UNIT –IV (PART-II)

Information management :

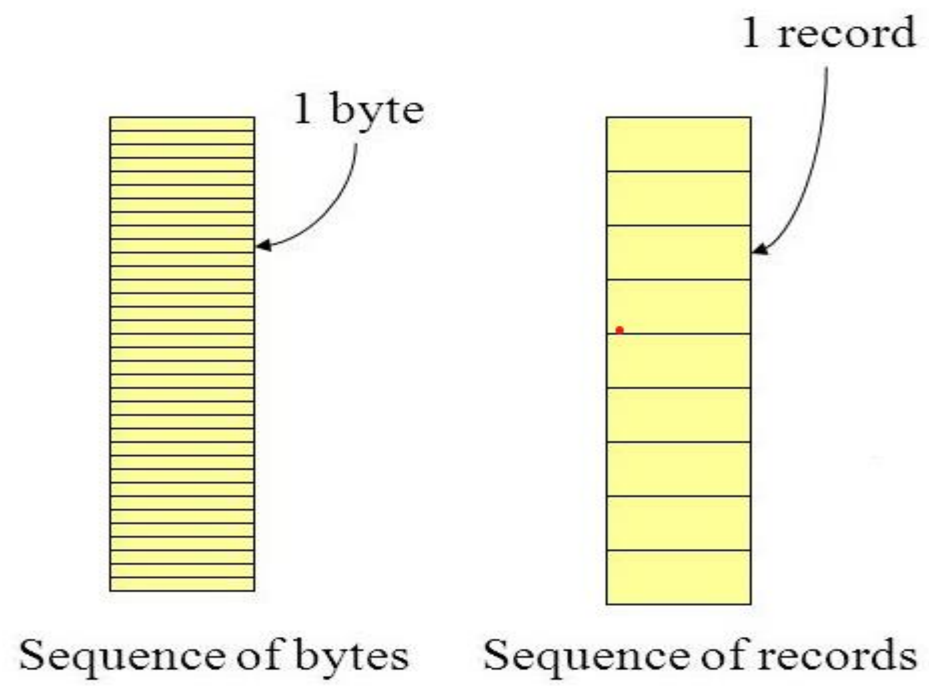
- Files Concept and Systems
- File Implementation (Allocation methods)
  - Contiguous Allocation
  - Linked Allocation
  - Indexed Allocation
- Free-Space Management
- Directory structures
- Directory implementation
  - linear list
  - hash table

Information management

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

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.



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** – data for usage monitoring

File Operations

All operations involve some system calls

1. Create: using a specific system call.

Two steps are necessary to create a file.

1) Space in the file system must be found for the file.

2) 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

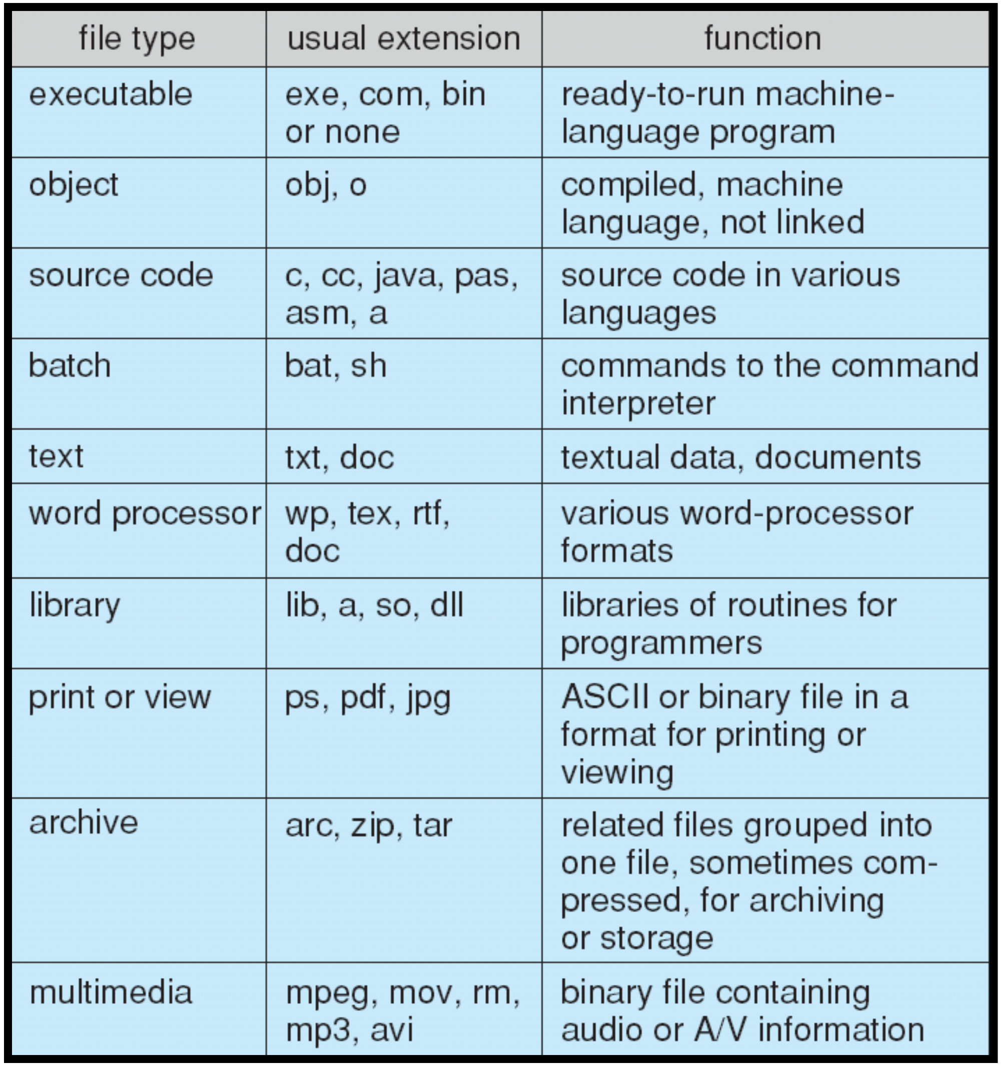
3. Write:

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

4. Delete

5. Truncate

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**

- There should be minimum redundancy in data
- Redundancy can be used to speed up the access

2. Simple

3. Maintenance

4. Reliability

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
  - **Storage Management:** Allocating space for files on secondary storage devices.
  - **File Management:** Provide mechanism for files to be stored, shared and secured.
  - **File Integrity Mechanism:** Guaranteeing that file information is not corrupted

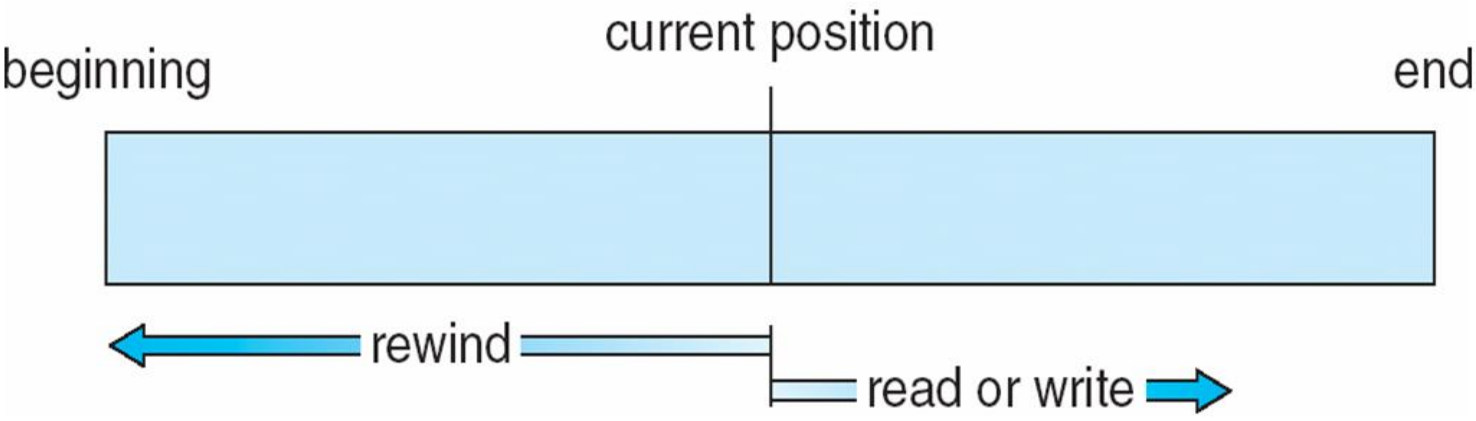
Access Methods

- 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.

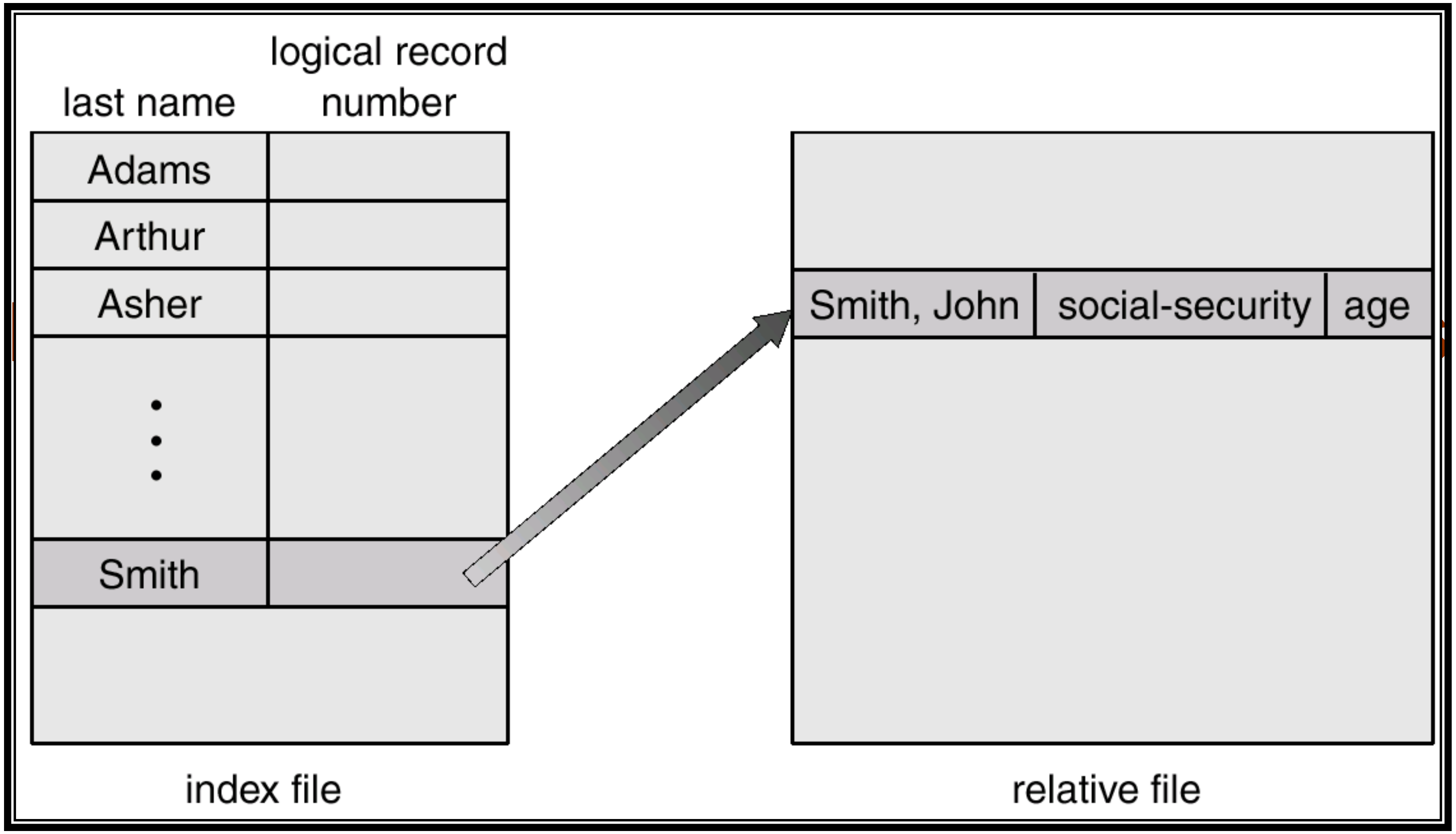
```
read n
write n
position to n (jump to record n)
read next
write next
```

n = relative block number

3.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.
- 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**

Example of Index and Relative Files



File Management

- Allocation methods
  - Contiguous Allocation
  - Linked Allocation
  - Indexed Allocation
- Free-Space Management

Allocation Methods

- It is the mechanism of keeping track of which disk blocks go which files

- 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

1. Contiguous Allocation

- Store each file as a contiguous block of data on the disk.

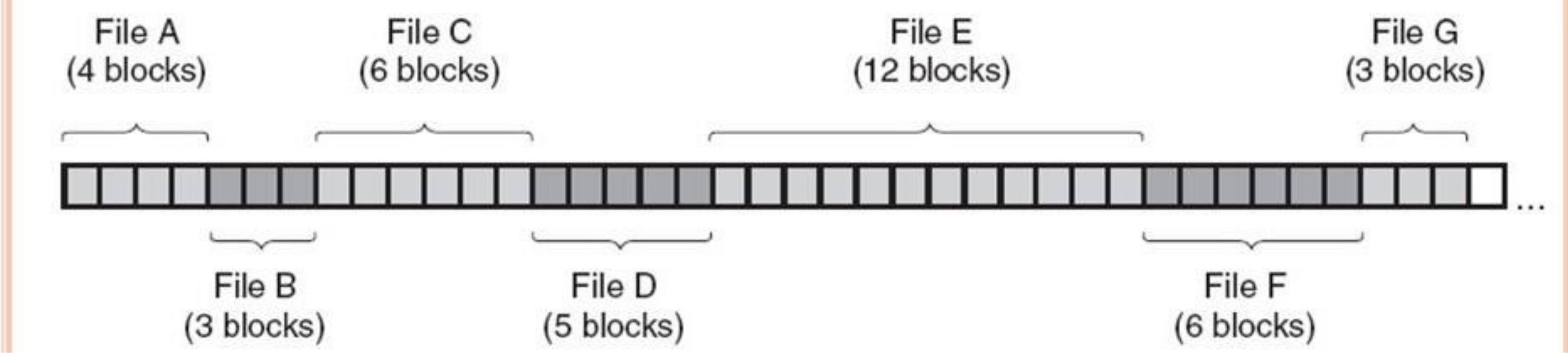
- Performance is very fast

  - Reading successive blocks of the same file generally requires no movement of the disk heads

- It is easy to implement

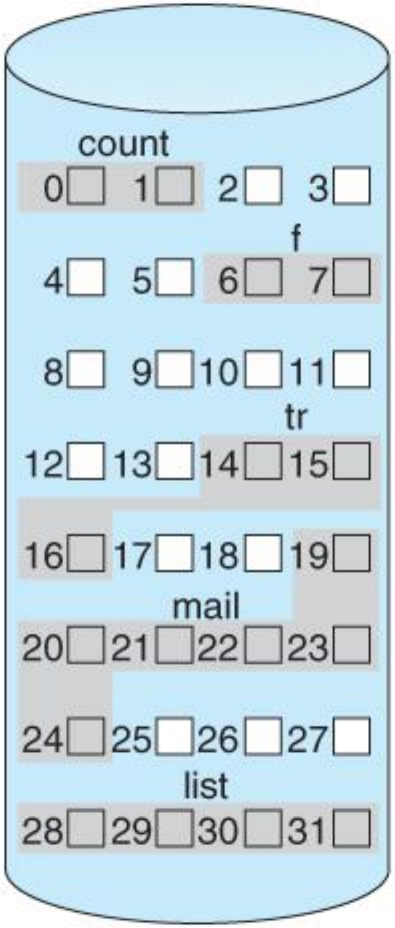
- Contiguous allocation of a file is defined by the **disk address and length of first block.**

Contiguous Allocation



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



directory

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

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 an entry as a pointer to first disk block of file.
- Pointer is initialized to NIL and size / data part is set to 0

Linked Allocation

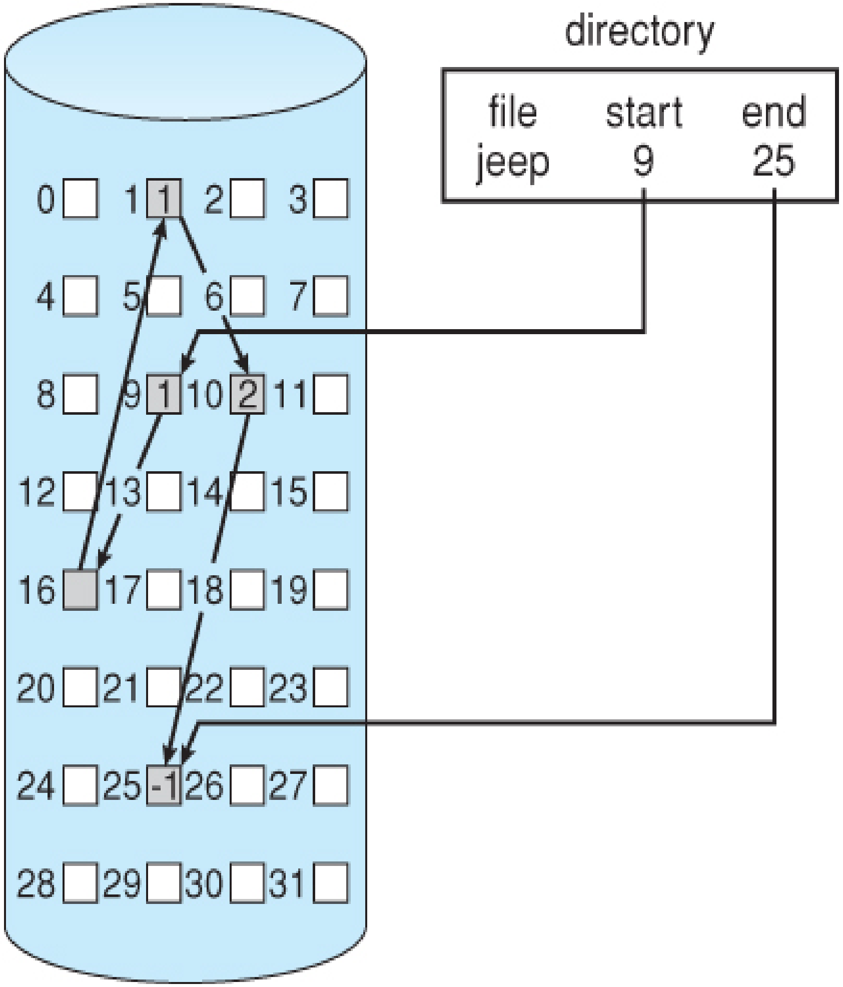
- Disk files can be stored as **linked lists**
- Linked allocation **does not require pre-known file sizes** and allows files to **grow dynamically** at any time.

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

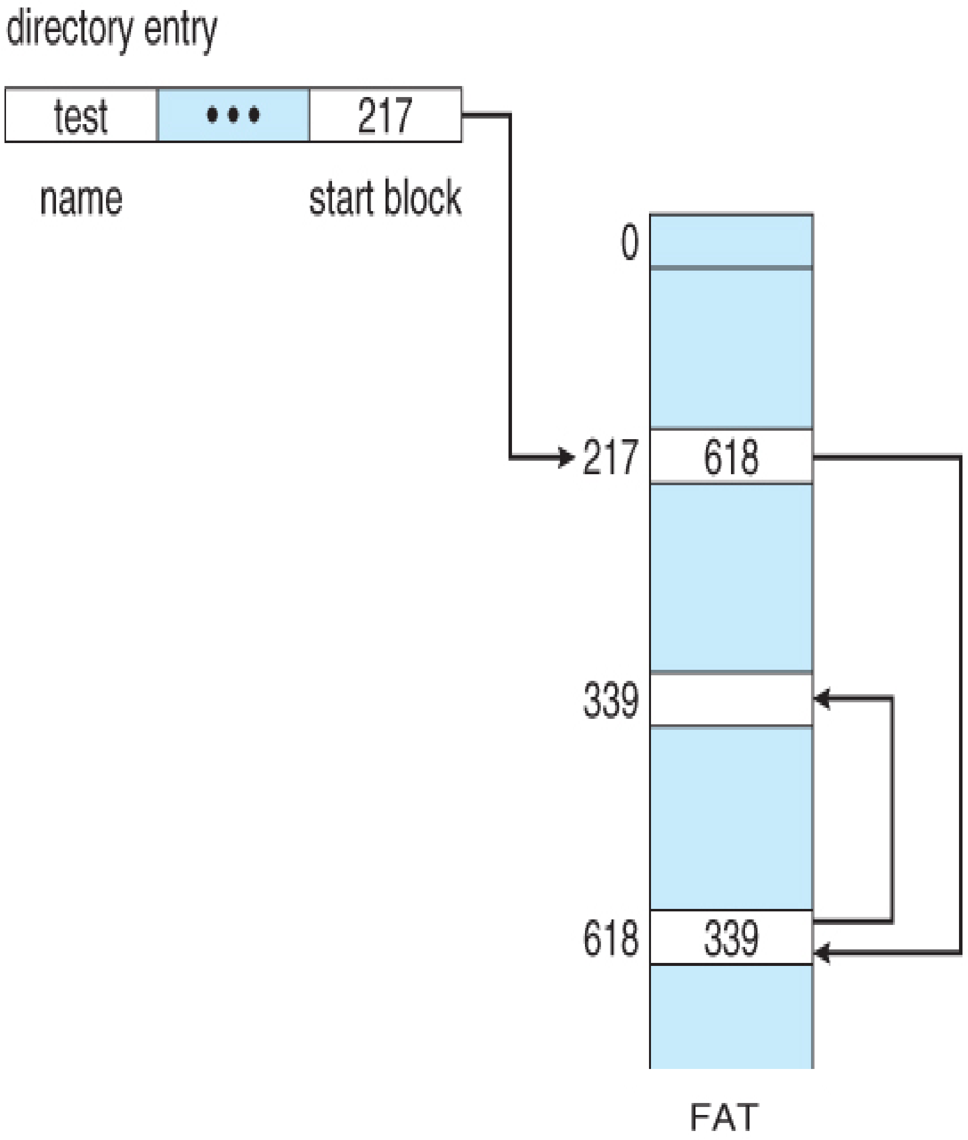


directory

file	start	end
jeep	9	25

File Allocation Table

- Used by DOS
- It is a variation of linked allocation
- All the links are stored in a separate table at the beginning of the disk.
- The benefit of this approach is that the FAT table can be cached in memory, and improving random access speeds.



3. Indexed Allocation

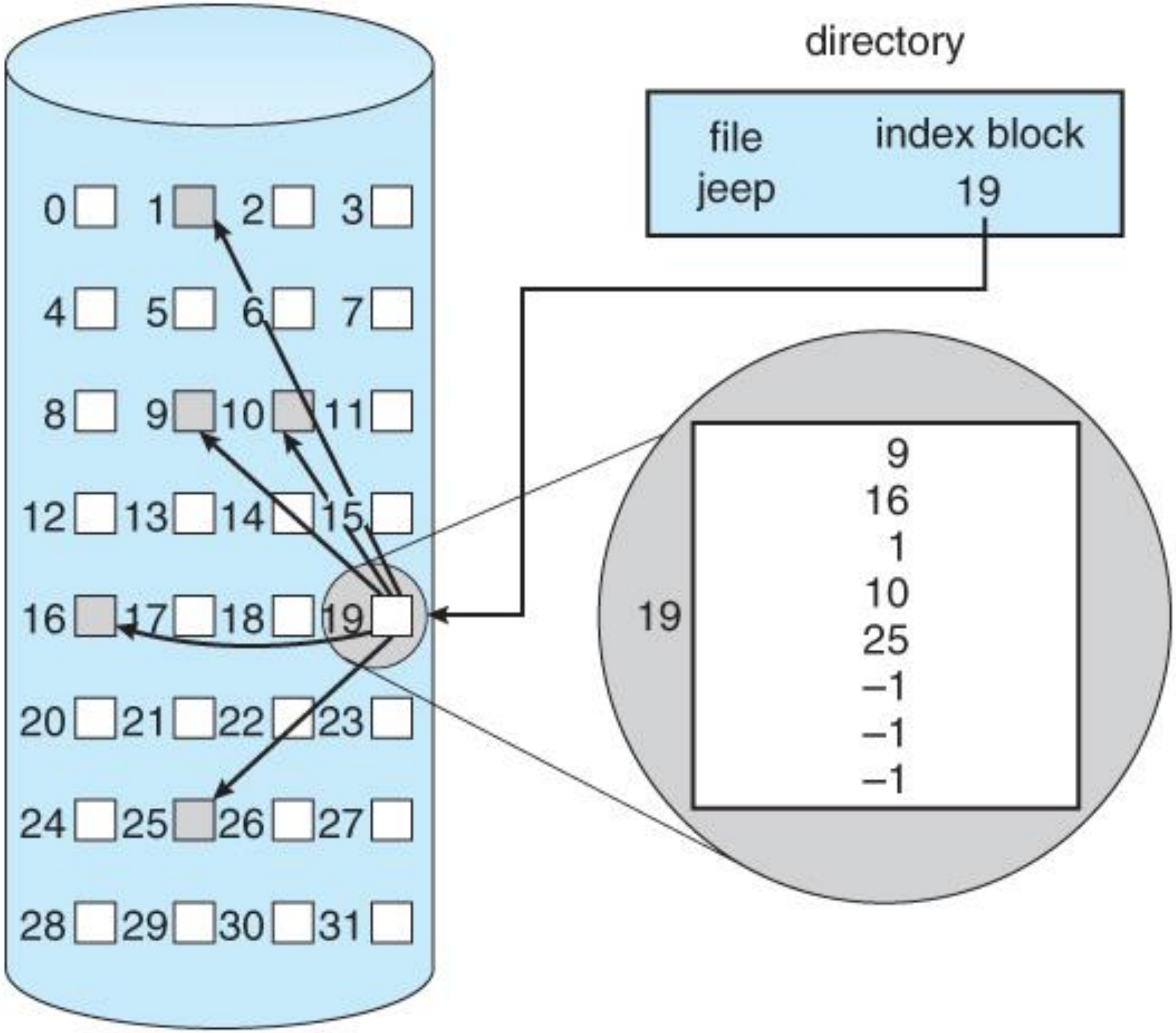
- **Indexed Allocation** combines all of the indexes for accessing file in a file.
- **Create a table of indexes**
- Index file contains pointers to blocks.
- Bring all the pointers together into one location : **Index Block**
- Each ith entry in index block points to ith block of the file
- Create index of linked locations
- Indexed allocation support **Direct Access**.

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



Approaches to implement Index Block

- For large files several indexes can be combined and maintained.

- The first index block contains:
  - Header information
  - First N block addresses
  - Pointer to linked index blocks.

- **Multi-Level Index -**

index block contains → pointers to secondary index blocks → which further contain pointers to the actual data blocks.

Approaches to implement Index Block

- **Combined Scheme -**
- pointers provide access to more data blocks
- The advantage of this scheme:
  - for small files the data blocks are readily accessible

Free Space Management

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

Bit Map or Bit Vector

- A Bitmap or Bit Vector is series or collection of bits where each bit corresponds to a disk block.
- The bit can take two values: 0 and 1: *0 indicates that the block is allocated* and 1 indicates a free block
- *Figure 1* (where green blocks are allocated) can be represented by a bitmap of 16 bits as: **0000111000000110**.

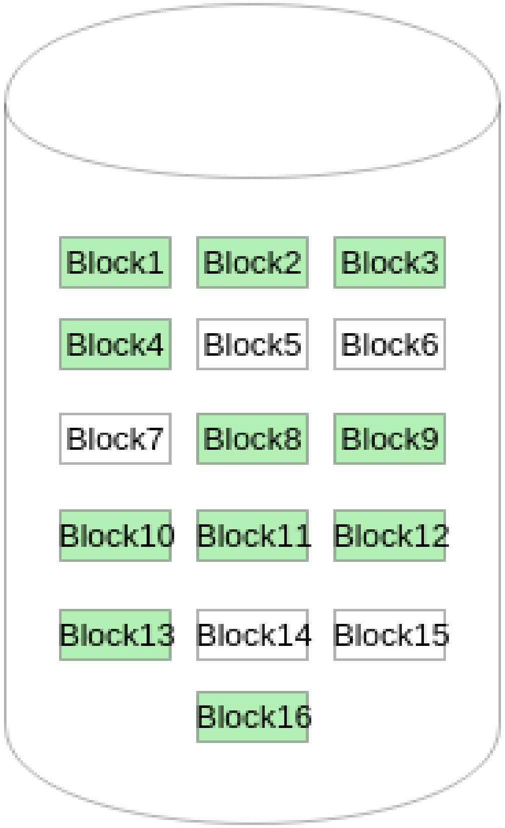


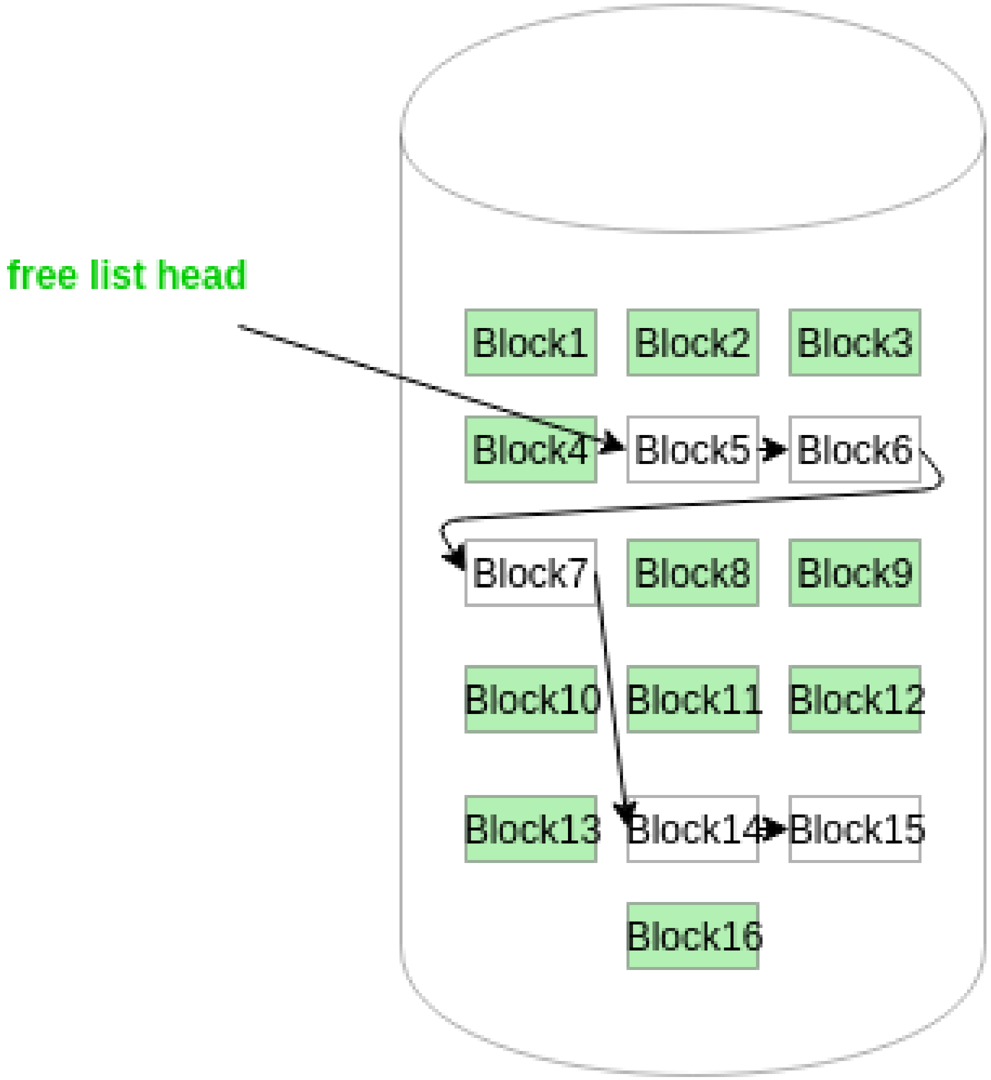
Figure - 1

Bit Map or Bit Vector

- Find the location of the first free block.

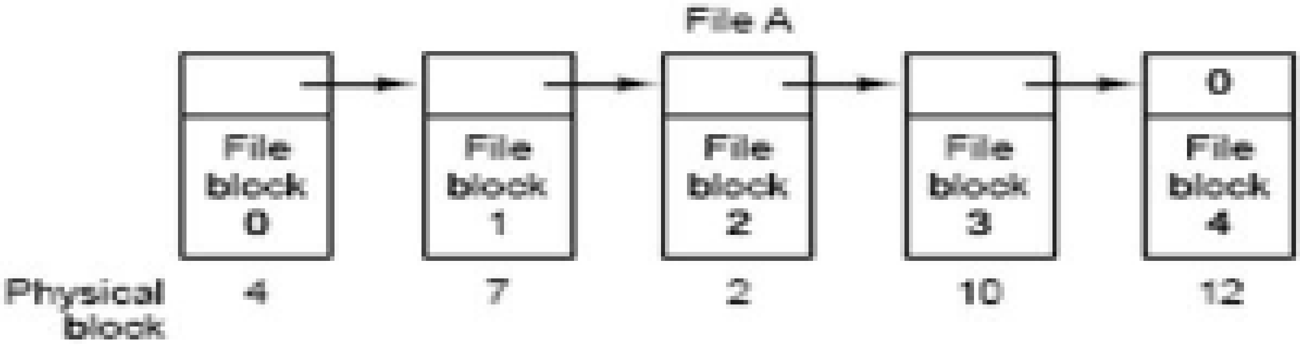
- (number of bits per word) x (number of 0-value words) + offset of first 1 bit.

Linked List


free list head
Block1
Block2
Block3
Block4
Block5
Block6
Block7
Block8
Block9
Block10
Block11
Block12
Block13
Block14
Block15
Block16

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

- This approach stores the address of the free blocks in the first free block.
- The first free block stores the address of some, say n free blocks.
- Out of these n blocks, the first n-1 blocks are actually free and the last block contains the address of next free n blocks.

Counting

- This approach stores the address of the first free disk block and a number n of free contiguous disk blocks that follow the first block.

Every entry in the list would contain:

- Address of first free disk block
- A number n

Implementing File-System

File System Structure

- File System resides on Secondary Storage

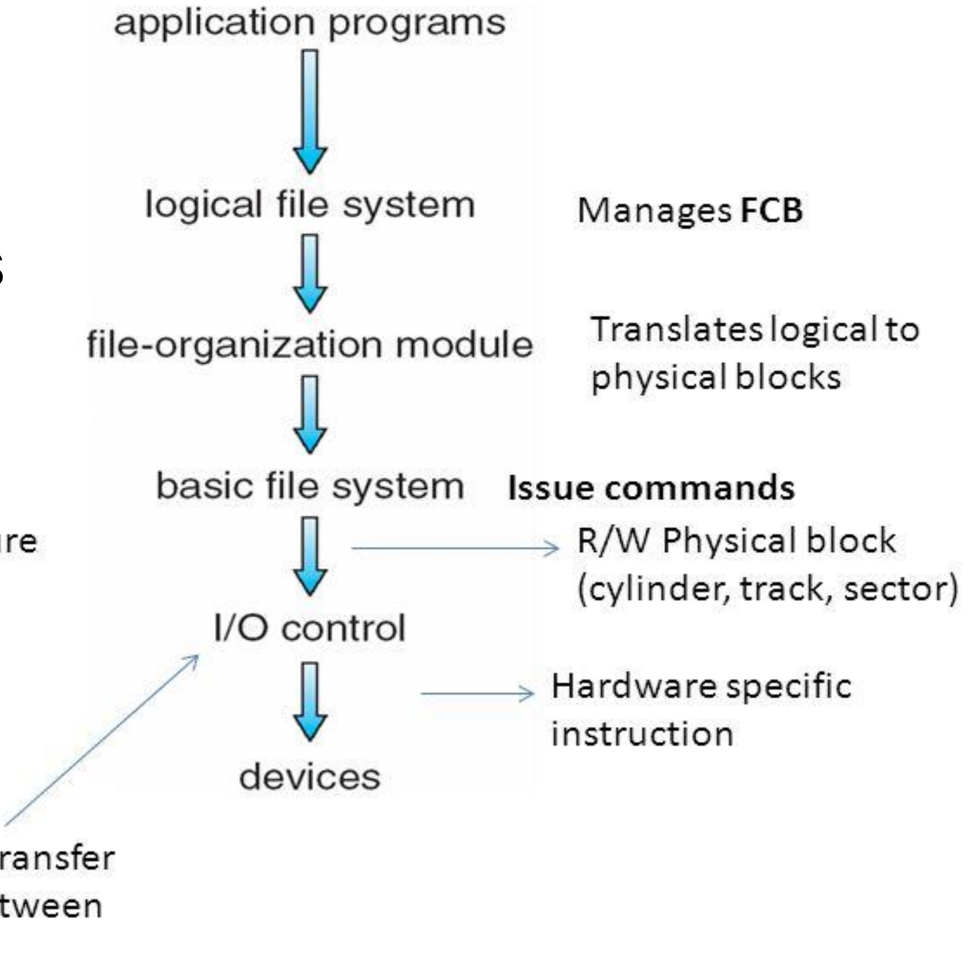
- Disk provides bulk storage on which a file system resides

- 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 an algorithm and data structure** to map the logical file system to physical file system.

Layered File System

Logical File System contains info. Meta Data, and manages file structure



application programs

logical file system

Manages **FCB**

file-organization module

Translates logical to physical blocks

basic file system

Issue commands

R/W Physical block (cylinder, track, sector)

I/O control

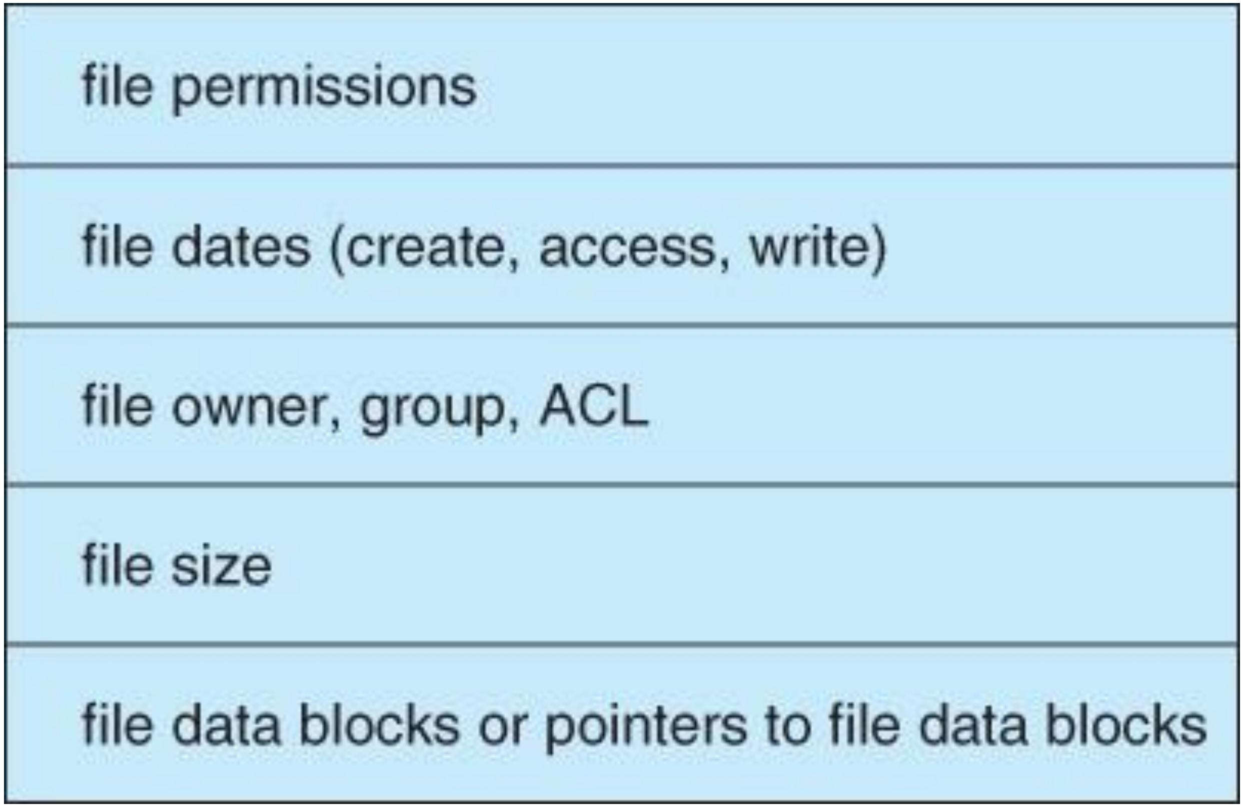
Hardware specific instruction

devices

- Each level uses the feature of low level
- Create new features for higher level

Device driver, transfer information between memory/disk

File Control Block



file permissions

file dates (create, access, write)

file owner, group, ACL

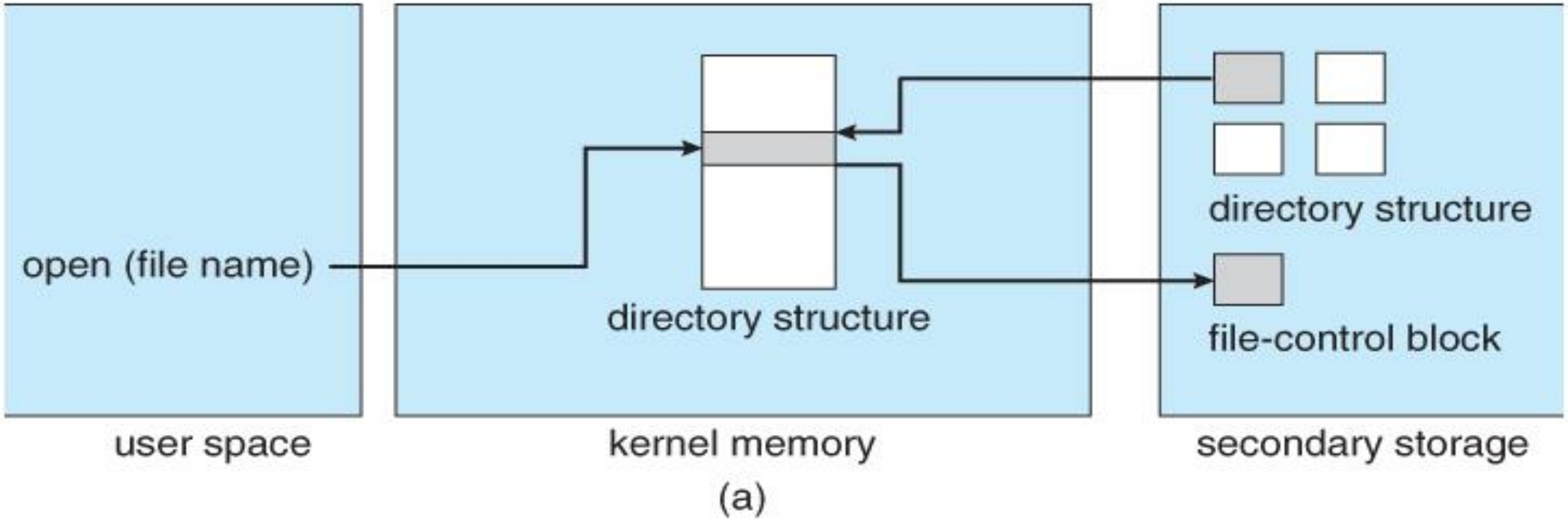
file size

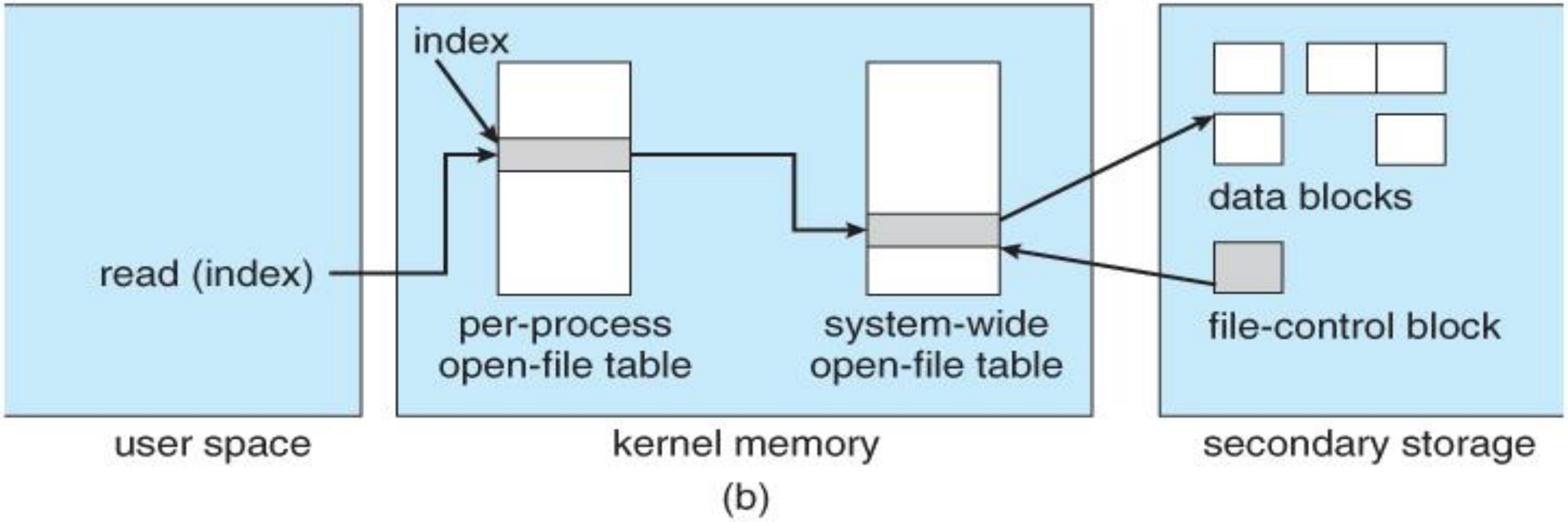
file data blocks or pointers to file data blocks

File System Implementation

- On disk, file System may contain information about
  - **a) Boot Control Block:** Contains info. Needed by the system to boot an OS. This block is empty if there is no OS.
  - **b) Volume Control Block:** Contain partition details:
    - **No. of blocks in partition**
    - **Size of blocks**
    - **Block pointers etc.**
  - **c) Directory Structure:** Used to organize files
  - **d) File Control Block:** Contains info. About file:
    - **File Permissions**
    - **Modification dates**
    - **File owner, File size and Pointers**

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





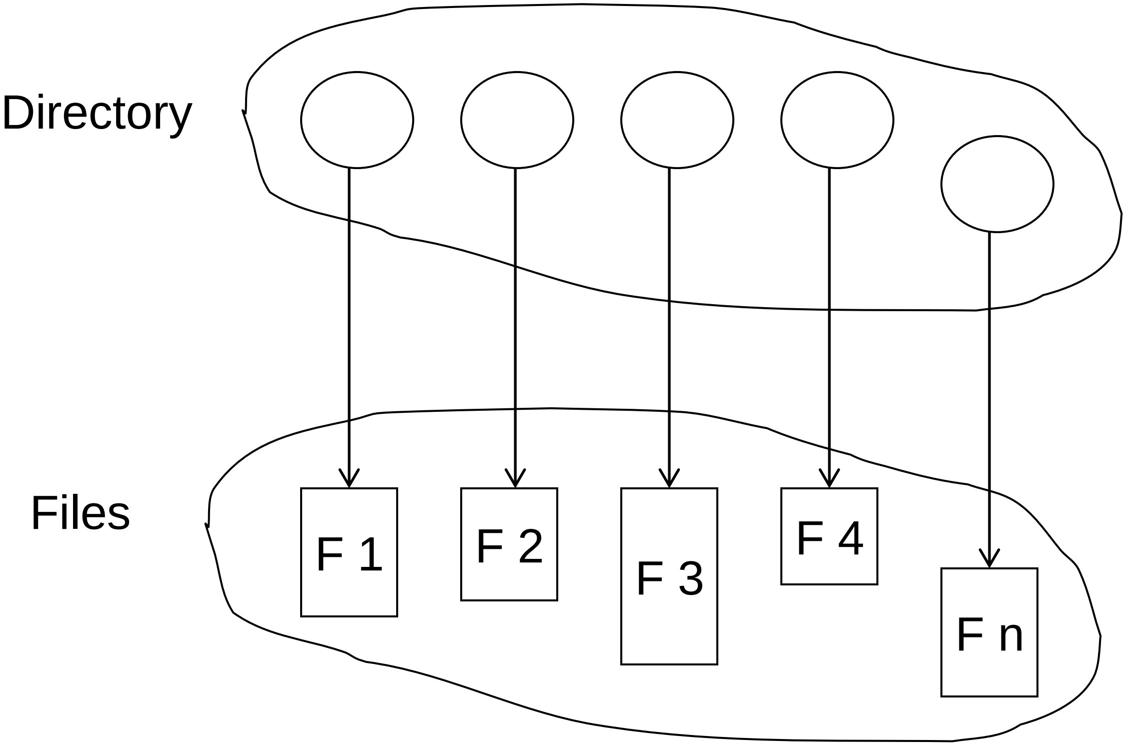
UNIT –IV (PART-II)

Information management

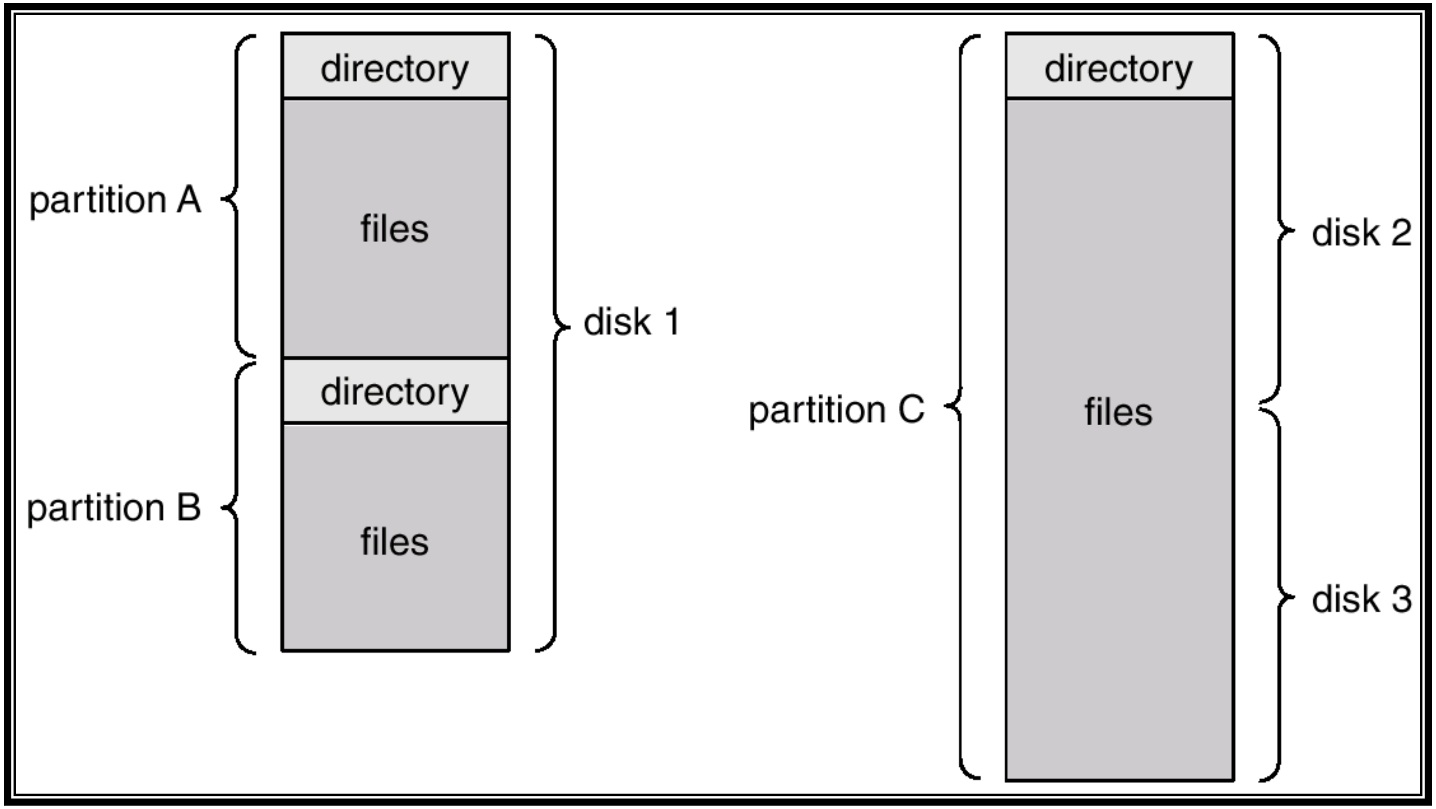
- Files Concept and Systems
- File Implementation (Allocation methods)
  - Contiguous Allocation
  - Linked Allocation
  - Indexed Allocation
- Free-Space Management
- File System Implementation
- Directory structures
- Directory implementation
  - linear list
  - hash table

Directory Structure

- A collection of nodes containing information about all files.



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


directory
partition A
files
disk 1
directory
partition B
files
directory
disk 2
partition C
files
disk 3

Operations Performed on Directory

Directory: collection of files or directories

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

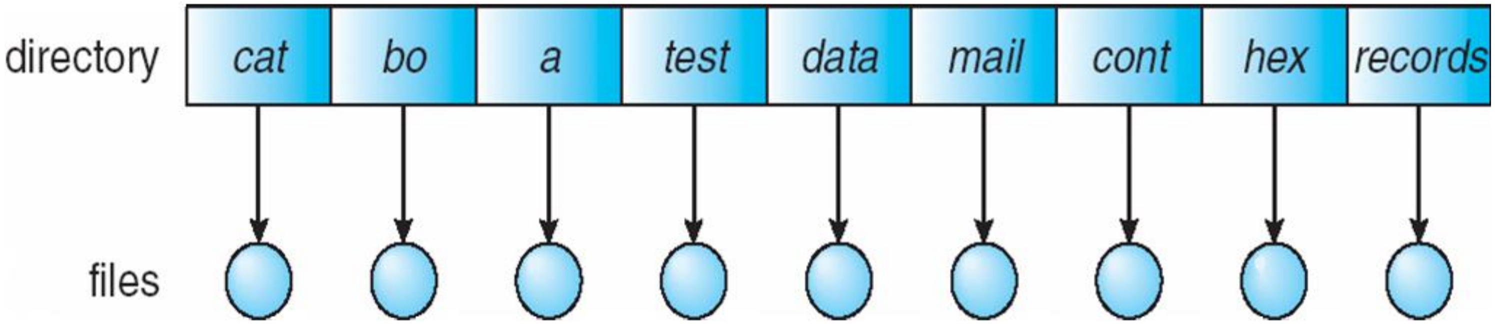
Operations:

- 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



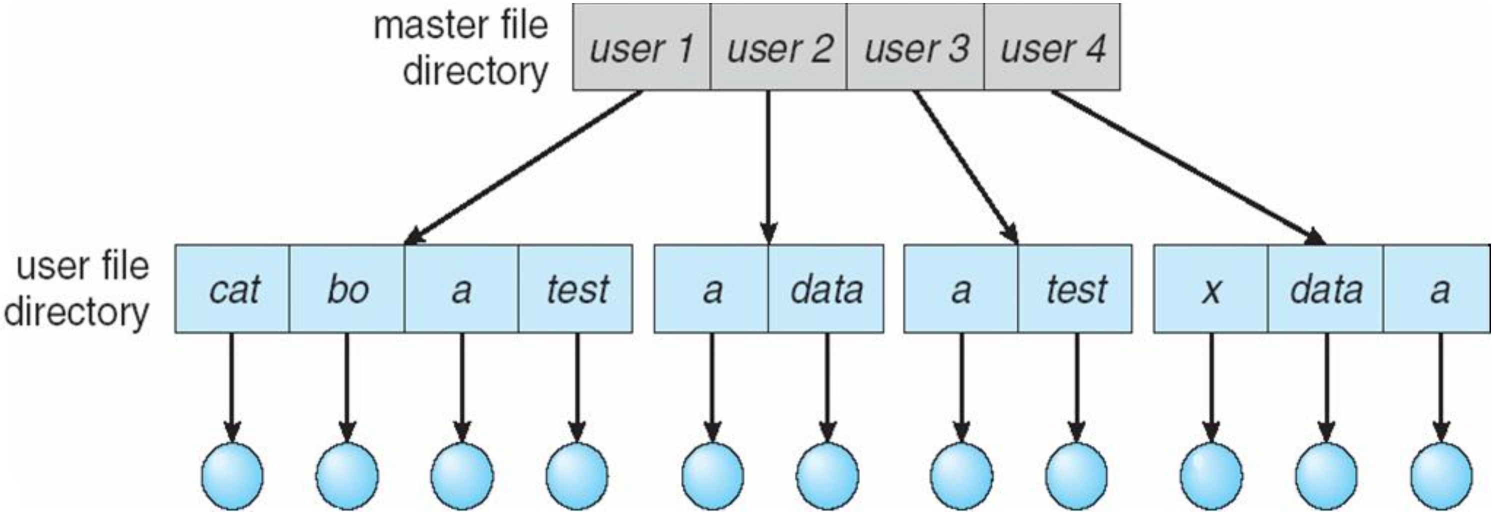
Disadvantage:

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

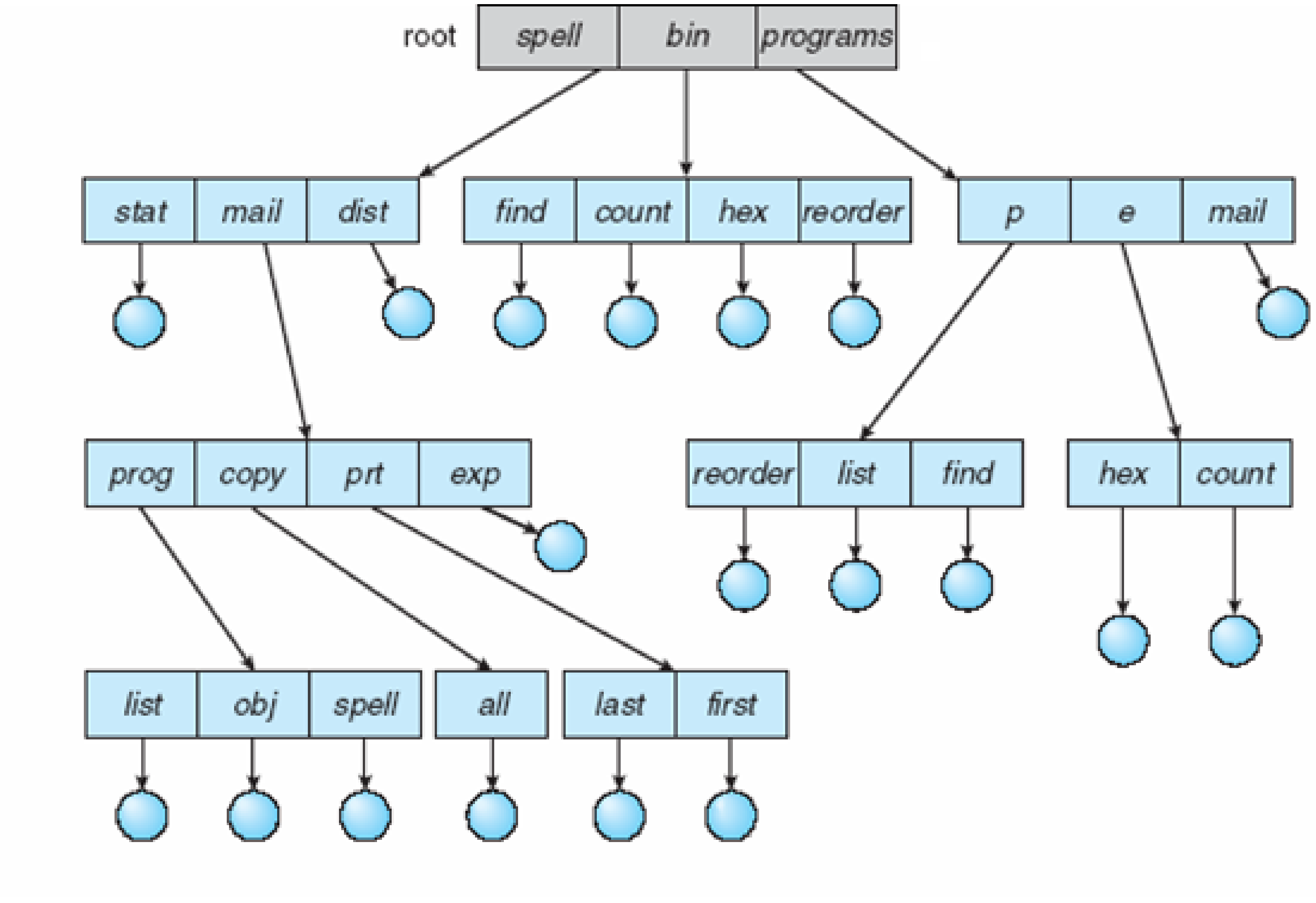
- **2. Two level directory**, **each user has his own user file directory(UFD).**

- 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



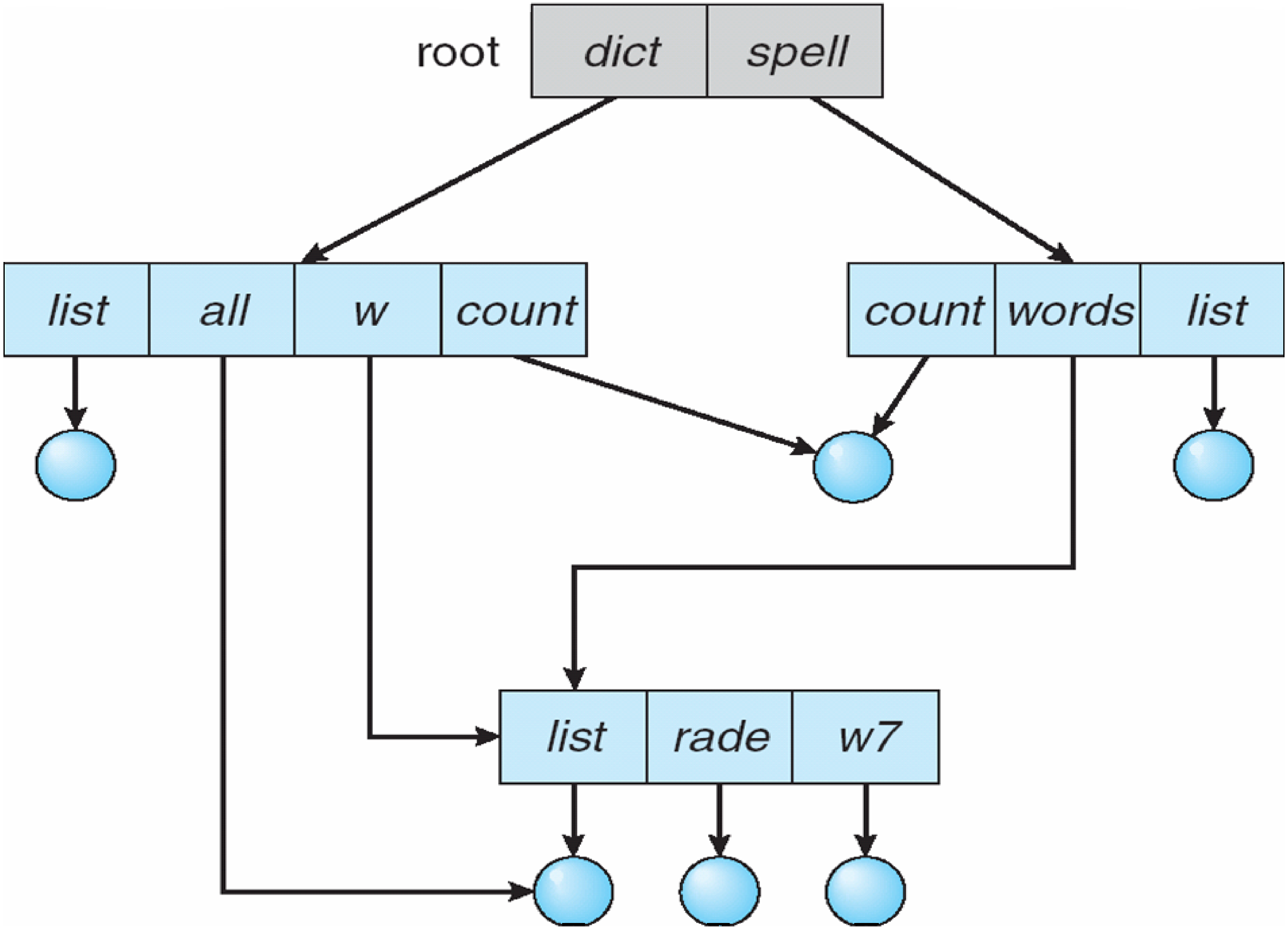
Acyclic-Graph Directories

- 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

- These kind of directory graphs can be made using links.
- Links can either be symbolic (logical) or hard link (physical).
- If a file gets deleted then,
  - In case of soft link , the file just gets deleted, and we are left with a **dangling pointers**.
  - In case of hard link, the actual file will be deleted only if all the references to it gets deleted.

Acyclic-Graph Directories



General Graph Directories

- Cycles are allowed within a directory structure where multiple directories can be derived from more than one parent directory.
- The main problem with this kind of directory structure is to calculate the total size or space that has been taken by the files and directories.

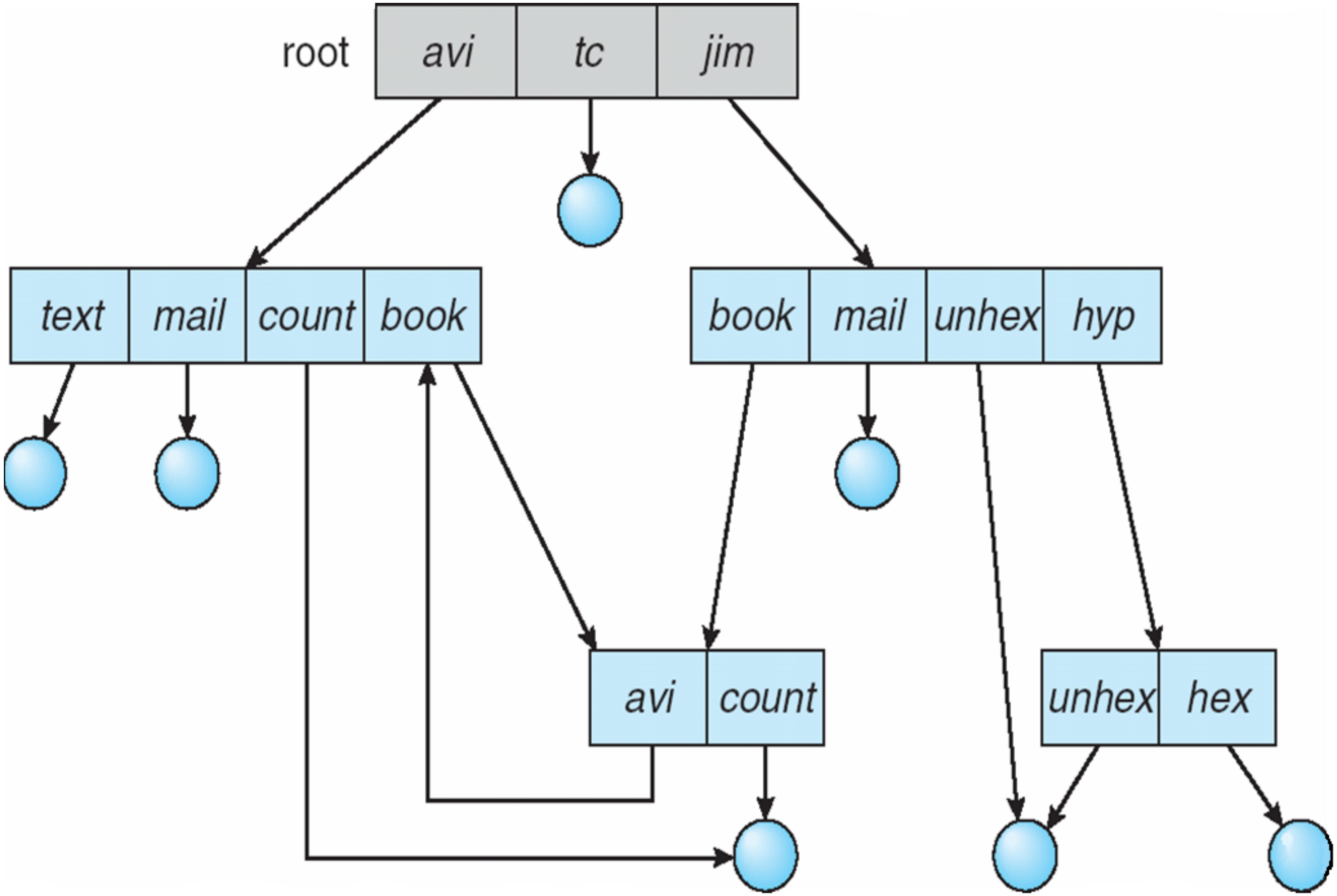
General Graph Directories

- Advantages:
  - It allows cycles.
  - It is more flexible than other directory's structure.

- Disadvantages:
  - It is more costly than others.
  - It needs garbage collection.

General Graph Directories

- There can be cycle in the directory arrangement



Directory Implementation

Directory Implementation

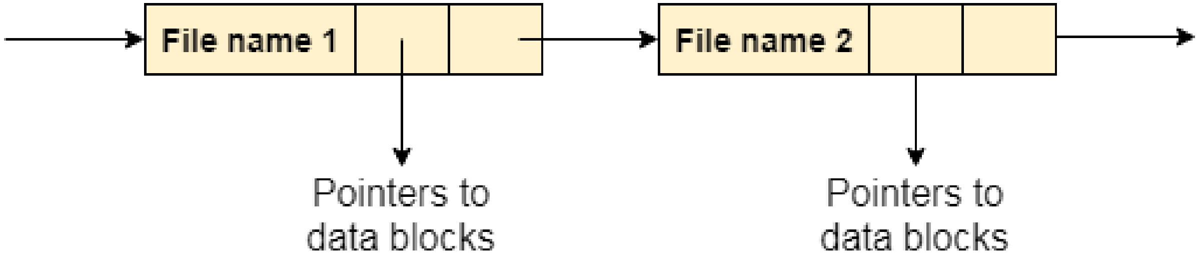
- There is the number of algorithms by using which, the directories can be implemented.
- The selection of an appropriate directory implementation algorithm may significantly affect the performance of the system.
- Directories need to be fast to search, insert, and delete, with a minimum of wasted disk space.
- Two methods
  1. Linear List

  2. Hash Table

Directory Implementation

1. Linear List

- all the files in a directory are maintained as singly lined list.
- Each file contains the pointers to the data blocks which are assigned to it and the next file in the directory.



Linear List

Directory Implementation

1. **Linear List**

- When a new file is created, then the entire list is checked whether the new file name is matching to an existing file name or not.

- In case, it doesn't exist, the file can be created at the beginning or at the end.

- Therefore, searching for a unique name is a big concern because traversing the whole list takes time

Directory Implementation

2. Hash Table

- This approach uses **hash table** along with the linked lists.
- A **key-value** pair for each file in the directory gets generated and stored in the hash table.
- The key can be determined by applying the **hash function** on the file name while the key points to the corresponding file stored in the directory.

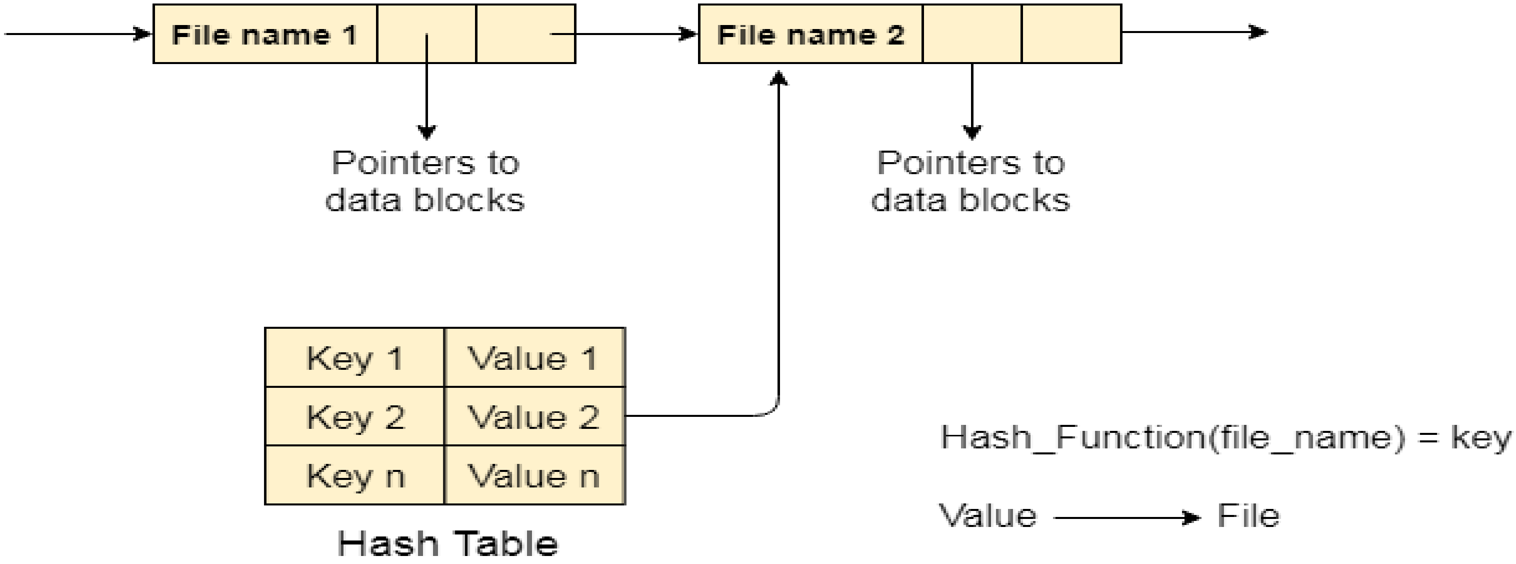
Directory Implementation

2. Hash Table

- This approach uses **hash table** along with the linked lists.
- A key-value pair for each file in the directory gets generated and stored in the hash table.
- The key can be determined by applying the hash function on the file name while the key points to the corresponding file stored in the directory.

Directory Implementation

2. Hash Table



File name 1

File name 2

Pointers to
data blocks

Pointers to
data blocks

Key 1	Value 1
Key 2	Value 2
Key n	Value n

Hash Table

Hash_Function(file_name) = key

Value → File

Directory Implementation

2. Hash Table

- Searching becomes efficient due to the fact that now, entire list will not be searched on every operating.
- Only hash table entries are checked using the key and if an entry found then the corresponding file will be fetched using the value.