**UNIT IV Storage and Memory Management**

**Swapping,contiguous memory allocation, paging, segmentation, segmentation with paging.** Virtual Memory: **Demand paging**, process creation, page replacement, allocation of frames, **thrashing**. File System Interface: File concept, access methods, directory structure, file system mounting, file sharing, protection. File-System Implementation: File system structure, file-system implementation,directory implementation, allocation methods, free space management, efficiency and performance.

**Storage Management in OS**

Storage Management refers to the **management of the data storage equipment’s that are used to store the user/computer generated data**. Hence it is a **tool or set of processes used by an administrator to keep your data and storage equipment’s safe**. Storage management is a process for users to optimize the use of storage devices and to protect the integrity of data. **Key attributes:**

1. Performance

2. Reliability

3. Recoverability

4. Capacity

**Feature of Storage management:**

1. Storage management is a process that is used to optimize the use of storage devices.
2. Storage management must be allocated and managed as a resource in order to truly benefit a corporation.
3. Storage management is generally a basic system component of information systems.
4. It is used to improve the performance of their data storage resources.

**Advantage of storage management:**

* It becomes very simple to manage a storage capacity.
* It generally reduces the time consumption.
* It improves the performance of system.
* In virtualization and automation technologies, it can help an organization improve its agility.

**Limitations of storage management:**

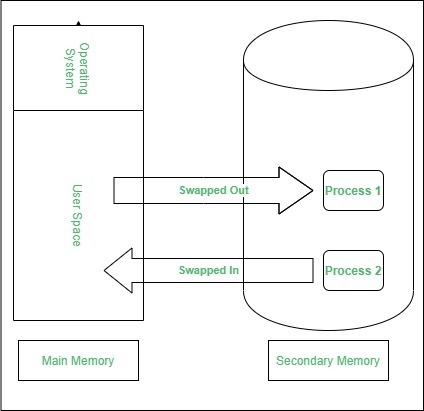
* Limited physical storage capacity: Operating systems can only manage the physical storage space that is available, and as such, there is a limit to how much data can be stored.
* Performance degradation with increased storage utilization: As more data is stored, the system’s performance can decrease due to increased disk access time, fragmentation, and other factors.
* Complexity of storage management: Storage management can be complex, especially as the size of the storage environment grows.
* Cost: Storing large amounts of data can be expensive, and the cost of additional storage capacity can add up quickly.
* Security issues: Storing sensitive data can also present security risks, and the operating system must have robust security features in place to prevent unauthorized access to this data.
* Backup and Recovery: Backup and recovery of data can also be challenging, especially if the data is stored on multiple systems or devices.

**Memory Management in Operating System**

The term memory can be defined as **a collection of data in a specific format**. It is **used to store instructions and process data**. The memory comprises a large array or group of words or bytes, each with its own location. The primary purpose of a computer system is to execute programs. These programs, along with the information they access, should be in the main memory during execution. The CPU fetches instructions from memory according to the value of the program counter. To achieve a degree of multiprogramming and proper utilization of memory, memory management is important.

**Swapping in Operating System**

Swapping is the **process of bringing a process into memory and then temporarily copying it to the disc after it has run for a while**. It is **used to increase CPU utilization in multiprogramming**. The **purpose** of swapping in an operating system is **to access data on a hard disc and move it to RAM so that application programs can use it**. It’s important to remember that **swapping is only used when data isn’t available in RAM**. Although the swapping process degrades system performance, it allows larger and multiple processes to run concurrently. Because of this, **swapping** is also known as **memory compaction**. The **CPU scheduler** determines which processes are swapped in and which are swapped out. Consider a multiprogramming environment that employs a priority-based scheduling algorithm. When a high-priority process enters the input queue, a low-priority process is swapped out so the high-priority process can be loaded and executed. When this process terminates, the low priority process is swapped back into memory to continue its execution. Below figure shows the swapping process in operating system:



**Swapping has been subdivided into two concepts: swap-in and swap-out.**

* Swap-out is a **technique for moving a process from RAM to the hard disc**.
* Swap-in is a **method of transferring a program from hard disc to main memory (RAM)**.

## **Advantages**

* If there is low main memory so some processes may have to wait for much long but by using swapping process do not have to wait long for execution on CPU.
* It utilizes the main memory.
* Using only single main memory, multiple process can be run by CPU using swap partition.
* The concept of virtual memory start from here and it utilize it in better way.
* This concept can be useful in priority based scheduling.

## **Disadvantages**

* If there is low main memory resource and user is executing too many processes and suddenly the power of system goes off there might be a scenario where data get erase of the processes which were involved in swapping.
* Chances of page faults.
* Low processing performance.

**Definition of contiguous memory allocation**

Contiguous 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, leading to faster access to memory and improved system performance. This results in faster access to memory, as contiguous blocks of memory can be accessed more quickly than non-contiguous blocks.

## **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.

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. This means that the memory allocated to 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.

**Paging in OS (Operating System)**

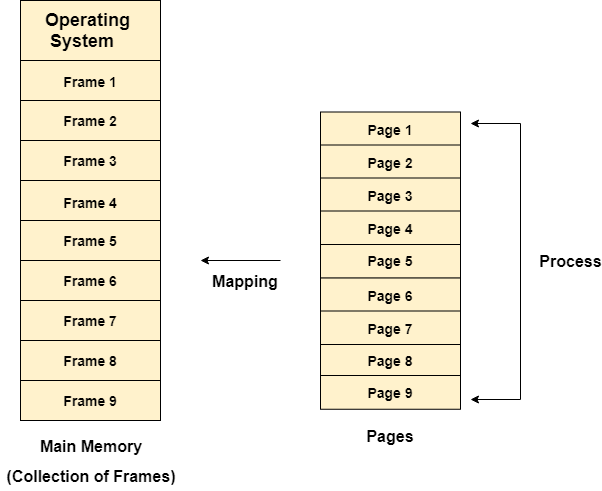
In Operating Systems, Paging is a **storage mechanism used to retrieve processes from the secondary storage into the main memory in the form of pages**.

The main idea behind the paging is to divide each process in the form of pages. The main memory will also be divided in the form of frames.

**One page of the process is to be stored in one of the frames of the memory**. The pages can be stored at the different locations of the memory but the priority is always to find the contiguous frames or holes.

Pages of the process are brought into the main memory only when they are required otherwise they reside in the secondary storage.

**Different operating system defines different frame sizes**. **The sizes of each frame must be equal**. Since pages are mapped to the frames in Paging, **page size and frame size must be equal.**



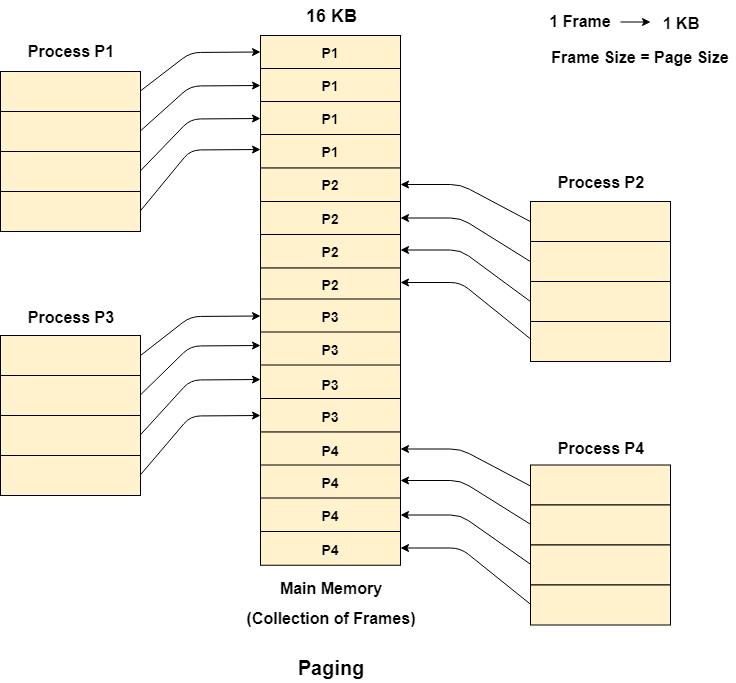
### **Example**

Let us consider the main memory size 16 Kb and Frame size is 1 KB therefore the main memory will be divided into the collection of 16 frames of 1 KB each.

There are 4 processes in the system that is P1, P2, P3 and P4 of 4 KB each. Each process is divided into pages of 1 KB each so that one page can be stored in one frame.

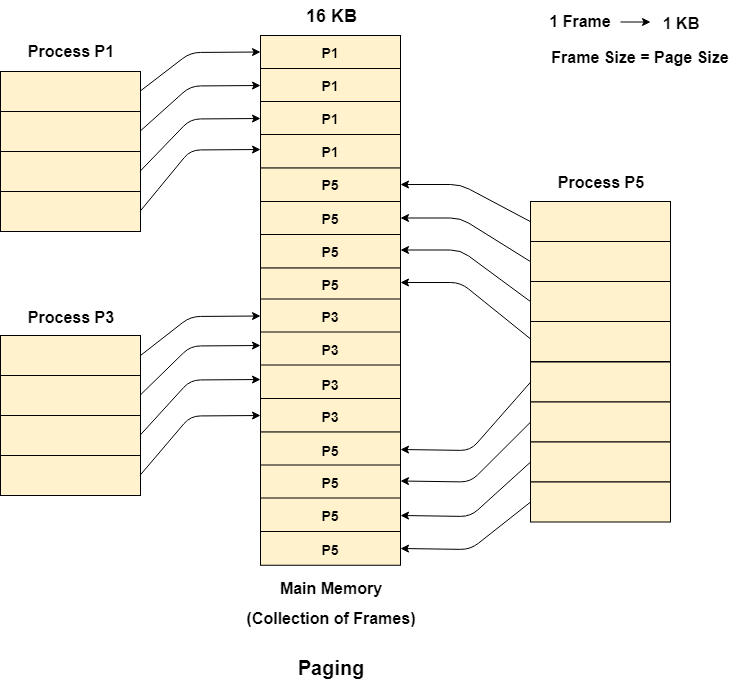
Initially, all the frames are empty therefore pages of the processes will get stored in the contiguous way.

Frames, pages and the mapping between the two is shown in the image below.



Let us consider that, P2 and P4 are moved to waiting state after some time. Now, 8 frames become empty and therefore other pages can be loaded in that empty place. The process P5 of size 8 KB (8 pages) is waiting inside the ready queue.

Given the fact that, we have 8 non contiguous frames available in the memory and paging provides the flexibility of storing the process at the different places. Therefore, we can load the pages of process P5 in the place of P2 and P4.



**Memory Management Unit**

The purpose of Memory Management Unit (MMU) is to **convert the logical address into the physical address**. The **logical address is the address generated by the CPU for every page while the physical address is the actual address of the frame where each page will be stored**.

When a page is to be accessed by the CPU by using the logical address, the operating system needs to obtain the physical address to access that page physically. Memory management unit of OS needs to convert the page number to the frame number.

The logical address has two parts :-

1. Page Number
2. Offset

**Segmentation in Operating System**

A **process is divided into Segments**. The chunks that a program is divided into which are not necessarily all of the exact sizes are called segments. Segmentation gives the user’s view of the process which paging does not provide. Here the user’s view is mapped to physical memory.

## Types of Segmentation in Operating System

* **Virtual Memory Segmentation:** Each process is divided into a number of segments, **but** the **segmentation is not done all at once**. This segmentation may or may not take place at the run time of the program.

1. **Simple Segmentation:** Each process is divided into a number of segments, **all of which are loaded into memory at run time**, though not necessarily contiguously.

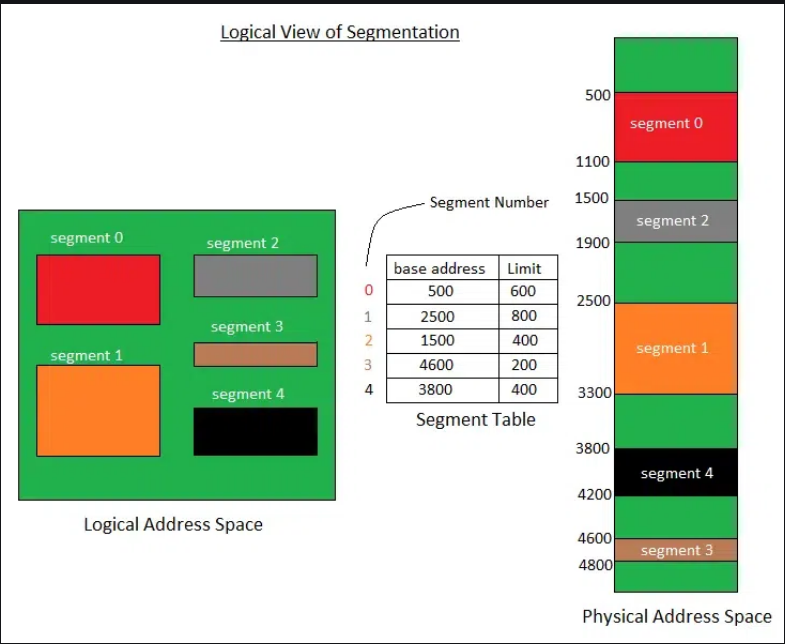
There is no simple relationship between logical addresses and physical addresses in segmentation.

## **What is Segment Table?**

A **process is divided into Segments**. A table that stores the information about all such segments is called Segment Table. It **maps a two-dimensional Logical address into a one-dimensional Physical address**. It has two components :-

* **Base Address:**Itcontains the starting physical address where the segments reside in memory.

1. **Segment Limit:** Also known as segment offset. It specifies the length of the segment.



The address generated by the CPU is divided into:

* **Segment number(s):** Number of bits required to represent the segment.

1. **Segment offset(d):** Number of bits required to represent the size of the segment.

## **Advantages of Segmentation in Operating System**

* No Internal fragmentation.

1. Segment Table consumes less space in comparison to Page table in paging.
2. As a complete module is loaded all at once, segmentation improves CPU utilization.
3. The user specifies the segment size, whereas, in paging, the hardware determines the page size.
4. Segmentation is a method that can be used to segregate data from security operations.
5. **Flexibility:** Segmentation provides a higher degree of flexibility than paging. Segments can be of variable size, and processes can be designed to have multiple segments, allowing for more fine-grained memory allocation.
6. **Sharing:** Segmentation allows for sharing of memory segments between processes. This can be useful for inter-process communication or for sharing code libraries.
7. **Protection:** Segmentation provides protection to the segments, preventing one process from accessing or modifying another process’s memory segment. This can help to increase the security and stability of the system.

## **Disadvantages of Segmentation in Operating System**

1. As processes are loaded and removed from the memory, the free memory space is broken into small segments causing External fragmentation. This can lead to wasted memory and decreased performance.
2. Due to the need for two memory accesses, one for the segment table and the other for main memory, access time to retrieve the instruction increases.
3. **Overhead:** Using a segment table can increase overhead and reduce performance. Each segment table entry requires additional memory and accessing the table to retrieve memory locations can increase the time needed for memory operations.

**Segmentation with Paging**

Pure segmentation is not very popular and not being used in many of the operating systems. However, **Segmentation can be combined with Paging** to get the best features out of both the techniques. Page Table is a data structure used by the virtual memory system to store the mapping between logical addresses and physical addresses.

**In Segmented Paging, the main memory is divided into variable size segments which are further divided into fixed size pages.**

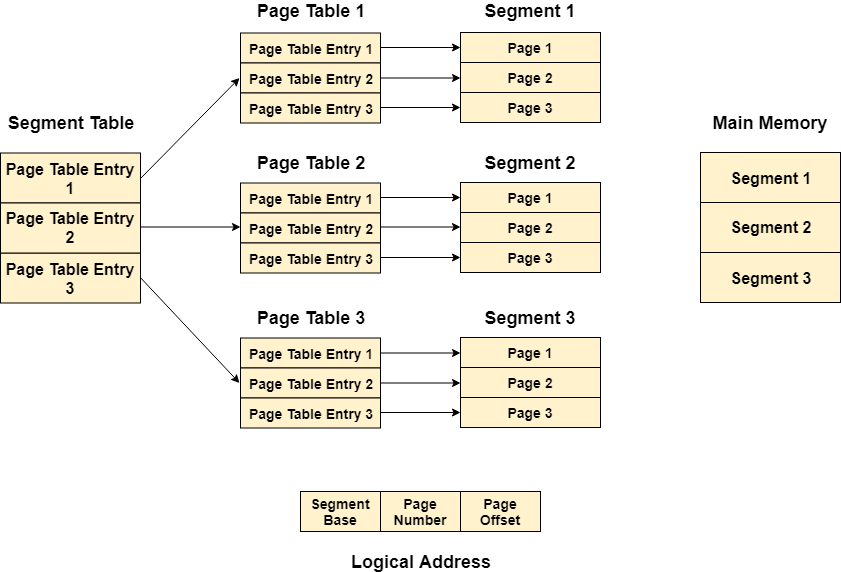
1. Pages are smaller than segments.
2. Each Segment has a page table which means every program has multiple page tables.
3. The logical address is represented as Segment Number (base address), Page number and page offset.

**Segment Number →** It points to the appropriate Segment Number.

**Page Number →** It Points to the exact page within the segment

**Page Offset →** Used as an offset within the page frame

Each Page table contains the various information about every page of the segment. The Segment Table contains the information about every segment. Each segment table entry points to a page table entry and every page table entry is mapped to one of the page within a segment.



## **Advantages of Segmented Paging**

## The page table size is reduced as pages are present only for data of segments, hence reducing the memory requirements.

## Gives a programmers view along with the advantages of paging.

## Reduces external fragmentation in comparison with segmentation.

## Since the entire segment need not be swapped out, the swapping out into virtual memory becomes easier .

## **Disadvantages of Segmented Paging**

## Internal fragmentation still exists in pages.

## Extra hardware is required

## Translation becomes more sequential increasing the memory access time.

## External fragmentation occurs because of varying sizes of page tables and varying sizes of segment tables in today’s systems.

**Virtual Memory**

**Advantages of Virtual Memory**

1. More processes may be maintained in the main memory: Because we are going to load only some of the pages of any particular process, there is room for more processes. This leads to more efficient utilization of the processor because it is more likely that at least one of the more numerous processes will be in the ready state at any particular time.
2. A process may be larger than all of the main memory: One of the most fundamental restrictions in programming is lifted. A process larger than the main memory can be executed because of demand paging. The OS itself loads pages of a process in the main memory as required.
3. It allows greater multiprogramming levels by using less of the available (primary) memory for each process.
4. It has twice the capacity for addresses as main memory.
5. It makes it possible to run more applications at once.
6. Users are spared from having to add memory modules when [RAM](https://www.geeksforgeeks.org/different-types-ram-random-access-memory/)space runs out, and applications are liberated from shared memory management.
7. When only a portion of a program is required for execution, speed has increased.
8. Memory isolation has increased security.
9. It makes it possible for several larger applications to run at once.
10. Memory allocation is comparatively cheap.
11. It doesn’t require outside fragmentation.
12. It is efficient to manage logical partition workloads using the CPU.
13. Automatic data movement is possible.

**Disadvantages of Virtual Memory**

1. It can slow down the system performance, as data needs to be constantly transferred between the physical memory and the hard disk.
2. It can increase the risk of data loss or corruption, as data can be lost if the hard disk fails or if there is a power outage while data is being transferred to or from the hard disk.
3. It can increase the complexity of the memory management system, as the operating system needs to manage both physical and virtual memory.

**Demand Paging**

Demand paging is a technique used in virtual memory systems where pages enter main memory only when requested or needed by the CPU. It is used to improve memory usage and system performance. In other words, the process of loading the page into memory on demand (whenever a page fault occurs) is known as demand paging.

In demand paging, the operating system loads only the necessary pages of a program into memory at runtime, instead of loading the entire program into memory at the start. A page fault occurred when the program needed to access a page that is not currently in memory. The operating system then loads the required pages from the disk into memory and updates the page tables accordingly. This process is transparent to the running program and it continues to run as if the page had always been in memory.

**Advantages of Demand Paging**

1. So in the Demand Paging technique, there are some benefits that provide efficiency of the operating system.
2. Efficient use of physical memory: Query paging allows for more efficient use because only the necessary pages are loaded into memory at any given time.
3. Support for larger programs: Programs can be larger than the physical memory available on the system because only the necessary pages will be loaded into memory.
4. Faster program start: Because only part of a program is initially loaded into memory, programs can start faster than if the entire program were loaded at once.
5. Reduce memory usage: Query paging can help reduce the amount of memory a program needs, which can improve system performance by reducing the amount of disk I/O required.

**Disadvantages of Demand Paging**

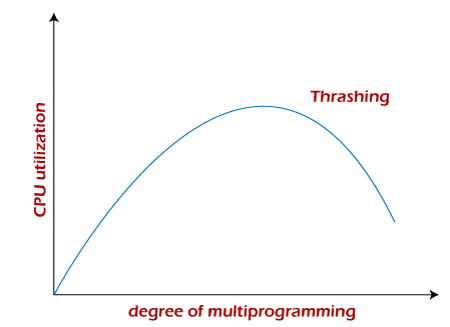
1. Page Fault Overload: The process of swapping pages between memory and disk can cause a performance overhead, especially if the program frequently accesses pages that are not currently in memory.
2. Degraded performance: If a program frequently accesses pages that are not currently in memory, the system spends a lot of time swapping out pages, which degrades performance.
3. Fragmentation: Query paging can cause physical memory [fragmentation](https://www.geeksforgeeks.org/what-is-fragmentation-in-operating-system/), degrading system performance over time.
4. Complexity: Implementing query paging in an operating system can be complex, requiring complex algorithms and [data structures](https://www.geeksforgeeks.org/introduction-to-data-structures/) to manage page tables and swap space.

**Thrashing**

*Thrashing* occurs 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.The situation can continue indefinitely until the user closes some running applications or the active processes free up additional virtual memory resources.

**Page fault:** Every program is divided into 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.



**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**

Global page replacement can bring any page, so it will try 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**

Local page replacement will select pages which only belong to that process. So there is a chance to reduce the thrashing. Therefore, local page replacement is just an alternative to global page replacement in a thrashing scenario.

**Causes of Thrashing**

* High degree of multiprogramming.
* Lack of frames.
* Page replacement policy.

**How to Eliminate Thrashing**

* **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**

**1. Locality Model**

**2. Working-Set Model**

**3. Page Fault Frequency**

**Process Creation in Operating Systems**

1. When a new process is created, the operating system assigns a unique Process Identifier (PID) to it and inserts a new entry in the primary process table.

2. Then required memory space for all the elements of the process such as program, data, and stack is allocated including space for its [Process Control Block](https://www.geeksforgeeks.org/process-table-and-process-control-block-pcb/) (PCB).

3. Next, the various values in PCB are initialized such as,

1. The process identification part is filled with PID assigned to it in step (1) and also its parent’s PID.
2. The processor register values are mostly filled with zeroes, except for the stack pointer and program counter. The stack pointer is filled with the address of the stack-allocated to it in step (ii) and the program counter is filled with the address of its program entry point.
3. The process state information would be set to ‘New’.
4. Priority would be lowest by default, but the user can specify any priority during creation.

4. Then the operating system will link this process to the scheduling queue and the process state would be changed from ‘New’ to ‘Ready’. Now the process is competing for the CPU.

5. Additionally, the operating system will create some other data structures such as log files or accounting files to keep track of processes activity.

**Allocation of Frames in OS**

The main memory of the operating system is divided into various frames. The process is stored in these frames, and once the process is saved as a frame, the CPU may run it. As a result, the operating system must set aside enough frames for each process. As a result, the operating system uses various algorithms in order to assign the frame.

Demand paging is used to implement virtual memory, an essential operating system feature. It requires the development of a page replacement mechanism and a frame allocation system. If you have multiple processes, the frame allocation techniques are utilized to define how many frames to allot to each one. A number of factors constrain the strategies for allocating frames:

1. You cannot assign more frames than the total number of frames available.
2. A specific number of frames should be assigned to each process. This limitation is due to two factors. The first is that when the number of frames assigned drops, the page fault ratio grows, decreasing the process's execution performance. Second, there should be sufficient frames to hold all the multiple pages that any instruction may reference.

There are mainly five ways of frame allocation algorithms in the OS. These are as follows:

1. **Equal Frame Allocation**
2. **Proportional Frame Allocation**
3. **Priority Frame Allocation**
4. **Global Replacement Allocation**
5. **Local Replacement Allocation**

## **Equal Frame Allocation**

In equal frame allocation, the frames are assigned equally among the processes in the OS. For example, if the system has 30 frames and 7 processes, each process will get 4 frames. The 2 frames that are not assigned to any system process may be used as a free-frame buffer pool in the system.

**Disadvantage**

In a system with processes of varying sizes, assigning equal frames to each process makes little sense. Many allotted empty frames will be wasted if many frames are assigned to a small task.

## **Proportional Frame Allocation**

The proportional frame allocation technique assigns frames based on the size needed for execution and the total number of frames in memory.

The allocated frames for a process **pi** of size **si** are **ai = (si/S)\*m**, in which **S** represents the total of all process sizes, and **m** represents the number of frames in the system.

**Disadvantage**

The only drawback of this algorithm is that it doesn't allocate frames based on priority. Priority frame allocation solves this problem.

## **Priority Frame Allocation**

Priority frame allocation assigns frames based on the number of frame allocations and the processes. Suppose a process has a high priority and requires more frames that many frames will be allocated to it. Following that, lesser priority processes are allocated.

## **Global Replacement Allocation**

When a process requires a page that isn't currently in memory, it may put it in and select a frame from the all frames sets, even if another process is already utilizing that frame. In other words, one process may take a frame from another.

**Advantages**

Process performance is not hampered, resulting in higher system throughput.

**Disadvantages**

The process itself may not solely control the page fault ratio of a process. The paging behavior of other processes also influences the number of pages in memory for a process.

## **Local Replacement Allocation**

When a process requires a page that isn't already in memory, it can bring it in and assign it a frame from its set of allocated frames.

**Advantages**

The paging behavior of a specific process has an effect on the pages in memory and the page fault ratio.

**Disadvantages**

A low priority process may obstruct a high priority process by refusing to share its frames.

**File Systems in Operating System**

A computer file is defined as a medium used for saving and managing data in the computer system. The data stored in the computer system is completely in digital format, although there can be various types of files that help us to store the data.

## What is a File System?

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

1. **FAT (File Allocation Table):** An older file system used by older versions of Windows and other operating systems.
2. **NTFS (New Technology File System):** A modern file system used by Windows. It supports features such as file and folder permissions, compression, and encryption.
3. **ext (Extended File System):** A file system commonly used on Linux and Unix-based operating systems.
4. **HFS (Hierarchical File System):** A file system used by macOS.
5. **APFS (Apple File System):** A new file system introduced by Apple for their Macs and iOS devices.

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

**File Access Methods in Operating System**

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

**1.Sequential Access –**It is the simplest access method. Information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, editor and compiler usually access the file in this fashion.   
Read and write make up the bulk of the operation on a file. A read operation *-read next-* read the next position of the file and automatically advance a file pointer, which keeps track I/O location. Similarly, for the -write*next-* append to the end of the file and advance to the newly written material.   
**Key points:**

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

**Advantages of Sequential Access Method :**

* It is simple to implement this file access mechanism.
* It uses lexicographic order to quickly access the next entry.
* It is suitable for applications that require access to all records in a file, in a specific order.
* It is less prone to data corruption as the data is written sequentially and not randomly.
* It is a more efficient method for reading large files, as it only reads the required data and does not waste time reading unnecessary data.
* It is a reliable method for backup and restore operations, as the data is stored sequentially and can be easily restored if required.

**Disadvantages of Sequential Access Method :**

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

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

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

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

**Key points:**

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

**4.Relative Record Access –**

Relative record access is a file access method used in operating systems where records are accessed relative to the current position of the file pointer. In this method, records are located based on their position relative to the current record, rather than by a specific address or key value.

**Key Points of Relative Record Access:**

Relative record access is a random access method that allows records to be accessed based on their position relative to the current record.

This method is efficient for accessing individual records but may not be suitable for files that require frequent updates or random access to specific records.

Relative record access requires fixed-length records and may not be flexible enough for some applications.

This method is useful for processing records in a specific order or for files that are accessed sequentially.

**Advantages of Relative Record Access:**

Random Access: Relative record access allows random access to records in a file. The system can access any record at a specific offset from the current position of the file pointer.

Efficient Retrieval: Since the system only needs to read the current record and any records that need to be skipped, relative record access is more efficient than sequential access for accessing individual records.

Useful for Sequential Processing: Relative record access is useful for processing records in a specific order. For example, if the records are sorted in a specific order, the system can access the next or previous record relative to the current position of the file pointer.

**Disadvantages of Relative Record Access:**

Fixed Record Length: Relative record access requires fixed-length records. If the records are of varying length, it may be necessary to use padding to ensure that each record is the same length.

Limited Flexibility: Relative record access is not very flexible. It is difficult to insert or delete records in the middle of a file without disrupting the relative positions of other records.

Limited Application: Relative record access is best suited for files that are accessed sequentially or with some regularity, but it may not be appropriate for files that are frequently updated or require random access to specific records.

# Structures of Directory in Operating System

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



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

### **1) Single-level directory:**

 The single-level directory is the **simplest directory structure**. In it, all files are contained in the same directory which makes it easy to support and understand.   
A single level directory has a significant limitation, however, when the number of files increases or when the system has more than one user. Since all the files are in the same directory, they must have a **unique name**. If two users call their dataset test, then the unique name rule violated.



**Advantages:**

* Since it is a single directory, so its implementation is very easy.

1. If the files are smaller in size, searching will become faster.
2. The operations like file creation, searching, deletion, updating are very easy in such a directory structure.
3. **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.
4. **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.
5. **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.
6. **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.
7. **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.

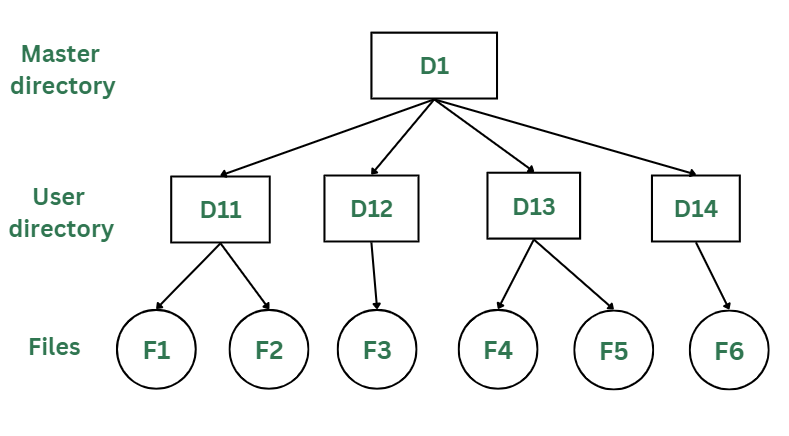
**Disadvantages:**

* There may chance of name collision because two files can have the same name.

1. Searching will become time taking if the directory is large.
2. 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.

1. A security would be there which would prevent user to access other user’s files.
2. 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.

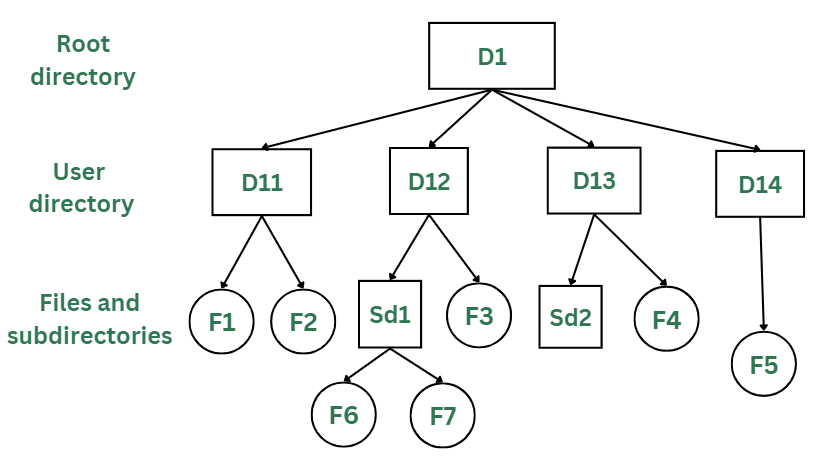
1. Unlike the advantage users can create their own files, users don’t have the ability to create subdirectories.
2. Scalability is not possible because one use can’t group the same types of files together.

### **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.

1. The searching is easier.
2. File sorting of important and unimportant becomes easier.
3. 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.

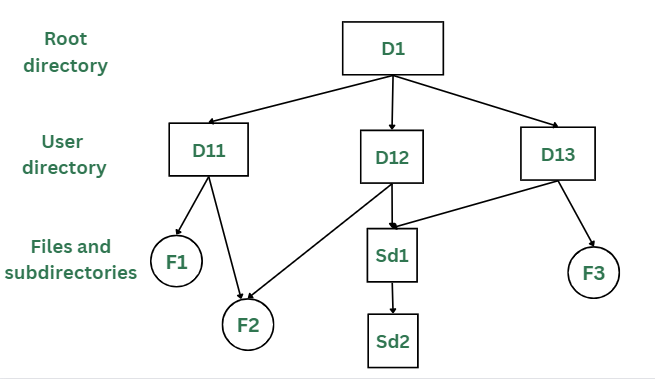
1. As the user has the capability to make subdirectories, if the number of subdirectories increase the searching may become complicated.
2. Users cannot modify the root directory data.
3. If files do not fit in one, they might have to be fit into other directories.

### **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.

1. Searching becomes too easy.
2. 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.

1. The user must be very cautious to edit or even deletion of file as the file is accessed by multiple users.
2. If we need to delete the file, then we need to delete all the references of the file inorder to delete it permanently.

# File System Mounting in OS

In this article, we are going to discuss the most important concept in Operating Systems which allows the users to organize and access files from different storage devices. If we compare it to a real-life scenario, it is closely related to “Connecting puzzles to get a complete picture of data”.

## 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.

1. **Device name:** It is a name/identifier given to a storage partition. In windows, for example, “D:” in windows.
2. **Mount point**: It is an empty directory in which we are adding the file system during the process of mounting.

## Mounting Indifferent Operating Systems

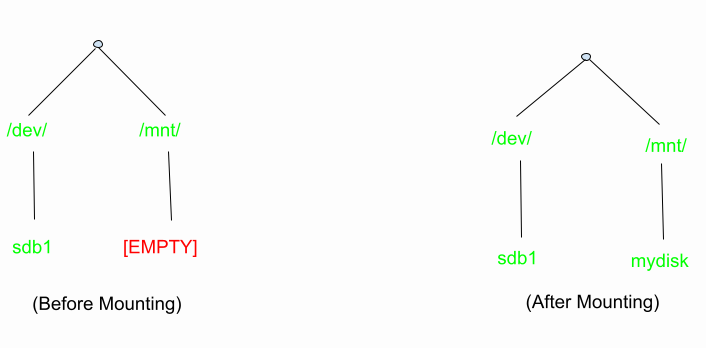
### **1. Linux-Unix based OS**

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.

1. Windows detects the file system on the drive (e.g., FAT32 or NTFS) and assigns it a drive letter, such as “E:”.
2. You can access the derive by going through, THIS PC –> FILE EXPLORER –>”E:” drive
3. Access the data.

### 3. Mac OS

In Mac OS when we connect an external storage it will automatically mount, and it will be accessible via Finder. As an advanced mounting method user can also use the command **diskutil** in Terminal.

#### Method 1:

Steps:

* Connect an external storage device to your MAC.

1. MS OS detects the file system and automatically mount it.
2. You can access the drive by opening Finder, and it will appear in the sidebar.

#### Method 2(Using diskutil):

To mount a drive with a known identifier: **disk2s1**

diskutil mount /dev/disk2s1

To unmount:

diskutil unmount /dev/disk2s1

# File Sharing in OS

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.

Now, file sharing is nothing like a magical bridge between Computer A to Computer B allowing them to swap some files with each other.

## Primary Terminology Related to File Sharing

Let’s see what are the various ways to achieve this, but there are some important terminologies one should know beforehand. Let’s discuss those primary terminologies first:

* **Folder/Directory:** It is basically like a container for all of our files on a computer. The folder can contain files and even other folders maintaining like hierarchical structure for organizing data.

1. **Networking:** It is involved in connecting computers or devices where we need to share the resources. Networks can be local (LAN) or global (Internet).
2. **IP Address:** It is numerical data given to every connected device on the network
3. **Protocol:** It is given as the set of rules which drives the communication between devices on a network. In the context of file sharing, protocols define how files are transferred between computers.
4. **File Transfer Protocol (FTP):** FTP is a standard network protocol used to transfer files between a client and a server on a computer network.

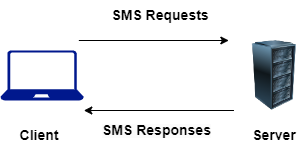
## Various Ways to Achieve File Sharing

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*

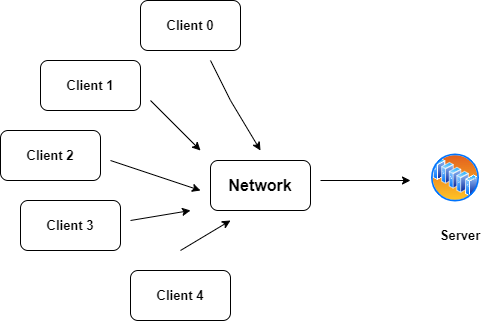
Read more about SMB in the article :

[SMB and it’s implementation](https://www.geeksforgeeks.org/introduction-to-microsoft-smb-a-network-file-sharing-protocol/)

### 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*

Read more about NFS in the article:

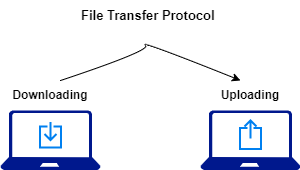
[NFS and it’s architecture](https://www.geeksforgeeks.org/network-file-system-nfs/)

### 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.

Read more about FTP: [FTP and it’s implementation](https://www.geeksforgeeks.org/file-transfer-protocol-ftp/)



*FTP File Sharing*

### 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.



*Cloud Based File Sharing*

These all file sharing methods serves different purpose and needs according to the requirements and flexibility of the users based on the operating system.

**Protection in File System**

In computer systems, alot of user’s information is stored, the objective of the operating system is to keep safe the data of the user from the improper access to the system. Protection can be provided in number of ways. For a single laptop system, we might provide protection by locking the computer in a desk drawer or file cabinet. For multi-user systems, different mechanisms are used for the protection.

**Types of Access :**

The files which have direct access of the any user have the need of protection. The files which are not accessible to other users doesn’t require any kind of protection. The mechanism of the protection provide the facility of the controlled access by just limiting the types of access to the file. Access can be given or not given to any user depends on several factors, one of which is the type of access required. Several different types of operations can be controlled:

* **Read –** Reading from a file.
* **Write –** Writing or rewriting the file.
* **Execute –** Loading the file and after loading the execution process starts.
* **Append –** Writing the new information to the already existing file, editing must be end at the end of the existing file.
* **Delete –** Deleting the file which is of no use and using its space for the another data.
* **List –** List the name and attributes of the file.

Operations like renaming, editing the existing file, copying; these can also be controlled. There are many protection mechanism. each of them mechanism have different advantages and disadvantages and must be appropriate for the intended application.

**Access Control :**

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

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

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

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

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

**Other Protection Approaches:**

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

The use of passwords has a few disadvantages:

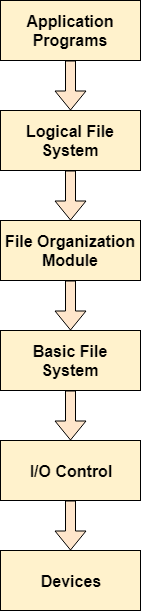
* The number of passwords are very large so it is difficult to remember the large passwords.
* If one password is used for all the files, then once it is discovered, all files are accessible; protection is on all-or-none basis.

File System Structure

File System provide efficient access to the disk by allowing data to be stored, located and retrieved in a convenient way. A file System must be able to store the file, locate the file and retrieve the file.

Most of the Operating Systems use layering approach for every task including file systems. Every layer of the file system is responsible for some activities.

The image shown below, elaborates how the file system is divided in different layers, and also the functionality of each layer.



* When an application program asks for a file, the first request is directed to the logical file system. The logical file system contains the Meta data of the file and directory structure. If the application program doesn't have the required permissions of the file then this layer will throw an error. Logical file systems also verify the path to the file.
* Generally, files are divided into various logical blocks. Files are to be stored in the hard disk and to be retrieved from the hard disk. Hard disk is divided into various tracks and sectors. Therefore, in order to store and retrieve the files, the logical blocks need to be mapped to physical blocks. This mapping is done by File organization module. It is also responsible for free space management.
* Once File organization module decided which physical block the application program needs, it passes this information to basic file system. The basic file system is responsible for issuing the commands to I/O control in order to fetch those blocks.
* I/O controls contain the codes by using which it can access hard disk. These codes are known as device drivers. I/O controls are also responsible for handling interrupts.

# Directory Implementation

There is the number of algorithms by using which, the directories can be implemented. However, the selection of an appropriate directory implementation algorithm may significantly affect the performance of the system.

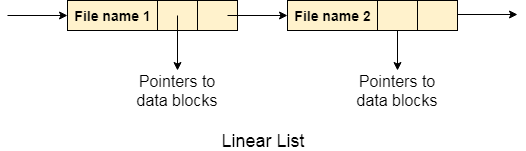
The directory implementation algorithms are classified according to the data structure they are using. There are mainly two algorithms which are used in these days.

### **1. Linear List**

In this algorithm, all the files in a directory are maintained as singly lined list. Each file contains the pointers to the data blocks which are assigned to it and the next file in the directory.

**Characteristics**

1. When a new file is created, then the entire list is checked whether the new file name is matching to a existing file name or not. In case, it doesn't exist, the file can be created at the beginning or at the end. Therefore, searching for a unique name is a big concern because traversing the whole list takes time.
2. The list needs to be traversed in case of every operation (creation, deletion, updating, etc) on the files therefore the systems become inefficient.

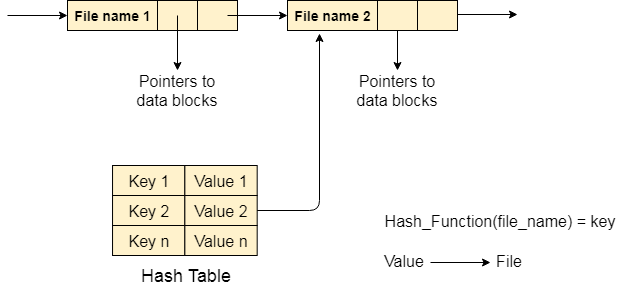


### **2. Hash Table**

To overcome the drawbacks of singly linked list implementation of directories, there is an alternative approach that is hash table. This approach suggests to use hash table along with the linked lists.

A key-value pair for each file in the directory gets generated and stored in the hash table. The key can be determined by applying the hash function on the file name while the key points to the corresponding file stored in the directory.

Now, searching becomes efficient due to the fact that now, entire list will not be searched on every operating. Only hash table entries are checked using the key and if an entry found then the corresponding file will be fetched using the value.



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

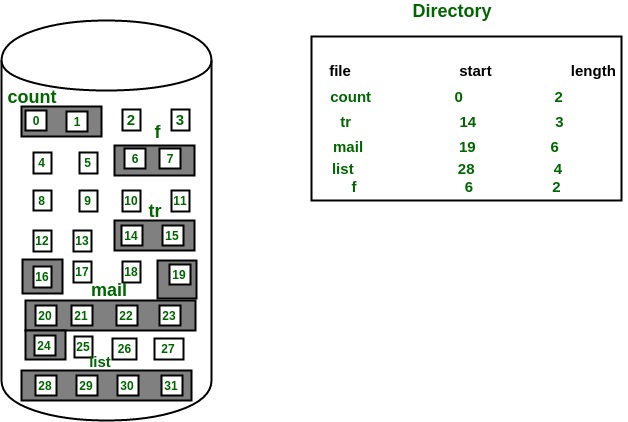
The main idea behind these methods is to provide:

* 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.[](https://media.geeksforgeeks.org/wp-content/uploads/Contiguous-Allocation.jpg)

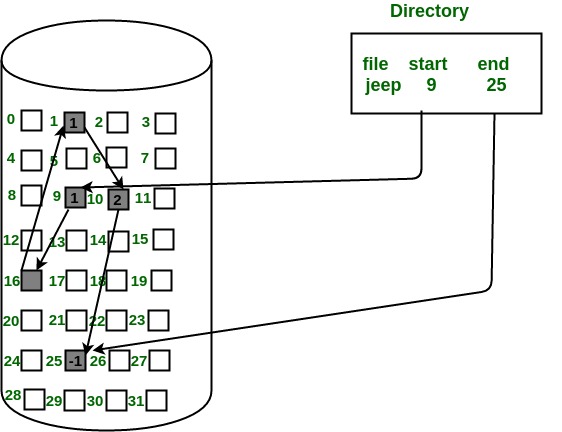
**Advantages:**

* 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.*



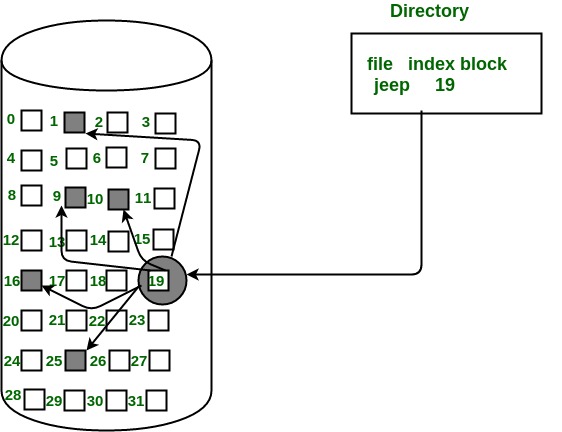
**Advantages:**

* 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:**

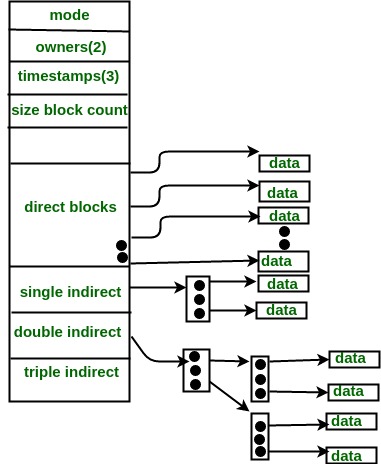
* This supports direct access to the blocks occupied by the file and therefore provides fast access to the file blocks.
* It overcomes the problem of external fragmentation.

**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.

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

1. **Linked scheme:** This scheme links two or more index blocks together for holding the pointers. Every index block would then contain a pointer or the address to the next index block.
2. **Multilevel index:** In this policy, a first level index block is used to point to the second level index blocks which inturn points to the disk blocks occupied by the file. This can be extended to 3 or more levels depending on the maximum file size.
3. **Combined Scheme:** In this scheme, a special block called the **Inode (information Node)** contains all the information about the file such as the name, size, authority, etc and the remaining space of Inode is used to store the Disk Block addresses which contain the actual file*as shown in the image below.* The first few of these pointers in Inode point to the **direct blocks** i.e the pointers contain the addresses of the disk blocks that contain data of the file. The next few pointers point to indirect blocks. Indirect blocks may be single indirect, double indirect or triple indirect. **Single Indirect block** is the disk block that does not contain the file data but the disk address of the blocks that contain the file data. Similarly, **double indirect blocks** do not contain the file data but the disk address of the blocks that contain the address of the blocks containing the file data.

[](https://media.geeksforgeeks.org/wp-content/uploads/Combined-Scheme.jpg)

# Free Space Management

A file system is responsible to allocate the free blocks to the file therefore it has to keep track of all the free blocks present in the disk. There are mainly two approaches by using which, the free blocks in the disk are managed.

## **1. Bit Vector**

In this approach, the free space list is implemented as a bit map vector. It contains the number of bits where each bit represents each block.

If the block is empty then the bit is 1 otherwise it is 0. Initially all the blocks are empty therefore each bit in the bit map vector contains 1.

LAs the space allocation proceeds, the file system starts allocating blocks to the files and setting the respective bit to 0.

## **2. Linked List**

It is another approach for free space management. This approach suggests linking together all the free blocks and keeping a pointer in the cache which points to the first free block.

Therefore, all the free blocks on the disks will be linked together with a pointer. Whenever a block gets allocated, its previous free block will be linked to its next free block.