UNIT – 5

FILE SYSTEM INTERFACE AND OPERATIONS-

Access methods,

Directory Structure,

Protection,

File System Structure,

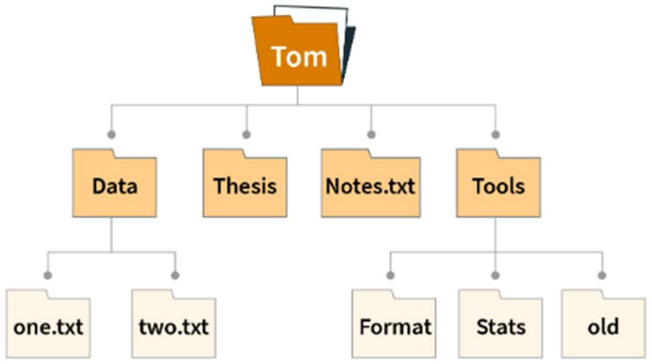
Allocation methods,

Free-space Management.

Usage of open, create, read, write, close, seek, stat, ioctl system calls.

5.1 File System concepts:

- A file system in OS dictates how the contents of a storage medium are stored and organized.
- These storage media (such as secondary memory, external drives, etc) could be computer secondary memory, flash memory, etc.
- The contents are either files or directories.
- Most of the time, a storage device has a number of partitions.
- Each of these partitions is formatted with an empty filesystem for that device.
- A filesystem helps in separating the data on the storage into comparatively smaller and simpler segments.
- These chunks are files and directories.
- The filesystem also provides for storing data related to files, such as their name, extension, permissions, etc.



a) Attributes:

A file is named, for the convenience of its human users, and is referred to by its name. A name is usually a string of characters, such as example.c. Some systems differentiate between uppercase and lowercase characters in names,

A file's attributes vary from one operating system to another but typically consist of these:

• Name - The symbolic file name is the only information kept in human readable form.
• Identifier - This unique tag, usually a number, identifies the file within the file system; it is the non-human-readable name for the file.
• Type - This information is needed for systems that support different types of files.
• Location - This information is a pointer to a device and to the location of the file on that device.
• Size - The current size of the file (in bytes, words, or blocks) and possibly the maximum allowed size are included in this attribute.
• Protection - Access-control information determines who can do reading, writing, executing, and so on.
• Time, date, and user identification - This information may be kept for creation, last modification, and last use. These data can be useful for protection, security, and usage monitoring.

b) File operations:

A file is an abstract data type. To define a file properly, we need to consider the operations that can be performed on files. The operating system can provide system calls to create, write, read, reposition, delete, and truncate files. Let's examine what the operating system must do to perform each of these six basic file operations. It should then be easy to see how other similar operations, such as renaming a file, can be implemented.

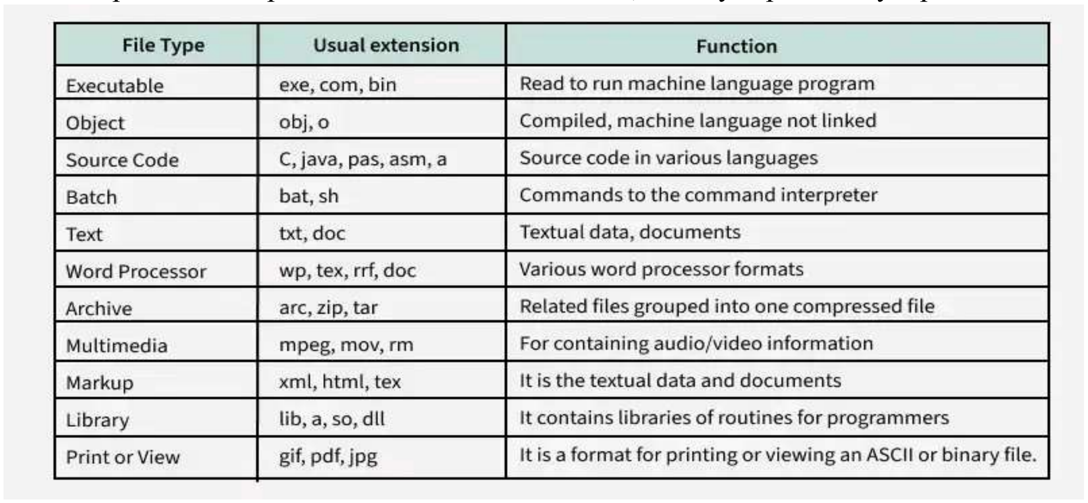
• Creating a file:

- Two steps are necessary to create a file. First, space in the file system must be found for the file.

- **Writing a file.**
  - To write a file, we make a system call specifying both the name of the file and the information to be written to the file.
  - Given the name of the file, the system searches the directory to find the file's location.
  - The system must keep a write pointer to the location in the file where the next write is to take place.
  - The write pointer must be updated whenever a write occurs.
- **Reading a file.**
  - To read from a file, we use a system call that specifies the name of the file and where (in memory) the next block of the file should be put.
  - Again, the directory is searched for the associated entry, and the system needs to keep a read pointer to the location in the file where the next read is to take place.
  - Once the read has taken place, the read pointer is updated.
  - Because a process is usually either reading from or writing to a file, the current operation location can be kept as a per-process current file - position pointer.
  - Both the read and write operations use this same pointer, saving space and reducing system complexity.
- **Repositioning within a file.**
  - The directory is searched for the appropriate entry, and the current-file-position pointer is repositioned to a given value.
  - Repositioning within a file need not involve any actual I/O. This file operation is also known as a file seek.
- **Deleting a file.**
  - To delete a file, we search the directory for the named file. Having found the associated directory entry,
  - we release all file space, so that it can be reused by other files, and erase the directory entry.
- **Truncating a file.**
  - The user may want to erase the contents of a file but keep its attributes.
  - Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged.

c) File Types:

- If an operating system recognizes the type of a file, it can then operate on the file in reasonable ways.
- A common technique for implementing file types is to include the type as part of the file name.
- The name is split into two parts-a name and an extension, usually separated by a period character.



File Type	Usual extension	Function
Executable	exe, com, bin	Read to run machine language program
Object	obj, o	Compiled, machine language not linked
Source Code	C, java, pas, asm, a	Source code in various languages
Batch	bat, sh	Commands to the command interpreter
Text	txt, doc	Textual data, documents
Word Processor	wp, tex, rrf, doc	Various word processor formats
Archive	arc, zip, tar	Related files grouped into one compressed file
Multimedia	mpeg, mov, rm	For containing audio/video information
Markup	xml, html, tex	It is the textual data and documents
Library	lib, a, so, dll	It contains libraries of routines for programmers
Print or View	gif, pdf, jpg	It is a format for printing or viewing an ASCII or binary file.

d) File Structure:

- A File Structure needs to be predefined format in such a way that an operating system understands
- It has an exclusively defined structure, which is based on its type.
- Three types of files structure in OS:
- A text file: It is a series of characters that is organized in lines.
- An object file: It is a series of bytes that is organized into blocks.
- A source file: It is a series of functions and processes.

5.2 File Access Methods:

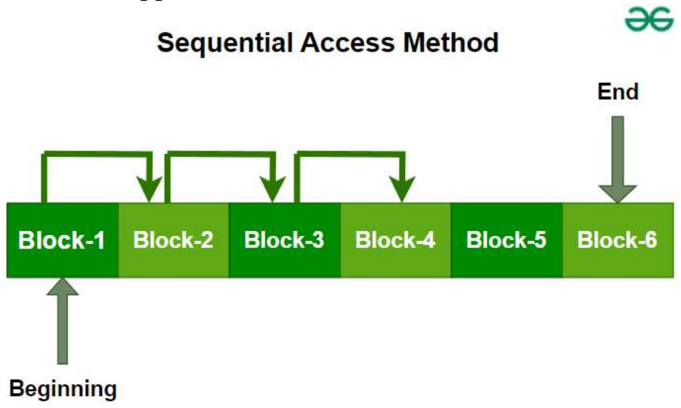
- File access methods in an operating system are the techniques and processes used to read from and write to files stored on a computer's storage devices.
- There are several ways to access this information in the file. Some systems provide only one access method for files.
- Other systems, such as those of IBM, support many access methods, and choosing the right one for a particular application is a major design problem.
- These methods determine how data is organized, retrieved, and modified within a file system. Understanding file access methods is crucial for efficient data management and system performance.
- In this article, we are going to discuss different types of methods to access the file.

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

1. Sequential-Access
2. Direct Access
3. Index sequential Method

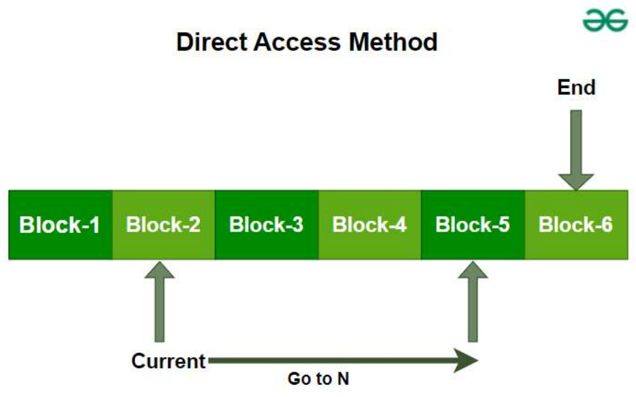
Sequential Access:

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



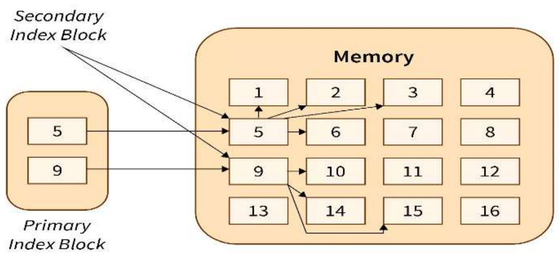
Direct Access Method:

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



Index Sequential method:

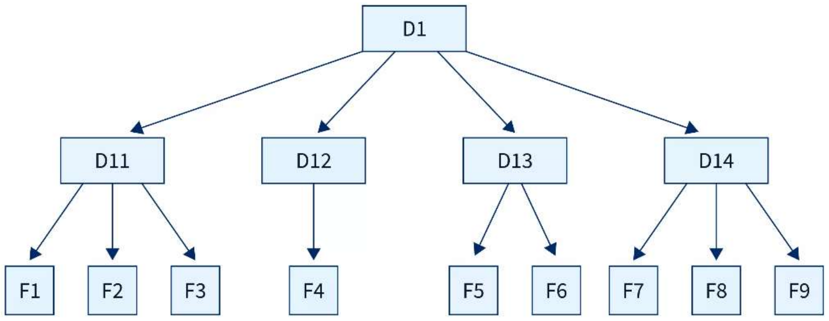
- It is the other method of accessing a file that is built on the top of the sequential access method.
- These methods construct an index for the file.
- The index, like an index in the back of a book, contains the pointer to the various blocks.
- To find a record in the file, we first search the index,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



Indexed Sequential Access Of File

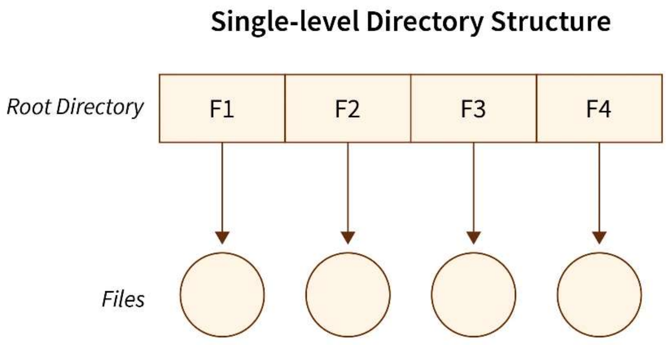
5.3 Directory Structure:

- A directory is a container that stores files and folders, organizing them hierarchically.
- The Directory Structure in OS manages entries of files, including file names, locations, protection info, and more. This structure enables efficient file retrieval.



a) Single Level Directory:

- The single-level directory structure is the simplest and easiest directory structure out of all the other directories.
- In this directory structure, all the folders/files are contained under the same directory which is called the root directory.
- The single-level directory structure gathers all the files under one directory or the root directory, this makes it easy to support and understand.



The Advantages

- The implementation of a single-level directory structure is simple and easier as compared to other directory structures in OS.
- If the file size is smaller, then the searching of such files with the single-level directory structure becomes simpler.
- The single-level directory structure allows the operations such as searching, creation, deletion and updating as well.

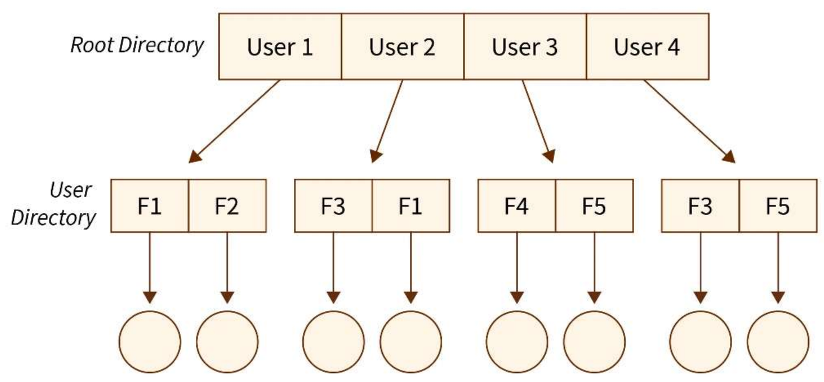
The Disadvantages

- As several users can log in at the same system to log their files maintaining a unique name becomes difficult leading to a collision.
- This also means that if the file with the same name is created then the old file will get destroyed first, then the new file (having the same name) created will replace it.
- If the size of the files is bigger then searching the files in one root directory of the single-level directory structure will become time-consuming and hence difficult.
- The single-level directory structure restricts the grouping of the same type of files together.

b) 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-level Directory Structure



The Advantages:

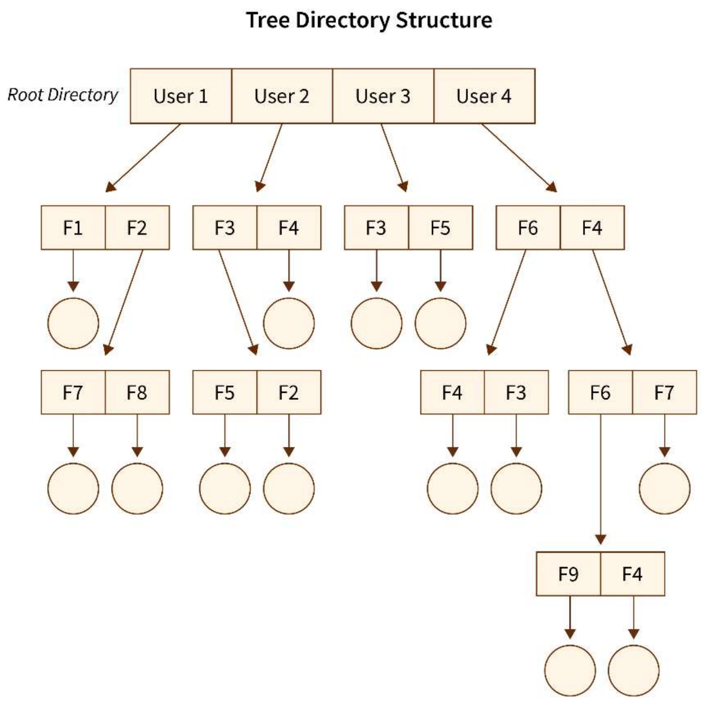
- In the two-level directory structure in OS different users have the right to have the same directory as well as a file name as the user has its own USD which can give a filename that can match other users but won't cause an issue.
- We can also see that searching for files become much simpler.
- As we have a user-defined directory this also provides privacy related to files stored as no user can enter the other user's directory without permission.
- In a two-level directory structure, we cannot group the files which are having the same name into a single directory for a specific user.

The Disadvantages:

- In a two-level directory structure, a user is not allowed to share files with other users.
- We also find that scalability is not present in a two-level directory structure as two files of the same type cannot be grouped together in the same user.
- Here users cannot create subdirectories only one user file directory can be defined under one master file directory.

c) **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.



The Advantages:

- In the Tree-structured directory structure searching is quite effective where we use the current working concept that is we can access the file by using two kinds of paths that are either absolute or relative.
- Here we can group the same type of files into one directory.
- In this directory structure the chances of collision of names/types etc are less and hence we can say that the directory structure in OS is scalable.

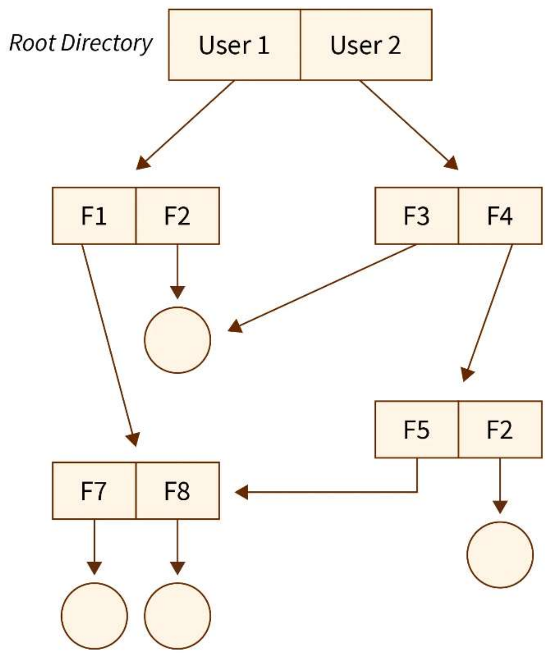
The Disadvantages:

- In the tree-structure directory structure in OS the files cannot be shared between users. Also, the users cannot modify/update the root directory of other users.
- This directory structure in OS as we have to go under multiple directories to access a file we can say that it is said to be inefficient.
- Here each file does not fit into the hierarchal model and so we have to save the files into multiple directories.

d) 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.

Acyclic Graph Directory Structure



The Advantages:

- ♦ In the Acyclic Graph directory structure in OS we can share files between users.
- ♦ Here we can search the files easily as compared to the tree-structured directory structure as here we have different-different paths to one file.

The Disadvantages:

- ♦ In the Acyclic Graph directory structure in OS as we can share the files via linking, so there are chances that in the case when we want to delete a file in a directory it may create a problem.
  - o Also, even if the link is a soft link then after deleting the file we are left with a dangling/suspended point of the link.
  - o But in the case of hardlink, when we delete a file we have to vanish all the references associated with it, which can lead to issues associated with referring back to files in case a requirement arises.

e) General-Graph Directory Structure

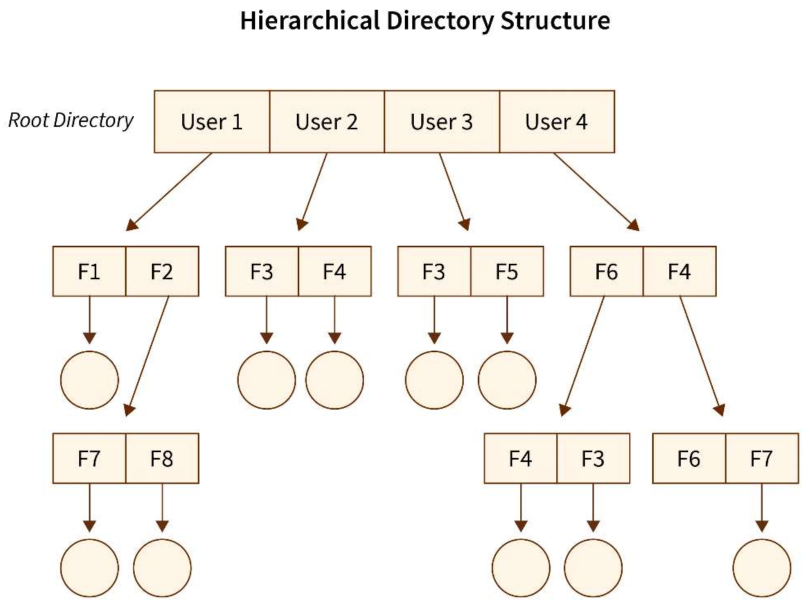
- ♦ Unlike the acyclic-graph directory, which avoids loops, the general-graph directory can have cycles, meaning a directory can contain paths that loop back to the starting point.
- ♦ This can make navigating and managing files more complex.

The Absolute Path:

Here, the path of the desired files can be determined by considering the root directory as the base directory.

The Relative Path:

Here, the path of the desired files can be determined by two choices that are, either the file that needs to be retrieved from its directory is considered the base directory, or the user's directory is considered as the base directory.



The Advantages:

- ♦ The General Graph directory structure allows the cycle or creation of a directory within a directory.
- ♦ This directory structure is known to be a flexible version compared to other directory structures.

The Disadvantages:

- ♦ The main issue that can overpower this structure is to calculate the total size or space that the directories will take up.
- ♦ As this directory structure allows the creation of multiple sub-directories a lot of garbage collection can be required.
- ♦ If compared to the other directory structure in OS the General Graph directory structure is a costly structure to be chosen.

5.4 Protection:

Protection refers to a mechanism which controls the access of programs, processes, or users to the resources defined by a computer system. We can take protection as a helper to multi programming operating system, so that many users might safely share a common logical name space such as directory or files.

Need for Protection:

- To prevent the access of unauthorized users
- To ensure that each active programs or processes in the system uses resources only as the stated policy
- To improve reliability by detecting latent errors

Role of Protection:

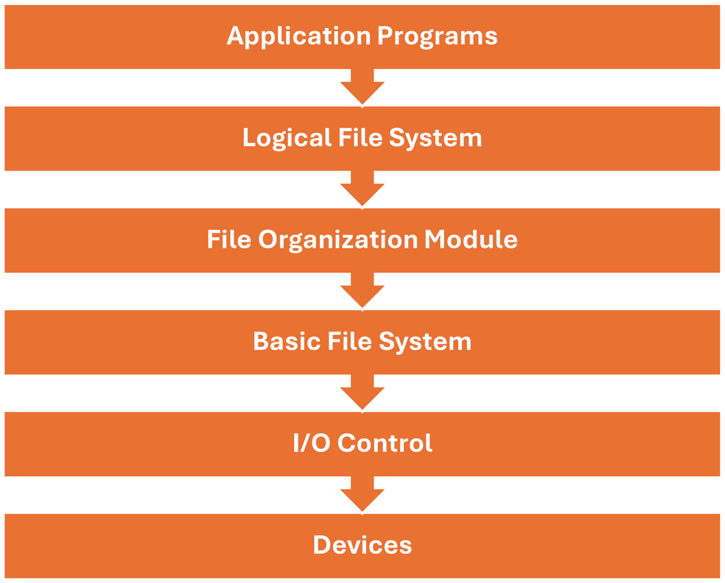
- The role of protection is to provide a mechanism that implement policies which defines the uses of resources in the computer system.
- Some policies are defined at the time of design of the system, some are designed by management of the system and some are defined by the users of the system to protect their own files and programs.
- Every application has different policies for use of the resources and they may change over time so protection of the system is not only concern of the designer of the operating system.
- Application programmer should also design the protection mechanism to protect their system against misuse.
- Policy is different from mechanism. Mechanisms determine how something will be done and policies determine what will be done.
- Policies are changed over time and place to place. Separation of mechanism and policy is important for the flexibility of the system.

Advantages of system protection in an operating system:

- Ensures the security and integrity of the system
- Prevents unauthorized access, misuse, or modification of the operating system and its resources
- Protects sensitive data
- Provides a secure environment for users and applications
- Prevents malware and other security threats from infecting the system
- Allows for safe sharing of resources and data among users and applications
- Helps maintain compliance with security regulations and standards
- Disadvantages of system protection in an operating system:
- Can be complex and difficult to implement and manage
- May slow down system performance due to increased security measures
- Can cause compatibility issues with some applications or hardware
- Can create a false sense of security if users are not properly educated on safe computing practices
- Can create additional costs for implementing and maintaining security measures.

5.5 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.



- 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.

5.6 File Allocation methods:

- When a hard drive is formatted, a system has numerous tiny spaces to store files called blocks.
- The operating system stores data in memory blocks using various file allocation methods, which enables the hard disk to be used efficiently and the file to be accessed.

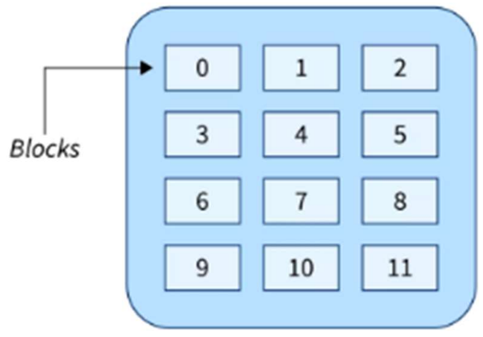
a) Contiguous File Allocation:

In contiguous file allocation, the block is allocated in such a manner that all the allocated blocks in the hard disk are adjacent.

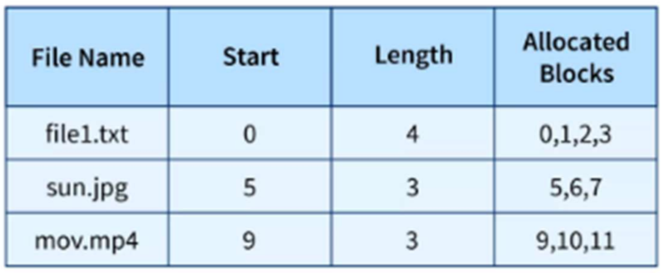
Assuming a file needs 'n' number of blocks in the disk and the file begins with a block at position 'x', the next blocks to be assigned to it will be x+1,x+2,x+3,...,x+n-1 so that they are in a contiguous manner.

Example:

Contiguous Allocation



Hard Disk



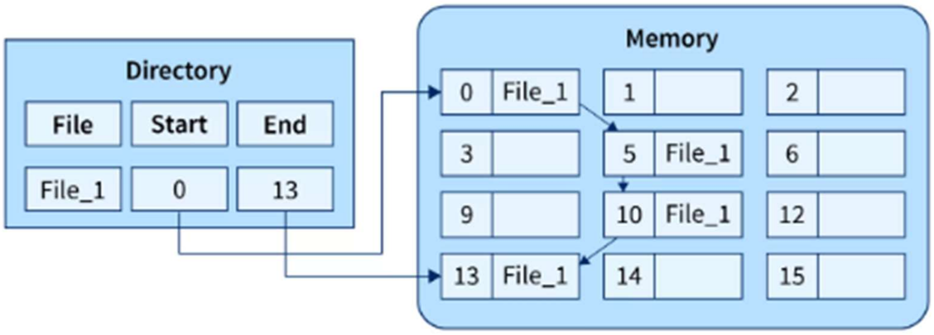
File Name	Start	Length	Allocated Blocks
file1.txt	0	4	0,1,2,3
sun.jpg	5	3	5,6,7
mov.mp4	9	3	9,10,11

Directory

- In the above image on the left side, we have a memory diagram where we can see the blocks of memory.
- At first, we have a text file named file1.txt which is allocated using contiguous memory allocation, it starts with the memory block 0 and has a length of 4 so it takes the 4 contiguous blocks 0,1,2,3. Similarly,
- We have an image file and video file named sun.jpg and mov.mp4 respectively, which you can see in the directory that they are stored in the contiguous blocks. 5,6,7 and 9,10,11 respectively.

b) Linked File Allocation:

- The Linked file allocation overcomes the drawback of contiguous file allocation.
- Here the file which we store on the hard disk is stored in a scattered manner according to the space available on the hard disk.
- Now, you must be thinking about how the OS remembers that all the scattered blocks belong to the same file.
- So as the name linked File Allocation suggests, the pointers are used to point to the next block of the same file, therefore along with the entry of each file each block also stores the pointer to the next block.

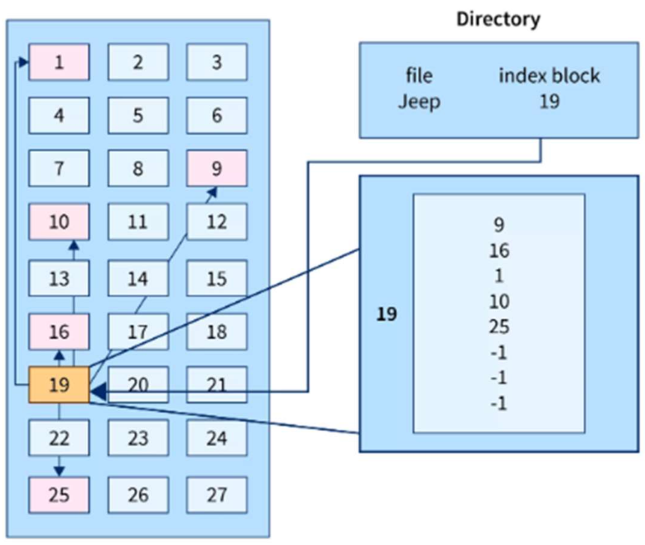


- In the above image on the right, we have a memory diagram where we can see memory blocks.
- On the left side, we have a directory where we have the information like the address of the first memory block and the last memory block.
- In this allocation, the starting block given is 0 and the ending block is 15, therefore the OS searches the empty blocks between 0 and 15 and stores the files in available blocks, but along with that it also stores the pointer to the next block in the present block.
- Hence it requires some extra space to store that link.

c) Indexed File Allocation:

- The indexed file allocation is somewhat similar to linked file allocation as indexed file allocation also uses pointers but the difference is here all the pointers are put together into one location which is called index block.
- That means we will get all the locations of blocks in one index file.
- The blocks and pointers were spread over the memory in the Linked Allocation method, where retrieval was accomplished by visiting each block sequentially.
- But here in indexed allocation, it becomes easier with the index block to retrieve.

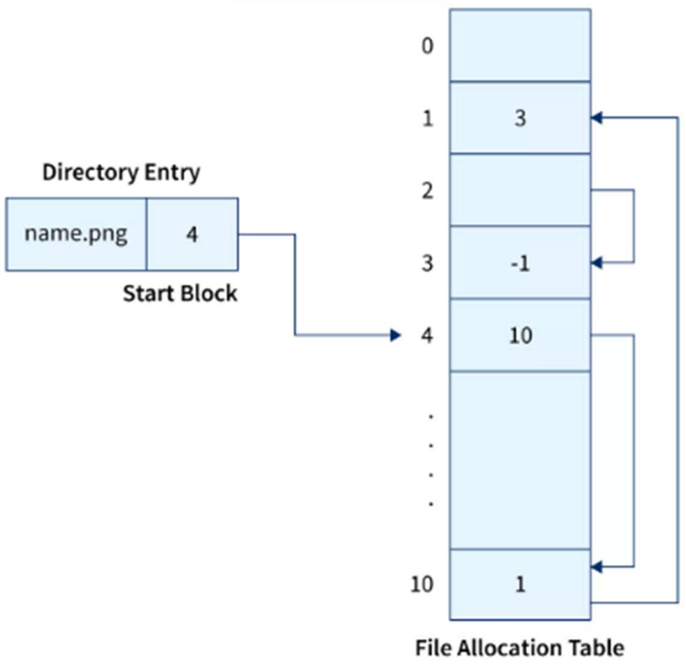
Example:



- As shown in the diagram below block 19 is the index block which contains all the addresses of the file named text1.
- In order, the first storage block is 9, followed by 16, 1, then 10, and 25.
- The negative number -1 here denotes the empty index block list as the file text1 is still too small to fill more blocks.

d) File Allocation Table (FAT):

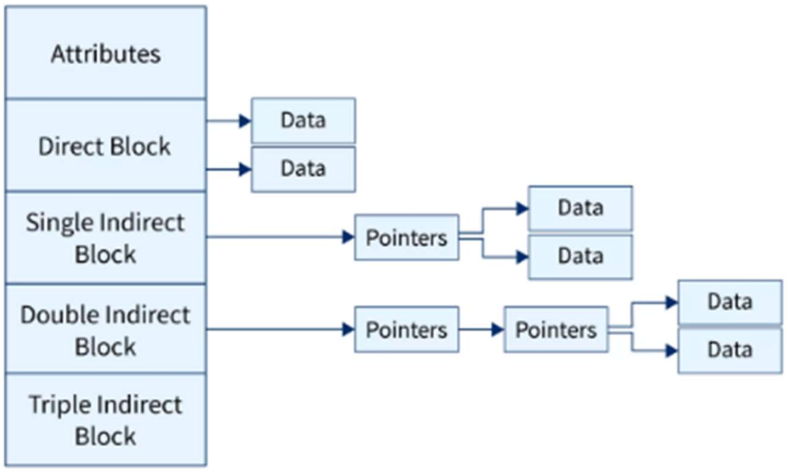
- The File Allocation Table (FAT) overcomes the drawback of Linked File allocation.
- The random access of a particular block is not possible in the linked file allocation.
- To access a particular block, it is necessary to access all its previous blocks.
- Let's see how File Allocation Table works.



- In the file allocation table, all disk block links are collected and maintained.
- Here the number of head seeks is reduced by caching the file allocation table so that the head does not need to go through all the disk blocks to access one particular block.
- The whole process of randomly accessing any block using FAT is completed by reading the desired entry of a block from the file allocation table and accessing that particular block.

e) Inode:

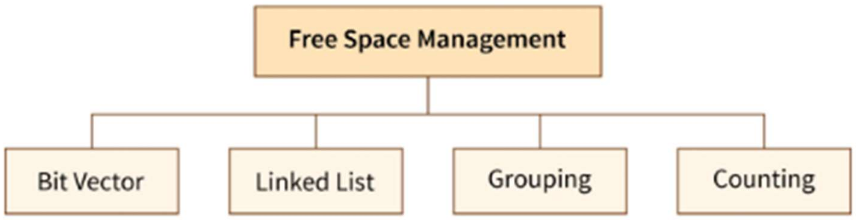
- In Operating systems based on UNIX, every file is indexed using Inode(information-node).
- The inode is the special disk block that is created with the creation of the file system.
- This special block is used to store all the information related to the file like name, authority, size, etc along with the pointers to the other blocks where the file is stored.



- In this special block, some of its space is used to store the information related to the field as mentioned above and the remaining space is used to store the addresses of blocks that contain the actual file.
- The first few pointers in Inode point to direct blocks, i.e they contain the addresses of the disk blocks containing the file data.
- A few pointers in inode point to the indirect blocks.
- There are three types of indirect blocks: single indirect, double indirect, and triple indirect.
- A single indirect block contains nothing but the address of the block containing the file data, not the file itself as shown in the figure below.
- Furthermore, the double indirect block points to the pointers which again point to the pointers which point to the data blocks.
- Further, it goes in a similar way for triple indirect block.

5.7 Free Space Management:

- There is a system software in an operating system that manipulates and keeps a track of free spaces to allocate and de-allocate memory blocks to files, this system is called a file management system in an operating system".
- There is a free space list in an operating system that maintains the record of free blocks.
- When a file is created, the operating system searches the free space list for the required space allocated to save a file.
- While deletion a file, the file system frees the given space and adds this to the free space list.

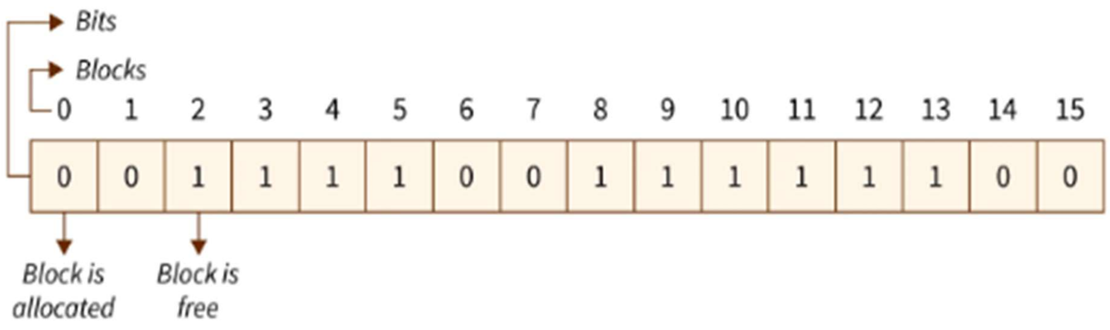


a) Bitmap or Bit Vector:

- A bit vector is a most frequently used method to implement the free space list. A bit vector is also known as a Bit map.
- It is a series or collection of bits in which each bit represents a disk block.
- The values taken by the bits are either 1 or 0.
- If the block bit is 1, it means the block is empty and if the block bit is 0, it means the block is not free.
- It is allocated to some files.
- Since all the blocks are empty initially so, each bit in the bit vector represents 0.

Example:

- Given below is a diagrammatic representation of a disk in which there are 16 blocks.
- There are some free and some occupied blocks present.
- The upper part is showing block number. Free blocks are represented by **1** and occupied blocks are represented by **0**.



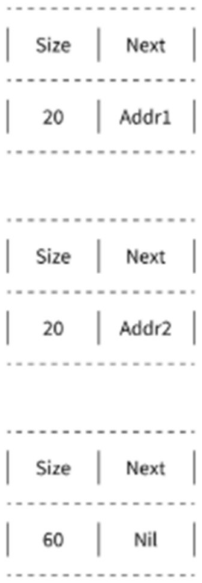
"Free block number" can be defined as that block which does not contain any value, i.e., they are free blocks. The formula to find a free block number is :

[Block number = (number of bits per words)*(number of 0-value word) + Offset of first 1 bit]

- We will consider the first 8 bits group (0011110) to constitute a non-zero word since all bits are not 0 here.
- Non-zero word is that word that contains the bit value 1 (block that is not free).
- Here, the first non-zero word is the third block of the group. So, the offset will be 3.

Hence, the block number =8∗0+3=3

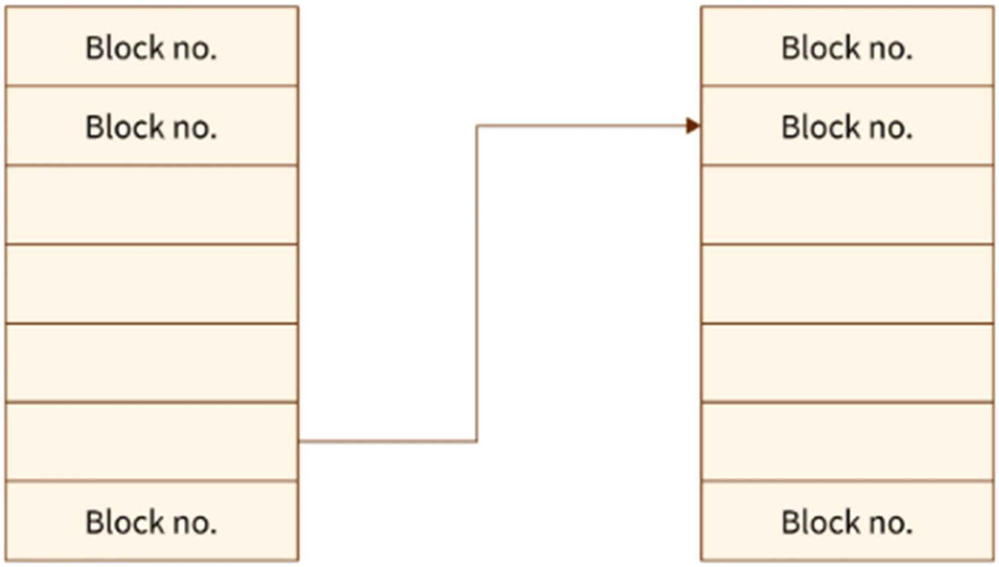
b) Linked List
- A linked list is another approach for free space management in an operating system.
- In it, all the free blocks inside a disk are linked together in a linked list.
- These free blocks on the disk are linked together by a pointer.
- These pointers of the free block contain the address of the next free block and the last pointer of the list points to null which indicates the end of the linked list.
- This technique is not enough to traverse the list because we have to read each disk block one by one which requires I/O time.



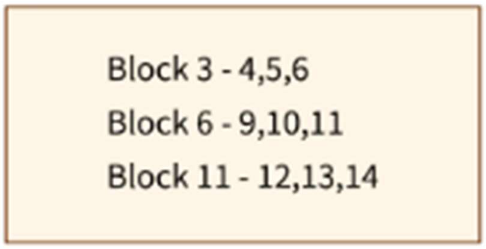
- In the above example, we have three blocks of free memory, each represented by a node in the linked list.
- The first block has 20 bytes of free memory, the second block has 20 bytes of free memory, and the third block has 60 bytes of free memory.
- The operating system can use this linked list to allocate memory blocks to processes as needed.

c) **Grouping:**

- The grouping technique is also called the "modification of a linked list technique".
- In this method, first, the free block of memory contains the addresses of the n-free blocks.
- And the last free block of these n free blocks contains the addresses of the next n free block of memory and this keeps going on.
- This technique separates the empty and occupied blocks of space of memory.



- Suppose we have a disk with some free blocks and some occupied blocks.
- The free block numbers are 3,4,5,6,9,10,11,12,13,, and 14 and occupied block numbers are 1,2,7,8,15, and 16 i.e., they are allocated to some files.



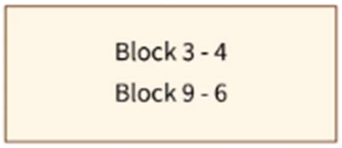
- When the "grouping technique" is applied, block 3 will store the addresses of blocks 4, 5, and 6 because block 3 is the first free block.
- In the same way, block 6 will store the addresses of blocks 9, one 0, and one 1 because block 6 is the first occupied block.

d) Counting

In memory space, several files are created and deleted at the same time.
For which memory blocks are allocated and de-allocated for the files.
Creation of files occupy free blocks and deletion of file frees blocks.
When there is an entry in the free space, it consists of two parameters- "address of first free disk block (a pointer)" and "a number 'n'".

Example

Let us take an example where a disk has 16 blocks in which some blocks are empty and some blocks are occupied as given below :



Block 3 - 4

Block 9 - 6

- When the "counting technique" is applied, the block number 3 will represent block number 4 because block 3 is the first free block.
- Then, the block stores the number of free blocks
- i.e. - there are 4 free blocks together. In the same way, the first occupied block number 9 will represent block number 10 and keeps the number of rest occupied blocks i.e.- there are 6 occupied blocks as shown in the above figure.

5.8 Usage of open, create, read, write, close, seek, stat, ioctl system calls:

Types of System Calls	**Windows**	**Linux**

Process Control	CreateProcess()
	ExitProcess()
	WaitForSingleObject()
		fork()
		exit()
		wait()

File Management	CreateFile()
	ReadFile()
	WriteFile()
	CloseHandle()
		open()
		read()
		write()
		close()

Device Management	SetConsoleMode()
	ReadConsole()
	WriteConsole()
		ioctl()
		read()
		write()

Information Maintenance	GetCurrentProcessID()
	SetTimer()
	Sleep()
		getpid()
		alarm()
		sleep()

Communication	CreatePipe()
	CreateFileMapping()
	MapViewOfFile()
		pipe()
		shmget()
		mmap()