1. Introduction: File System

A file is a collection of related information that is recorded on secondary storage. Commonly, file represents programs and data. Data files may be numeric, alphabetic or binary. In general, a file is a sequence of bits, bytes, lines or records. There are many different types of files such as *data files, text files, program files, directory files,* and so on. Different types of files store different types of information. For example, program files store programs, whereas text files store text.

A **file system** is used to control how data is stored and retrieved. The file system consists of two distinct parts:

- A collection of ***files***, each storing related data and
- A ***directory structure***, which organizes and provides information about all the files in the system.

The file system permits users to create data collections, called files, with desirable properties, such as

- **Long-term existence:** Files are stored on disk or other secondary storage for long term and do not disappear when a user logs off.
- **Sharable between processes:** Files have names and can be shared among different processes with associated access permissions that permit controlled sharing.
- **Structure:** Depending on the file system, a file can have an internal structure that is convenient for particular applications. In addition, files can be organized into hierarchical or more complex structure to reflect the relationships among files.

A **file management system** is that set of system software that provides services to users and applications related to the use of files i.e. a user or application may access files through the file management system. It has following objectives:

- To meet the data-management needs and requirements of the user, which includes storage of data
- To optimize performance, both from the system point of view (through put) and the users point of view (response time)
- To provide I/O support for a varieties of storage device.
- To minimize or eliminate the potential for lost or destroy of data.
- To provide I/O support for multiple users in case of multiple-user system.

The minimal sets of requirements of file management system are:

- Each user should be able to create, delete and change files.
- May have controlled access to other user's files.
- Control what types of accesses are allowed to the user's files.
- Should be able to restructure the user's files in a form appropriate to the problem
- Should be able to move data between files.
- Should be able to backup and recover the user's files in case of damage.
- Should be able to access the user's files by a symbolic name.

A Window Operating system supports following file systems:

File System	Meaning
FAT	The MS-DOS operating system introduced the File Allocation Table system of keeping track of file entries and free clusters. Filenames where restricted to eight characters with an addition three characters signifying the file type.

	The FAT tables were stored at the beginning of the storage space.
FAT32	An updated version of the FAT system designed for Windows 98. It supports file compression and long filenames
NTFS	Windows NT introduced the NT File System, designed to be more efficient at handling files than the FAT system. It spreads file tables throughout the disk, beginning at the center of the storage space. It supports file compression and long filenames

Linux operating system uses EXT file system. It uses indexed allocation where each file has its own index block or inode. A file's inod links to other indirect blocks.

1.1 File Naming

A filename is a string used to uniquely identify a file stored on the file system of a computer. Files are an abstraction mechanism. They provide a way to store information on the disk and read it back later. This must be done in such a way as to shield the user from the details of how and where the information is stored, and how the disks actually work. Probably the most important characteristic of any abstraction mechanism is the way the objects being managed are named, when a process creates a file; it gives the file a name. When the process terminates, the file continues to exist and can be accessed by other processes using its name.

The name of a file is called **filename**. All files have names. Different operating system imposes different restrictions on filenames. Many system, allow a filename extension that consists of one or more characters following the proper filename. The *filename extension* usually indicates what type of file it is. Every filename has two parts: *primary filename* and *secondary name* and both are separated by dot (.) symbol. Primary file names are given by the user and the secondary name are optional, generally by the system. The secondary filename indicates the nature of file i.e. indicates the type of file.

Example: balance_sheet.xls, operating_system.doc, etc Some extension names are listed as follows:

Extension name	**Full forms**	**Meanings**
File.doc	Document	Word document file
File.xls	Excel	Excel spreadsheet file
File.ppt	PowerPoint	PowerPoint file
File.bak	Backup	Backup file
File.c		C source Program
File.gif	Graphical interchange format	Photo file
File.jpg	Joint photographic group	Photo file
File.jpeg	Joint photographic expert group	Photo file
File.txt	Text	General Text format files
File.pdf	Portable document format	Portable document format
File.ps	PostScript	PostScript
File.zip		Compressed archive

Rules for naming a file:

Most operating systems prohibit the use of certain characters in a filename and impose a limit on the length of a filename. File naming is an important component of file management. Here are the basic rules and recommendations.

- Filename should consist different symbols such as alphabets (Capital or small), digits (0 to 9) or combination of both.
- Only use hyphens and underscores. Avoid any other punctuation marks, accented letters, non-Latin letters, and other non-standard characters such as forward and back slashes, colon, semi-colon, asterisks, angle brackets or brackets.
- Number of characters in a filename (primary name) should not exceed 8 characters (in DOS) or 256 characters (in GUI mode OS). However it should not too length.
- File names should end in a three-letter file extension preceded by a period, such as .CR2, .JPG, .TIFF, etc.

1.2 File Attributes

Every file has a name and its data. In addition, all operating systems associate other information with each file, for example, the date and time the file was created and the file's size. We will call these extra items the file's **attributes** although some people called them **metadata**. The list of attributes varies considerably from system to system.

Attribute	**Meaning**
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in
	listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed

Time of last change	Date and time the file has last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

1.3 File Operations

Files exist to store information and allow it to be retrieved later. Different systems provide different operations to allow storage and retrieval. Below is a discussion of the most common system calls relating to files.

I. **Create.** The file is created with no data. The purpose of the call is to announce that the file is coming and to set some of the attributes.

II. **Delete**. When the file is no longer needed, it has to be deleted to free up disk space. A system call for this purpose is always provided.

III. **Open.** Before using a file, a process must open it. The purpose of the open call is to allow the system to fetch the attributes and list of disk addresses into main memory for rapid access on later calls.

IV. **Close.** When all the accesses are finished, the attributes and disk addresses are no longer needed, so the file should be closed to free up some internal table space. Many systems encourage this by imposing a maximum number of open files on processes. A disk is written in blocks, and closing a file forces writing of the file's last block, even though that block may not be entirely full yet.

V. **Read.** Data are read from file. Usually, the bytes come from the current position. The caller must specify how much data are needed and must also provide a buffer to put them in.

VI. **Write.** Data are written to the file, again, usually at the current position. If the current position is the end of the file, the file's size increases. If the current position is in the middle of the file, existing data are overwritten and lost forever.

VII. **Append.** This call is a restricted form of write. It can only add data to the end of the file. Systems that provide a minimal set of system calls do not generally have append, but many systems provide multiple ways of doing the same thing, and these systems sometimes have append.

VIII. **Seek**. For random access files, a method is needed to specify from where to take the data. One common approach is a system call, seek, that repositions the file pointer to a specific place in the file. After this call has completed, data can be read from, or written to, that position.

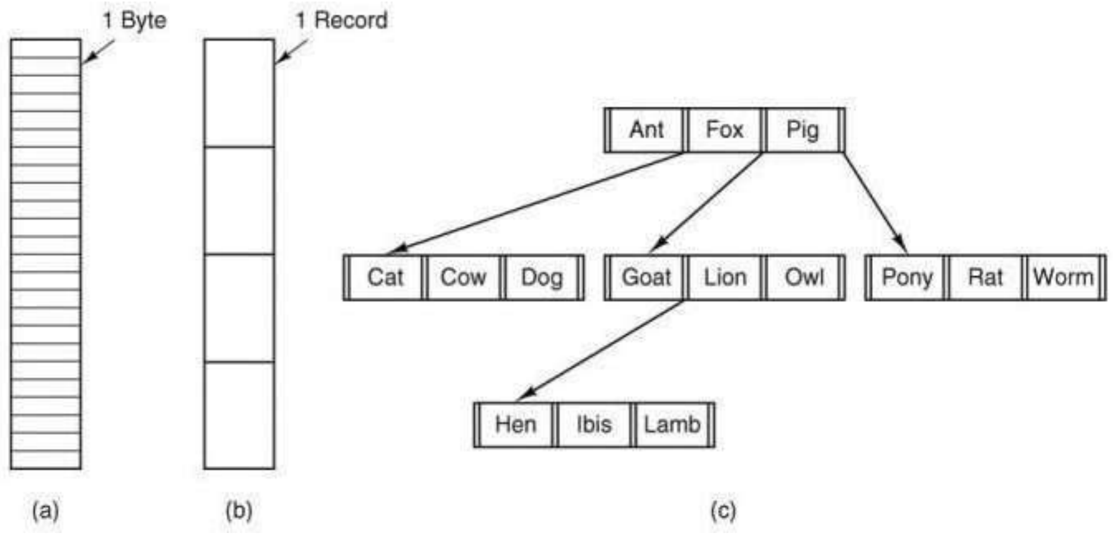
IX. **Get attributes**. Processes often need to read file attributes to do their work. For example, the UNIX make program is commonly used to manage software development projects consisting of many source files. When make is called, it examines the modification times of all the source and object files and arranges for the minimum number of compilations required to bring everything up to date. To do its job, it must look at the attributes, namely, the modification times.

X. **Set attributes.** Some of the attributes are user settable and can be changed after the file has been created. This system call makes that possible. The protection mode information is an obvious example. Most of the flags also fall in this category.

XI. **Rename.** It frequently happens that a user needs to change the name of an existing file. This system call makes that possible. It is not always strictly necessary, because the file can usually be copied to a new file with the new name, and the old file then deleted.

XII. **Lock.** Locking a file or a part of a file prevents multiple simultaneous access by different process. For an airline reservation system, for instance, locking the database while making a reservation prevents reservation of a seat for two different travelers.

1.4 File Structure:



Three kinds of files. (a) Byte sequence. (b) Record sequence. (c) Tree.

a. Byte Sequence:

The file in Figure (a) is just an unstructured sequence of bytes. In effect, the operating system does not know or care what is in the file. All it sees are bytes. Any meaning must be imposed by user-level programs. Both UNIX and Windows 98 use this approach.

b. Record Sequence:

In this model, a file is a sequence of fixed-length records, each with some internal structure. Central to the idea of a file being a sequence of records is the idea that the read operation returns one record and the write operation overwrites or appends one record. As a historical note, when the 80-column punched card was king many (mainframe) operating systems based their file systems on files consisting of 80-character records, in effect, card images

c. Record Sequence:

In this organization, a file consists of a tree of records, not necessarily all the same length, each containing a key field in a fixed position in the record. The tree is sorted on the key field, to allow rapid searching for a particular key.

1.5 File Descriptor (FD)

File descriptor is an abstract indicator for accessing a file i.e. it is an index for an entry in a kernel-resident array data structure containing the details of open files. In other words a file descriptor is an unsigned integer used by a process to identify an open file. In POSIX (an OS) this data structure is called a ***file descriptor table***, and each process has its own file descriptor table. The process passes the file descriptor to the kernel through a system call, and the kernel

will access the file on behalf of the process. The process itself cannot read or write the file descriptor table directly.
On Linux, the set of file descriptors open in a process can be accessed under the path:

/proc/PID/fd/

where PID is the process identifier.

In simple words, when we open a file, the operating system creates an entry to represent that file and store the information about that opened file. So if there are 10 files opened then there will be 10 entries in OS (somewhere in kernel). These entries are represented by integers like (0, 1, 2.). This entry number is the file descriptor i.e. it is a unique integer that identifies an open *file* within a process.

File descriptor values: when the shell runs a program, it opens three files with file descriptors 0, 1, and 2. The default assignments for these descriptors are as follows:

0	Represents standard input.
1	Represents standard output.
2	Represents standard error.

These default file descriptors are connected to the terminal, so that if a program reads file descriptor 0 and writes file descriptors 1 and 2, the program collects input from the terminal and sends output to the terminal. As the program uses other files, file descriptors are assigned in ascending order.

2. File Organization

A file is a collection of data, usually stored on disk. A file enables us to divide our data into meaningful groups, for example, we can use one file to hold all of a company's product information and another to hold all of its personnel information.

The term "file organization" refers to the way in which data is stored in a file and the method by which it can be accessed i.e. File organization refers primarily to the logical arrangement (structuring) of data in a file system. The physical organization of the file on secondary storage depends upon the blocking strategy and the file allocation strategy.

While selecting a file organization, following criteria's is important:

- Rapid access for effective information retrieval
- Ease of update to aid in having up-to-date information.
- Economy of storage to reduce storage capacity.
- Simple maintenance to reduce cost and potential for error
- Reliability to assure confidence in the data.

There are three types of file organizations. They are;

- Sequential
- Relative and
- Indexed

I. Sequential

A sequential file is one in which the individual records can only be accessed sequentially, that is, in the same order as they were originally written to the file. New records are

always added to the end of the file. The order of the records is fixed. The records are stored and sorted in physical, contiguous blocks within each block the records are in sequence.
Records in these files can only be read or written sequentially.

Three types of sequential file are supported by this COBOL system:

- Record sequential
- Line sequential
- Printer sequential

II. **Relative**

A relative file is a file in which each record is identified by its ordinal position within the file (record 1, record 2 and so on). This means that records can be accessed randomly as well as sequentially. Because records can be accessed randomly, access to relative files is fast.

Within a relative file are numbered positions, called *cells*. These cells are of fixed equal length and are consecutively numbered from 1 to *n*, where 1 is the first cell, and *n* is the last available cell in the file. Each cell either contains a single record or is empty. Records in a relative file are accessed according to cell number. A cell number is a record's relative record number; its location relative to the beginning of the file.

III. **Indexed**

An indexed file is a file in which each record includes a primary key. To distinguish one record from another, the value of the primary key must be unique for each record. Records can then be accessed randomly by specifying the value of the record's primary key. Indexed file records can also be accessed sequentially. As well as a primary key, indexed files can contain one or more additional keys known as alternate keys. The value of a record's alternate key(s) does not have to be unique.

3. Blocking and Buffering

Block: It is a smallest amount of data that can be read from or written to secondary storage at one time i.e. any chunk of data that can be treated as a unit (for reading, writing, organizing). The process of grouping several components into one block is called **blocking**. **Clustering** is a grouping file components according to access behavior
Factors affecting block size:

- size of available main memory
- space reserved for programs (and their internal data space) that use the files
- size of one component of the block
- characteristics of the external storage device used

4. File Access Methods

When a file is used then the stored information in the file must be accessed and read into the memory of a computer system. Various mechanisms are provided to access a file from the operating system.

- Sequential access
- Direct Access
- Index Access

I. **Sequential Access:**

It is the simplest access mechanism, in which information's stored in a file are accessed in an order such that one record is processed after the other. For example editors and compilers usually access files in this manner.

II. Direct Access:

It is an alternative method for accessing a file, which is based on the disk model of a file, since disk allows random access to any block or record of a file. For this method, a file is viewed as a numbered sequence of blocks or records which are read / written in an arbitrary manner, i.e. there is no restriction on the order of reading or writing. It is well suited for Database management System.

III. Index Access:

In this method an index is created which contains a key field and pointers to the various blocks. To find an entry in the file for a key value, we first search the index and then use the pointer to directly access a file and find the desired entry. With large files, the index file itself may become too large to be keeping in memory. One solution is to create an index for the index file. The primary index file would contain pointers to secondary index files, which would point to the actual data items.

5. File Sharing

File sharing is the practice of sharing or offering access to digital information or resources, including documents, multimedia (audio/video), graphics, computer programs, images and e-books. It is the private or public distribution of data or resources in a network with different levels of sharing privileges.

File sharing is the public or private sharing of computer data or space in a network with various levels of access privilege.

File sharing allows a number of people to use the same file or file by some combination of being able to read or view it, write to or modify it, copy it, or print it. Typically, a file sharing system has one or more administrators. Users may all have the same or may have different levels of access privilege. File sharing can also mean having an allocated amount of personal file storage in a common file system.

File sharing can be done using several methods. The most common techniques for file storage, distribution and transmission include the following:

- Removable storage devices
- Centralized file hosting server installations on networks
- World Wide Web-oriented hyperlinked documents
- Distributed peer-to-peer networks

However there are two main issues while sharing files and they are: ***access rights*** and ***the management of simultaneous access.***

a) Access Right (Access Privilege)

The file system should provide a flexible tool for allowing extensive file sharing among users. There should be a number of options so that the way in which a particular file is accessed can be controlled. Typically, users or group of users are granted certain access rights to a file. Some access rights that can be assigned to a particular user for a particular file are:

- ***None:*** It means that the user may not even learn the existence of the file i.e. the user is not allowed to read the user directory that includes this file.
- ***Knowledge:*** It means that a user can determine that the file exists and who is the owner. The user is then able to request the owner for additional access rights.
- ***Execution:*** The user can load and execute a program but cannot copy it.
- ***Reading:*** The user can read the file for any purpose, including copying and execution.

- ***Appending:*** The user can add data to the file, often only at the end, but cannot modify or delete any of the file's contents. This right is useful in collecting data from a number of sources.
- ***Updating:*** The user can modify, delete, and add to the file's data. Updating normally includes writing the file initially, rewriting it completely or in part, and removing all or a portion of the data.
- ***Changing Protection:*** The user can change the access rights granted to other users. Typically this right is held only by the owner of the file.
- ***Deletion:*** The user can delete the file form the file system.

Access can be provided to the following class of the user:

- ***Specific users:*** Individual users who are designated by user ID.
- ***User groups:*** A set of users who are not individually defined.
- ***All:*** All users who have access to this system. These are public files.

b) Simultaneous access:

When access is granted to append or update a file to more than one user, the operating system or file management system must have a mechanism to control the simultaneous access to the file. A brute-force approach is to allow a user to lock the entire file when it is to be updated.

6. File control Block (FCB)

File control blocks (FCB) are data structures that hold information about a file. When an operating system needs to access a file, it creates an associated file control block to manage the file.

The structure of the file control block differs between operating systems, but most file control blocks include the following parts

- Filename
- Location of file on secondary storage
- Length of file
- Date and time or creation or last access

7. File permissions in linux:

- There are 3 access permissions used in file. They are read, write and execution.
- In linux system access permissions on files (read, write and execution) are given as numbers shown below;
- 4 represents read (r) permission on file.
- 2 represents write (w) permission on file.
- 1 represents execution (x) permission on file.

To change the file permission chmod command is generated. For example:

chmod 754 filename/directoryname

Above command put the read, write and execution permission for user. Read and execution permission for group and read permission for others.

8. ACM (Access Control Matrix)

The *access control matrix* is a matrix with each subject represented by a row, and each object represented by a column. An access control matrix is a tool that can describe the current protection state. The access matrix model is the policy for user authentication, and has several implementations such as access control lists (ACLs) and capabilities. It is used to describe which users have access to what objects.

The access matrix model consists of four major parts:

- A list of objects (file, piece of hardware, process)
- A list of subjects (processes and rights)
- A function T which returns an object's type
- The matrix itself, with the objects making the columns and the subjects making the rows

The access controls provided with an operating system typically authenticate principals using some mechanism such as passwords or Kerberos, and then mediate their access to files, communications ports, and other system resources. Their effect can often be modeled by a matrix of access permissions, with columns for files and rows f permission to read, w for permission to write, x for permission to execute a program, and (–)
for no access at all, as shown in Figure below.

	Operating system	Accounts program	Accounting data	Audit trail
Surya	rwx	rwx	rw	r
Daman	x	X	rw	-
Sunita	rx	R	r	r

In above example:

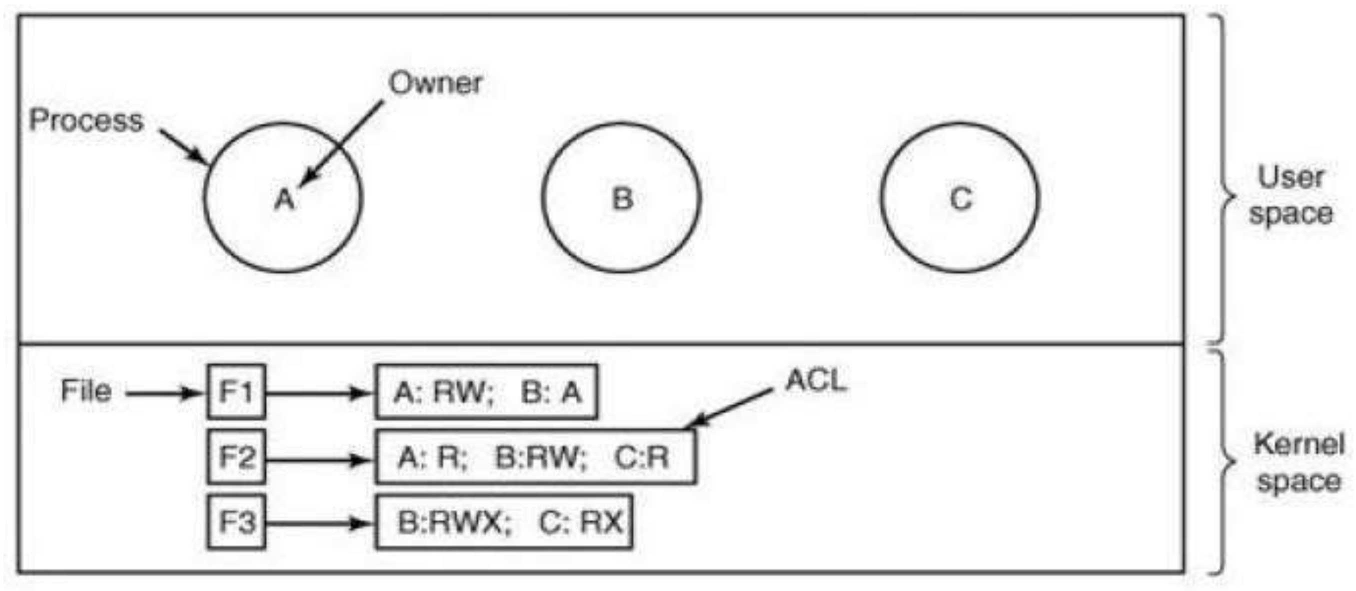
- Surya is the system administrator, and has universal access (except to the audit trail, which even he should only be able to read).
- Daman, the manager, needs to execute the operating system and application, but only through the approved interfaces — he mustn't have the abilit She also needs to read and write the data.
- Sunita, the auditor, can read everything.

Access control matrices (whether in two or three dimensions) can be used to implement protection mechanisms, as well as just model them. But they do not scale well. For instance, a bank with 50,000 staff and 300 applications would have an access control matrix of 15 million entries. This is inconveniently large. It might not only impose a performance problem but also be vulnerable to administrators' mi of storing and managing this information. The two main ways of doing this are to use groups
or roles to manage the privileges of large sets of users simultaneously, or to store the access control matrix either by columns (access control lists) or rows (capabilities, sometimes known as ―tickets‖) or certificates.

9. ACL (Access Control List)

In multi-user operating systems, files may be accessed by multiple users. Permission rights associated with folders (directories) and files are used to protect or restrict access to files. An ACL consists of a set of ACL entries. An ACL entry specifies the access permissions on the associated object for an individual user or a group of users as a combination of reading write and execute permissions.

In other words, The Access Control List (ACL) is the list of a system object's (file, folder or other network resources) security information that defines access rights for resources like users, groups, processes or devices. The object's security information is known as a permission, which controls resource access to view or modify system object contents. The most common privileges include the ability to read a file (or all the files in a directory), to write to the file or files, and to execute the file. Each ACL has one or more access control entries (ACEs) consisting of the name of a user or group of users. Generally, the system administrator or the object owner creates the access control list for an object.



Each file has an ACL associated with it.

- File F1 has two entries in its ACL (separated by a semicolon). The first entry says that any process owned by user "A" may read and write the file. The second entry says that any process owned by user "B" may read the file. All other accesses by these users and all accesses by other users are forbidden. Note that the rights are granted by user, not by process. As far as the protection system goes, any process owned by user "A" can read and write file F1. It does not matter if there is one such process or 100 of them. It is the owner, not the process ID that matters.

- File F2 has three entries in its ACL: "A", "B", and "C" can all read the file, and in addition B can also write it. No other accesses are allowed.

- File F3 is apparently an executable program, since "B" and "C" can both read and execute it. B can also write it.

10. Directories

Directory is a location for storing files in our computer i.e. it is a *filing cabinet* of data storage device in which data (file) is grouped and listed in a hierarchical manner for allowing direct user access to any file. It is used to organize files and folders into a hierarchical structure. The directory contains information about the files, including attributes, location, and ownership. It keeps track of file. The directory is itself a file, owned by the operating system and accessible by various file management routines.
A directory contains following information's:

- ***Basic information's:*** file name, file type, file organization

- ***Address information:*** volume (storage device e.g. C:, D:, etc), starting address, size used (current size of the file in bytes, words, or blocks), size allocated (maximum size of the file)
- ***Access control information's:*** Owner, Access information (user name, password), permitted actions (Read, Write, Execute, transmitting over a network i.e. share)
- ***Usage information's:*** date created, identity of creator, date last read access, identity of last reader, date last modified, identity of last modifier, date of last backup, current usage (information about current activity such as process or processes that have the file open, whether it is locked bya a process, etc)

I. Directory Hierarchy

Files are grouped into directories, and directories are organized in a hierarchy. It is also known as directory tree. At the top of the hierarchy is the ―root‖ d ―/‖. It can be shown in following figure.

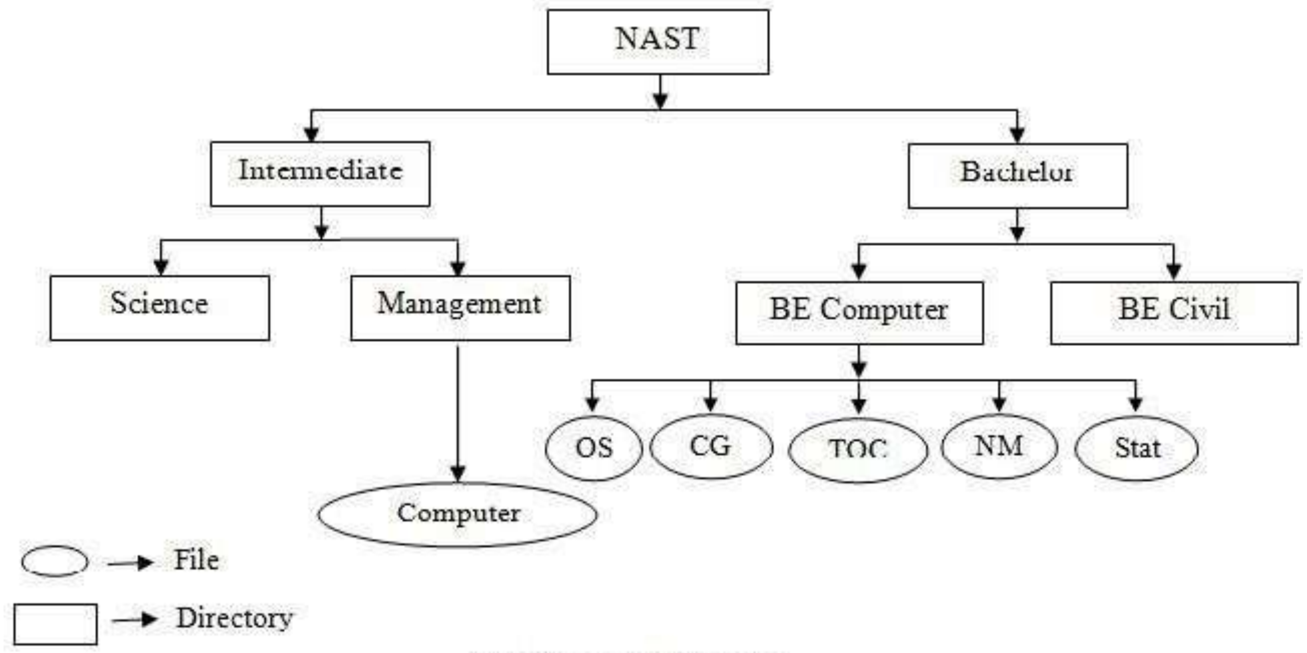


Fig: Directory Hierarchy

In above figure, "NAST" is a root directory which contains two sub directories named: *Intermediate* and *Bachelor.* At the bottom of "BE Computer" directory, there are 5 files: OS, CG, TOC, NM and Stat.

All subdirectory names and file names within a directory must be unique. However, names within different directories can be the same. Creating an arbitrary number of sub directories provides a powerful structuring tool for users to organize their work. The use of tree-structured directory minimizes the difficulty in assigning unique names. Any file in the system can be located by following a path from the root, or master directory down various branches until the file is reached.

II. Path names:

A path is the route through a file system to a particular file. A pathname (or *path name*) is the specification of that path. Each operating system has its own format for specifying a pathname. The path name of a file describes that file's place within the file-system hierarchy. There are two common methods: absolute path name and relative path name.

- **Absolute path name**

An absolute pathname is a pathname that starts from the root path and specifies all directories between the root path and the specified directory or file.
Example: D:\NAST\Bachelor\BE Computer/OS

It means that the root directory contains a subdirectory *NAST*, which in turn contains a subdirectory *Bachelor,* which in turn again contains another sub directory *BE Computer* which contains the file *OS*.

- **Relative path name**

  A relative pathname is a pathname that specifies some abbreviations for traversing up the directory structure and down to another directory or file in the same directory tree. In most file systems the . . (Two periods) abbreviation indicates "go up one directory level". Relative path do not begin with "\". It specifies location relative to our current working directory and it can be used as a shorter way to specify a file name.
  Example: ..\NAST\Bachelor\BE computer\OS

III. Directory Operation

The system call form managing directories may vary from system to system. Some of the basic operations on directory are listed below:

i) ***Create:*** When a directory is created, it is emp directories.

ii) ***Delete:*** A created directory can be deleted from the system in order to free up the storage. However only empty directory are deleted.

iii) ***Open:*** Before a directory can be read, it must be opened, it should be opened. A listing program opens the directory and the files inside it.

iv) ***Close:*** When a directory has been read, it should be closed to free up internal table space.

v) ***Read:*** This call returns the next entry in an open directory.

vi) ***Rename:*** It can be renamed like as files, because in many cases it is assumed as a file

vii) ***Link:*** Linking is a technique that allows a file to appear in more than one directory. This system call specifies an existing file and a path name, and creates a link from the existing file to the name specified by the path. In this way, the same file may appear in multiple directories.

viii) ***Unlink:*** A directory entry is removed. If the file is unlinked then it only present in one directory i.e. normal case.

11. File system Implementation:

I. Contiguous Allocation

It is simplest allocation scheme to store each file as a contiguous run of the disk blocks. It requires each file to occupy a set of contiguous addresses on a disk. It sores each file as a contiguous run of disk blocks. Thus on a disk with 1-KB blocks, a 50-KB file would be allocated 50 consecutive blocks. Both sequential and direct access is supported by the contiguous allocation method.

Advantages:

Contiguous disk space allocation has two significant advantages.

- **It is simple to implement** because keeping track of where a file's blocks are is reduced to remembering two numbers: the disk address of the first block and the number of blocks in the file. Given the number of the first block, the number of any other block can be found by a simple addition.

- **The read performance is excellent** because the entire file can be read from the disk in a single operation. Only one seek is needed (to the first block). After that, no more seeks or rotational delays are needed so data come in at the full bandwidth of the disk.

Thus contiguous allocation is simple to implement and has high performance.

Disadvantage:`

- In time, the disk becomes fragmented, consisting of files and holes. It needs compaction to avoid this.

Example of contiguous allocation: CD and DVD ROMs

II. Linked List Allocation

This method keeps each file as a linked list of disk blocks as shown in the following figure. The first word of each block is used as a pointer to the next one. The rest of the block is for data.

Advantage:

- Unlike contiguous allocation, every disk block can be used in this method. No space is lost to disk fragmentation.

Disadvantage:

- It can be used only for sequential access files. To find the ith block of a file, we must start at the beginning of that file, and follow the pointers until we get the ith block. It is inefficient to support direct access capability for linked allocation of files.
- Another problem of linked list allocation is reliability. Since the files are linked together with the pointer scattered all over the disk. Consider what will happen if a pointer is lost or damaged.

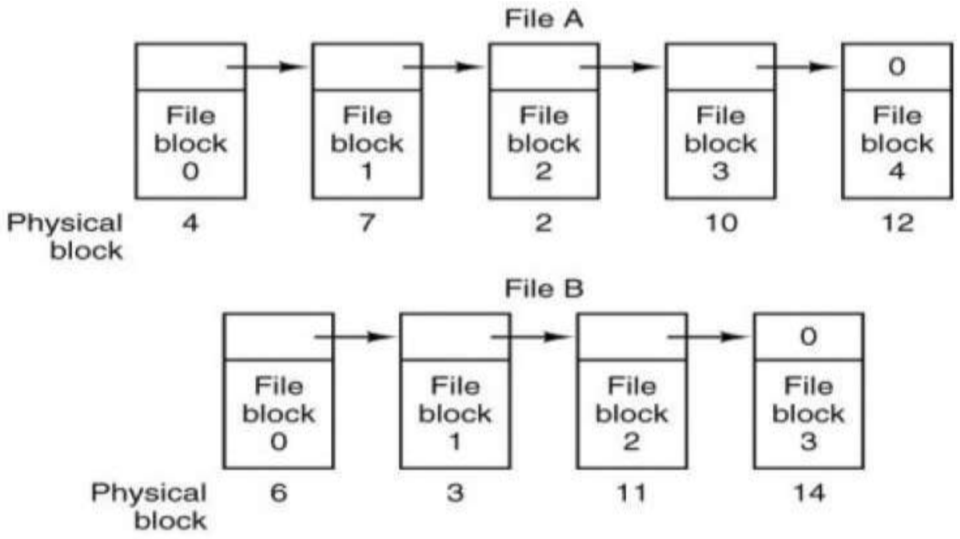


Fig: Storing a file as a linked list of disk blocks.

Linked List allocation using a table in memory:

File "**A**" uses disk blocks 4, 7, 2, 10, and 12 and file **"B"** uses disk blocks 6, 3, 11, and 14, in order. Using the following table we can start with block 4 and follow the chain all the way to the end. Same process is done with block 6 (File "B"). Both chains are terminated with a special marker (e.g. -1) that is not a valid block number. Such table in the main memory is called FAT (File Allocation Table).

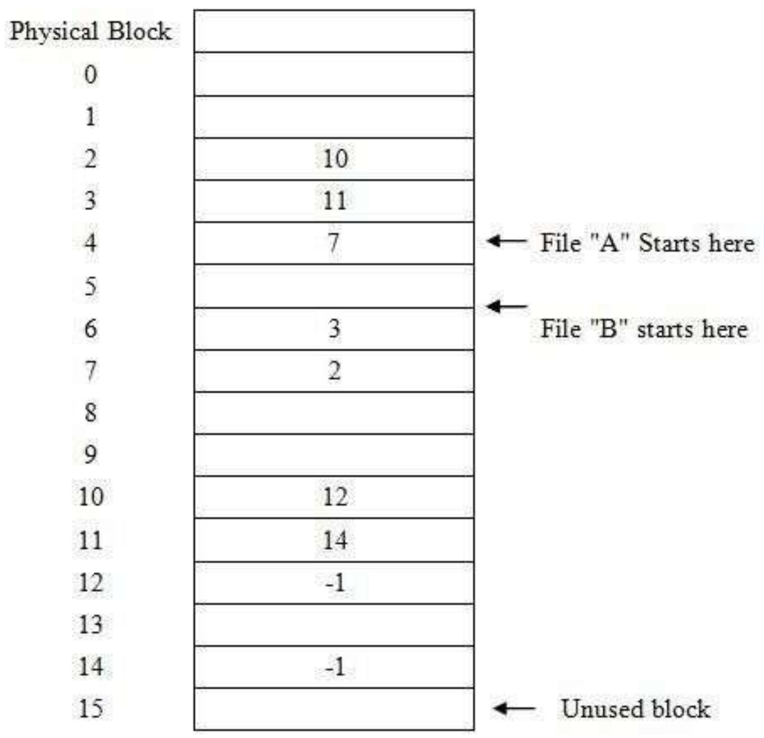


Fig: Linked list allocation using a FAT in main memory

III. **I-nodes**

It is one of the modern method for keeping tracks of each block belongs to which file is to associate with each file within table, called i-node table. I-node is a unique number given to a file in an operating system i.e. it is a data structure that stores all the information's about a file except its name and its actual data. When a file is created both name and an I node number is assigned to it, which is unique within the file system. Both name and their corresponding I node numbers are stored as entries in their directory that appears to the user to contain the file i.e. the directory associates file names and i-nodes.

Whenever a user or a program refers to a file by name, the operating system uses that name to look up the corresponding I node, which then enables the system to obtain the information it needs about the file to perform further operations.

A i-node contains following metadata:

- Size of file (in bytes) and its physical location
- The file's owner and group
- The file's access permissions (read, write, execute)
- Timestamps telling when the i-node was created, last modified and last accessed and
- A reference count telling how many hard links points to the i-node.

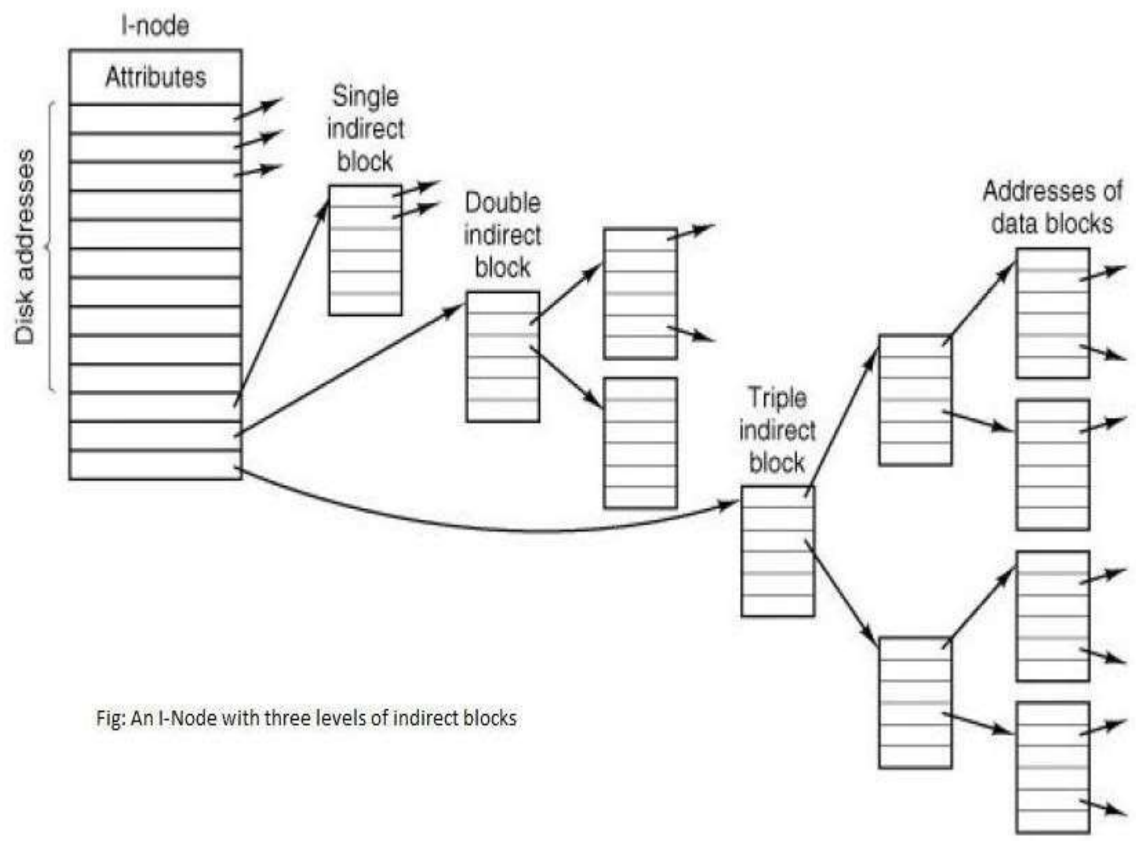


Fig: An I-Node with three levels of indirect blocks

It solves the external fragmentation and size declaration problems of contiguous allocation.

12. Security and Multi-media Files

I. Security

Files are securable objects, access to them is regulated by the access-control model that governs access to all other securable objects in Windows.

Files with sensitive information are stored everywhere these days. Servers, workstations, backup tapes, USB drives, laptops etc, they all contain sensitive and mission critical information. No file is secure by workstation or USB drive, waiting to be used. What if that media was lost or stolen?

The information on that media is then compromised and can severely damage the company. With federal laws and regulations in place, it would be quite expensive too.

File encryption: has been around for as long as computers have. Yet, the use of file encryption almost never goes beyond the very large corporations that have many pending patents, a large database of customer & client information, etc. Regardless of the size of your organization, your information is your most valuable asset. If you lose or compromise that information, it could damage your company severely.

II. Multi-media files

A file that is capable of holding two or more multimedia elements such as text, audio, video, image, animations. Multimedia is usually recorded and played, displayed, or accessed. Multimedia presentations may be viewed by person on stage, projected, transmitted, or played locally with a media player.

Multimedia data and information must be stored in a disk file using formats similar to image file formats. Multimedia formats, however, are much more complex than most other file formats because of the wide variety of data they must store. Such data includes text, image data, audio and video data, computer animations, and other forms of binary data, such as Musical Instrument Digital Interface (MIDI), control information, and graphical fonts. Typical multimedia formats do not define new methods for storing these types of data. Instead, they offer the ability to store data in one or more existing data formats that are already in general use.

Some extension of media files:

.asx, .wax, .wvx, .wmx, mp3, mp4, etc

Security in Windows 2000

Windows 2000 is an operating system for use on both client and server computers. Windows 2000 security is based on a simple model of authentication and authorization. *Authentication* identifies the user when s/he logs on and makes network connections to services. Once identified, the user is authorized access to a specific set of network resources based on permissions. *Authorization* takes place through the mechanism of access control, using entries stored in Active Directory as well as access control lists that define permissions for objects including printers, files, and network file and print shares.

The key features of Windows 2000 distributed security services are:

- Smart Card authentication support: This card stores a user's public key, private key, and certificate. These cards are a secure way to protect and control a user's keys, instead of storing them on a computer.
- Kerberos for network authentication: It is the primary security protocol that allows users to use a single logon to access all resources. The protocol verifies both the identity of the user and the integrity of the session data.
- File encryption and certificate services
- Strong user authentication and authorization.
- Secure communications between internal and external resources.
- The ability to set and manage required security policies.
- Automated security auditing.
- Interoperability with other operating systems and security protocols.
- An extensible architecture to support application development that uses Windows 2000 security features.

To immediately secure your Windows 2000 system, we have to take following three steps:

- **Install anti-virus software**
  We have to install third party antivirus software.

- **Install a personal firewall**
  A personal firewall protects our machine against Internet attacks and random network scans.

- **Run Windows Update and Enable Automatic Updates**
  We should run Windows Update on our system to install all Critical and Recommended update