A file is an abstract data type, a "thing", which is defined and implemented by the operating system. The main task of the operating system is to map the logical file concept onto physical storage devices, such as disks or tapes.

1. A file is a named collection of related information recorded on secondary storage
    a. Secondary storage is usually nonvolatile
2. Files have contiguous logical address spaces
3. File types:
    a. Data
        i. Numeric
        ii. Text
        iii. Binary
    b. Program
        i. Source
        ii. Binary object file
4. None - just a sequence of words, bytes
5. Simple record structure
    a. Lines of fixed length or variable length
6. Complex structures for specific use
    a. Formatted document
    b. Relocatable load file
7. Can simulate last two with first method by inserting appropriate control characters and layout requirements
8. Who decides:
    a. Operating system
    b. Program (application developer)

A **file** ...

- Has a **type** (.com,.bat,exe,...) which can either be known and managed by the operating system, or leaving the interpretation to the application process.
    - Some systems allow different file operations based on type. The file may have attributes (name, creator, date, type, permissions)

For general purpose systems it is more effective to implement only basic types and grant maximum freedom to the processes.
For specialized systems, i.e. a database system it "may" be more efficient to implement the logic in the operating system.

A file can have various kinds of structure

- None - sequence of words, bytes
- Simple record structure
  - Lines
  - Fixed length
  - Variable length
- Complex Structures
  - Formatted document
  - Relocatable load file
- Who interprets this structure?
  - Operating system
  - Program

Attributes of a File

- **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 is kept in the directory structure, which is maintained on the disk.

Basic file **operations** ...

- **create** .. find space for the file and make an entry in the directory.
- **open** ... find file and determine if it has already been opened. If not open search directory, cache information, add entry in per-process open-file table. If open check lock and cache information if lock can be acquired. Increment the open count.
- **close** ... decrement the open count and remove the file's entry from the open-file table if count reaches zero.
- **read** ... read data from the file.
- **write** ... write data to the file.
- **delete** ... search directory, release file space and erase directory entry.
- **reposition** ... reposition the file position pointer. This is more commonly known as seek.
- **truncate** ... delete content of a file, but keep file properties.
- **lock** ... file locks provide concurrency control. A shared lock allows multiple "readers" to acquire a lock concurrently, while exclusive lock ensures only one "writer" can modify a file.
  With **mandatory locking** the operating system ensures locking integrity, while with **advisory locking** the application process ensures that the correct locking strategy is followed. The Windows operating system uses the mandatory locking strategy.

Open Files

- Several pieces of data are needed to manage open files:
  - **File pointer**: pointer to last read/write location, per process that has the file open

- **File-open count**: counter of number of times a file is open – to allow removal of data from open-file table when last processes closes it
- **Disk location of the file**: cache of data access information
- **Access rights**: per-process access mode information

Open File Locking

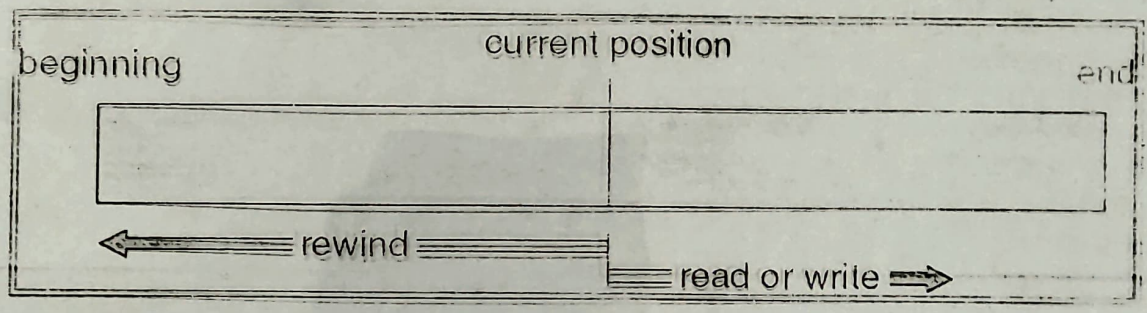
- Provided by some operating systems and file systems
- Mediates access to a file
- Mandatory or advisory locking:
- **Mandatory** – access is denied depending on locks held and requested
- **Advisory** – processes can find status of locks and decide what to do

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

- **Sequential Access**
  - read next
  - write next
  - reset (rewind to begin)
  - no read after last write
- **Direct Access** (or **relative access**)
  - read *n*
  - write *n*
  - position to *n*
    - read next
    - write next
  - rewrite *n*
- *n* ≡ **relative block number**

Sequential-access File

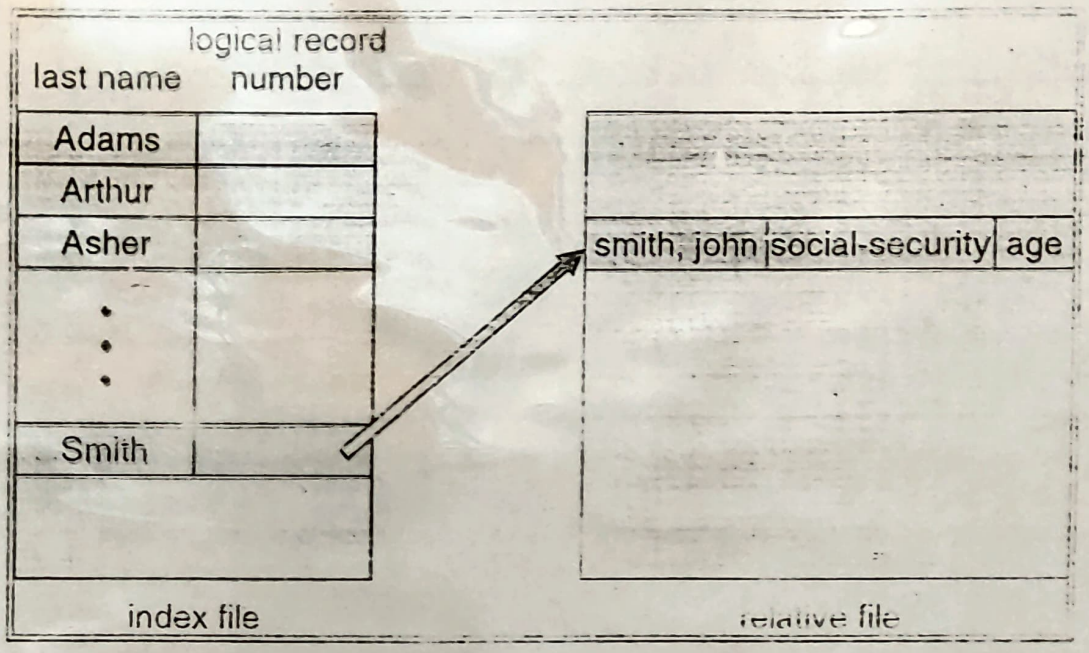


Simulation of Sequential Access on a Direct-access File

sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

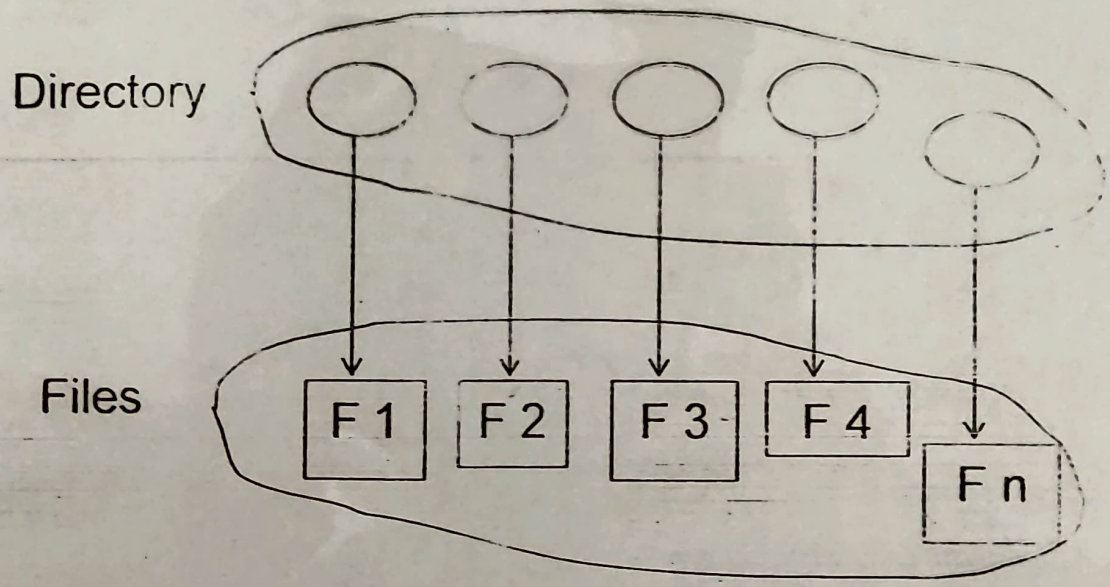
cp ≡ current position

Index and Relative Files



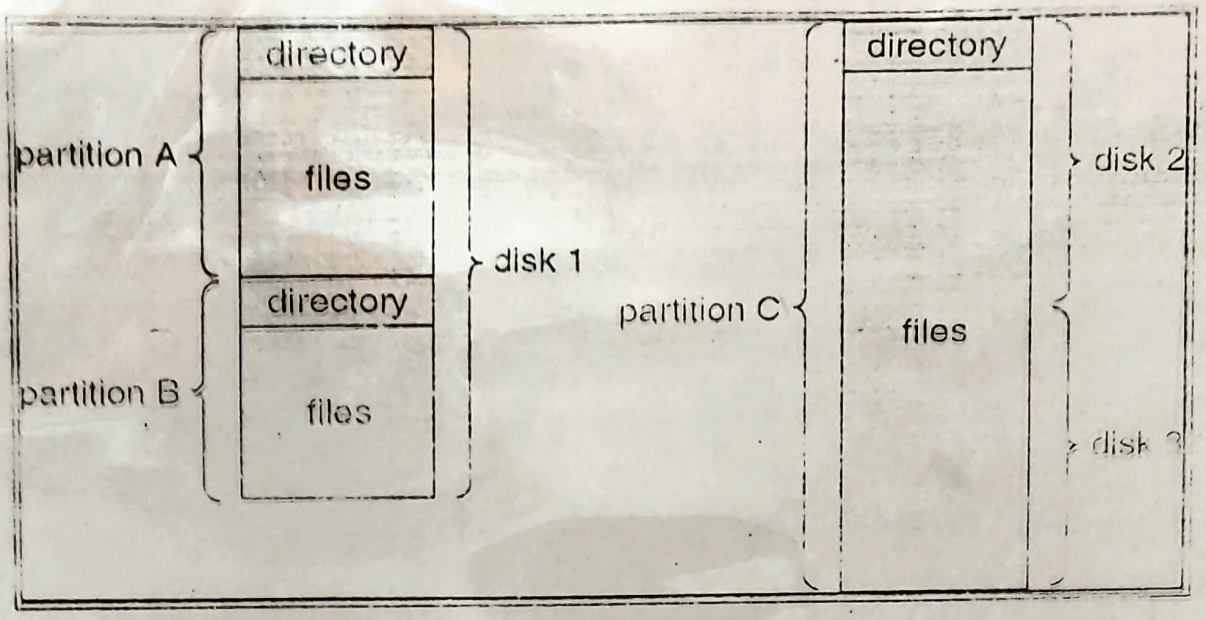
Directory Structure

- 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

A Typical File-system Organization



Operations Performed on Directory

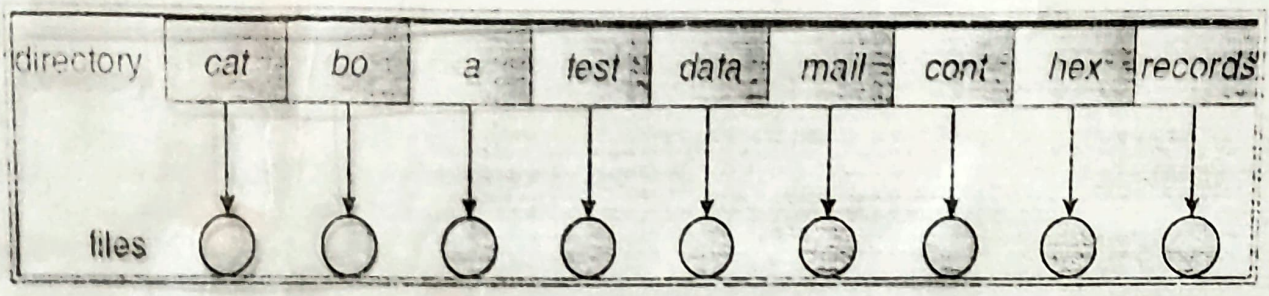
- **Search for a file** - find files whose names match a pattern
- **Create a file**
- **Delete a file**
- **List a directory** - list the names of the files in a directory with their attributes
- **Rename a file** - or move within directory structure
- **Traverse the file system** - need support for walking directory trees while other processes are changing contents

Directory Organization Design Issues

- Allow efficient searching – locating a file quickly
- Enable naming – convenient to users
  - Two users can have same name for different files
  - The same file can have several different names

- Grouping – logical grouping of files by properties, (e.g., all Java programs, all games, ...)

Single-Level Directory

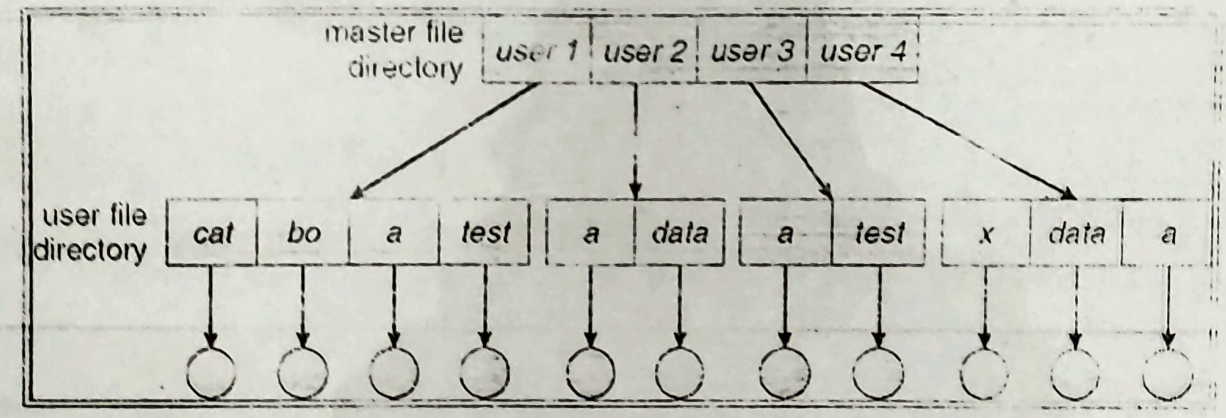


Naming problem

Grouping problem

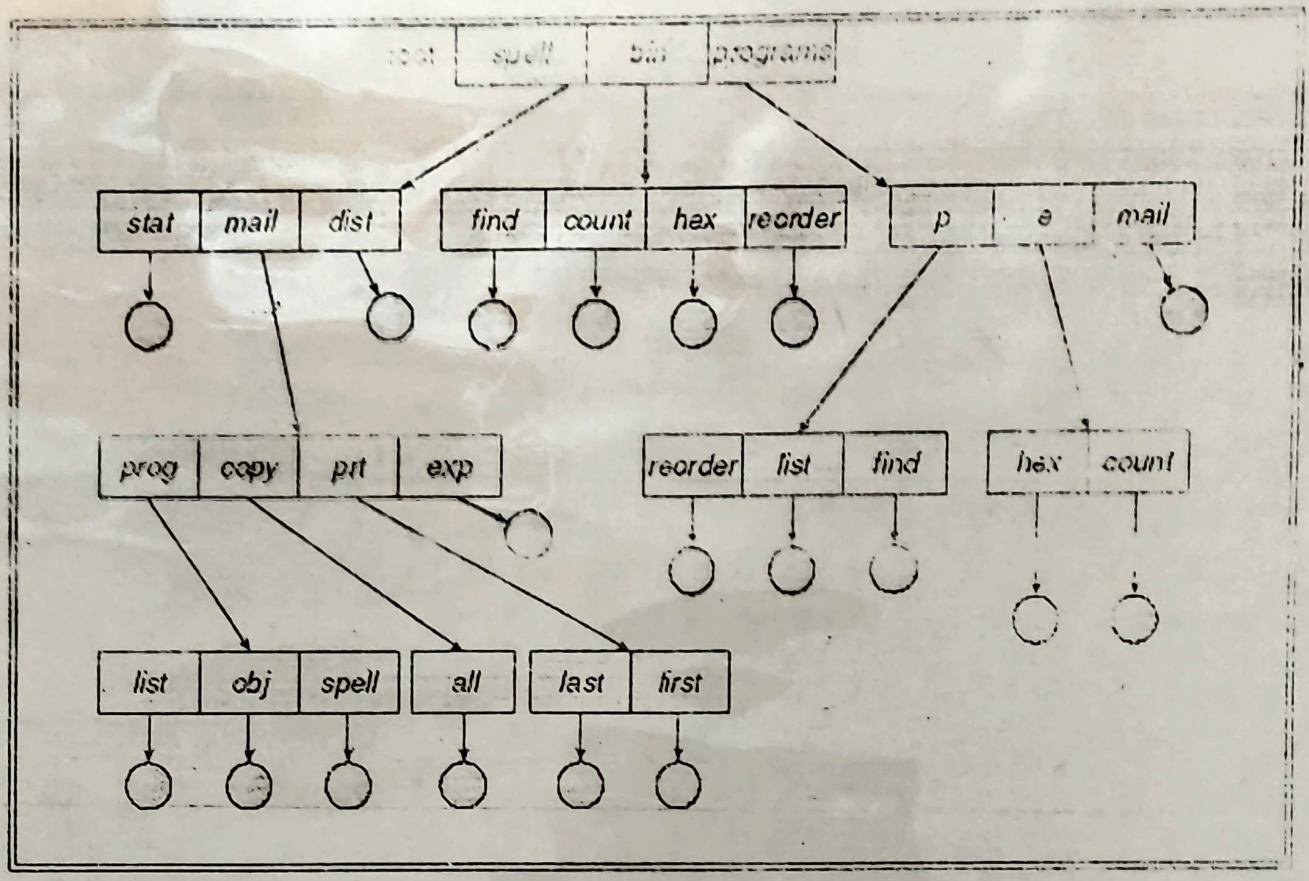
Two-Level Directory

- Separate directory for each user



- Path name
- Can have the same file name for different users
- Efficient searching
- No grouping capability

Tree-Structured Directories



- Efficient searching
- Grouping Capability
- Current directory (working directory)
  - cd /spell/mail/prog
  - type list
- **Absolute** or **relative** path name
- Creating a new file is done in current directory
- Delete a file

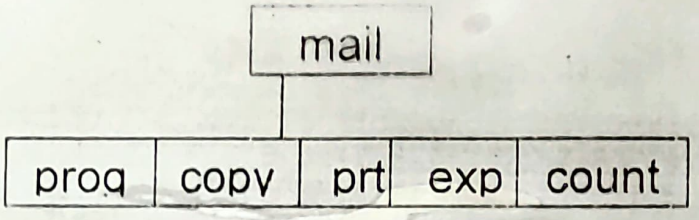
  rm <file-name>

- Creating a new subdirectory is done in current directory

  mkdir <dir-name>

- Example: if in current directory /mail

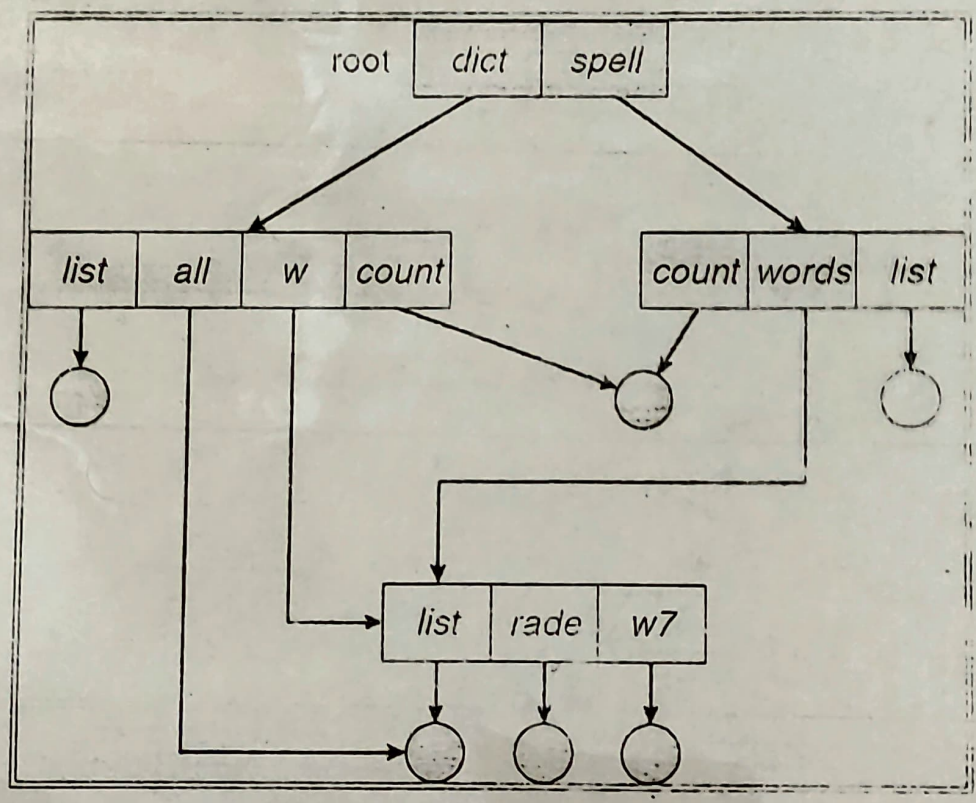
  mkdir count



Deleting "mail" ⇒ deleting the entire subtree rooted by "mail"

Acyclic-Graph Directories

- Allow shared subdirectories and files

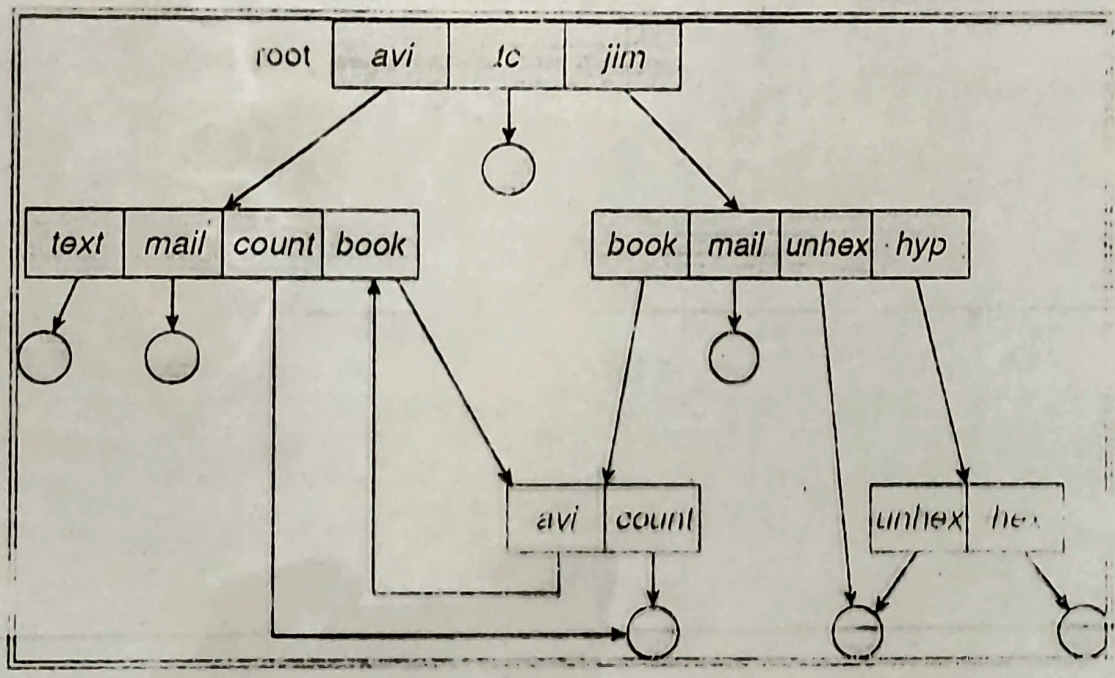


- More than one name for different files and directories (aliasing)
- New directory entry type
- **Link** – another name (pointer) to an existing file
- **Resolve the link** – follow pointer to locate the file

- Problem: if *dict* deletes *list* ⇒ dangling pointer

- Solutions:
- Backpointers, so we can delete all pointers, but variable size records is a problem
- Entry-hold-count solution
- Garbage collection - marks everything that can be accessed

General Graph Directory



- Problem
  - Cycles in directory structure may cause non-termination of depth-first searches and other traversals
  - Cycles are a problem for garbage collection:
    - Files in cycles may never get deleted
- How do we guarantee no cycles?
  - Allow only links to file not subdirectories
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK
    - Expensive