converts them to a digital format that may be saved on a disc. Before they are printed, these images can be modified.

**Web Camera**

Because a web camera records a video image of the scene in front of it, a webcam is an input device. It is either built inside the computer (for example, a laptop) or attached through a USB connection. A webcam is a computer-connected tiny digital video camera. It's also known as a web camera because it can take images and record video. These cameras come with software that must be installed on the computer in order to broadcast video in real-time over the Internet. It can shoot images and HD videos, however, the video quality isn't as good as other cameras (In Mobiles or other devices or normal cameras).

**Microphone**

The microphone works as an input device that receives input voice signals and also has the responsibility of converting it also to digital form. It is a very common device that is present in every device which is related to music.

**Output Devices**

Output Devices are the devices that show us the result after giving the input to a computer system. Output can be of many different forms like image, graphic audio, video, etc. Some of the output devices are described below.

**Monitor**

Monitors, also known as Visual Display Units (VDUs), are a computer's primary output device. It creates images by arranging small dots, known as pixels, in a rectangular pattern. The amount of pixels determines the image's sharpness.
The two kinds of viewing screens used for monitors are described below.

- **Cathode-Ray Tube (CRT) Monitor:** Pixels are minuscule visual elements that make up a CRT display. The higher the image quality or resolution, the smaller the pixels.
- **Flat-Panel Display Monitor:** In comparison to the CRT, a flat-panel display is a type of video display with less volume, weight, and power consumption. They can be hung on the wall or worn on the wrist.

Flat-panel displays are currently used in calculators, video games, monitors, laptop computers, and graphical displays.

**Television**

Television is one of the common output devices which is present in each and every house. It portrays video and audio files on the screen as the user handles the television. Nowadays, we are using plasma displays as compared to CRT screens which we used earlier.

**Printer**

Printers are output devices that allow you to print information on paper. There are certain types of printers which are described below.

- Impact Printers
- Character Printers
- Line Printers
- Non-Impact Printers
- Laser Printers
- Inkjet Printers

**Impact Printer**

Characters are printed on the ribbon, which is subsequently crushed against the paper, in impact printers. The following are the characteristics of impact printers:

- Exceptionally low consumable cost.
- Quite noisy

- Because of its low cost, it is ideal for large-scale printing.
- To create an image, there is physical contact with the paper.

**Character Printers**

Character Printer has the capability to print only one character at a time. It is of two types.

- Dot Matrix Printer
- Daisy Wheel

**Line Printers**

Line Printers are printers that have the capability to print one line at a time. It is of two types.

- Drum Printer
- Chain Printer

**Non-Impact Printers**

Characters are printed without the need for a ribbon in non-impact printers. Because these printers print a full page at a time, they're also known as Page Printers. The following are the characteristics of non-impact printers:

- Faster
- They don't make a lot of noise.
- Excellent quality
- Supports a variety of typefaces and character sizes

**Laser Printers**

Laser Printers use laser lights for producing dots which will produce characters on the page.

**Inkjet Printers**

Inkjet printers are printers that use spray technology for printing papers. High-quality papers are produced in an Inkjet printer. They also do color printing.

**Speakers**

Speakers are devices that produce sound after getting a command from a computer. Nowadays, speakers come with wireless technology also like Bluetooth speakers.

**Projector**

Projectors are optical devices that have the work to show visuals on both types of screens, stationary and moving both. It helps in displaying images on a big screen. Projectors are generally used in theatres, auditoriums, etc.

**Global Positioning System (GPS)**

Global Positioning System helps the user in terms of directions, as it uses satellite technology to track the geometrical locations of the users. With continuous latitudinal and longitudinal calculations, GPS gives accurate results. Nowadays, all smart devices have inbuilt GPS.

**Headphones**

Headphones are just like a speaker, which is generally used by a single person or it is a single-person usable device and is not commonly used in large areas. These are also called headsets having a lower sound frequency.

**The Input and Output Devices of a Computer**

There are so many devices that contain the characteristics of both input and output. They can perform both operations as they receive data and provide results. Some of them are mentioned below.

**USB Drive**

USB Drive is one of the devices which perform both input and output operations as a USB Drive helps in receiving data from a device and sending it to other devices.

**Modem**

Modems are one of the important devices that helps in transmitting data using telephonic lines.

**CD and DVD**

CD and DVD are the most common device that helps in saving data from one computer in a particular format and send data to other devices which works as an input device to the computer.

**Headset**

The headset consists of a speaker and microphone where a speaker is an output device and a microphone works as an input device.

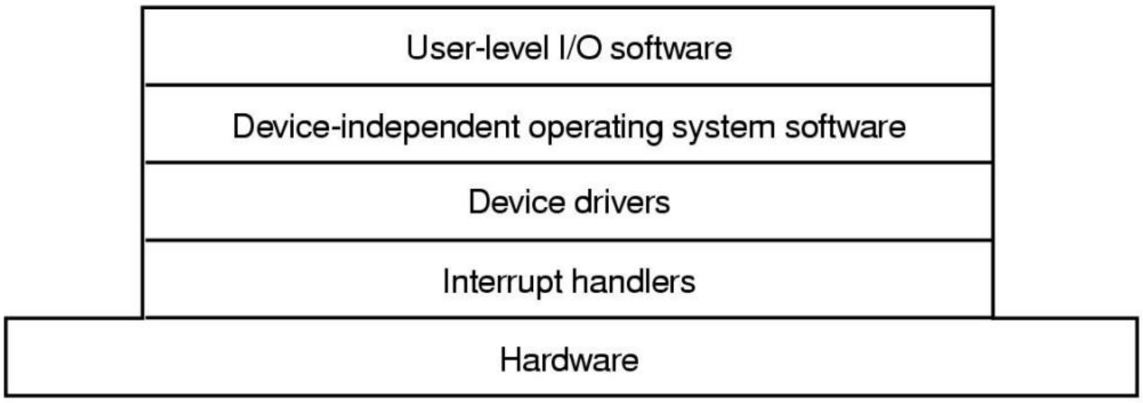
<u>**OS Design Issues for I/O Management**</u>

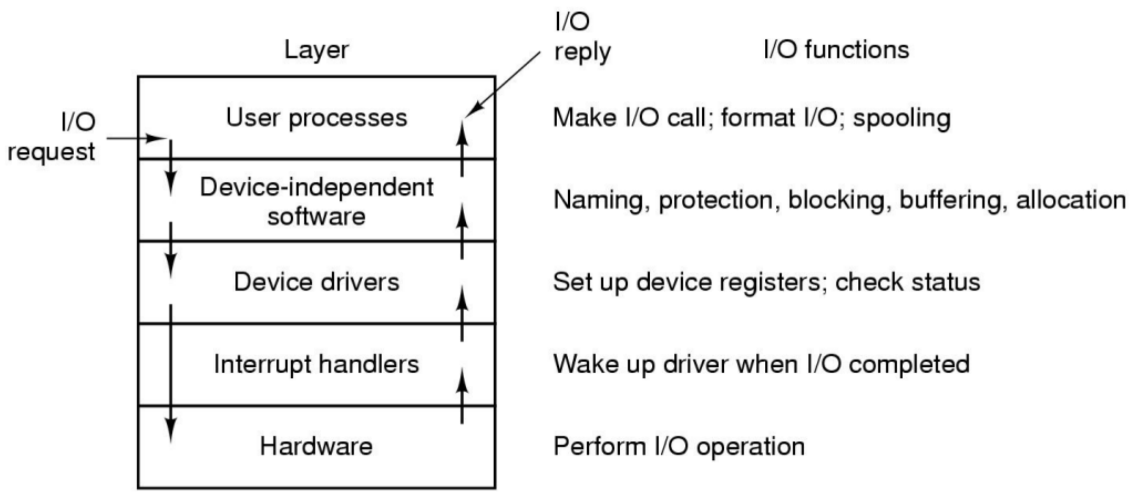
1. **Efficiency**:
   - I/O devices are typically slower than main memory and the CPU, which can cause bottlenecks in system performance.
   - To optimize efficiency, especially in Disk and Network I/O, operating systems often use techniques like **multiprogramming**. This allows one process to be executing while others wait for I/O operations to complete.
   - Additionally, systems may use **swapping** to bring in additional ready processes, increasing the number of I/O operations performed simultaneously.
2. **Generality and Uniformity**:
   - The goal is to handle all I/O devices uniformly, so both the operating system and user applications interact with devices in a standardized way. This can include abstracting device interactions using general functions like read(), write(), open(), and close().
   - However, the diversity of I/O devices (e.g., different access methods, data rates) often means that generality can compromise efficiency. To address this, lower-level routines are used to hide the specific details of device operations.
3. **Interrupt Handling**:

- Efficient I/O management involves handling **interrupts** efficiently, as they notify the CPU about I/O events. Properly managing interrupts is crucial to avoid concurrency issues within the kernel and applications.
- Interrupt handlers are generally designed to unblock drivers waiting for I/O completion and often operate in the context of the currently running process to avoid costly context switches.

4. **Device Drivers**:
    - Drivers translate higher-level I/O requests into device-specific operations, managing hardware directly. They need to be **reentrant** to handle multiple concurrent requests.
    - Modern systems often use dynamically loaded drivers (e.g., Linux modules) to support a wide range of hardware without recompiling the kernel.
5. **Buffering Techniques**:
    - Buffering is used to match the speed difference between devices and system memory. Several strategies include:
        - **Single and Double Buffering**: Uses kernel buffers to allow overlap between I/O operations and processing, improving efficiency.
        - **Circular Buffering**: Employs multiple buffers in a circular manner, which is useful for high-speed data streams to prevent data loss.

**I/O Software Layers**




I/O reply
Layer
I/O functions
I/O request
User processes
Make I/O call; format I/O; spooling
Device-independent software
Naming, protection, blocking, buffering, allocation
Device drivers
Set up device registers; check status
Interrupt handlers
Wake up driver when I/O completed
Hardware
Perform I/O operation

**<u>I/O Buffering</u>**

I/O buffering is a technique used in computer systems to improve the efficiency of input and output (I/O) operations. It involves the temporary storage of data in a buffer, which is a reserved area of memory, to reduce the number of I/O operations and manage the flow of data between fast and slow devices or processes.

**Types of I/O Buffering Techniques**

**1. Single Buffer**

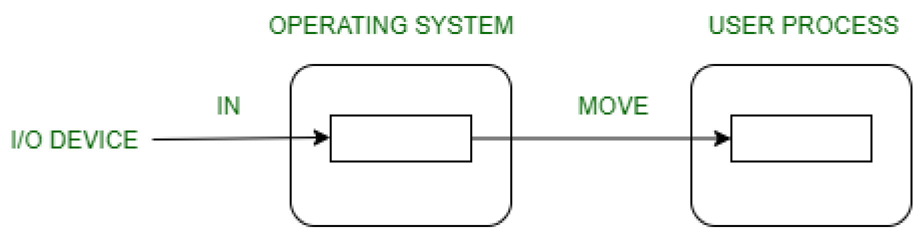
Using one buffer to store data temporarily. A buffer is provided by the operating system to the system portion of the main memory.

**Block Oriented Device**

- The system buffer takes the input.
- After taking the input, the block gets transferred to the user space by the process and then the process requests for another block.
- Two blocks work simultaneously, when one block of data is processed by the user process, the next block is being read in.
- OS can swap the processes.
- OS can record the data of the system buffer to user processes.

**Stream Oriented Device**

- Line- at a time operation is used for scroll-made terminals. The user inputs one line at a time, with a carriage return signaling at the end of a line.
- Byte-at-a-time operation is used on forms mode, terminals when each keystroke is significant.



SINGLE I/O BUFFERING

**2. Double Buffer**

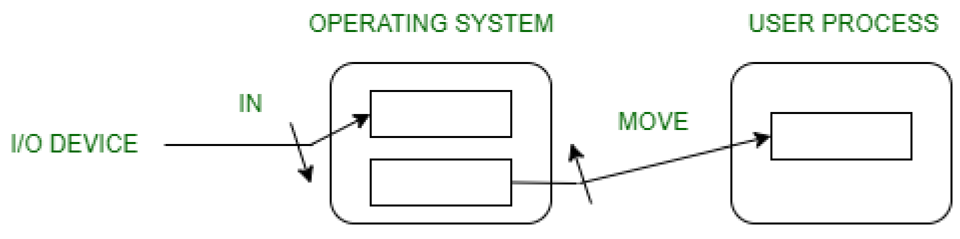
In this technique the operating system Uses two buffers to allow continuous data transfer between two process or two devices.

**Block Oriented**

- There are two buffers in the system.
- One buffer is used by the driver or controller to store data while waiting for it to be taken by higher level of the hierarchy.
- Other buffer is used to store data from the lower level module.
- Double buffering is also known as buffer swapping.
- A major disadvantage of double buffering is that the complexity of the process get increased.
- If the process performs rapid bursts of I/O, then using double buffering may be deficient.

**Stream Oriented**

- Line- at a time I/O, the user process need not be suspended for input or output, unless process runs ahead of the double buffer.
- Byte- at a time operations, double buffer offers no advantage over a single buffer of twice the length.



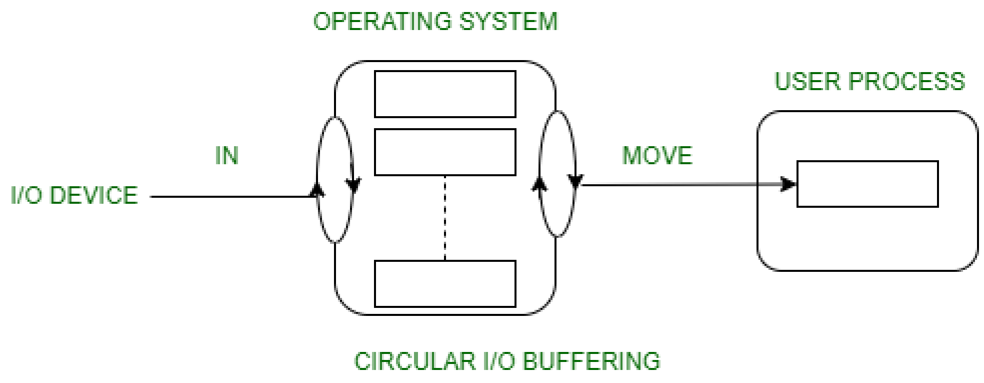
DOUBLE I/O BUFFERING

**3. Circular Buffer**

In this technique the operating system Uses a circular buffer to manage continuous data streams efficiently.

- When more than two buffers are used, the collection of buffers is itself referred to as a circular buffer.
- In this, the data do not directly passed from the producer to the consumer because the data would change due to overwriting of buffers before they had been consumed.

- The producer can only fill up to buffer i-1 while data in buffer i is waiting to be consumed.



<u>**File Access Methods**</u>

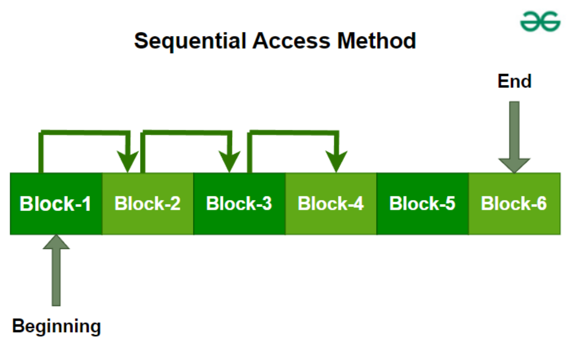
There are three ways to access a file in a computer system:

- 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, the 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*- reads the next position of the file and automatically advances a file pointer, which keeps track of the I/O location. Similarly, for the -write *next*- append to the end of the file and advance to the newly written material.



*Sequential Access Method*

**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.

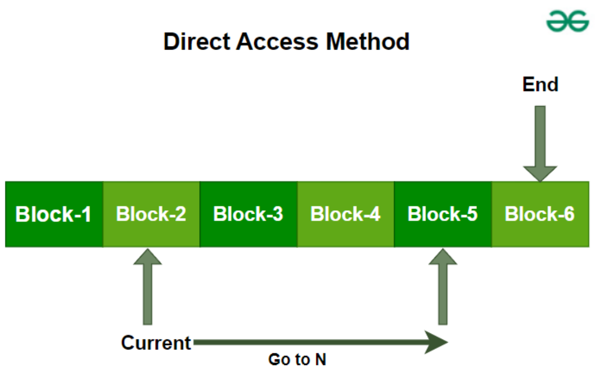
- It is a more efficient method for reading large files, as it only reads the required data and does not waste time reading unnecessary data..

**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 Method**

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.



*Direct Access Method*

**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.

**Disadvantages of Direct Access Method**

- **Complex Implementation** : Implementing direct access can be complex, requiring sophisticated algorithms and data structures to manage and locate records efficiently.
- **Higher Storage Overhead** : Direct access methods often require additional storage for maintaining data location information (such as pointers or address tables), which can increase the overall storage requirements.

**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 Related to Index Sequential Method**

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

**Advantages of Index Sequential Method**

- **Efficient Searching** : Index sequential method allows for quick searches through the index.
- **Balanced Performance** : It combines the simplicity of sequential access with the speed of direct access, offering a balanced approach that can handle various types of data access needs efficiently.
- **Flexibility** : This method allows both sequential and random access to data, making it versatile for different types of applications, such as batch processing and real-time querying.
- **Improved Data Management** : Indexing helps in better organization and management of data. It makes data retrieval faster and more efficient, especially in large databases.
- **Reduced Access Time** : By using an index to directly locate data blocks, the time spent searching for data within large datasets is significantly reduced.

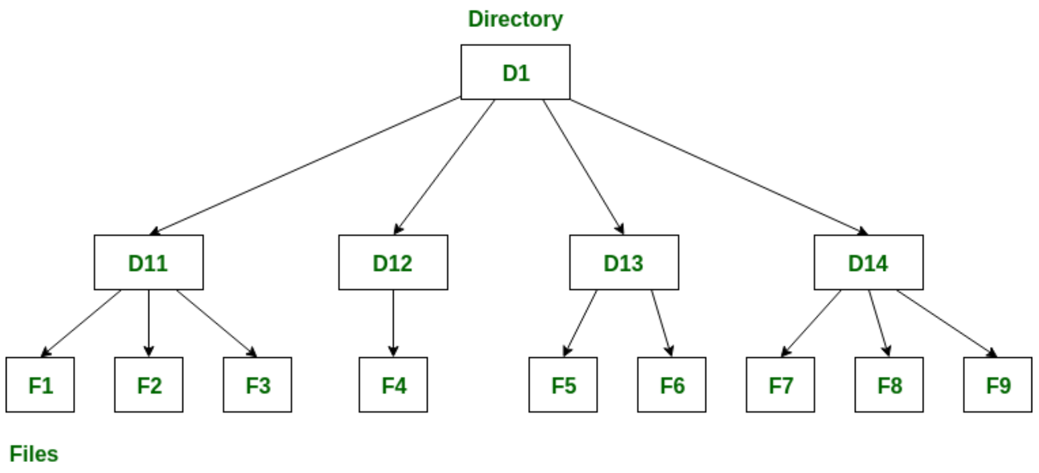
**Disadvantages of Index Sequential Method**

- **Complex Implementation** : The index sequential method is more complex to implement and maintain compared to simple sequential access methods.
- **Additional Storage** : Indexes require additional storage space, which can be significant for large datasets. This extra space can sometimes offset the benefits of faster access.
- **Update Overhead** : Updating the data can be more time-consuming because both the data and the indexes need to be updated. This can lead to increased processing time for insertions, deletions, and modifications.
- **Index Maintenance** : Keeping the index up to date requires regular maintenance, especially in dynamic environments where data changes frequently. This can add to the system’s overhead.

## <u>File Directories</u>

A **directory** is a container that is used to contain folders and files. It organizes files and folders in a hierarchical manner. In other words, directories are like folders that help organize files on a computer. Just like you use folders to keep your papers and documents in order, the operating system uses directories to keep track of files and where they are stored. 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. Following are the logical structures of a directory, each providing a solution to the problem faced in the previous type of directory structure.



### Different Types of Directory in OS

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

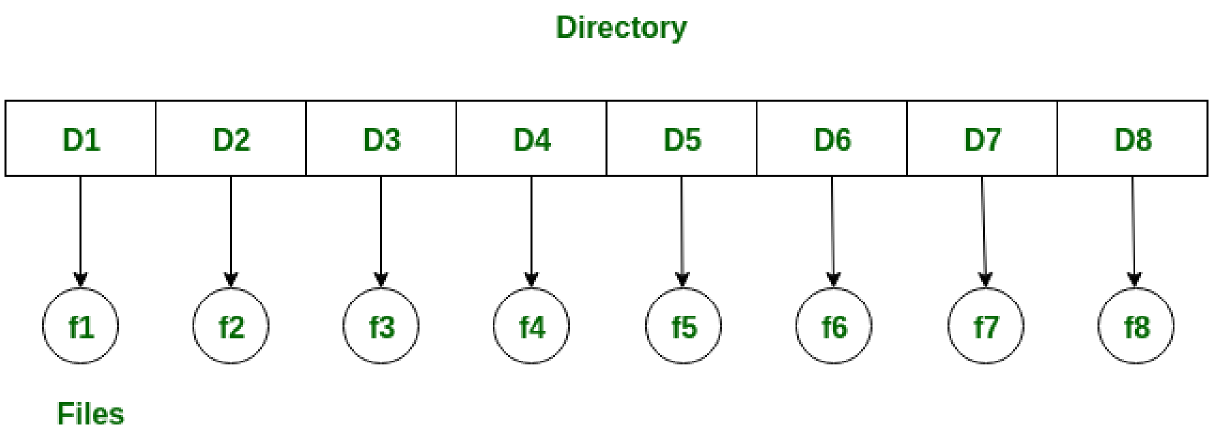
Each type of directory has its own way of arranging files and directories, offering unique benefits and features. These are:

- Single-Level Directory
- Two-Level Directory
- Tree Structure/ Hierarchical Structure
- Acyclic Graph Structure
- General-Graph 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.

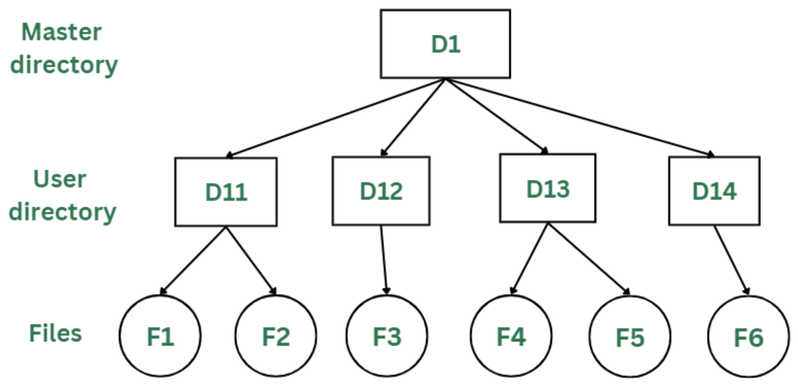
**Disadvantages**

- 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**

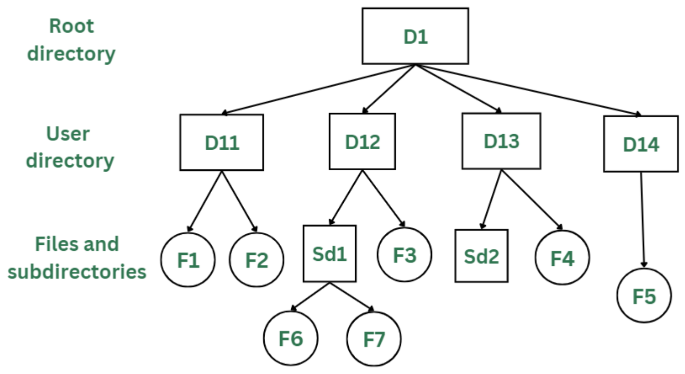
- As there is advantage of security, there is also disadvantage that the user cannot share the file with the other users.
- Unlike the advantage users can create their own files, users don't have the ability to create subdirectories.
- Scalability is not possible because one user can't group the same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

**3) Tree Structure/ Hierarchical Structure**

Tree directory structure of operating system is most commonly used in our **personal computers**. 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**

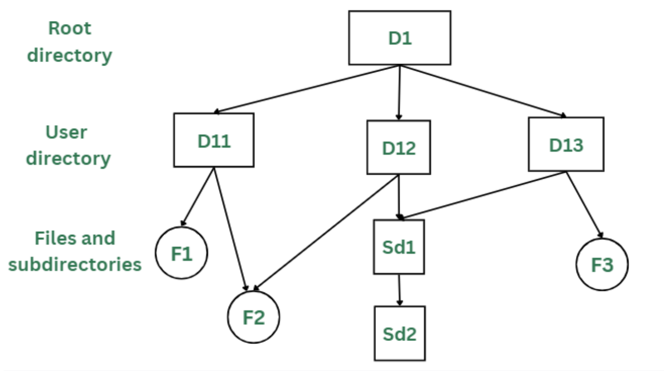
- As the user isn't allowed to access other user's directory, this prevents the file sharing among users.
- As the user has the capability to make subdirectories, if the number of subdirectories increase the searching may become complicated.
- Users cannot modify the root directory data.
- If files do 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**

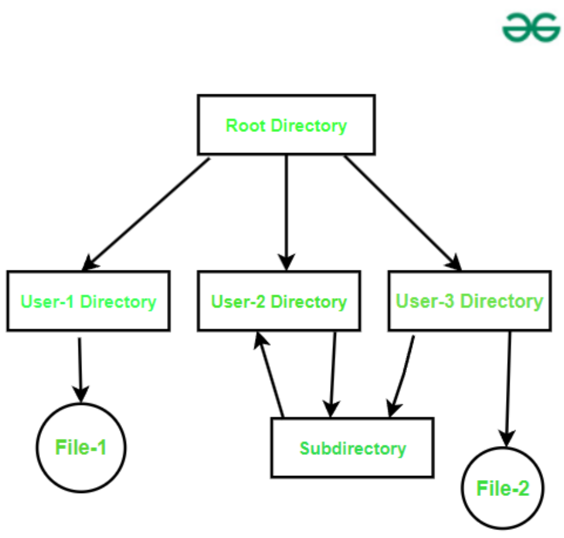
- Sharing of files and directories is allowed between multiple users.
- Searching becomes too easy.
- Flexibility is increased as file sharing and editing access is there for multiple users.

**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.

**5) General-Graph Directory Structure**

Unlike the acyclic-graph directory, which avoids loops, the general-graph directory can have cycles, meaning a directory can contain paths that loop back to the starting point. This can make navigating and managing files more complex.



*General Graph Directory Structure*

In the above image, you can see that a cycle is formed in the User 2 directory. While this structure offers more flexibility, it is also more complicated to implement.

**Advantages of General-Graph Directory**

- More flexible than other directory structures.
- Allows cycles, meaning directories can loop back to each other.

**Disadvantages of General-Graph Directory**

- More expensive to implement compared to other solutions.
- Requires garbage collection to manage and clean up unused files and directories.

<u>**FILE SHARING**</u>

File Sharing in an Operating System(OS) denotes how information and files are shared between different users, computers, or devices on a network; and files are units of data that are stored in a computer in the form of documents/images/videos or any others types of information needed.

**For Example:** Suppose letting your computer talk to another computer and exchange pictures, documents, or any useful data. This is generally useful when one wants to work on a project with others, send files to friends, or simply shift stuff to another device. Our OS provides ways to do this like email attachments, cloud services, etc. to make the sharing process easier and more secure.

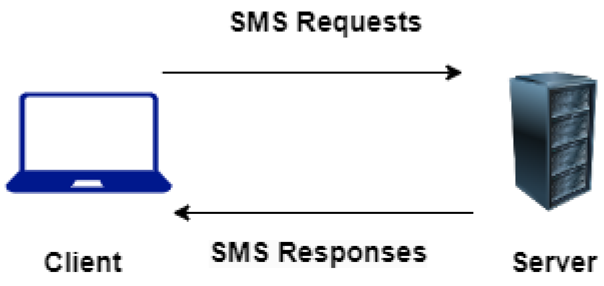
**Various Ways to Achieve File Sharing**

Let's see the various ways through which we can achieve file sharing in an OS.

**1. Server Message Block (SMB)**

SMB is like a network based file sharing protocol mainly used in windows operating systems. It allows our computer to share files/printer on a network. SMB is now the standard way for seamless file transfer method and printer sharing.

**Example:** Imagine in a company where the employees have to share the files on a particular project . Here SMB is employed to share files among all the windows based operating system.orate on projects. SMB/CIFS is employed to share files between Windows-based computers. Users can access shared folders on a server, create, modify, and delete files.

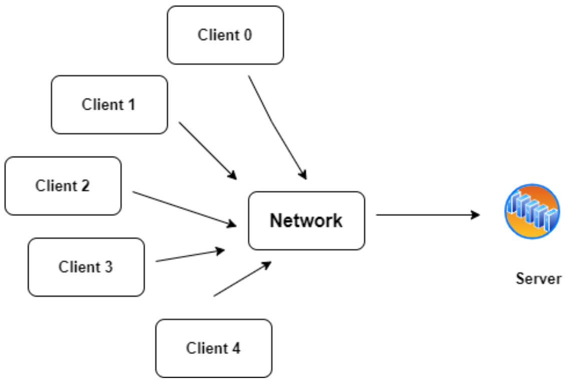


SMB File Sharing

**2. Network File System (NFS)**

NFS is a distributed based file sharing protocol mainly used in Linux/Unix based operating System. It allows a computer to share files over a network as if they were based on local. It provides a efficient way of transfer of files between servers and clients.

**Example:** Many Programmer/Universities/Research Institution uses Unix/Linux based Operating System. The Institutes puts up a global server datasets using NFS. The Researchers and students can access these shared directories and everyone can collaborate on it.

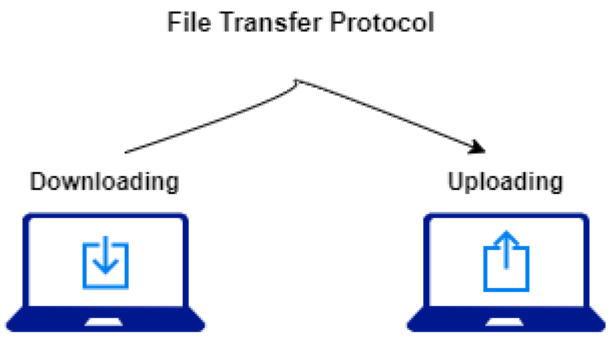


NFS File Sharing

**3. File Transfer Protocol (FTP)**

It is the most common standard protocol for transferring of the files between a client and a server on a computer network. FTPs supports both uploading and downloading of the files, here we can download,upload and transfer of files from Computer A to Computer B over the internet or between computer systems.

**Example:** Suppose the developer makes changes on the server. Using the FTP protocol, the developer connects to the server they can update the server with new website content and updates the existing file over there.



**4. Cloud-Based File Sharing**

It involves the famous ways of using online services like Google Drive, DropBox , One Drive ,etc. Any user can store files over these cloud services and they can share that with others, and providing access from many users. It includes collaboration in realtime file sharing and version control access.

Ex: Several students working on a project and they can use Google Drive to store and share for that purpose. They can access the files from any computer or mobile devices and they can make changes in realtime and track the changes over there.

## FILE ALLOCATION

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.

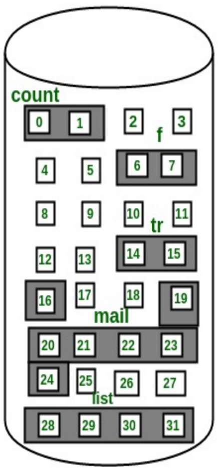
All the three methods have their own advantages and disadvantages as discussed below:

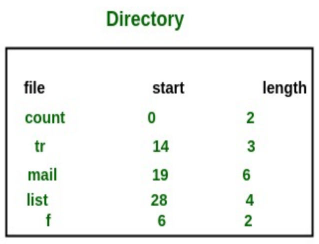
### 1. Contiguous Allocation

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.

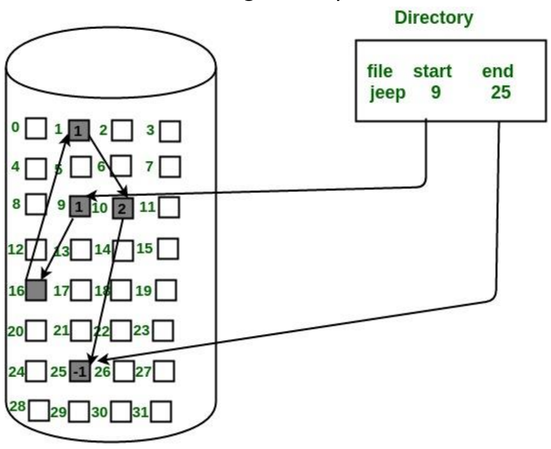
**Disadvantages:**

- 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.

**2. 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:**

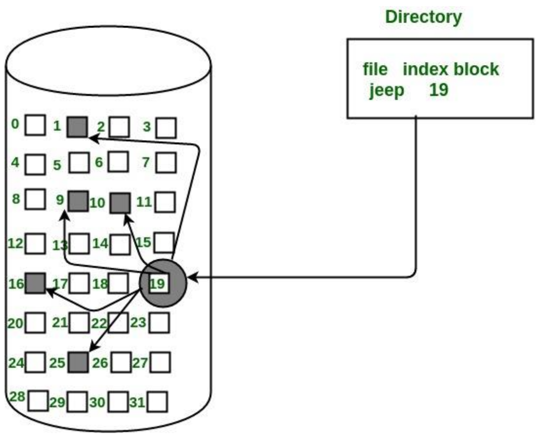
- This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
- This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.

**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.

**3. 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 ith entry in the index block contains the disk address of the ith 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.

## FREE SPACE MANAGEMENT

Free space management is a critical aspect of operating systems as it involves managing the available storage space on the hard disk or other secondary storage devices. The operating system uses various techniques to manage free space and optimize the use of storage devices. Here are some of the commonly used free space management techniques:

### Free Space Management Techniques

- **Linked Allocation:** In this technique, each file is represented by a linked list of disk blocks. When a file is created, the operating system finds enough free space on the disk and links the blocks of the file to form a chain. This method is simple to implement but can lead to fragmentation and waste of space.
- **Contiguous Allocation:** In this technique, each file is stored as a contiguous block of disk space. When a file is created, the operating system finds a contiguous block of free space and assigns it to the file. This method is efficient as it minimizes fragmentation but suffers from the problem of external fragmentation.
- **Indexed Allocation:** In this technique, a separate index block is used to store the addresses of all the disk blocks that make up a file. When a file is created, the operating system creates an index block and stores the addresses of all the blocks in the file. This method is efficient in terms of storage space and minimizes fragmentation.
- **File Allocation Table (FAT):** In this technique, the operating system uses a file allocation table to keep track of the location of each file on the disk. When a file is created, the operating

system updates the file allocation table with the address of the disk blocks that make up the file. This method is widely used in Microsoft Windows operating systems.

- **Volume Shadow Copy:** This is a technology used in Microsoft Windows operating systems to create backup copies of files or entire volumes. When a file is modified, the operating system creates a shadow copy of the file and stores it in a separate location. This method is useful for data recovery and protection against accidental file deletion.

**Windows File Systems: FAT, FAT32, NTFS, and ReFS**

Windows operating systems support several file systems, each designed to optimize storage, access, and data management. The most common file systems are **FAT**, **FAT32**, **NTFS**, and **ReFS**. Here's a brief overview of each:

---

### 1. FAT (File Allocation Table)

- **Introduced:** 1977, initially for floppy disks.
- **Structure:** Simple and lightweight, making it suitable for low-storage devices.
- **Max File Size:** 4 GB
- **Max Partition Size:** 2 GB
- **Features:**
  - Lacks modern features like security permissions and file encryption.
  - Uses a table to keep track of file locations.
  - Commonly used in older systems, BIOS firmware, and embedded systems.
- **Use Cases:**
  - Ideal for legacy systems, small storage devices like floppy disks, and bootable media.

---

### 2. FAT32 (File Allocation Table 32)

- **Introduced:** 1996 with Windows 95 OSR2.
- **Structure:** An improved version of FAT with a 32-bit allocation table.
- **Max File Size:** 4 GB
- **Max Partition Size:** 2 TB
- **Features:**
  - Widely supported across various platforms (Windows, macOS, Linux).
  - No built-in security features (e.g., file permissions).
  - Efficient for smaller drives due to low overhead.
- **Use Cases:**

    - Commonly used for USB flash drives, SD cards, and portable storage devices.
    - Suitable for cross-platform file sharing due to wide compatibility.

---

**3. NTFS (New Technology File System)**

- **Introduced:** 1993 with Windows NT 3.1.
- **Structure:** A robust and feature-rich file system designed for modern storage needs.
- **Max File Size:** 16 TB (theoretical limit of up to 16 EB with newer versions)
- **Max Partition Size:** 256 TB (practical limit on Windows systems)
- **Features:**
    - Supports advanced features like file compression, encryption (EFS), disk quotas, and file permissions (ACLs).
    - Uses a **Master File Table (MFT)** to store file metadata.
    - Provides fault tolerance with journaling, reducing data corruption risks.
- **Use Cases:**
    - The default file system for modern Windows operating systems (e.g., Windows 10, 11).
    - Ideal for internal hard drives, SSDs, and external drives requiring security and reliability.

---

**4. ReFS (Resilient File System)**

- **Introduced:** 2012 with Windows Server 2012.
- **Structure:** Built for high data integrity, scalability, and resilience.
- **Max File Size:** 35 PB (Petabytes)
- **Max Partition Size:** 4.7 YB (Yottabytes)
- **Features:**
    - Focuses on data integrity with features like **automatic data corruption detection and repair**.
    - Supports **Storage Spaces**, offering advanced redundancy and fault tolerance.
    - Optimized for large-scale data storage (e.g., virtual machines, data centers).
    - Does not support some NTFS features like file compression or encryption.
- **Use Cases:**
    - Primarily used in server environments, data centers, and storage pools requiring high reliability and performance.

- Suitable for virtualized environments, large-scale file systems, and data archiving.

---

**Summary Comparison**

| Feature | FAT | FAT32 | NTFS | ReFS |
|---|---|---|---|---|
| **Max File Size** | 4 GB | 4 GB | 16 TB (up to 16 EB) | 35 PB |
| **Max Partition** | 2 GB | 2 TB | 256 TB | 4.7 YB |
| **Security** | No | No | Yes (ACL, EFS) | Limited (focus on integrity) |
| **Journaling** | No | No | Yes | Yes |
| **Compatibility** | Legacy | Cross-platform | Windows (default) | Windows Server |
| **Use Cases** | Floppy disks | USB drives | Internal drives, OS | Servers, data centers |

## Linux File Systems: ext2, ext3, ext4, ReiserFS, XFS, and JFS

Linux supports a variety of file systems, each optimized for different performance, reliability, and scalability needs. Here's an overview of some of the most popular Linux file systems, including **ext2**, **ext3**, **ext4**, **ReiserFS**, **XFS**, and **JFS**:

---

### 1. ext2 (Second Extended File System)

- **Introduced:** 1993
- **Journaling:** No
- **Max File Size:** 2 TB
- **Max Partition Size:** 32 TB
- **Features:**
    - Known for its simplicity and efficiency.
    - Lacks a journaling feature, making it less resilient to sudden power failures but faster in write operations.
    - Still used in some embedded systems and USB drives due to low overhead.
- **Use Cases:**
    - Suitable for systems where journaling is unnecessary, like portable storage.

---

### 2. ext3 (Third Extended File System)

- **Introduced:** 2001
- **Journaling:** Yes

- **Max File Size:** 2 TB
- **Max Partition Size:** 32 TB
- **Features:**
  - Extends ext2 with journaling, improving data integrity by tracking changes before committing them.
  - Offers three journaling modes: **Writeback**, **Ordered**, and **Journal**, balancing between performance and reliability.
  - Backward-compatible with ext2, allowing easy upgrades.
- **Use Cases:**
  - Ideal for general-purpose Linux systems requiring a balance of performance and reliability.

---

**3. ext4 (Fourth Extended File System)**

- **Introduced:** 2008
- **Journaling:** Yes
- **Max File Size:** 16 TB
- **Max Partition Size:** 1 EB (Exabyte)
- **Features:**
  - Enhanced version of ext3 with support for larger volumes and files.
  - Supports **extents**, which improve performance by reducing fragmentation.
  - Includes features like **delayed allocation**, **multiblock allocation**, and **fast fsck**, making it more efficient.
  - Backward compatible with ext3 and ext2.
- **Use Cases:**
  - The default file system for many Linux distributions (e.g., Ubuntu, Debian).
  - Suitable for desktops, laptops, and servers requiring robust performance.

---

**4. ReiserFS**

- **Introduced:** 2001
- **Journaling:** Yes
- **Max File Size:** 8 TB
- **Max Partition Size:** 16 TB

- **Features:**
    - Known for efficient small file storage and fast write speeds.
    - Uses a balanced tree structure for improved performance in handling large numbers of small files.
    - Lacks active development since 2008, reducing its popularity.
- **Use Cases:**
    - Best suited for systems that handle many small files, like mail servers.

---

**5. XFS**

- **Introduced:** 1994 by SGI (Silicon Graphics, Inc.)
- **Journaling:** Yes (metadata journaling)
- **Max File Size:** 8 EB (Exabytes)
- **Max Partition Size:** 8 EB
- **Features:**
    - High-performance, 64-bit file system designed for scalability.
    - Supports **allocation groups**, allowing parallel I/O and making it suitable for multi-threaded environments.
    - Excellent for large files and high-throughput environments, like media streaming or databases.
- **Use Cases:**
    - Commonly used in enterprise environments, data centers, and high-performance servers (e.g., CentOS/RHEL defaults to XFS).

---

**6. JFS (Journaled File System)**

- **Introduced:** 1990 by IBM
- **Journaling:** Yes
- **Max File Size:** 4 PB (Petabytes)
- **Max Partition Size:** 32 PB
- **Features:**
    - Designed for stability and performance with minimal CPU usage.
    - Uses **dynamic inode allocation**, reducing the space used by inodes.
    - Quick file system checks, making it ideal for large partitions.

- **Use Cases:**
    - Suitable for systems needing efficient handling of large files, such as archival and backup servers.

---

**Summary Comparison**

| File System | Journaling | Max File Size | Max Partition Size | Notable Features | Use Cases |
|---|---|---|---|---|---|
| **ext2** | No | 2 TB | 32 TB | Simple, efficient | USB drives, embedded systems |
| **ext3** | Yes | 2 TB | 32 TB | Extends ext2 with journaling | General-purpose Linux systems |
| **ext4** | Yes | 16 TB | 1 EB | Extents, delayed allocation | Default for many Linux distros |
| **ReiserFS** | Yes | 8 TB | 16 TB | Efficient for small files | Systems with many small files |
| **XFS** | Yes | 8 EB | 8 EB | High performance, allocation groups | Enterprise servers, data centers |
| **JFS** | Yes | 4 PB | 32 PB | Stability, low CPU usage | Archival, large storage servers |