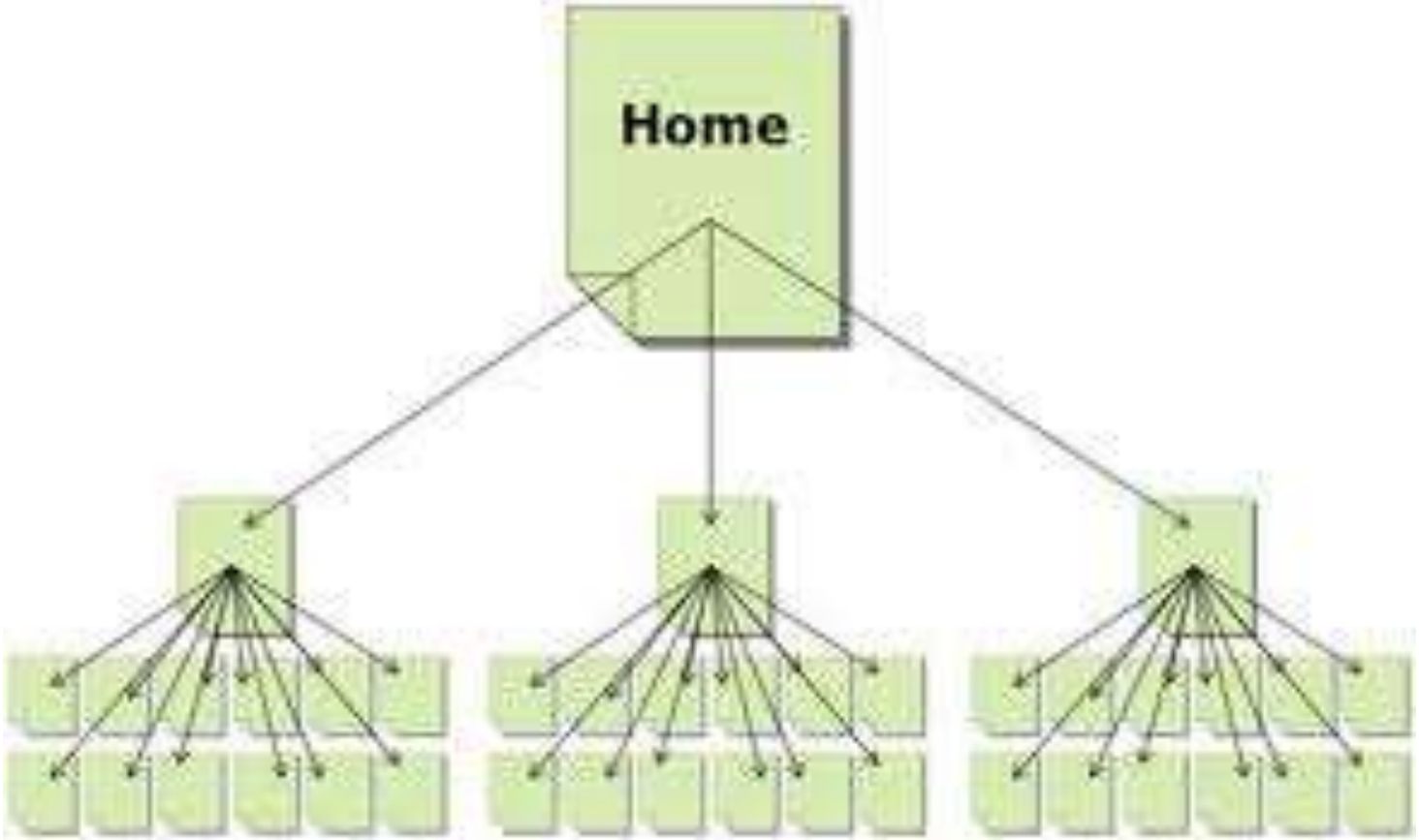

Home

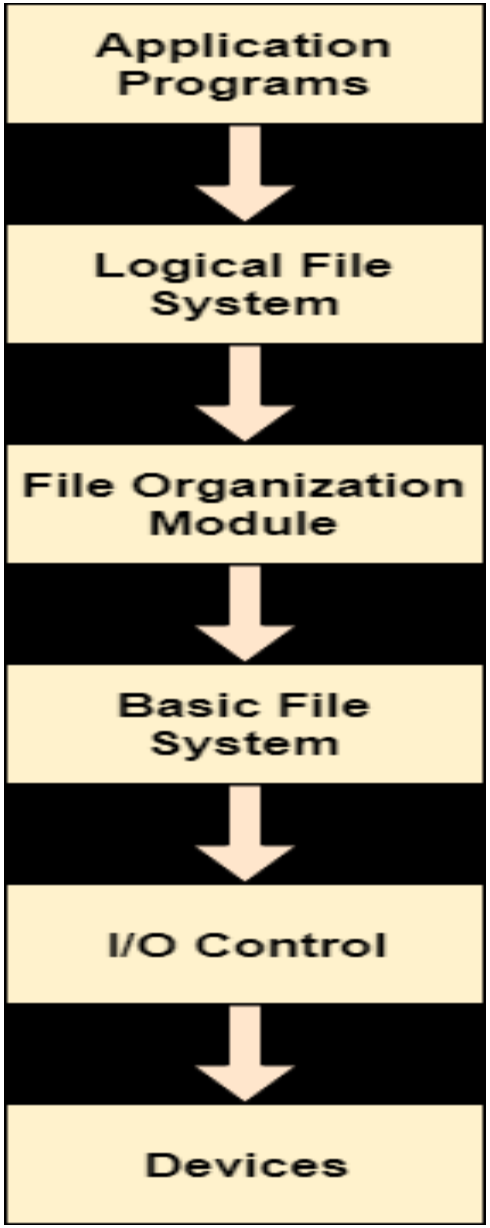
File Concept

- A file is a named collection of related information that is recorded on secondary storage.
- Data can NOT be written to secondary storage unless they are within a file.
- A file is a collection of related information that is recorded on secondary storage. Or file is a collection of logically related entities.
- From user's perspective a file is the smallest allotment of logical secondary storage.

File System Structure

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.


Application Programs
Logical File System
File Organization Module
Basic File System
I/O Control
Devices

- 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.
-

File Structure

- A file has a certain defined **structure** which depends on its types:
  - A text file is a sequence of characters organized into lines.
  - A source file is a sequence of subroutines and function.
  - An object file is a sequence of bytes organized into blocks understandable by the system's linker.
  - An executable file is a series of code sections that the loader can bring into memory and execute.

File Attributes

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring
- Information about files are kept in the directory structure, which is maintained on the disk

File Operations

- File is an **abstract data type**
- **Create**
- **Write**
- **Read**
- **Reposition within file**
- **Delete**
- **Truncate**
- *Open(F_i)* – search the directory structure on disk for entry F_i, and move the content of entry to memory
- *Close (F_i)* – move the content of entry F_i in memory to directory structure on disk

FILE DIRECTORIES:

- Collection of files is a file directory.
- The directory contains information about the files, including attributes, location and ownership.
- Much of this information, especially that is concerned with storage, is managed by the operating system.
- The directory is itself a file, accessible by various file management routines.

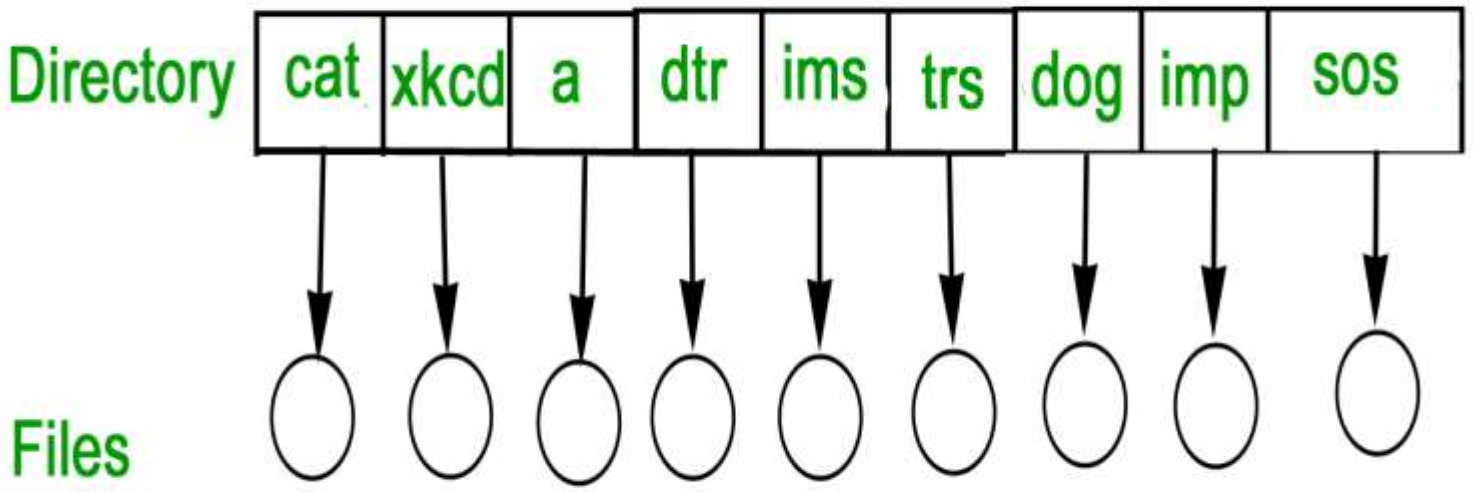
- **Information contained in a device directory**
- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed
- Date last updated
- Owner id
- Protection information

- **Operation performed on directory are:**
- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

- **Advantages of maintaining directories are:**
- **Efficiency:** A file can be located more quickly.
- **Naming:** It becomes convenient for users as two users can have same name for different files or may have different name for same file.
- **Grouping:** Logical grouping of files can be done by properties e.g. all java programs, all games etc.

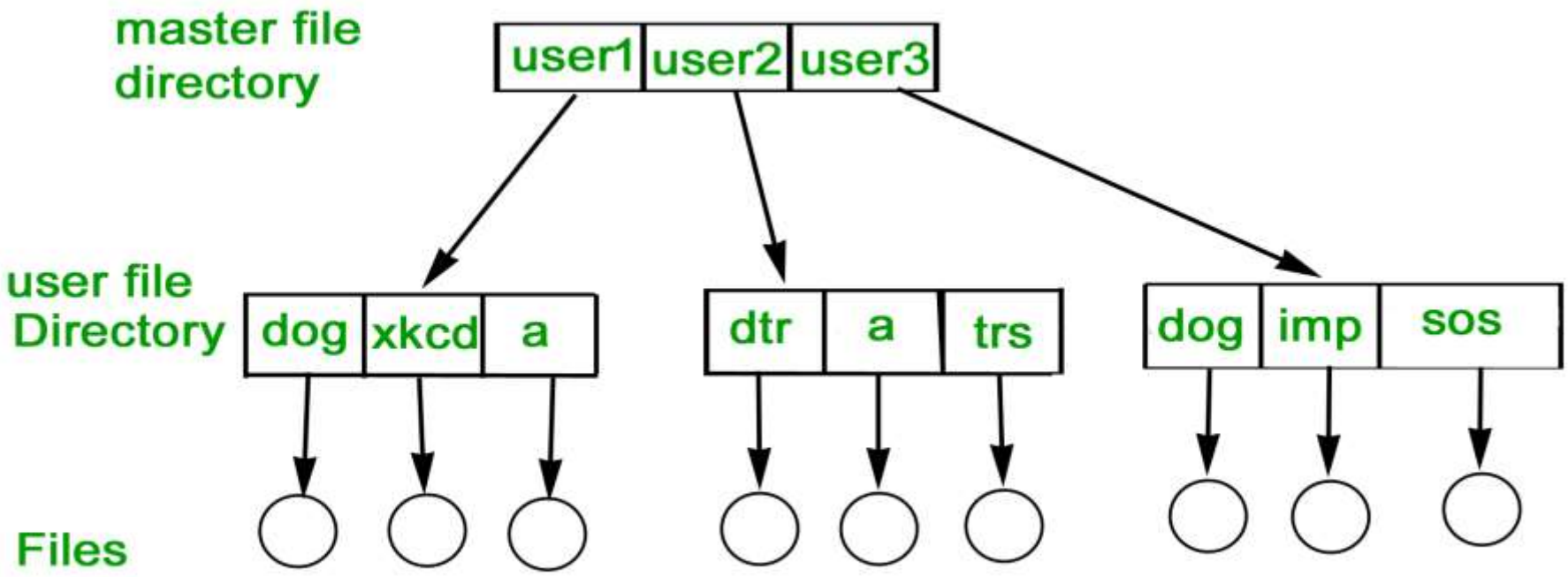
- **SINGLE-LEVEL DIRECTORY**
  In this a single directory is maintained for all the users.
- **Naming problem:** Users cannot have same name for two files.
- **Grouping problem:** Users cannot group files according to their need.

SINGLE-LEVEL DIRECTORY



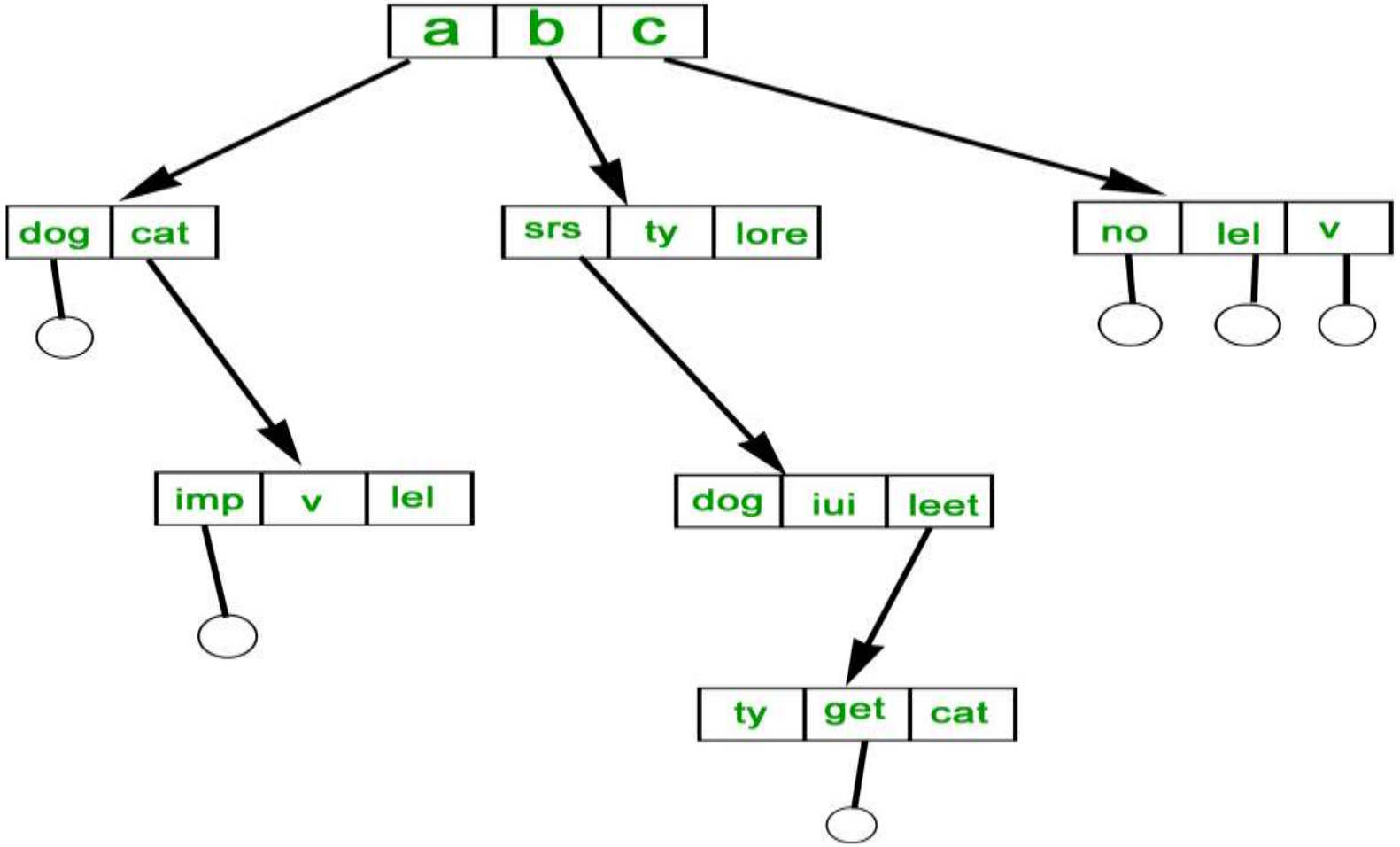
- **TWO-LEVEL DIRECTORY**
  In this separate directories for each user is maintained.
- Path name: Due to two levels there is a path name for every file to locate that file.
- Now,we can have same file name for different user.
- Searching is efficient in this method.

TWO-LEVEL DIRECTORY



- **TREE-STRUCTURED DIRECTORY :**
  Directory is maintained in the form of a tree. Searching is efficient and also there is grouping capability. We have absolute or relative path name for a file.

TREE-STRUCTURED DIRECTORY :



File System Implementation

- File System Structure
- File System Implementation
- Directory Implementation
- Allocation Methods
- Free-Space Management

File-System Structure

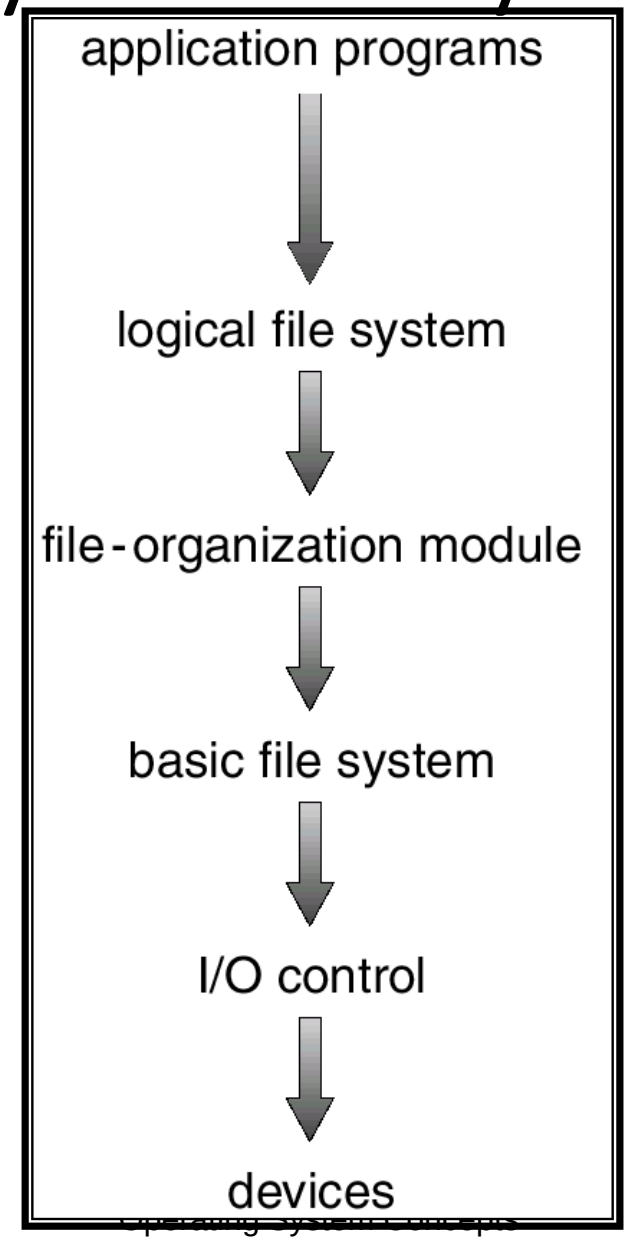
- File
  - Collection of related information
- File system resides on secondary storage (disks).
- Windows- NTFS(New Technology File system)
- Linux- Ext(Extendable File system)
- File system organized into layers.
- File control block – storage structure consisting of information about a file.

- Boot control block
  - Contains information needed by the system to boot an operating system from a partition
- Partition control block
  - Contains partition details such as the number of blocks in the partition, size of the blocks, free block count
  - In NTFS, it is called as Master File Table (MFT)

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.

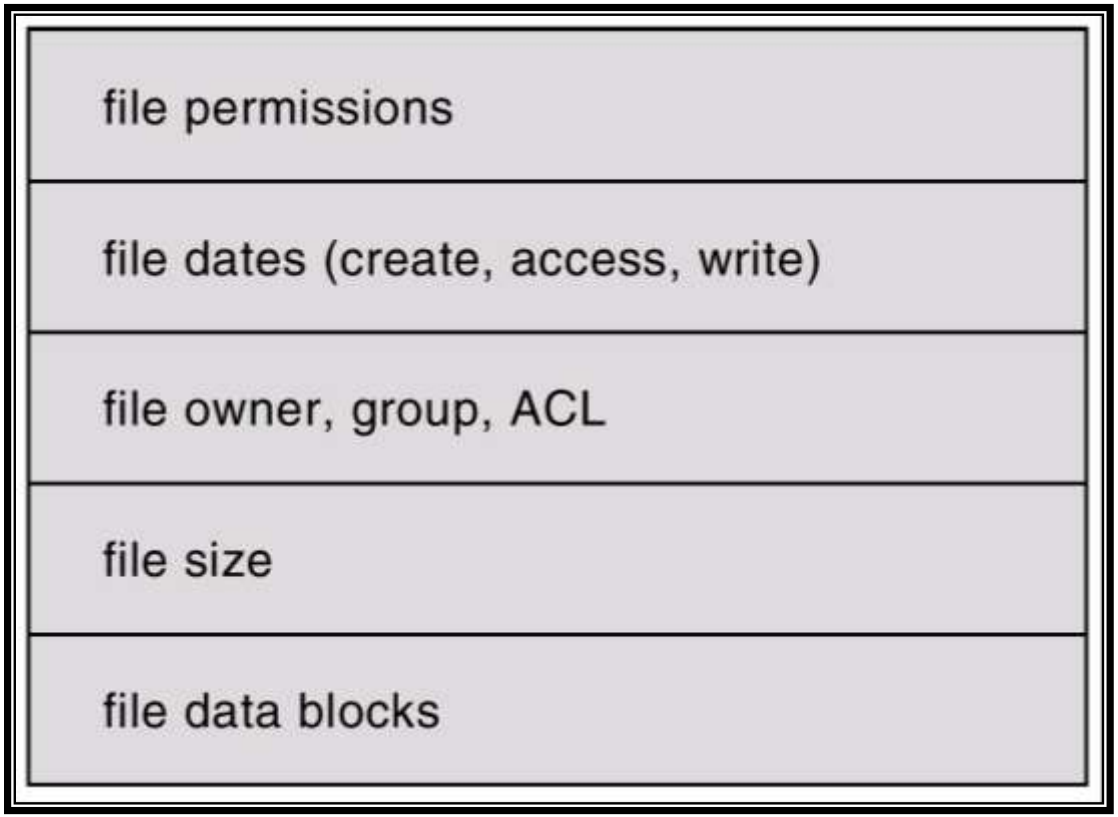
Layered File System



- 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.

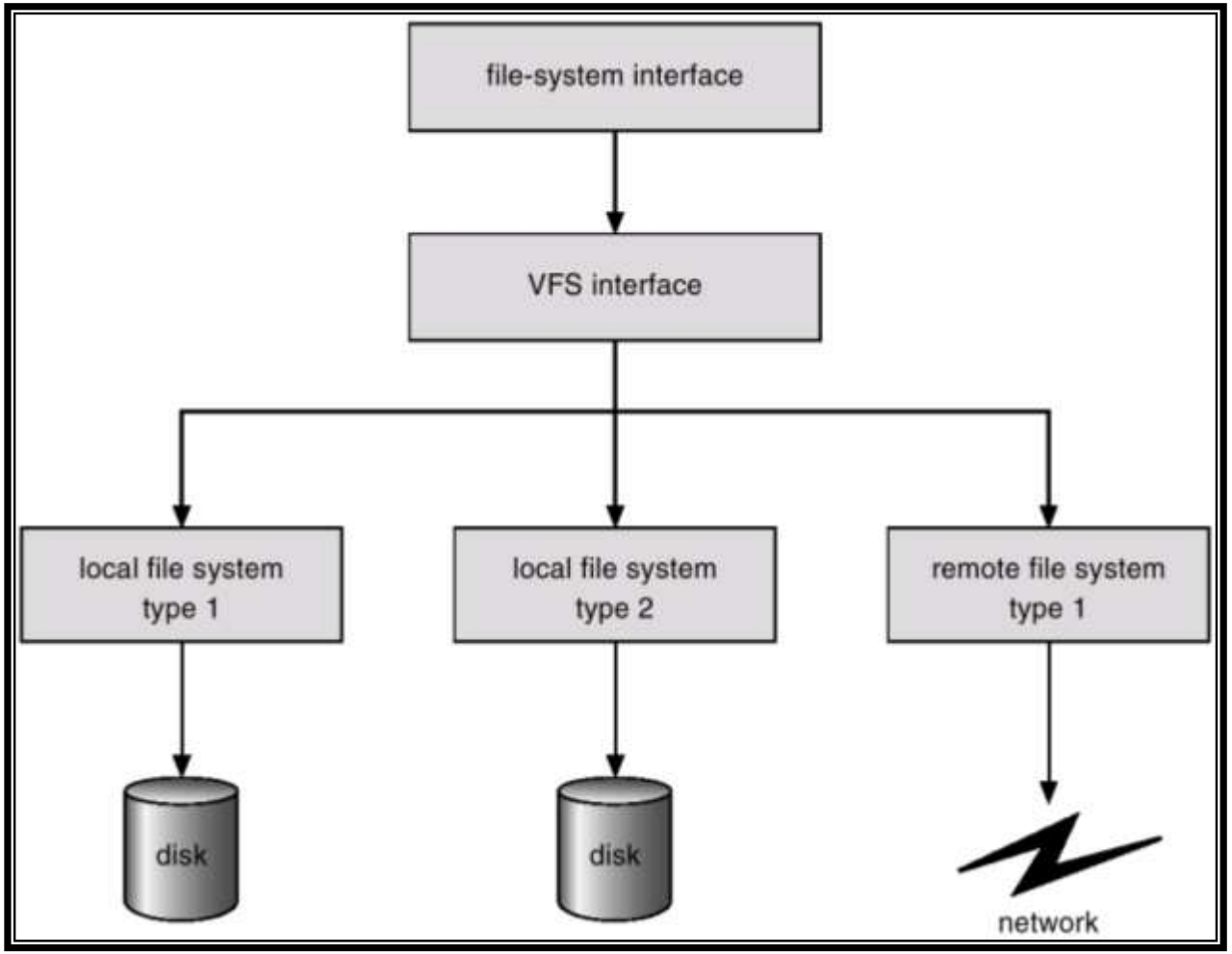
A Typical File Control Block



Virtual File Systems

- A virtual file system (VFS) is programming that forms an interface between an operating system's kernel and a more concrete file system.
- The VFS serves as an abstraction layer that gives applications access to different types of file systems and local and network storage devices. For that reason, a VFS may also be known as a *virtual file system switch*.
- It also manages the data storage and retrieval between the operating system and the storage sub-system

Schematic View of Virtual File System



Directory Implementation

- Linear list
  - Simplest method of implementing a directory
  - Linear list contains file name with pointers to the data blocks
  - time-consuming to execute

- Hash Table – linear list with hash data structure.
  - decreases directory search time
  - collisions – situations where two file names hash to the same location
  - fixed size

Allocation Methods

- An allocation method refers to how disk blocks are allocated to files
  - 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.

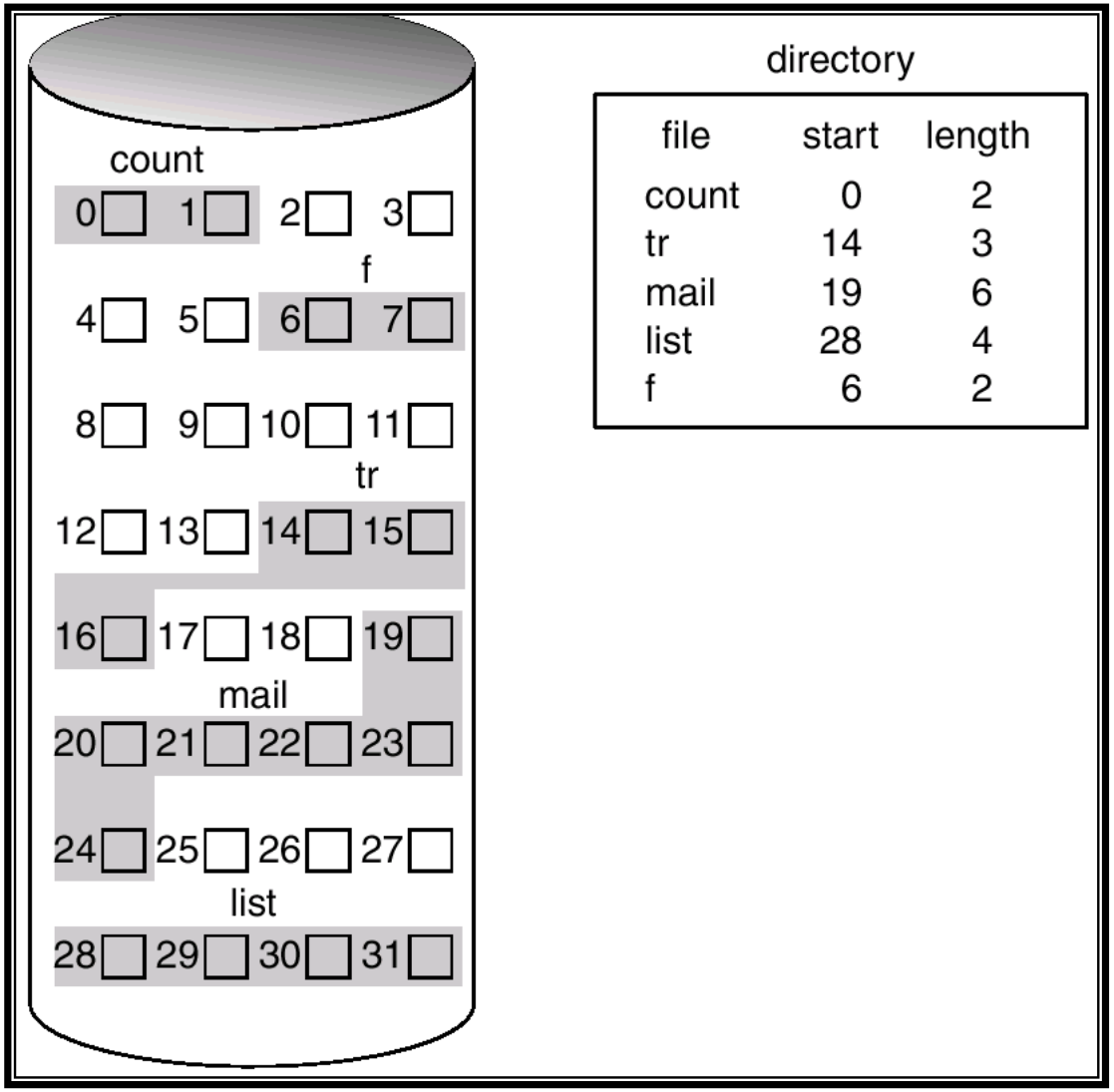
Contiguous Allocation

- 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.

Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk.
- Simple – only starting location (block #) and length (number of blocks) are required.

Contiguous Allocation of Disk Space



- **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.

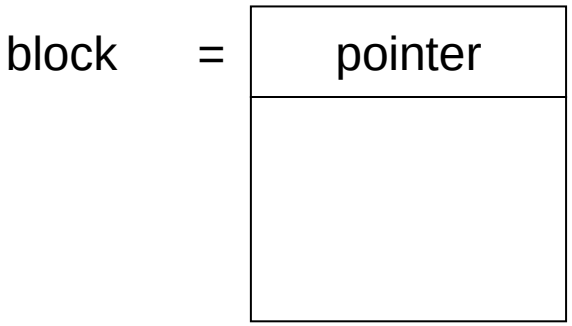
- **Disadvantages:**
- This method suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization.
- Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

2. Linked List Allocation

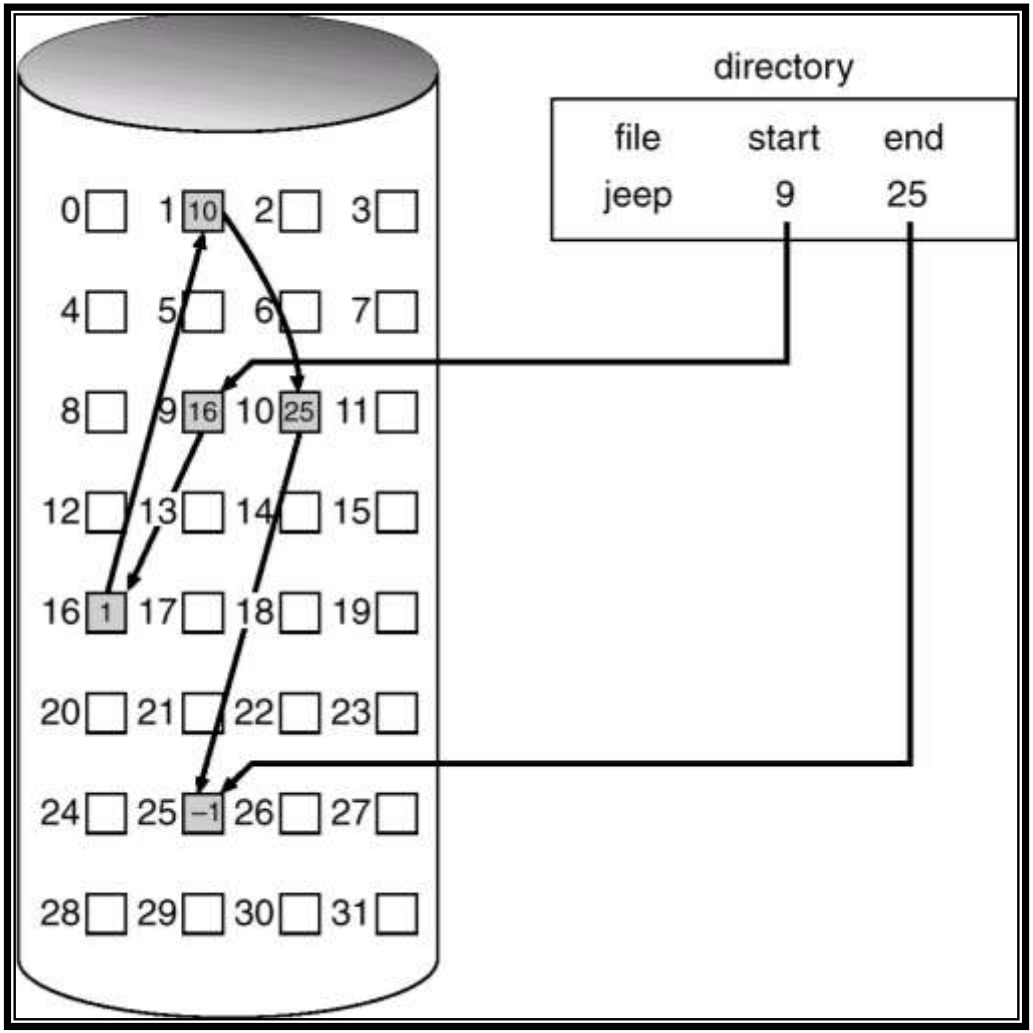
- In this scheme, each file is a linked list of disk blocks which **need not be** contiguous. The disk blocks can be scattered anywhere on the disk. The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.

Linked Allocation

- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.



Linked Allocation



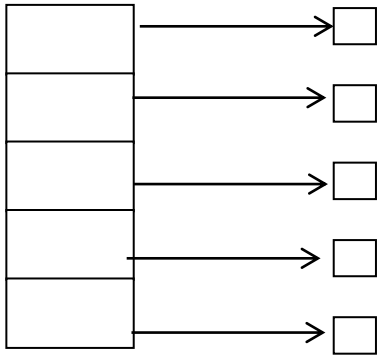
- **Advantages:**
- This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
- This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.

- **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:

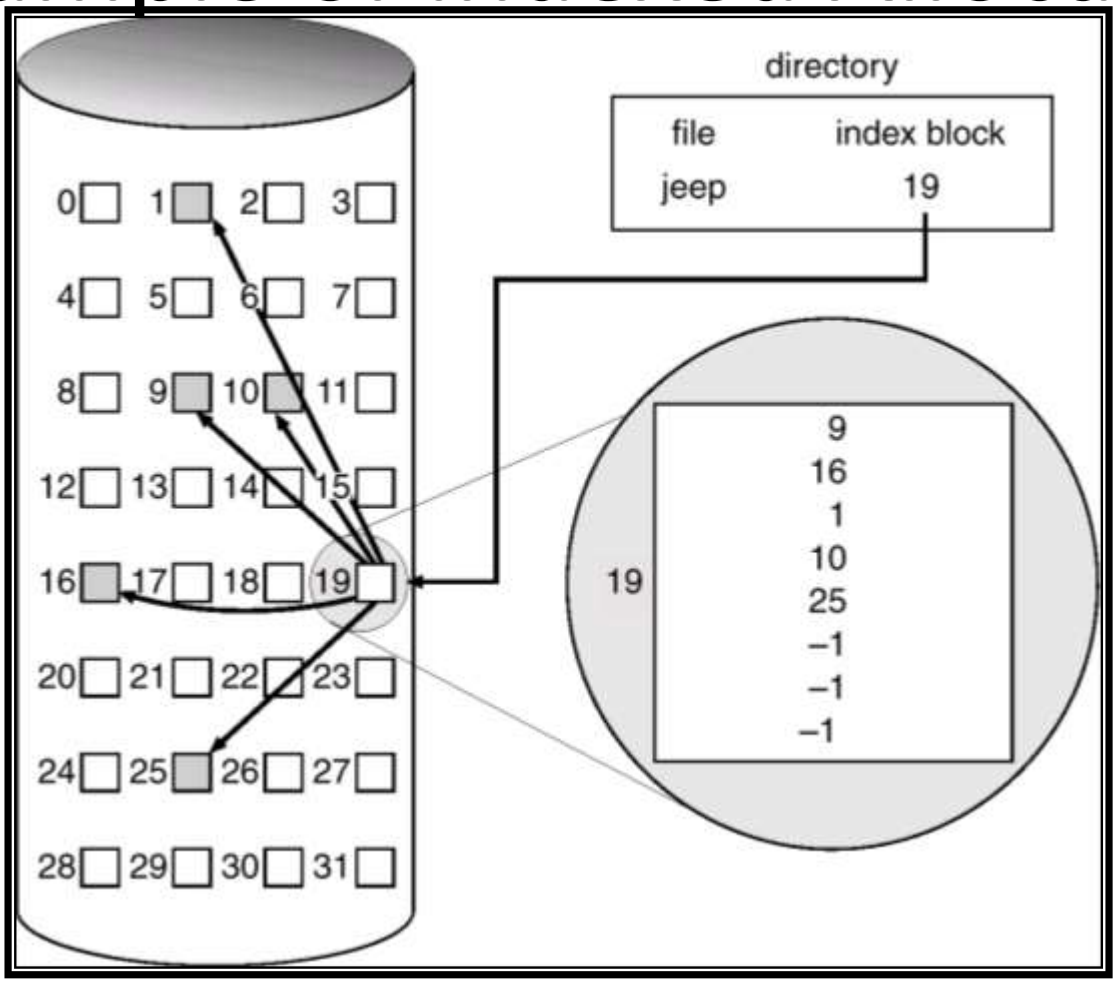
Indexed Allocation

- Brings all pointers together into the index block.
- Logical view.



index table

Example of Indexed Allocation



- **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.
- lose the space of only 1 pointer per block.

- **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