In computers, an address is used to identify a location in the [**computer memory**](https://www.tutorialspoint.com/computer_fundamentals/computer_memory.htm). In [**operating systems**](https://www.tutorialspoint.com/operating_system/index.htm), there are two types of addresses, namely,**logical address** and **physical address**. A logical address is the virtual address that is generated by the [**CPU**](https://www.tutorialspoint.com/computer_fundamentals/computer_cpu.htm). A user can view the logical address of a computer program. On the other hand, a physical address is one that represents a location in the computer memory. A user cannot view the physical address of a program.

Read this article to find out more about logical and physical address and how they are different from each other.

What is a Logical Address?

The**logical address** is a virtual address created by the CPU of the computer system. The logical address of a program is generated when the program is running. A group of several logical address is referred to a**logical address space**. The logical address is basically used as a reference to access the physical memory locations.

In computer systems, a hardware device named **memory management unit**(MMU) is used to map the logical address to its corresponding physical address. However, the logical address of a program is visible to the computer user.

What is a Physical Address?

The **physical address**of a computer program is one that represents a location in the memory unit of the computer. The physical address is not visible to the computer user. The MMU of the system generates the physical address for the corresponding logical address.

The physical address is accessed through the corresponding logical address because a user cannot directly access the physical address. For running a computer program, it requires a physical memory space. Therefore, the logical address has to be mapped with the physical address before the execution of the program.

Difference between Logical and Physical Address in Operating System

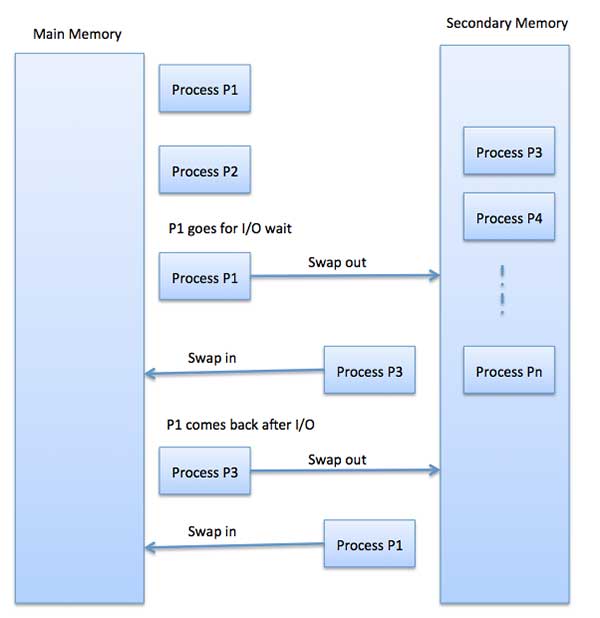
The following table highlights all the major differences between logical and physical address in operating system −

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Logical Address** | **Physical Address** |
| **1.** | This address is generated by the CPU. | This address is a location in the memory unit. |
| **2.** | The address space consists of the set of all logical addresses. | This address is a set of all physical addresses that are mapped to the corresponding logical addresses. |
| **3.** | These addresses are generated by CPU with reference to a specific program. | It is computed using Memory Management Unit (MMU). |
| **4.** | The user has the ability to view the logical address of a program. | The user can’t view the physical address of program directly. |
| **5.** | The user can use the logical address in order to access the physical address. | The user can indirectly access the physical address. |

**Swapping**

Swapping is a mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. At some later time, the system swaps back the process from the secondary storage to main memory.

Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason **Swapping is also known as a technique for memory compaction**.



The total time taken by swapping process includes the time it takes to move the entire process to a secondary disk and then to copy the process back to memory, as well as the time the process takes to regain main memory.

Let us assume that the user process is of size 2048KB and on a standard hard disk where swapping will take place has a data transfer rate around 1 MB per second. The actual transfer of the 1000K process to or from memory will take

2048KB / 1024KB per second

= 2 seconds

= 2000 milliseconds

Now considering in and out time, it will take complete 4000 milliseconds plus other overhead where the process competes to regain main memory.

**Contiguous memory allocation**

Contiguous memory allocation is a memory management technique used by operating systems to allocate memory to processes in contiguous blocks. In this technique, a process is allocated a single block of memory that is contiguous or adjacent to each other. This ensures that memory is efficiently utilized, with minimal fragmentation and wasted memory. Contiguous memory allocation is a widely used technique in modern operating systems and has several advantages, including efficient memory utilization, fast access to memory, and simple management. However, it also has some limitations, such as the possibility of external fragmentation, large block requirements, and fixed block size limitations. Despite its limitations, contiguous memory allocation remains an important and widely used technique in modern operating systems.

## Definition of contiguous memory allocation

Continuous memory allocation is a memory management technique in which memory is allocated to processes in contiguous blocks. This ensures that memory is utilized efficiently, with minimal fragmentation and wasted memory. The technique simplifies memory management by allowing the operating system to manage memory in larger blocks, leading to faster access to memory and improved system performance. Overall, continuous memory allocation is an important technique used by operating systems to efficiently manage memory resources and ensure that memory is effectively utilized.

### **Purpose of memory management techniques**

The purpose of contiguous memory allocation in operating systems is to efficiently manage the memory resources available on a computer system. By allocating memory to processes in contiguous blocks, this technique ensures that the memory is utilized efficiently, with minimal fragmentation and wasted memory. This results in faster access to memory, as contiguous blocks of memory can be accessed more quickly than non-contiguous blocks.

Additionally, contiguous memory allocation simplifies memory management by allowing the operating system to manage memory in larger blocks rather than small, fragmented pieces. This reduces the overhead associated with managing multiple memory blocks, leading to faster system performance.

Overall, the purpose of contiguous memory allocation is to ensure that memory is efficiently utilized and effectively managed, resulting in improved system performance and faster access to memory. This is particularly important for applications with high memory requirements, such as multimedia editing and gaming, and for systems with a small number of processes that require fast access to memory.

## Characteristics of Contiguous Memory Allocation

Contiguous memory allocation is a memory management technique that divides memory into contiguous blocks, where each block is assigned to a single process. Here are some of the characteristics of contiguous memory allocation −

Contiguous blocks: Memory is divided into contiguous blocks, with each block being assigned to a single process. This means that the memory allocated to a process is a single, contiguous block of memory.

* **Efficient use of memory** − Contiguous memory allocation is efficient in terms of memory utilization, as there is no internal fragmentation within a process's allocated memory block.
* **Easy to manage** − Contiguous memory allocation is easy to manage, as the operating system can quickly allocate and deallocate memory to processes by assigning contiguous blocks.
* **External fragmentation** − One of the main drawbacks of contiguous memory allocation is external fragmentation, which occurs when small gaps of free memory are scattered throughout the memory space. Over time, these small gaps can accumulate and result in larger portions of memory becoming unusable, even though the total amount of free memory may be sufficient to satisfy a request for memory.
* **Compaction** − To address external fragmentation, operating systems may use techniques such as compaction, where the operating system rearranges memory blocks to eliminate gaps and consolidate free memory into larger contiguous blocks.

Overall, contiguous memory allocation is an efficient and easy-to-manage memory management technique, but it may suffer from external fragmentation over time. Operating systems may use various techniques to reduce external fragmentation and ensure that memory is utilized effectively.

### **Advantages of Contiguous Memory Allocation**

* **Efficient memory utilization** − Contiguous memory allocation is efficient in terms of memory utilization, as there is no internal fragmentation within a process's allocated memory block.
* **Simple and easy to manage** − This technique is simple and easy to manage, as the operating system can quickly allocate and deallocate memory to processes by assigning contiguous blocks.
* **Fast access** − Since the memory is allocated in contiguous blocks, access to the memory is faster than other memory management techniques.

### **Disadvantages of Contiguous Memory Allocation**

* **External Fragmentation** − One of the main disadvantages of contiguous memory allocation is external fragmentation, which occurs when small gaps of free memory are scattered throughout the memory space.
* **Limited memory capacity** − Contiguous memory allocation is limited by the size of the memory blocks available on the system, which may limit the total amount of memory that can be allocated to a process.
* **Difficulty in sharing memory** − This technique makes it difficult to share memory between multiple processes, as each process is assigned a contiguous block of memory that cannot be shared with other processes.
* **Lack of flexibility** − Contiguous memory allocation lacks flexibility in allocating and deallocating memory, as the operating system can only allocate memory in contiguous blocks.

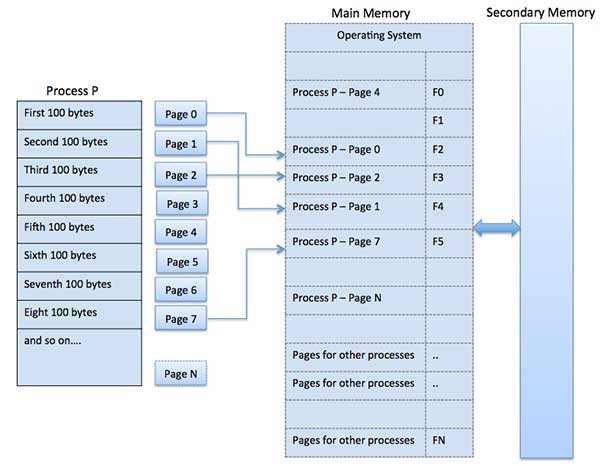
In summary, while contiguous memory allocation has advantages such as efficient memory utilization and simple management, it also has disadvantages such as external fragmentation and a lack of flexibility in allocating memory. As a result, operating systems must carefully consider the needs of the processes and system requirements when selecting memory management techniques.

## Paging

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard that's set up to emulate the computer's RAM. Paging technique plays an important role in implementing virtual memory.

Paging is a memory management technique in which process address space is broken into blocks of the same size called **pages** (size is power of 2, between 512 bytes and 8192 bytes). The size of the process is measured in the number of pages.

Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called **frames** and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.



### **Address Translation**

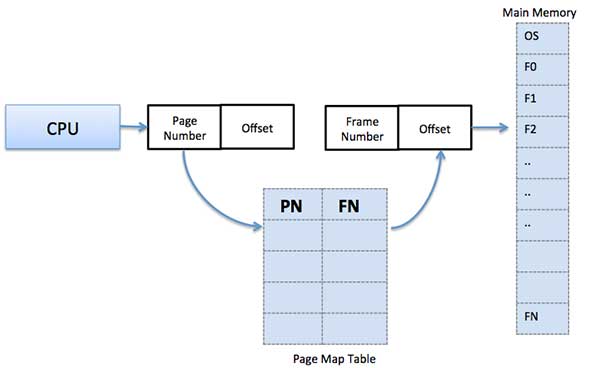
Page address is called **logical address** and represented by **page number** and the **offset**.

Logical Address = Page number + page offset

Frame address is called **physical address** and represented by a **frame number** and the **offset**.

Physical Address = Frame number + page offset

A data structure called **page map table** is used to keep track of the relation between a page of a process to a frame in physical memory.



When the system allocates a frame to any page, it translates this logical address into a physical address and create entry into the page table to be used throughout execution of the program.

When a process is to be executed, its corresponding pages are loaded into any available memory frames. Suppose you have a program of 8Kb but your memory can accommodate only 5Kb at a given point in time, then the paging concept will come into picture. When a computer runs out of RAM, the operating system (OS) will move idle or unwanted pages of memory to secondary memory to free up RAM for other processes and brings them back when needed by the program.

This process continues during the whole execution of the program where the OS keeps removing idle pages from the main memory and write them onto the secondary memory and bring them back when required by the program.

### **Advantages and Disadvantages of Paging**

Here is a list of advantages and disadvantages of paging −

* Paging reduces external fragmentation, but still suffer from internal fragmentation.
* Paging is simple to implement and assumed as an efficient memory management technique.
* Due to equal size of the pages and frames, swapping becomes very easy.
* Page table requires extra memory space, so may not be good for a system having small RAM.

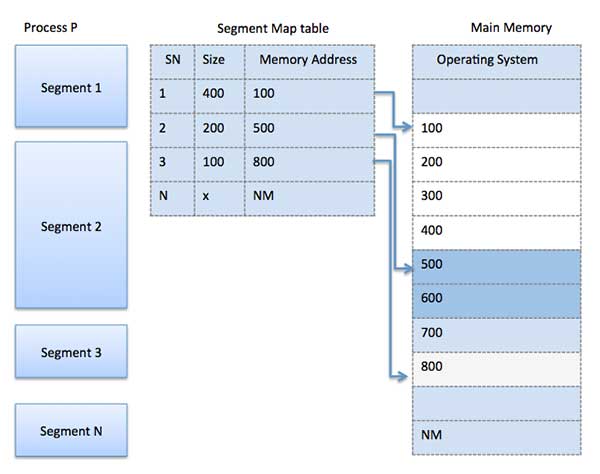
## Segmentation

Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.

When a process is to be executed, its corresponding segmentation are loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory.

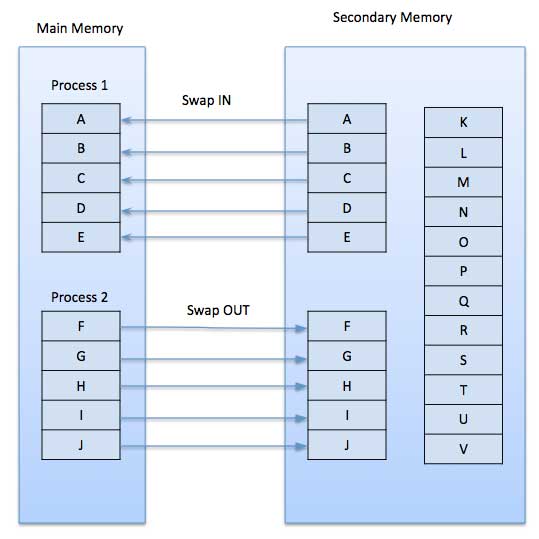
Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.

A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a **segment map table** for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.



## Demand Paging

A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance. When a context switch occurs, the operating system does not copy any of the old program’s pages out to the disk or any of the new program’s pages into the main memory Instead, it just begins executing the new program after loading the first page and fetches that program’s pages as they are referenced.



While executing a program, if the program references a page which is not available in the main memory because it was swapped out a little ago, the processor treats this invalid memory reference as a **page fault** and transfers control from the program to the operating system to demand the page back into the memory.

### **Advantages**

Following are the advantages of Demand Paging −

* Large virtual memory.
* More efficient use of memory.
* There is no limit on degree of multiprogramming.

### **Disadvantages**

* Number of tables and the amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.

## Page Replacement Algorithm

Page replacement algorithms are the techniques using which an Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated. Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.

When the page that was selected for replacement and was paged out, is referenced again, it has to read in from disk, and this requires for I/O completion. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm.

A page replacement algorithm looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses, while balancing it with the costs of primary storage and processor time of the algorithm itself. There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults,

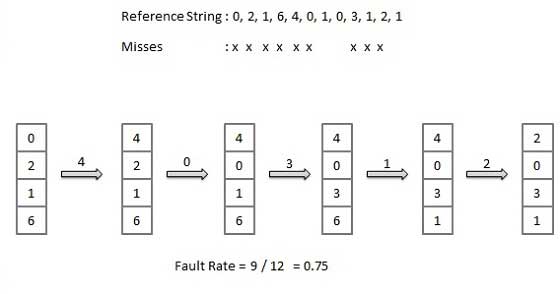
## Reference String

The string of memory references is called reference string. Reference strings are generated artificially or by tracing a given system and recording the address of each memory reference. The latter choice produces a large number of data, where we note two things.

* For a given page size, we need to consider only the page number, not the entire address.
* If we have a reference to a page **p**, then any immediately following references to page **p** will never cause a page fault. Page p will be in memory after the first reference; the immediately following references will not fault.
* For example, consider the following sequence of addresses − 123,215,600,1234,76,96
* If page size is 100, then the reference string is 1,2,6,12,0,0

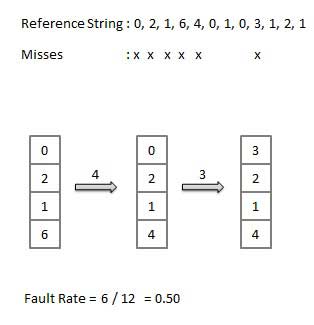
## First In First Out (FIFO) algorithm

* Oldest page in main memory is the one which will be selected for replacement.
* Easy to implement, keep a list, replace pages from the tail and add new pages at the head.



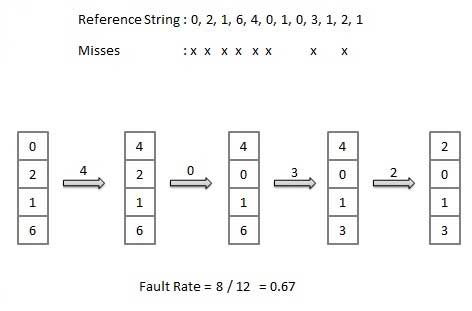
## Optimal Page algorithm

* An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN.
* Replace the page that will not be used for the longest period of time. Use the time when a page is to be used.



## Least Recently Used (LRU) algorithm

* Page which has not been used for the longest time in main memory is the one which will be selected for replacement.
* Easy to implement, keep a list, replace pages by looking back into time.



## Page Buffering algorithm

* To get a process start quickly, keep a pool of free frames.
* On page fault, select a page to be replaced.
* Write the new page in the frame of free pool, mark the page table and restart the process.
* Now write the dirty page out of disk and place the frame holding replaced page in free pool.

## Least frequently Used(LFU) algorithm

* The page with the smallest count is the one which will be selected for replacement.
* This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.

## Most frequently Used(MFU) algorithm

* This algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used

**Frame allocation algorithm**. Frame allocation algorithms are used if you have multiple processes; it helps decide how many frames to allocate to each process.

There are various constraints to the strategies for the allocation of frames:

* You cannot allocate more than the total number of available frames.
* At least a minimum number of frames should be allocated to each process. This constraint is supported by two reasons. The first reason is, as less number of frames are allocated, there is an increase in the page fault ratio, decreasing the performance of the execution of the process. Secondly, there should be enough frames to hold all the different pages that any single instruction can reference.

**Frame allocation algorithms –**  
The two algorithms commonly used to allocate frames to a process are:

1. **Equal allocation:**In a system with x frames and y processes, each process gets equal number of frames, i.e. x/y. For instance, if the system has 48 frames and 9 processes, each process will get 5 frames. The three frames which are not allocated to any process can be used as a free-frame buffer pool.
   * **Disadvantage:** In systems with processes of varying sizes, it does not make much sense to give each process equal frames. Allocation of a large number of frames to a small process will eventually lead to the wastage of a large number of allocated unused frames.
2. **Proportional allocation:** Frames are allocated to each process according to the process size.  
   For a process pi of size si, the number of allocated frames is **ai = (si/S)\*m**, where S is the sum of the sizes of all the processes and m is the number of frames in the system. For instance, in a system with 62 frames, if there is a process of 10KB and another process of 127KB, then the first process will be allocated (10/137)\*62 = 4 frames and the other process will get (127/137)\*62 = 57 frames.
   * **Advantage:** All the processes share the available frames according to their needs, rather than equally.

# **What is Thrash?**

In computer science, **thrash** is the poor performance of a virtual memory (or paging) system when the same pages are being loaded repeatedly due to a lack of main memory to keep them in memory. Depending on the configuration and algorithm, the actual throughput of a system can degrade by multiple orders of magnitude.

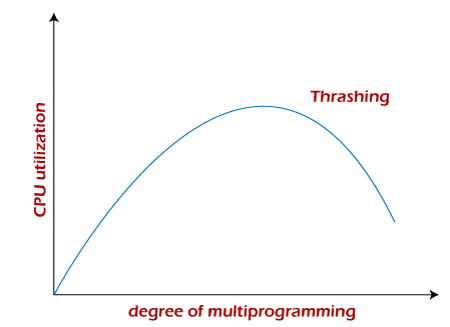
In computer science, **thrashing** occurs when a computer's virtual memory resources are overused, leading to a constant state of paging and page faults, inhibiting most application-level processing. It causes the performance of the computer to degrade or collapse. The situation can continue indefinitely until the user closes some running applications or the active processes free up additional virtual memory resources.

To know more clearly about thrashing, first, we need to know about page fault and swapping.

* **Page fault:** We know every program is divided into some pages. A page fault occurs when a program attempts to access data or code in its address space but is not currently located in the system RAM.
* **Swapping:** Whenever a page fault happens, the operating system will try to fetch that page from secondary memory and try to swap it with one of the pages in RAM. This process is called swapping.

**Thrashing** is when the page fault and swapping happens very frequently at a higher rate, and then the operating system has to spend more time swapping these pages. This state in the operating system is known as thrashing. Because of thrashing, the CPU utilization is going to be reduced or negligible.

ADVERTISEMENT



The basic concept involved is that if a process is allocated too few frames, then there will be too many and too frequent page faults. As a result, no valuable work would be done by the CPU, and the CPU utilization would fall drastically.

The long-term scheduler would then try to improve the CPU utilization by loading some more processes into the memory, thereby increasing the degree of multiprogramming. Unfortunately, this would result in a further decrease in the CPU utilization, triggering a chained reaction of higher page faults followed by an increase in the degree of multiprogramming, called thrashing.

### **Algorithms during Thrashing**

Whenever thrashing starts, the operating system tries to apply either the Global page replacement Algorithm or the Local page replacement algorithm.

**1. Global Page Replacement**

Since global page replacement can bring any page, it tries to bring more pages whenever thrashing is found. But what actually will happen is that no process gets enough frames, and as a result, the thrashing will increase more and more. Therefore, the global page replacement algorithm is not suitable when thrashing happens.

**2. Local Page Replacement**

Unlike the global page replacement algorithm, local page replacement will select pages which only belong to that process. So there is a chance to reduce the thrashing. But it is proven that there are many disadvantages if we use local page replacement. Therefore, local page replacement is just an alternative to global page replacement in a thrashing scenario.

### **Causes of Thrashing**

Programs or workloads may cause thrashing, and it results in severe performance problems, such as:

* If CPU utilization is too low, we increase the degree of multiprogramming by introducing a new system. A global page replacement algorithm is used. The CPU scheduler sees the decreasing CPU utilization and increases the degree of multiprogramming.
* CPU utilization is plotted against the degree of multiprogramming.
* As the degree of multiprogramming increases, CPU utilization also increases.
* If the degree of multiprogramming is increased further, thrashing sets in, and CPU utilization drops sharply.
* So, at this point, to increase CPU utilization and to stop thrashing, we must decrease the degree of multiprogramming.

### **How to Eliminate Thrashing**

Thrashing has some negative impacts on hard drive health and system performance. Therefore, it is necessary to take some actions to avoid it. To resolve the problem of thrashing, here are the following methods, such as:

* **Adjust the swap file size:**If the system swap file is not configured correctly, disk thrashing can also happen to you.
* **Increase the amount of RAM:** As insufficient memory can cause disk thrashing, one solution is to add more RAM to the laptop. With more memory, your computer can handle tasks easily and don't have to work excessively. Generally, it is the best long-term solution.
* **Decrease the number of applications running on the computer:** If there are too many applications running in the background, your system resource will consume a lot. And the remaining system resource is slow that can result in thrashing. So while closing, some applications will release some resources so that you can avoid thrashing to some extent.
* **Replace programs:** Replace those programs that are heavy memory occupied with equivalents that use less memory.

### **Techniques to Prevent Thrashing**

The Local Page replacement is better than the Global Page replacement, but local page replacement has many disadvantages, so it is sometimes not helpful. Therefore below are some other techniques that are used to handle thrashing:

**1. Locality Model**

A locality is a set of pages that are actively used together. The locality model states that as a process executes, it moves from one locality to another. Thus, a program is generally composed of several different localities which may overlap.

For example, when a function is called, it defines a new locality where memory references are made to the function call instructions, local and global variables, etc. Similarly, when the function is exited, the process leaves this locality.

**2. Working-Set Model**

This model is based on the above-stated concept of the Locality Model.

The basic principle states that if we allocate enough frames to a process to accommodate its current locality, it will only fault whenever it moves to some new locality. But if the allocated frames are lesser than the size of the current locality, the process is bound to thrash.

According to this model, based on parameter A, the working set is defined as the set of pages in the most recent 'A' page references. Hence, all the actively used pages would always end up being a part of the working set.

The accuracy of the working set is dependent on the value of parameter A. If A is too large, then working sets may overlap. On the other hand, for smaller values of A, the locality might not be covered entirely.

If D is the total demand for frames and WSSi is the working set size for process i,

D = ⅀ WSSi

Now, if 'm' is the number of frames available in the memory, there are two possibilities:

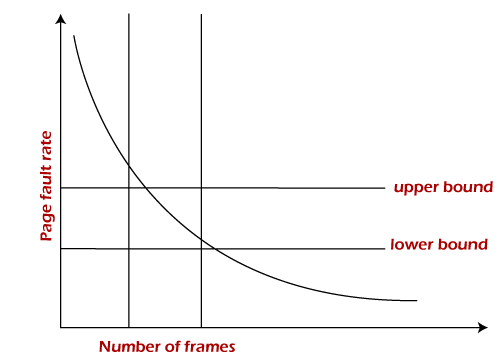
* D>m, i.e., total demand exceeds the number of frames, then thrashing will occur as some processes would not get enough frames.
* D<=m, then there would be no thrashing.

If there are enough extra frames, then some more processes can be loaded into the memory. On the other hand, if the summation of working set sizes exceeds the frames' availability, some of the processes have to be suspended (swapped out of memory).

This technique prevents thrashing along with ensuring the highest degree of multiprogramming possible. Thus, it optimizes CPU utilization.

**3. Page Fault Frequency**

A more direct approach to handle thrashing is the one that uses the Page-Fault Frequency concept.



The problem associated with thrashing is the high page fault rate, and thus, the concept here is to control the page fault rate.

If the page fault rate is too high, it indicates that the process has too few frames allocated to it. On the contrary, a low page fault rate indicates that the process has too many frames.

Upper and lower limits can be established on the desired page fault rate, as shown in the diagram.

If the page fault rate falls below the lower limit, frames can be removed from the process. Similarly, if the page faults rate exceeds the upper limit, more frames can be allocated to the process.

In other words, the graphical state of the system should be kept limited to the rectangular region formed in the given diagram.

If the page fault rate is high with no free frames, some of the processes can be suspended and allocated to them can be reallocated to other processes. The suspended processes can restart later.

## File Concept

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 files creator and user.

## File Structure

A File Structure should be according to a required format that the operating system can understand.

* A file has a certain defined structure according to its type.
* A text file is a sequence of characters organized into lines.
* A source file is a sequence of procedures and functions.
* An object file is a sequence of bytes organized into blocks that are understandable by the machine.
* When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

## 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 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.

**File access methods**

File access methods define how data is accessed and modified within a file. There are different file access methods with their own set of strengths and limitations. The three primary file access methods are sequential access, random access, and direct access. Sequential access reads and writes data in a linear order, random access allows direct access to specific data within the file, and direct access involves accessing data directly by its physical location in the file. Understanding the differences between each method is important for effective data management. In this article, we will explore the characteristics, advantages, and disadvantages of each file access method and provide guidance on which method is best suited for various applications.

## What is file access method

File access method is a way of accessing and manipulating data stored in a file. It determines how data is read and written in computer storage devices. There are several file access methods, each with its own characteristics, advantages, and disadvantages.

The three main types of file access methods are :

* sequential access
* random access
* direct access
* indexed access method

Sequential access reads and writes data in a linear order, while random access allows direct access to specific data within the file. Direct access involves accessing data directly by its physical location in the file. The choice of file access method depends on the specific needs of the application or device using the file. Understanding the differences between each method is important for efficient and effective data management.

## Importance of file access method in operating systems

The file access method is a critical component of an operating system because it determines how files are stored, organized, and accessed by applications and users.

The importance of the file access method in operating systems can be seen in the following ways −

* **Efficiency** − The file access method can significantly impact the efficiency of a computer system, as it determines how quickly files can be accessed and how efficiently data can be written or read.
* **Data integrity** − The file access method ensures that data is stored and accessed correctly, protecting the integrity of the data stored in files.
* **Security** − The file access method can help ensure the security of files, by controlling access to them, limiting who can view, modify, or delete files.
* **Resource management** − The file access method plays an important role in resource management, helping the operating system manage disk space and allocate resources efficiently.

### **Definition of sequential access**

Sequential access is a file access method in which data is accessed in a linear or sequential order. This means that data can only be accessed in the order in which it is stored in the file. Sequential access reads or writes data one after the other, starting from the beginning of the file and ending at the end of the file.

### **How data is read/written in sequential access**

In sequential access, data is accessed in a particular order. For example, to access the 10th record in a file, a program must first read the first nine records sequentially, starting from the beginning of the file, until it reaches the 10th record. The same is true for writing data in a sequential file. The data must be written in the order that it is to be stored in the file.

### **Advantages and disadvantages of sequential access**

Advantages of sequential access include that it is simple and easy to implement, it requires less memory, and it is suitable for storing large amounts of data. However, sequential access is not efficient for accessing specific data or making changes to the data. It is slow when it comes to reading or writing data in the middle of the file since the program must read or write all the data before the required data.

**Examples of devices that use sequential access** − Sequential access is commonly used in devices such as tape drives, which require reading or writing data in a linear or sequential order. Sequential access is also used in some types of disk storage systems, but random access is more commonly used for disk storage.

### **Definition of random access**

Random access is a file access method in which data can be accessed from any location within the file. This means that data can be read or written to any location in the file without having to read through all the data that comes before it. Random access provides the ability to directly access any record or data element in the file.

### **How data is read/written in random access**

In random access, data can be read or written at any location in the file without the need to read all the preceding data. This is possible because random access uses an index or address to locate the specific data required, making it faster and more efficient than sequential access.

### **Advantages and disadvantages of random access**

Random access provides fast and efficient access to specific data within the file. It is also efficient for editing and updating data in the file. However, random access requires more memory to store index or address information, which can make the file size larger than with sequential access. Additionally, if the index or address information becomes corrupted, data can become inaccessible.

**Examples of devices that use random access** − Random access is commonly used in devices such as hard drives, solid-state drives, and USB drives. These devices require fast and efficient access to specific data, which makes random access an ideal file access method. Random access is also commonly used in database systems, where fast access to specific records is required.

### **Definition of direct access**

Direct access is a file access method that allows data to be accessed directly by using the data's physical location within the file. In other words, data can be read or written to any location in the file, much like with random access. However, direct access does not use an index or address like random access, and instead relies on the physical location of the data within the file.

### **How data is read/written in direct access**

In direct access, data is read or written directly to the physical location in the file. The data can be accessed by using the record number, byte position, or block number. This allows for fast and efficient access to specific data within the file.

### **Advantages and disadvantages of direct access**

Direct access provides fast and efficient access to specific data within the file, similar to random access. It also does not require the additional memory needed for index or address information, making the file size smaller than with random access. However, direct access requires knowledge of the physical layout of the data within the file, and may require special hardware or software to access the data directly. Additionally, if data is deleted or moved, gaps can be left in the file which can impact performance.

**Examples of devices that use direct access** − Direct access is commonly used in devices such as magnetic disk drives, optical disk drives, and flash memory. These devices require fast and efficient access to specific data, which makes direct access an ideal file access method. Direct access is also commonly used in database systems, where fast access to specific records is required.

### **Indexed Access Method**

The indexed access method involves accessing files through an index or directory that contains a list of file names and their corresponding locations on the disk. This method is suitable for applications that need to access files by their names or attributes, such as file managers or search engines. The indexed access method provides a fast and efficient way to locate and access files.

The indexed access method uses a file index or directory to keep track of the locations of files on the disk. The file index is stored in a separate file or in a specific location on the disk. When a file is created, its name and location are added to the file index. To access a file, an application searches the file index for the file name and then uses the direct access method to read the file from its location on the disk.

### **Comparison of Access Methods**

Comparison of advantages and disadvantages

### **Sequential Access**

* **Advantages** − Simple and easy to implement, suitable for storing large amounts of data, requires less memory.
* **Disadvantages** − Not efficient for accessing specific data or making changes to the data, slow for reading or writing data in the middle of the file.

### **Random Access**

* **Advantages** − Provides fast and efficient access to specific data within the file, efficient for editing and updating data, suitable for devices that require fast access to specific data.
* **Disadvantages** − Requires more memory to store index or address information, file size can be larger than with sequential access, data can become inaccessible if index or address information becomes corrupted.

### **Direct Access**

* **Advantages** − Provides fast and efficient access to specific data within the file, suitable for devices that require fast access to specific data, file size is smaller than with random access.
* **Disadvantages** − Requires knowledge of the physical layout of the data within the file, may require special hardware or software to access the data directly, gaps can be left in the file which can impact performance.

### **Indexed access**

* **Advantages** − Provides fast and efficient access to files by name or attributes, making it suitable for applications that require searching and retrieving specific files quickly.
* **Disadvantages** − The index must be maintained, which can require additional disk space and processing time.

## Which method is best for certain situations?

The best file access method for a particular situation depends on the requirements of the application or device using the file.

* Sequential access is best suited for applications that require reading or writing data in a linear order, such as logging data, audio/video streaming, or processing large datasets in batches.
* Random access is best suited for applications that require fast access to specific data or records, such as database systems, search engines, or file systems used in operating systems.
* Direct access is best suited for applications that require fast access to specific data and use low-level disk operations, such as device drivers, file systems used in operating systems, or media streaming applications.

In summary, the choice of file access method depends on the specific needs of the application or device using the file. Sequential access is best for linear data processing, random access is best for fast access to specific data, and direct access is best for low-level disk operations.

A **directory** is a container that is used to contain folders and files. It organizes files and folders in a hierarchical manner https://media.geeksforgeeks.org/wp-content/uploads/111-11.png

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

### **1) Single-level directory:**

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

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

https://media.geeksforgeeks.org/wp-content/uploads/222-13.png

**Advantages:**

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

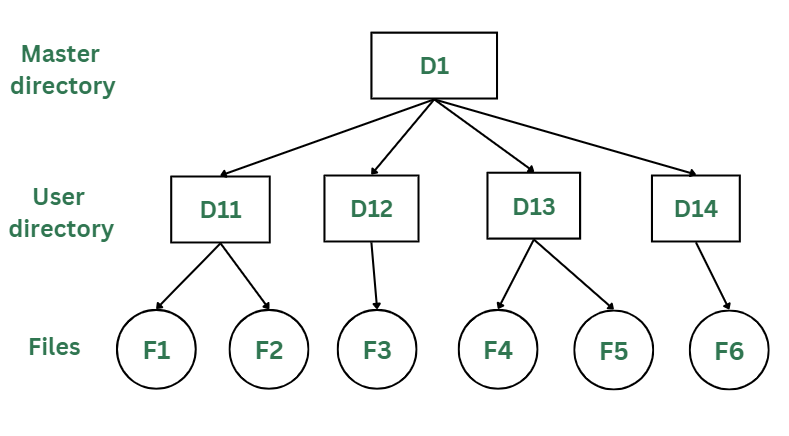
**Disadvantages:**

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

### **2) Two-level directory:**

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

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



*Two-Levels Directory Structure*

#### Advantages:

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

#### Disadvantages:

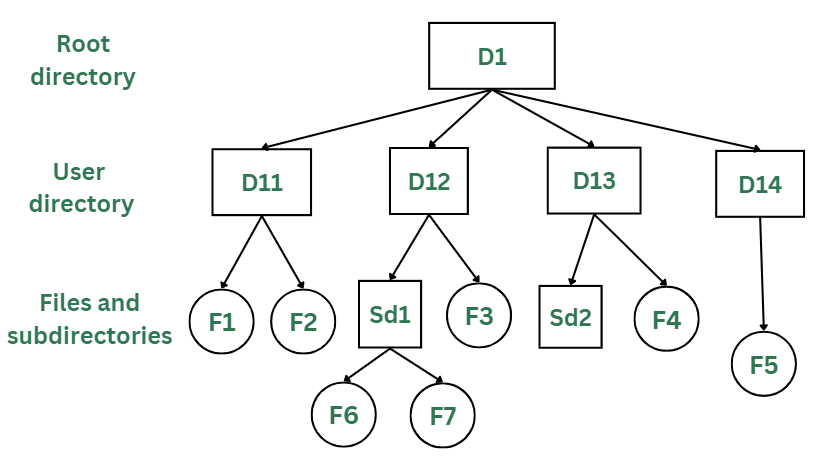
* As there is advantage of security, there is also disadvantage that the user cannot share the file with the other users.
* Unlike the advantage users can create their own files, users don’t have the ability to create subdirectories.
* Scalability is not possible because one use can’t group the same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

### 3) Tree Structure/ Hierarchical Structure:

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

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

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



*Tree/Hierarchical Directory Structure*

#### Advantages:

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

#### Disadvantages:

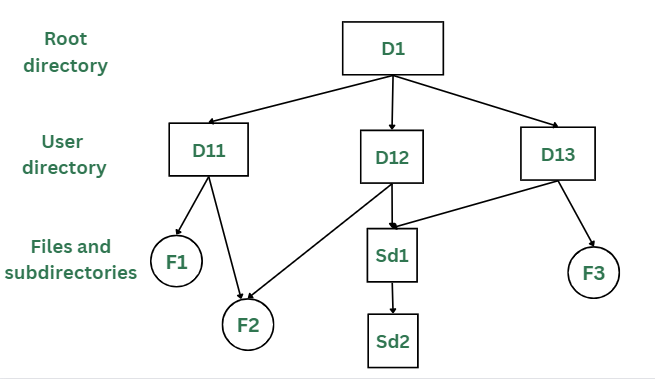
* As the user isn’t allowed to access other user’s directory, this prevents the file sharing among users.
* As the user has the capability to make subdirectories, if the number of subdirectories increase the searching may become complicated.
* Users cannot modify the root directory data.
* If files do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

### 4) Acyclic Graph Structure:

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

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

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



*Acyclic Graph Structure*

#### Advantages:

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

#### Disadvantages:

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

## What is File System Mounting?

Mounting is a process in which the operating system adds the directories and files from a storage device to the user’s computer file system. The file system is attached to an empty directory, by adding so the system user can access the data that is available inside the storage device through the system file manager. Storage systems can be internal hard disks, external hard disks, USB flash drivers, SSD cards, memory cards, network-attached storage devices, CDs and DVDs, remote file systems, or anything else.

### Terminologies used in File System Mounting

* **File System:** It is the method used by the operating system to manage data storage in a storage device. So, a user can access and organize the directories and files in an efficient manner.
* **Device name:** It is a name/identifier given to a storage partition. In windows, for example, “D:” in windows.
* **Mount point**: It is an empty directory in which we are adding the file system during the process of mounting.

## Mounting Indifferent Operating Systems

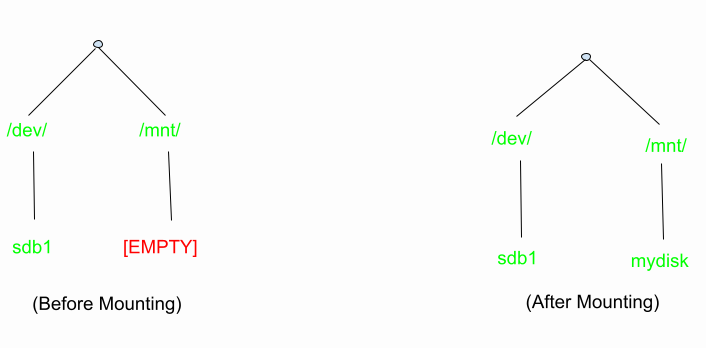
### **1. Linux-Unix based OS**

We want to mount ***/dev/sdb1*** to an existing directory ***/mnt***.

sudo mount /dev/sdb1 /mnt/mydisk

After mounting, we have to unmount after use

sudo umount /mnt/mydisk



*before and after mounting*

### 2. Windows OS

In windows mounting is very easy for a user. When we connect the external storage devices, windows automatically detect the file system and mount it to the drive letter. Drive letter may be**D:** or **E:**.

**Steps:**

* Connect an external storage device to your PC.
* Windows detects the file system on the drive (e.g., FAT32 or NTFS) and assigns it a drive letter, such as “E:”.
* You can access the derive by going through, THIS PC –> FILE EXPLORER –>”E:” drive
* Access the data.

### **Definition of file sharing**

File sharing refers to the process of sharing or distributing electronic files such as documents, music, videos, images, and software between two or more users or computers.

### **Importance of file sharing**

File sharing plays a vital role in facilitating collaboration and communication among individuals and organizations. It allows people to share files quickly and easily across different locations, reducing the need for physical meetings and enabling remote work. File sharing also helps individuals and organizations save time and money, as it eliminates the need for physical transportation of files.

### **Risks and challenges of file sharing**

File sharing can pose several risks and challenges, including the spread of malware and viruses, data breaches and leaks, legal consequences, and identity theft. Unauthorized access to sensitive files can also result in loss of intellectual property, financial losses, and reputational damage.

### **The need for file protection**

With the increase in cyber threats and the sensitive nature of the files being shared, it is essential to implement adequate file protection measures to secure the files from unauthorized access, theft, and cyberattacks. Effective file protection measures can help prevent data breaches and other cyber incidents, safeguard intellectual property, and maintain business continuity.

### **Types of File Sharing**

File sharing refers to the practice of distributing or providing access to digital files, such as documents, images, audio, and video files, between two or more users or devices. There are several types of file sharing methods available, and each method has its own unique advantages and disadvantages.

* **Peer-to-Peer (P2P) File Sharing** − Peer-to-peer file sharing allows users to share files with each other without the need for a centralized server. Instead, users connect to each other directly and exchange files through a network of peers. P2P file sharing is commonly used for sharing large files such as movies, music, and software.
* **Cloud-Based File Sharing** − Cloud-based file sharing involves the storage of files in a remote server, which can be accessed from any device with an internet connection. Users can upload and download files from cloud-based file sharing services such as Google Drive, Dropbox, and OneDrive. Cloud-based file sharing allows users to easily share files with others, collaborate on documents, and access files from anywhere.
* **Direct File Transfer** − Direct file transfer involves the transfer of files between two devices through a direct connection such as Bluetooth or Wi-Fi Direct. Direct file transfer is commonly used for sharing files between mobile devices or laptops.
* **Removable Media File Sharing** − Removable media file sharing involves the use of physical storage devices such as USB drives or external hard drives. Users can copy files onto the device and share them with others by physically passing the device to them.

Each type of file sharing method comes with its own set of risks and challenges. Peer-to-peer file sharing can expose users to malware and viruses, while cloud-based file sharing can lead to data breaches if security measures are not implemented properly. Direct file transfer and removable media file sharing can also lead to data breaches if devices are lost or stolen.

To protect against these risks, users should take precautions such as using encryption, password protection, secure file transfer protocols, and regularly updating antivirus and antimalware software. It is also essential to educate users on safe file sharing practices and limit access to files only to authorized individuals or groups. By taking these steps, users can ensure that their files remain secure and protected during file sharing.

## Risks of File Sharing

File sharing is a convenient and efficient way to share information and collaborate on projects. However, it comes with several risks and challenges that can compromise the confidentiality, integrity, and availability of files. In this section, we will explore some of the most significant risks of file sharing.

* **Malware and Viruses** − One of the most significant risks of file sharing is the spread of malware and viruses. Files obtained from untrusted sources, such as peer-to-peer (P2P) networks, can contain malware that can infect the user's device and compromise the security of their files. Malware and viruses can cause damage to the user's device, steal personal information, or even use their device for illegal activities without their knowledge.
* **Data Breaches and Leaks** − Another significant risk of file sharing is the possibility of data breaches and leaks. Cloud-based file sharing services and P2P networks are particularly vulnerable to data breaches if security measures are not implemented properly. Data breaches can result in the loss of sensitive information, such as personal data or intellectual property, which can have severe consequences for both individuals and organizations.
* **Legal Consequences** − File sharing copyrighted material without permission can lead to legal consequences. Sharing copyrighted music, movies, or software can result in copyright infringement lawsuits and hefty fines.
* **Identity Theft** − File sharing can also expose users to identity theft. Personal information, such as login credentials or social security numbers, can be inadvertently shared through file sharing if security measures are not implemented properly. Cybercriminals can use this information to commit identity theft, which can have severe consequences for the victim.

To protect against these risks, users should take precautions such as using trusted sources for file sharing, limiting access to files, educating users on safe file sharing practices, and regularly updating antivirus and anti-malware software. By taking these steps, users can reduce the risk of malware and viruses, data breaches and leaks, legal consequences, and identity theft during file sharing.

## File Sharing Protection Measures

* **Encryption** − Encryption is the process of converting data into a coded language that can only be accessed by authorized users with a decryption key. This can help protect files from unauthorized access and ensure that data remains confidential even if it is intercepted during file sharing.
* **Password protection** − Password protection involves securing files with a password that must be entered before the file can be accessed. This can help prevent unauthorized access to files and ensure that only authorized users can view or modify the files.
* **Secure file transfer protocols** − Secure file transfer protocols, such as SFTP (Secure File Transfer Protocol) and HTTPS (Hypertext Transfer Protocol Secure), provide a secure way to transfer files over the internet. These protocols use encryption and other security measures to protect files from interception and unauthorized access during transfer.
* **Firewall protection** − Firewall protection involves using a firewall to monitor and control network traffic to prevent unauthorized access to the user's device or network. Firewalls can also be configured to block specific file sharing protocols or limit access to certain users or devices, providing an additional layer of protection for shared files.

## Best Practices for Secure File Sharing

* **Use trusted sources for file sharing** − To reduce the risk of downloading malware or viruses, it is essential to use trusted sources for file sharing. Users should only download files from reputable sources and avoid downloading files from unknown or suspicious websites.
* **Limit access to files** − To minimize the risk of data breaches or leaks, users should limit access to files only to authorized individuals or groups. This can be done by using password protection, encryption, and other access control measures.
* **Educate users on safe file sharing practices** − Educating users on safe file sharing practices can help reduce the risk of security incidents. Users should be trained on how to identify and avoid phishing scams, how to recognize suspicious files or emails, and how to securely share files.
* **Regularly update antivirus and anti-malware software** − To ensure maximum protection against malware and viruses, it is essential to regularly update antivirus and anti-malware software. This will help to identify and remove any potential threats to the user's device or network.

**Consistency Semantics** is concept which is used by users to check file systems which are supporting file sharing in their systems. Basically, it is specification to check that how in a single system multiple user are getting access to same file and at same time. They are used to check various things in files, like when will modification by some user in some file is noticeable to others.

Consistency Semantics is concept which is in a direct relation with concept named process synchronization algorithms.

**Example :** When an atomic transaction to remote disk is performed by user, it involves network communications, disks read and write or both. System which is completing their task with full set of functionalities, had a poor performance.

To access same file by user process is always enclosed between open() and close() operations. When there are series of access take place for same file, then it makes up a file session.