# File System

- Users make use of computers to create, store, retrieve and manipulate information.
- The file abstraction provides a uniform logical view of physical contents from a wide variety of storage devices that have different characteristics.
- A file in a computer system has a name for its identification, space to store data, a location for convenient access, access restrictions and other attributes and support mechanisms.

- OS implements a software layer on top of the I/O subsystem (device drivers) for users to access data with ease from storage devices. The software layer is called the file management system or file system.
- The file management system contains files, directories and control information (metadata) and supports convenient and secured access on files and directories.

The functions of file system are

- Organization of files
- Execution of file operations
- Synchronization of file operations
- Protection of file contents
- Management of space in the file system.

# File concepts

File

- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks which normally represents programs and data.
- In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

Naming

- A symbolic file name given to a file, for the convenience of users, is a string of characters. When a file is named, it is independent of the process, the user and the system it created. It is the only information kept in human readable form.

File structure

- A file has a certain defined structure, which depends on its type of information stored in the file, like source program, object programs, executable programs, numeric data, text, payroll records, graphic images etc.
  - A text file is a sequence of characters organized into lines.
  - A source file is a sequence of procedures and functions.
  - An object file is a sequence of bytes organized into blocks that are understandable by the machine.
  - When operating system defines different file structures, it also contains the code to support these file structure.

**Attributes** define the characteristics of a file varying with respect to OS and useful for protection, security and usage monitoring.

The following are the file attributes:

- *Name*: The symbolic file name is the only information kept in human readable form
- *Type*: This information is needed for those systems that support different types.
- *Location*: It is a pointer to a device used 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 controls 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.

## Logical file structure

There are three possible forms of atomic units in a file.

- *Bytes*: File is a sequence of bytes called as flat file. (No internal structure)
- *Fixed length record*: Many OS require files to be divided into fixed length records. Each record is a collection of information about one thing. Easy to deal with but do not reflect the realities of data.
- *Variable length records*: These are used to meet the various workload lengths but the problem is unless the location of the file is known it is hard to perform search operation. To overcome this problem keyed file is used. Each record has a specified field which is the key field which is used for finding the location of the file.

# File Meta data

- A file contains information, but the file system also keeps information about the file,
- i.e. meta data (information about information) such as Name, Type, Size and Owner of the file; Group(s) of users that have special access to the file; Access rights; Last Read time; Last Written time; The time of the file was created; which disk the file is stored on and where the file is stored on disk.

# File Types

File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc. The file system supports various classes of file types. A file may or may not have certain internal structure depending on its type.

Many operating systems support many types of files. Operating system like MS-DOS and UNIX have the following types of files −

Ordinary files

- These are the files that contain user information.
- These may have text, databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the entire file.

- Directory files
  - These files contain list of file names and other information related to these files.

Special files

- These files are also known as device files.
- These files represent physical device like disks, terminals, printers, networks, tape drive etc.
- These files are of two types −
  - **Character special files** − data is handled character by character as in case of terminals or printers.
  - **Block special files** − data is handled in blocks as in the case of disks and tapes.

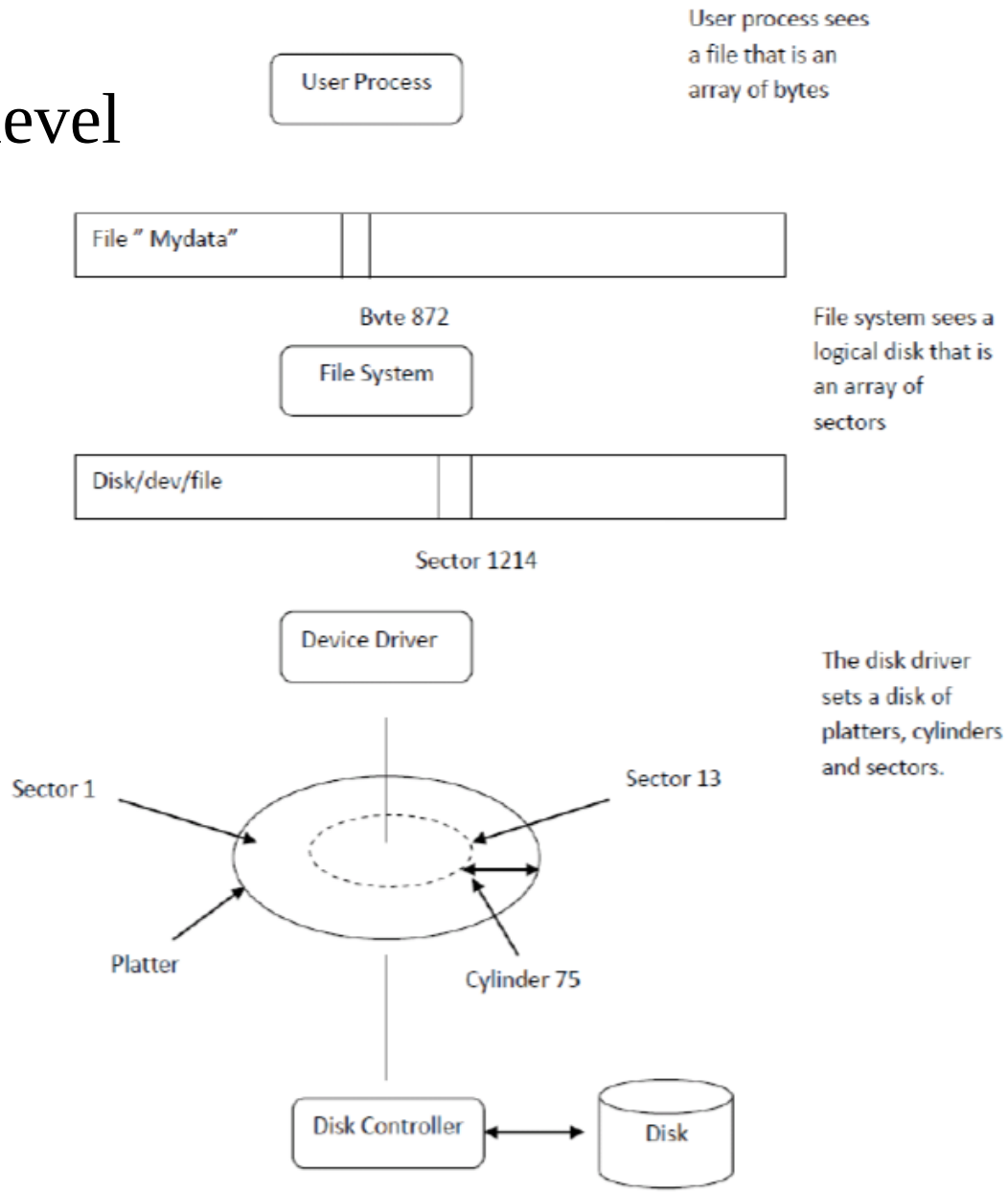
**File System**

The file system consists of two distinct parts:

i) a collection of files, each storing related data and

ii) a directory structure which organizes and provide information about all the files in the system.

# view of a file at each level



# File operations

Create:

- Essential if a system is to add files. Need not be a separate system call (can be merged with open).
- Requires the name of the file/directory being created, the name of the directory and some file attributes. Two steps are necessary to create a file:
- space in the file system must be found for the file.
- an entry for the new file must be made in the directory i.e. the directory entry records the name of the file and the location in the file system.

Delete:

- Essential if a system is to delete files.
- Requires the file/directory name to be deleted. To delete a file, the directory is searched for the named file and having found the associated directory entry, all file spaces are released and the directory entry is erased.

## Open

- Not essential. An optimization in which the translation from file name to disk locations is perform only once per file rather than once per access.
- Open operation requires a filename to be opened.  File system checks for permission to access the file and if the user has the permission, it creates a file descriptor that will be used by the application for future reference.

Close:

- Not essential. Free resources.
- The close operation requires a file descriptor that was obtained in the open operation. It releases the file descriptor and other related resources allocated for the open file session.

## Read

- Essential. Must specify filename, file location, number of bytes, and a buffer into which the data is to be placed. Several of these parameters can be set by other system calls and in many OS's they are.
- A system call is issued that specifies the name of the file and where in the memory the file should be put for reading.
- A read operation takes as parameters file descriptor, a positive integer number and a buffer address.
- The operation copies into the buffer those many number of consecutive bytes starting at the current file pointer position from the file. It repositions the file pointer past the last byte it read.

## Write

- Essential if updates are to be supported.
- A system call is made specifying both the name of the file and the information to be written to the file.
- A write operation takes as parameters a file descriptor, a byte string and the size of the string. The byte string is written in the file identified by the file descriptor.
- It overwrites the file content starting at the current file pointer position within the file. It allocates more space to the file when it needs to write past the current last byte in the file. It repositions the file pointer after the last byte written.

## Truncate

- To erase the contents of the file but keep attributes same, truncating can be used i.e. attributes of the file remain same but the length of the file is reset to zero.
- A truncate operation takes as its parameters a file descriptor and a positive integer number. It reduces the size of the corresponding file to the specified number. If needed, it frees up space from the file.

# Memory map

- Memory mapping a file creates a region in the process address space, and each byte in the region corresponds to a byte in the file.
- Conventional memory read and write operations on the mapped sections by applications are treated by the system as file read and write operations respectively.
- When the mapped file is closed, all the modified data are written back to the file and the file is unmapped from the process address space.

## File Pointer

- On systems that do not include a file offset as part of the read and write system calls, the system must track the last read / write location as current file position pointer.
- This pointer is unique to each process operating on the file, and therefore must be kept separate from the disk file attributes.

## Reposition

- adjusts a file pointer to a new offset. The operation takes a file descriptor and an offset as parameters and sets the associated file pointer to the offset. The directory is searched for the appropriate entry and current file position is set to a given value.

## Seek

- Not essential (could be in read/write). Specify the offset of the next (read or write) access to this file. Repositioning within a file does not needed to involve any actual I/O. This file operation is also known as file seek.