Chapter: File-System Interface

Chapter: File-System Interface

- File Concept
- Access Methods
- Directory Structure
- File-System Mounting
- File Sharing
- Protection

File Management

- Process of storing, Controlling, Managing data stored on secondary storage in the form of files.

- File Management:
  - **Ensure consistency** of data when multiple users access the files
  - Provides **measures for file security and protection**

- File system consists of 2 parts:
  - Collection of files
  - A Directory structure which organizes and provides all the information about all the files in your system.

File Concept

- File is a sequence of bits, bytes, lines, or records, the meaning of which is defined by the file's creator and user.
- Field
- Record
- File
- Data Base

File Attributes

- **Name** – Name is usually a string of characters
- **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** – Access Permissions, 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

All operations involve some system calls

1. Create: using a specific system call. **Two steps are necessary to create a file.**

a) First, space in the file system must be found for the file.
b) Second, an entry for the new file must be made in the directory.

2.Read:

- Open a file
- File pointer points to particular record to be read
- Once a record/character is read, the file pointer is increment

File Operations

3. Write:

- Pointer is at the end of the file
- Can be repositioned and is incremented after every write
- File pointer can be repositioned

File Operations

- Reposition within file
- Delete
- Truncate

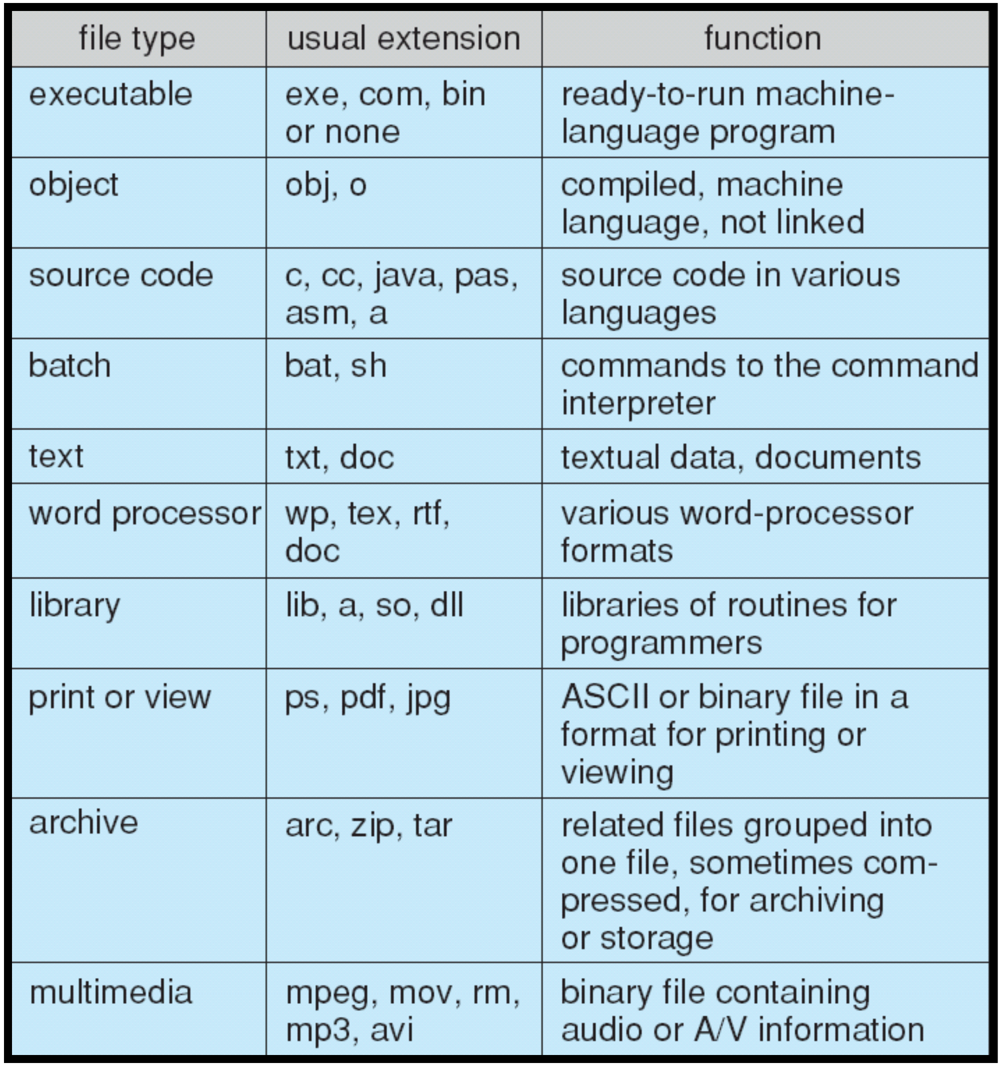
- *Open(F_i)* – **search** the file F_i from the **directory structure on disk**

- *Close (F_i)* – Save / move the content of file F_i in memory

Open Files

- Several pieces of data are needed to manage open files:
  - **File Pointer:** pointer to last read/write location
  - **File - Open Count:** counter of number of times a file is open
  - **Disk Location Of The File:** From where the is required to be opened, cache of data access information
  - **Access Rights:** per-process access mode information

File Types – Name, Extension



file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, 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, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

Criteria for File Organization

1. **Economy of Storage**

2. There should be minimum redundancy in data

3. Redundancy can be used to speed up the access

2. Simple

3. Maintenance

4. Reliability

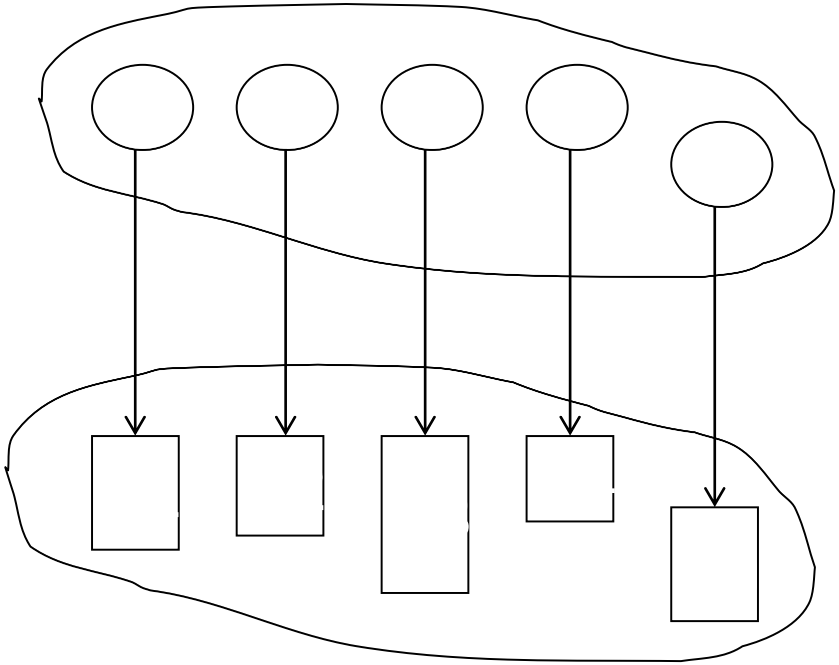
What kind of reliability is provided for a file

Unreliable:- if index file is lost / corrupted, actual file may be lost

Log File: Any record file is a log file

Directory Structure

- Symbol table of files that stores all related information about a file it holds with its contents



Both the directory structure and the files reside on disk
Backups of these two structures are kept on tapes

Operations Performed on Directory

Directory: collection of files or directories

- A Symbol Table that translates file names into their directory entry.

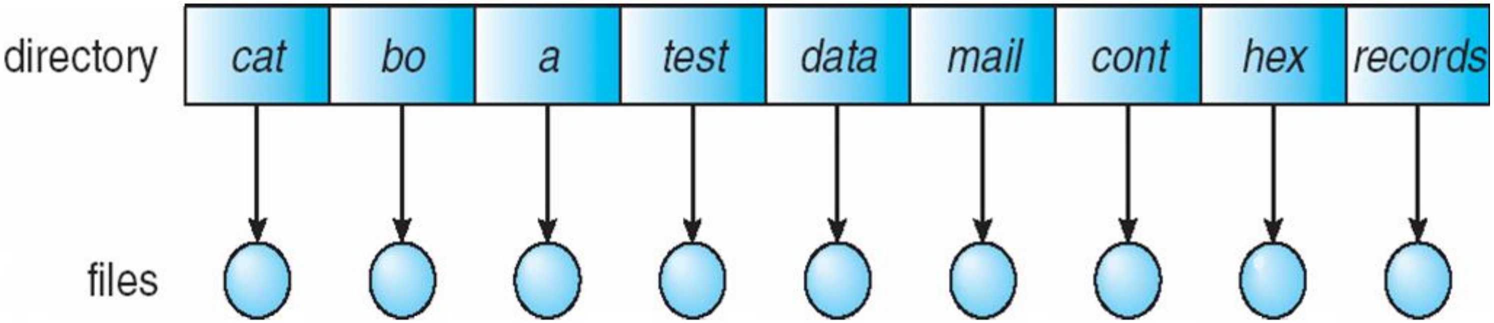
Operations:

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system : Search all directories/ sub directories and files

Directory Schemes

1. Single Level Directory

One directory many files



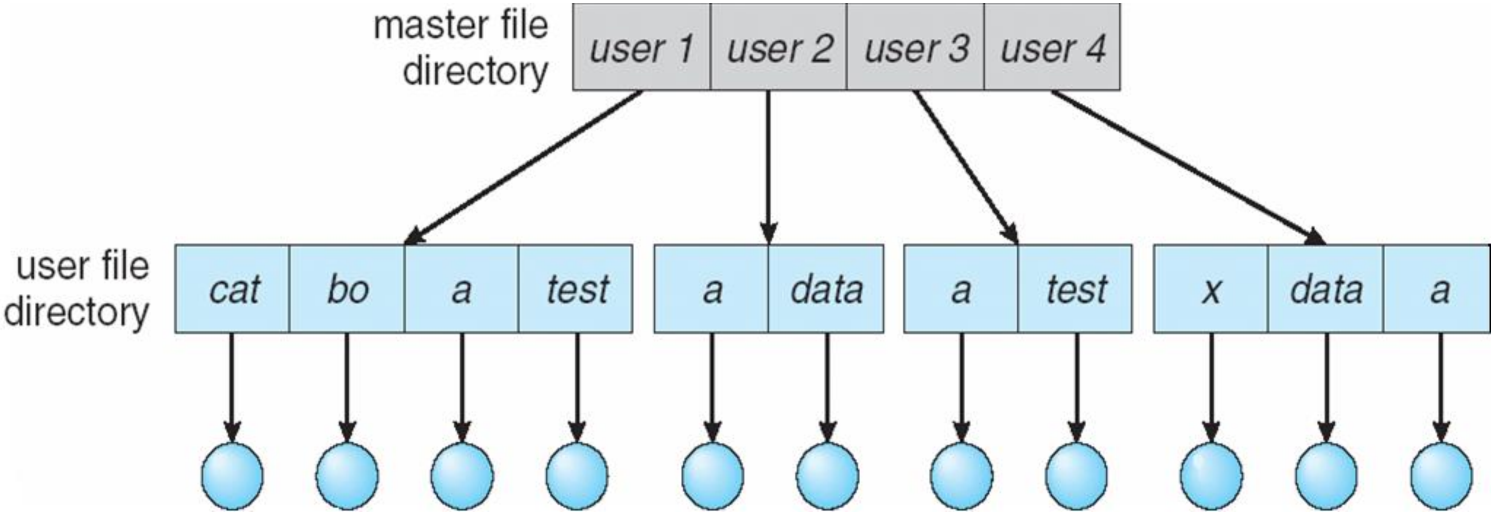
Disadvantage:

1. Difficult to remember the name of files when files increases
2. Single directory for all users
3. File names created by different users should be different.

Two Level

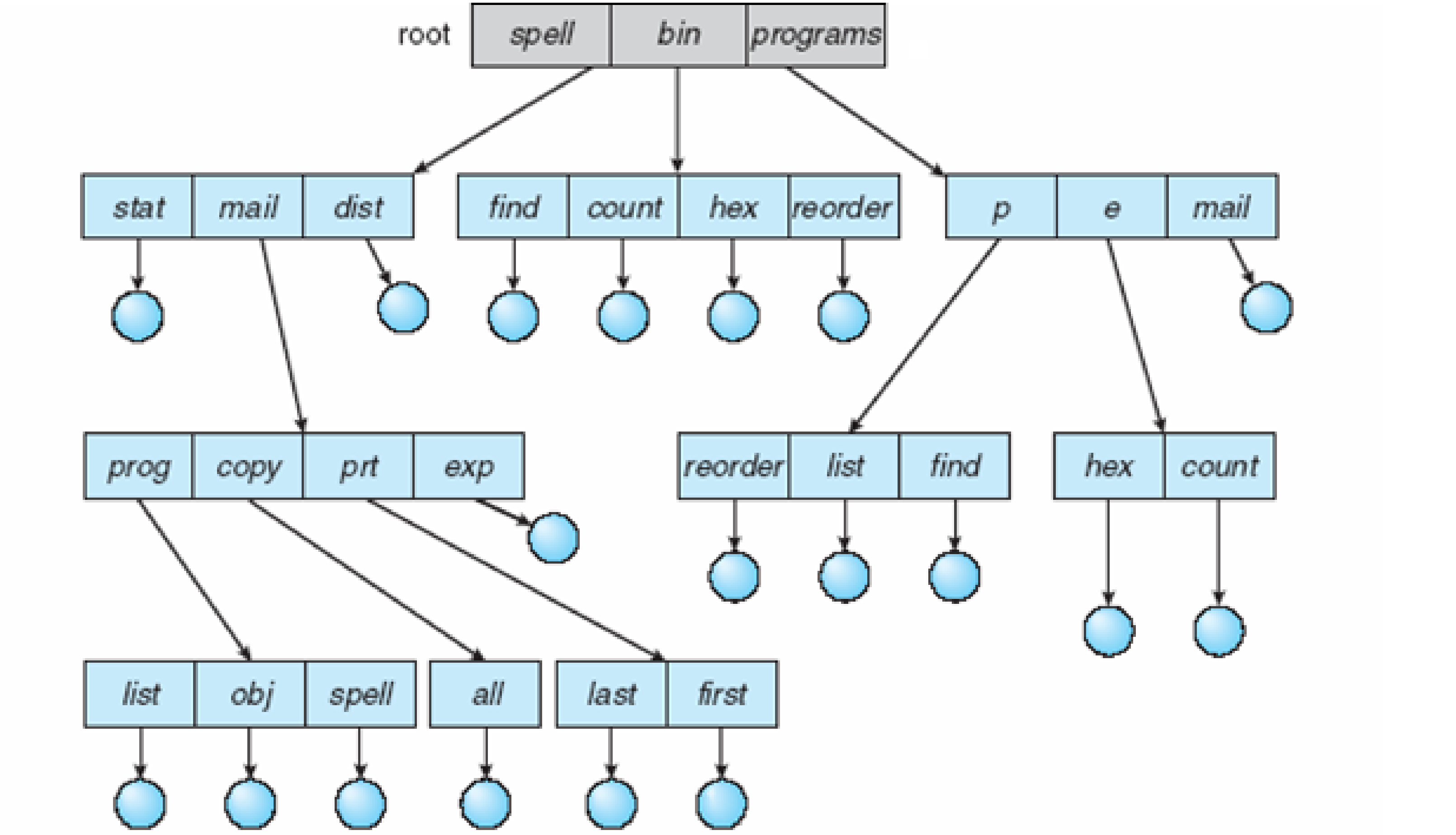
- **2. Two level directory**, **each user has his own user file directory(UFD).**

- UFDs have the similar structure, but each **lists files of a single user**.



Tree Structure

- Users can create their sub directories to manage the files.
- Three has Root directory and files have unique file names



Paths

- **Absolute Path:** Begins at the root and follows a path down to the specified file giving directory names on the path.
- **Relative Path:** Defines a path from the current directory.
- Creating a new file is done in current directory
- **Delete a file**

  rm <file-name>
- **Creating** a new subdirectory is done in current directory

  mkdir <dir-name>

Delete Directory:

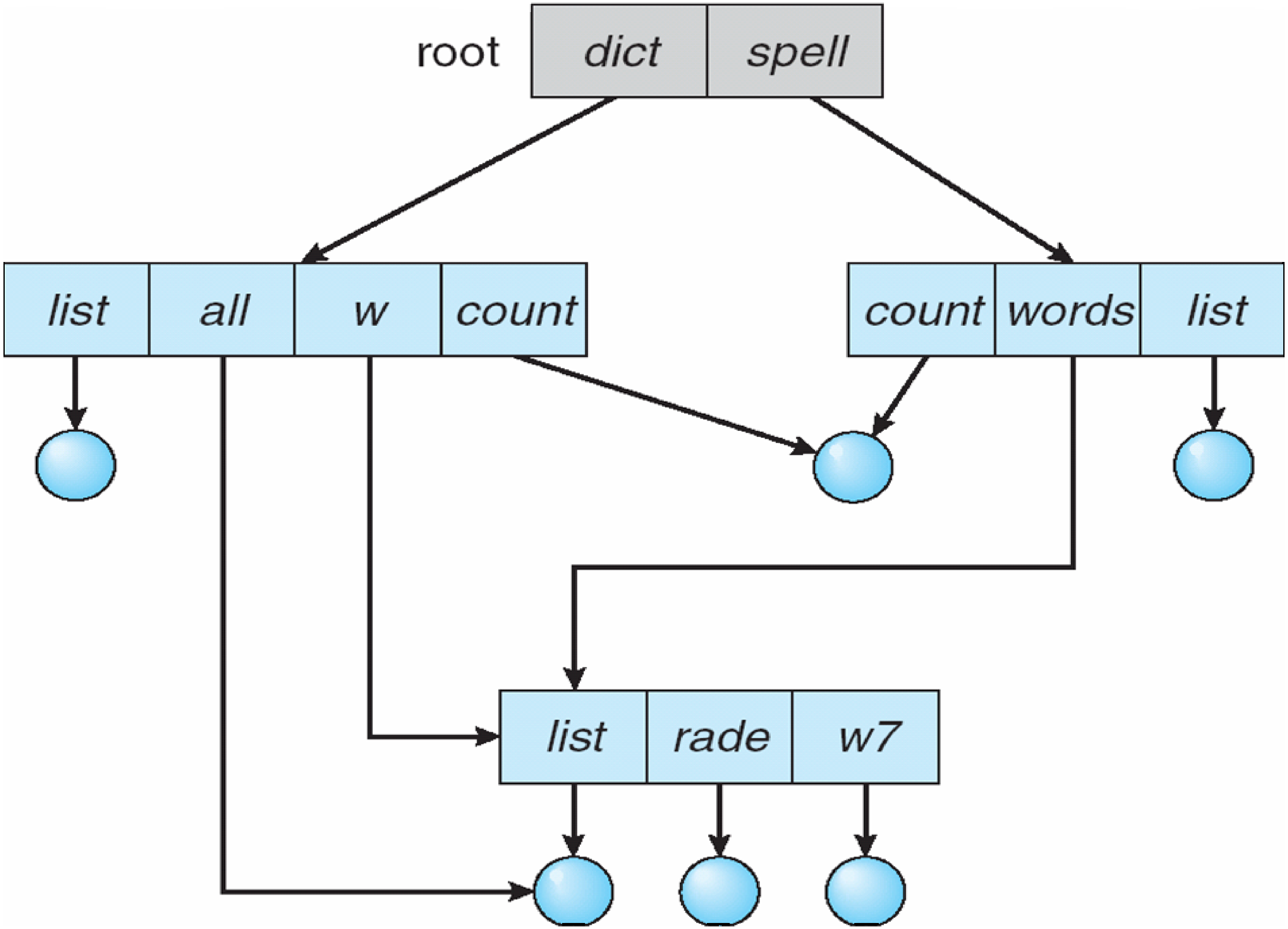
Empty: rm <dir. Name>

Not Empty: rm –r <dir. name?>

Acyclic-Graph Directories

- Multiple users can Have **shared subdirectories and files**
- **Users have their own working directory** and may have one shared directory
- Shared subdirectory created by one user in one directory is automatically visible to all users sharing that directory.
- Shared directory or file may exist at multiple places simultaneously
- Because of sharing, a file may have multiple absolute paths
- So different names can refer to same file

Acyclic-Graph Directories



General Graph Directories

- Created by adding links to the existing directory
- Allows cycles in the same directory

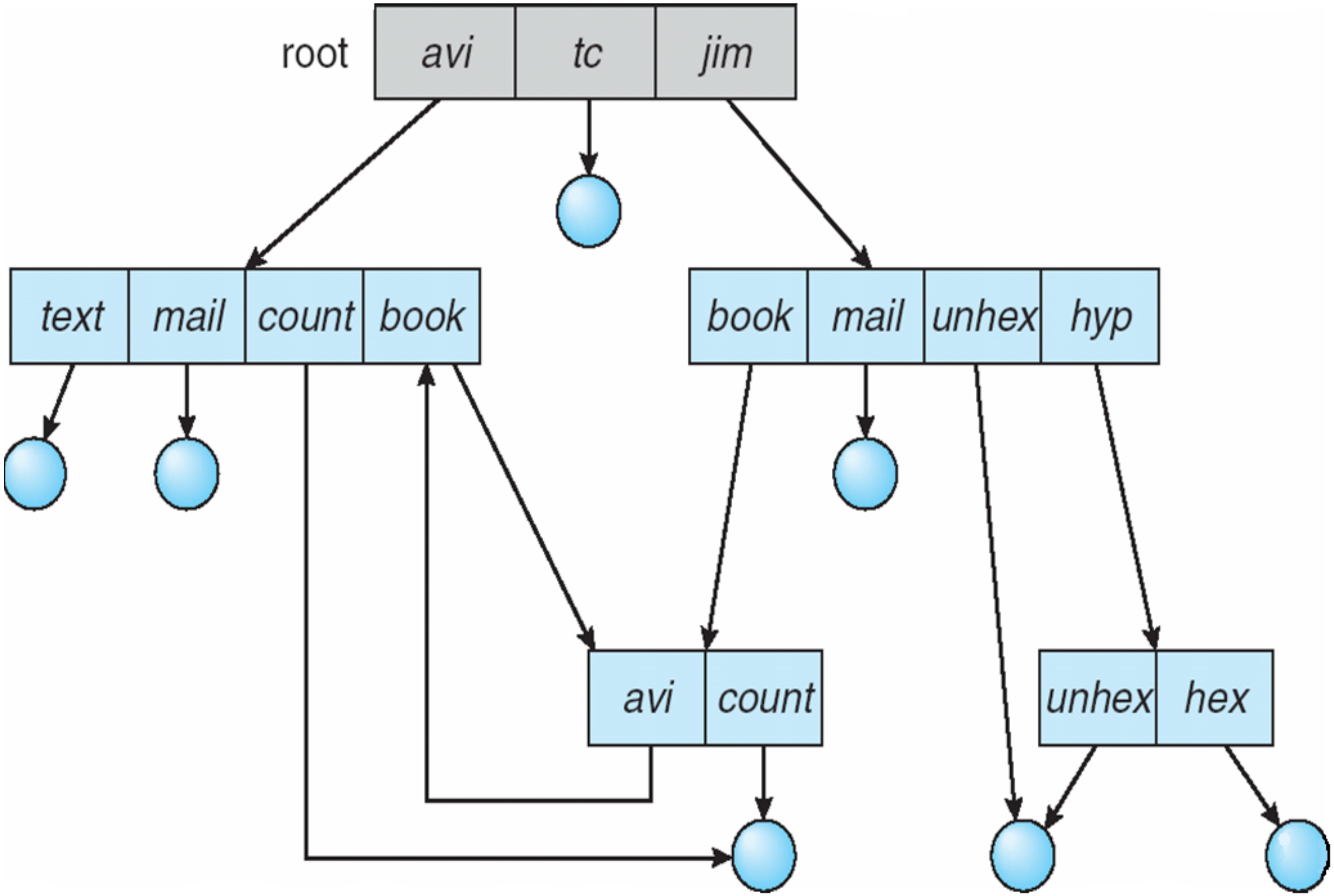
- As all files are dependent / linked deleting a main file may harm other files

- In case of deletion: Garbage Collection is used
- In First Pass: Traversing the entire file and marking everything that can be accessed

- In Second Pass: Collect everything that is not marked as the free space

General Graph Directories

- There can be cycle in the directory arrangement



Directory Implementation

- Directories need to be fast to search, insert, and delete, with a minimum of wasted disk space.

1 Linear List

- A linear list is the simplest and easiest directory structure
- Finding a file requires a linear search.
- Deletions can be done by moving all or one entry to vacant position and deleting the pointer.

2 Hash Table

- A hash table can also be used to speed up searches.
- Implementation is by using Hash value.
- **(Division/Variant Method)**

Implementing File-System

File System Structure

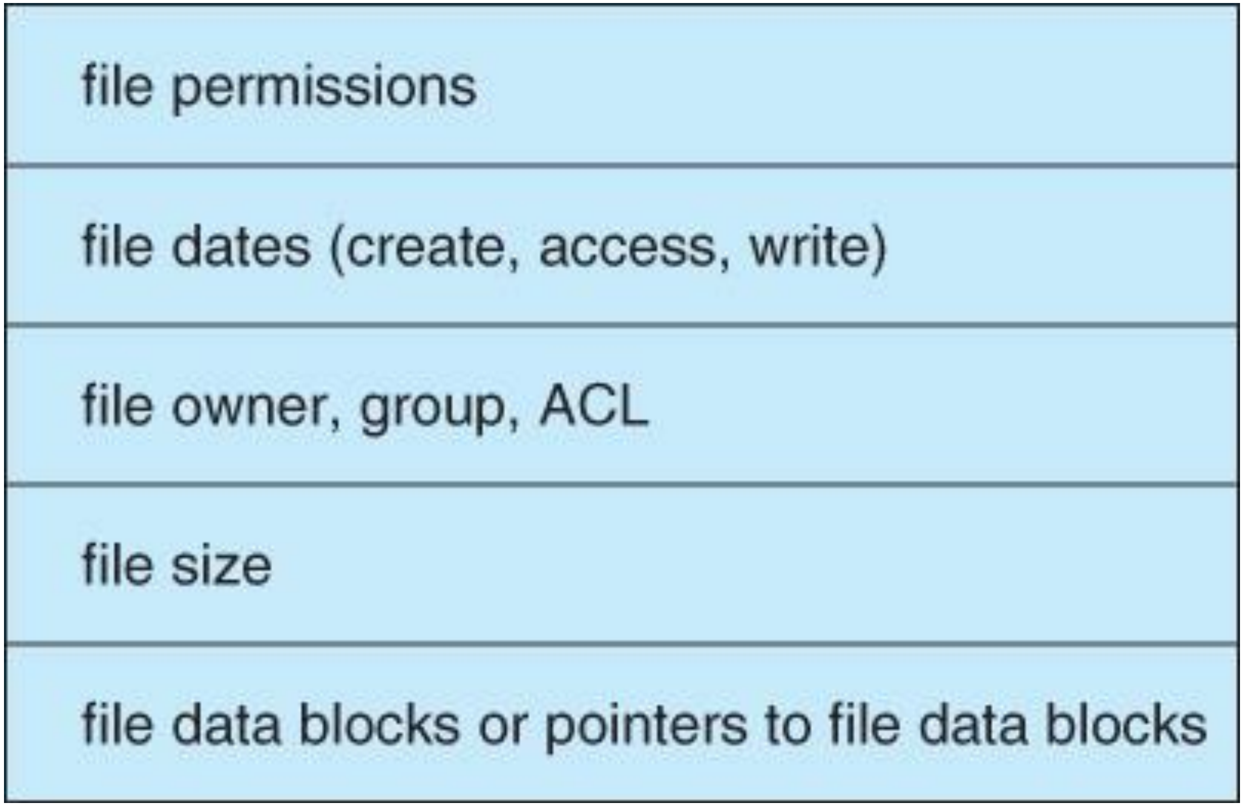
- File System resides on Secondary Storage

- Disk provides bulk storage on which a file system resides

- To improve I/O efficiency, **I/O transfers between memory and disk are performed in units of blocks.**

- File system has 2 design problems:
  - a) Defining **How** the file system **should look to user**
  - b) **Create a algorithm and data structure** to map the logical file system to physical file system.

File Control Block



file permissions

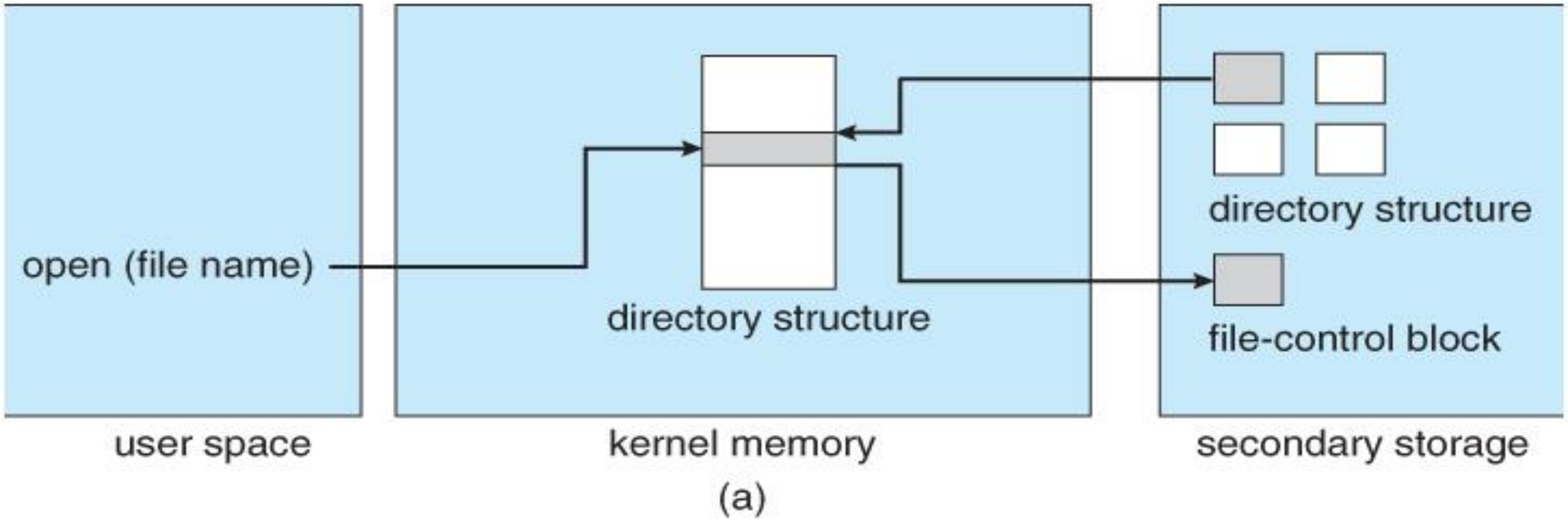
file dates (create, access, write)

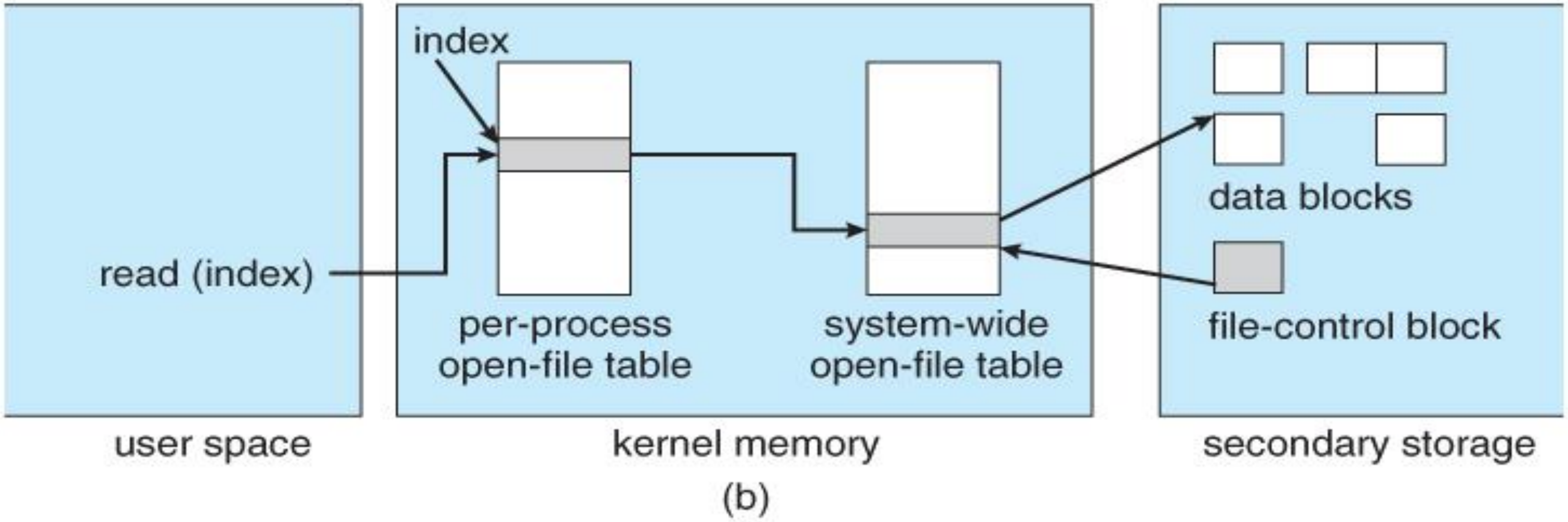
file owner, group, ACL

file size

file data blocks or pointers to file data blocks

In-memory file-system structures. (a) File open. (b) File read.





File System

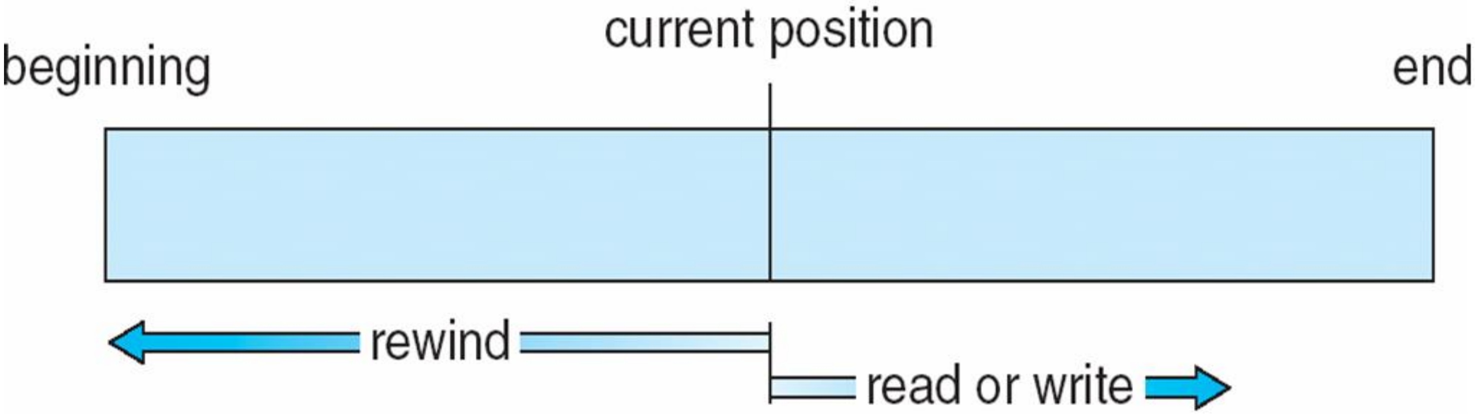
- It is a part of OS, responsible for controlling secondary storage space.

- File system consists of following:
  - **Access Methods:** Manner in which data stored in files is accessed
  - **File Management:** Provide mechanism for files to be stored, shared and secured.
  - **Storage Management:** Allocating space for files on secondary storage devices.
  - **File Integrity Mechanism:** Guaranteeing that file information is not corrupted

Access Methods

- 1. **Sequential Access:**
- Information of the file is processed in order, one record after the other.

read next: reads a portion of the file (read record) and automatically updates pointer. (move pointer to next location)
write next: Append to end of the file (write and update the pointer.
rewind or reset: to reset the file from beginning



Access Methods

2. **Direct Access: Or Relative access**

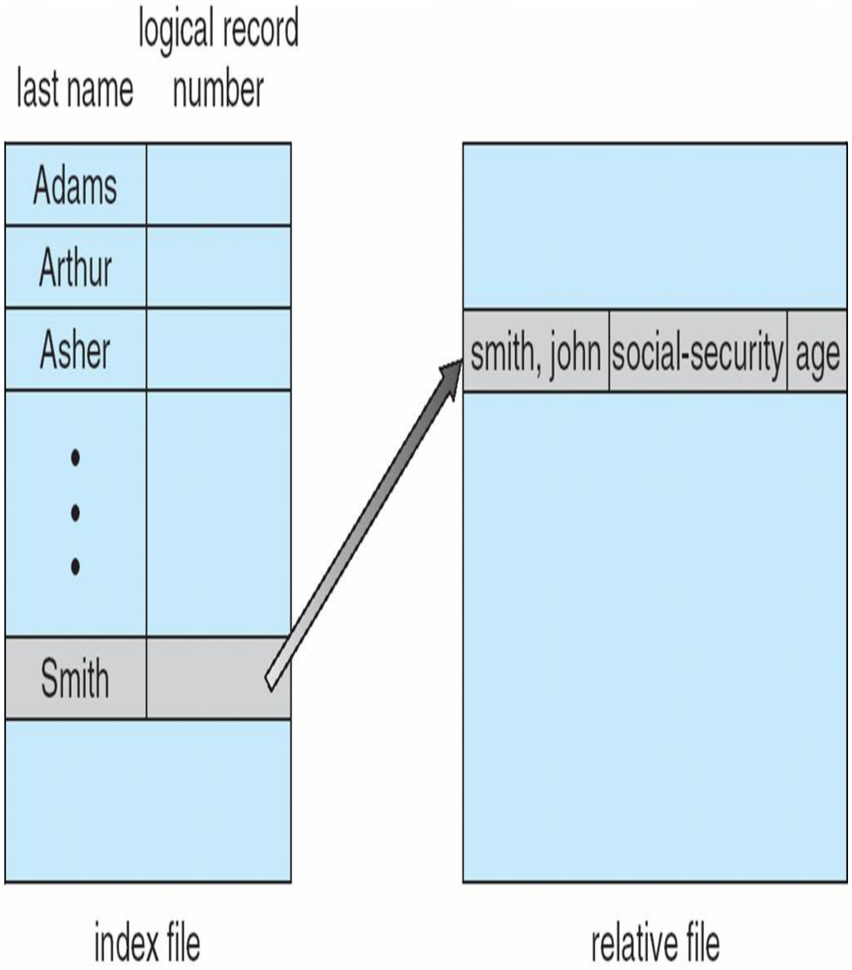
- A file is made up of fixed length logical records that allow programs to read and write records rapidly in no particular order.
- For Direct Access, File is a numbered sequence of blocks or records.

```
read n
write n
position to n (jump to record n)
read next
write next
        n = relative block number
```

- First Relative Block of file is 0, Second is 1 and so on. Absolute address could be anything.

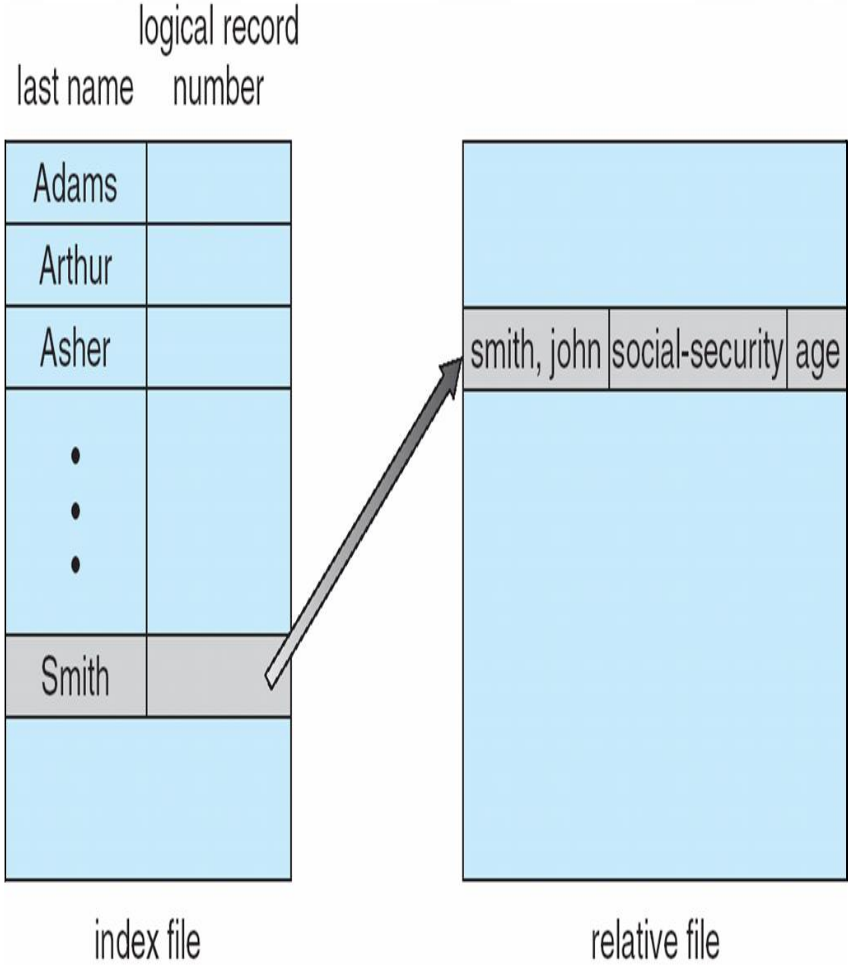
Indexed Access

- Index is created for file.
- Index contain pointers for various blocks of a file.
- To find the record in the file, search the index and then use the pointer to access the file directly and to find the desired record.



Indexed Access

- An **indexed file** is a computer file with an index that allows easy random access to any record given its file key.
- The key must be such that it uniquely identifies a record.
- A **relative file** is a **file** in which each record is identified by its ordinal position in the **file**



Indexed Sequential Access

- Advancement of Sequential Access
- Index of files is maintained in a sequential file
- Each entry contains:
  - Key Field
  - Pointer pointing to some record in a file

Allocation Methods

- Files are stored on disk

- Main Issue is: how to allocate space to these files so that disk space is utilized effectively and file can be accessed quickly.

- There are three major methods of storing files on disks:
  - Contiguous
  - Linked
  - Indexed

Allocation Methods

1. Contiguous Allocation

- Keep blocks of a file together in the continuous memory locations.
- Performance is very fast, because reading successive blocks of the same file generally requires no movement of the disk heads.

- Storage allocation involves issues:
  - first fit, best fit , worst fit etc

- Contiguous allocation of a file is defined by the **disk address and length of first block.**

Contiguous Allocation

- If file is **n blocks long and starts at location b**, the it occupies:

  b , b+1, b+2, ……. , b+n-1 locations

Directory entry for each file = Address of the starting block and length of area allocated.

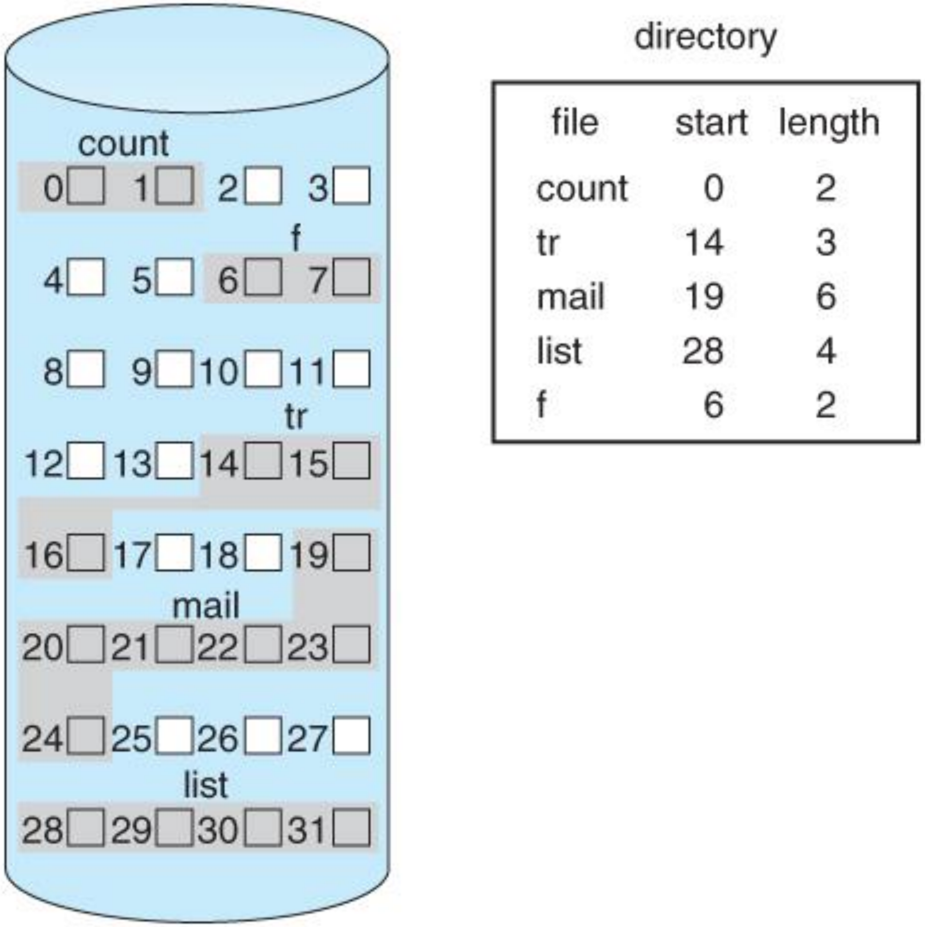
Best suited:

a) For Sequential Access: File system remembers disk address of the last block referenced and reads next block when required.

b) For Direct Access: Direct Access to **block i** of a file that starts at block b, block b+i can be accessed immediately.

Contiguous Allocation

- Problems can arise:
  - Finding space for a new file
  - When files grow
  - if the exact size of a file is unknown at creation time
  - Suffers from problem of external fragmentation.
  - Difficult to know how much space is needed for a file
  - Alot of space becomes unusable before the file fills the space



directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Linked Allocation

- Each file is a linked list of disk blocks
- **Disk blocks can be scattered any where in disk**
- Directory contains a pointer to first and last block of the file.
- Exp: if file of 5 blocks might start at block 9, continue at block 16, then block 1, block 10 and finally block 25.
- **Each block contains a pointer to next block.**
- If each block is 512 bytes and disk address requires 4 bytes then user see block of 508 bytes
- To create a new file , create a new entry in directory
- Each directory has a entry has a pointer to first disk block of file.
- Pointer is initialized to NIL and size / data part is set to 0

Linked Allocation

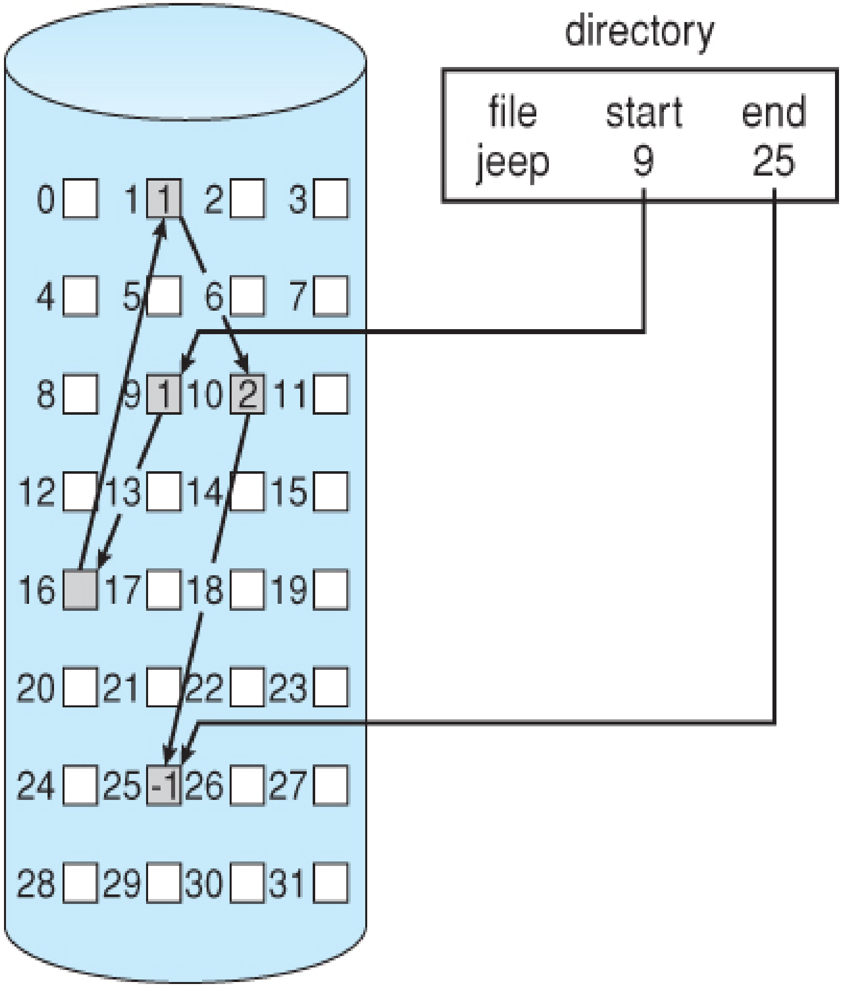
- Disk **files can be stored as linked lists**
- Linked allocation **does not require pre-known file sizes**, and allows files to grow dynamically at any time.

Drawback

- It is efficient for sequential access files
- To find ith block of a file, we must start at the beginning of that file and follow the pointers until we go to ith block.
- Requires extra space for pointers

- Problem with linked allocation is reliability:
  - **If a pointer is lost or damaged**

Linked Allocation



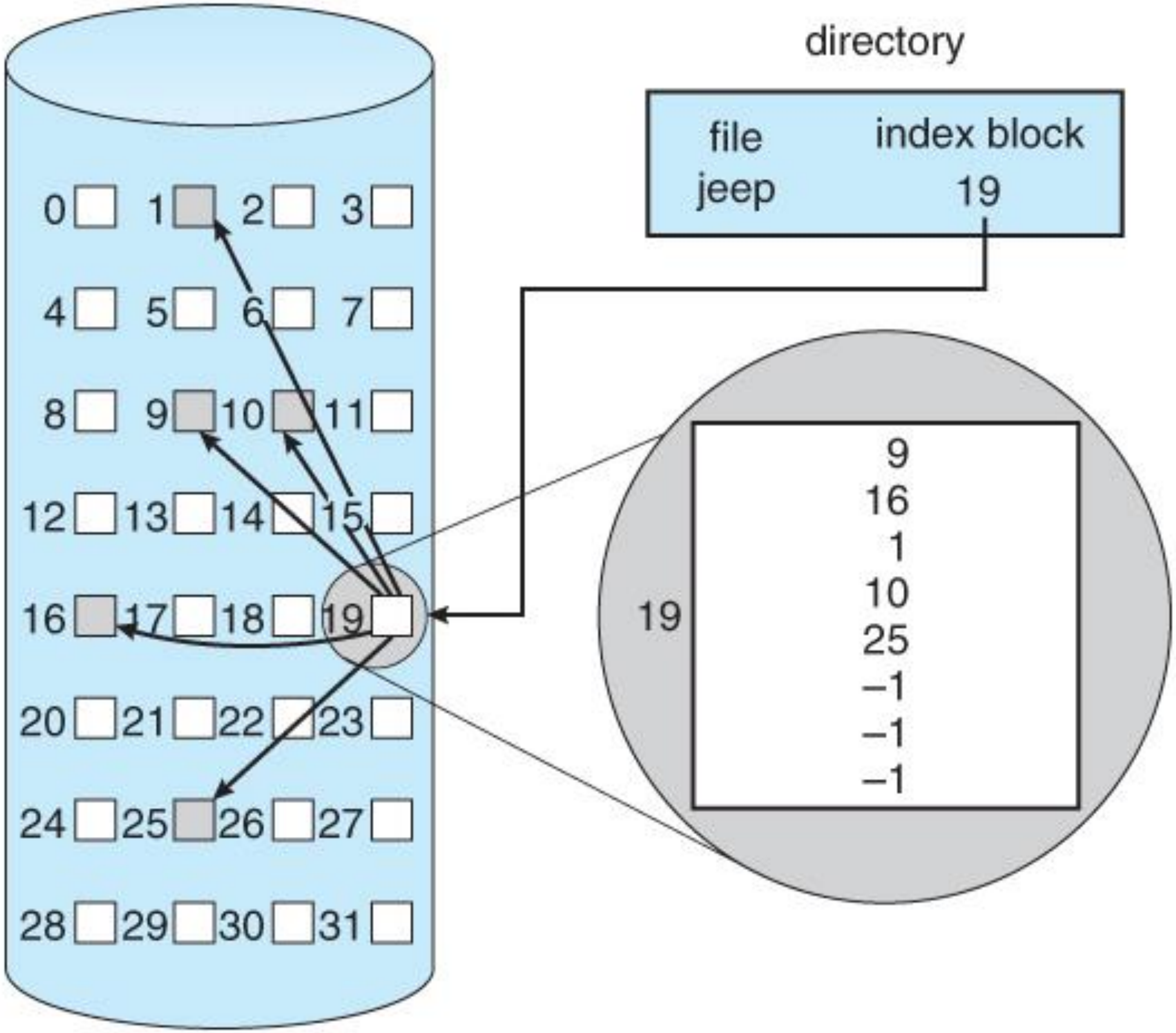
Indexed Allocation

- **Indexed Allocation** combines all of the indexes for accessing file in a file.
- **Create a table of indexes**
- Index file contains pointers to blocks.
- Bring all the pointers together into one location : Index Block
- Each ith entry in index block points to ith block of the file
- Create index of linked locations
- Indexed allocation support Direct Access.
  - Any free space on the disk can be used for allocation
  - Removes external fragmentation

Indexed Allocation

- **Advantages:**
  - Any free space on the disk can be used for allocation
  - Removes external fragmentation
- **Disadvantages:**
  - If index block is small, it will not be able to hold enough pointers
  - Entire index will have to be kept in main memory to make it work

Indexed Allocation



Free Space Management

- Process of looking after and managing unused blocks of disk
- Os maintains free space list: which records all blocks that are not allocated to any file

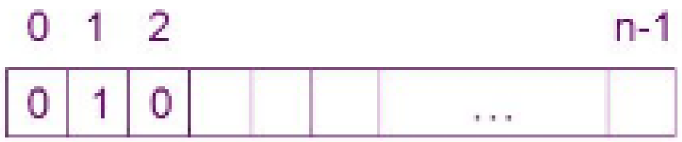
Methods to implement free space list:

- Bit Map or Bit Vector
- Linked List
- Grouping
- Counting

Methods to implement free space list

Bit Map or Bit Vector

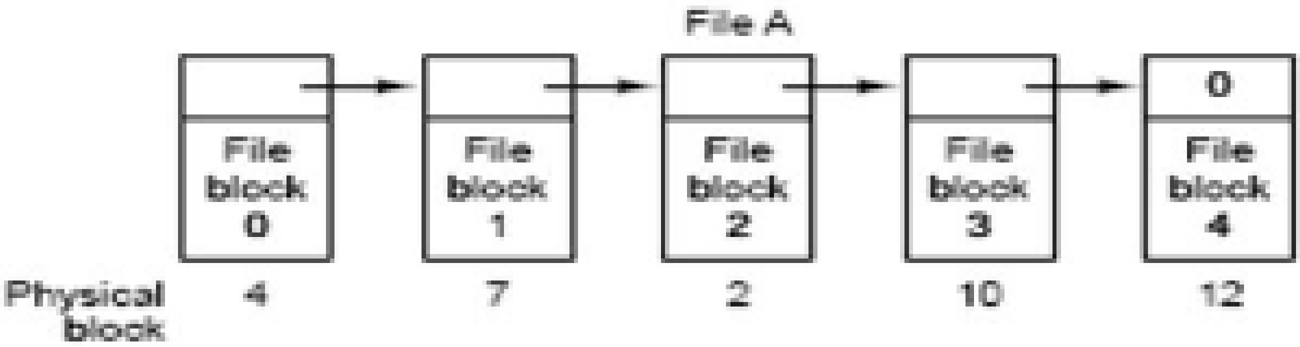
- Each block is represented by a bit
- If block is free, bit is set to 1 otherwise 0



$$\text{bit}[i] = \begin{cases} 0 \Rightarrow \text{block}[i] \text{ occupied} \\ 1 \Rightarrow \text{block}[i] \text{ free} \end{cases}$$

Linked List

- Linked list of all the free blocks is maintained
- First free block in the list can be pointed by the head pointer



- Disadvantage: Traversing is time consuming

Grouping

Grouping

- Maintained using linked list
- **Reserve few disk blocks for management**
- Modify linked list to:
  - **store address of next free blocks**
  - **A pointer to next block that contains free block pointers**
- In this address of n free blocks is stored in the first free blocks

Counting:

This method keeps the address of first free block and number of free contiguous blocks that follow the free block,

(X100 + 10 free blocks)

Mounting

Mount: Mounting makes file systems, files, directories, devices and special files **available for use** and available to the user.

Process that instructs the OS that a file system is:

- ready to use
- associates it with a particular point in the overall file system hierarchy
- sets options relating to its access