File-System Interface

Chapter 11

Adapted from the slides given by Silberschatz, Galvin, and Gagne
In "Operating System Concepts", 10th edition John Wiley and Sons, Inc 2018.

Dr.Siwar Rekik

Outline of this lecture

In this lecture, we will discuss the following:

- File Concept
- Access Methods
- Directory Structure
- Protection

File-System (FS) Structure

- File system consists of:
  - A collection of files – storing related data
  - Directory structure- provides information about all the files in the system.

- File system resides on secondary storage (disks).

File Concept

- A collection of data- normally is stored on a secondary storage device.
- File types refer to classifying the content of the file, such as:

- Types:
  - Data
    - numeric
    - character
    - binary
  - Program

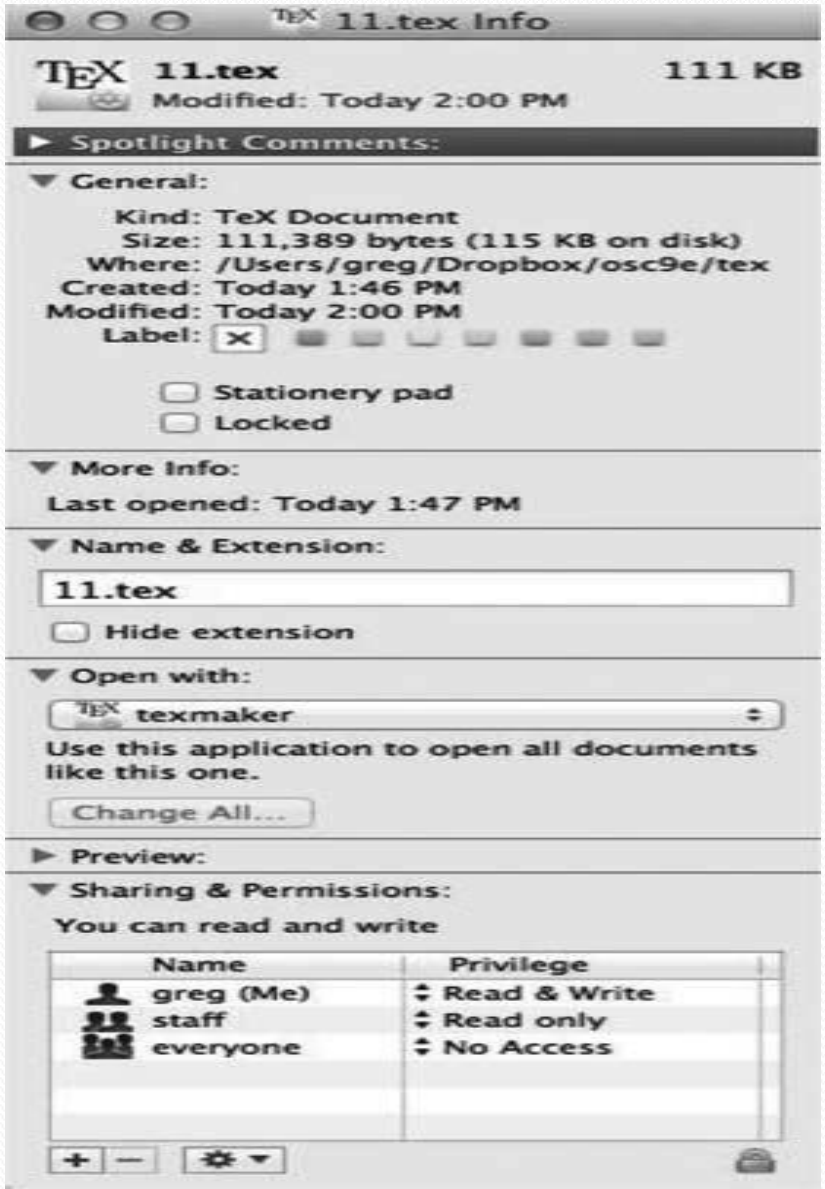
- Contents defined by file's creator
  - Many types
    - Consider **text file, source file, executable file**

File Attributes

A file has several attribute- vary from one O/S to another but consist of these:

- **Name** – only information kept in human-readable form.
- **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** – this information can be useful for protection, security, and usage monitoring.
- Information about files are kept in the directory structure, which is maintained on the disk.

File info Window on Mac OS X



File Operations

An OS must provide a number of operations associated with files so that users can safely store and retrieve data. Typical 6 basic operations are:

- *Create*- find space in the file system & entry for new file must be made in dir.
- *Write*- make system call specifying both the name of the file & inf to be written to it.
- *Read*- use system call specifying the file's name & where next block should be put.
- *Seek* -repositioning within file (searching the directory for the appropriate entry).
- *Delete*- search the directory for the named file, erase the directory entry.
- *Truncate*- erase the contents of file but keep its attributes.
- *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.

Open Files

- Most of the file operations involve **searching** the directory for the entry associated with the named file. To avoid searching, the OS keeps a table

- Several pieces of data are needed to **manage** open files:
  - **Open-file table**: contains info abt all open files
  - **File pointer:** pointer must track the last read/write location as a current file position pointer, per process that has the file open
  - **File-open count**: counter of number of times a file is open by many processes, when the open count reaches o, the file is no longer in use, and the file's entry is removed from the open file table
  - **Disk location of the file:** most file operations require the system to modify data within the file. The info needed to locate the file on a disk is kept in memory
  - **Access rights:** each process opens a file in an access mode. This info is stored on the per-process table so the OS can allow/deny I/O requests.

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 com-pressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

File Structure

- Files must conforms to a required structure understood by the OS. i.e. OS requires that the exe file have a specific structure, so that it can determine where in memory to load the file & what the location of the 1st instruction.
- None - sequence of words, bytes
- Simple record structure
  - Lines
  - Fixed length
  - Variable length
- Complex Structures
  - Formatted document
  - Relocatable load file
- Can simulate last two with first method by inserting appropriate control characters
- Who decides:
  - Operating system
  - Program

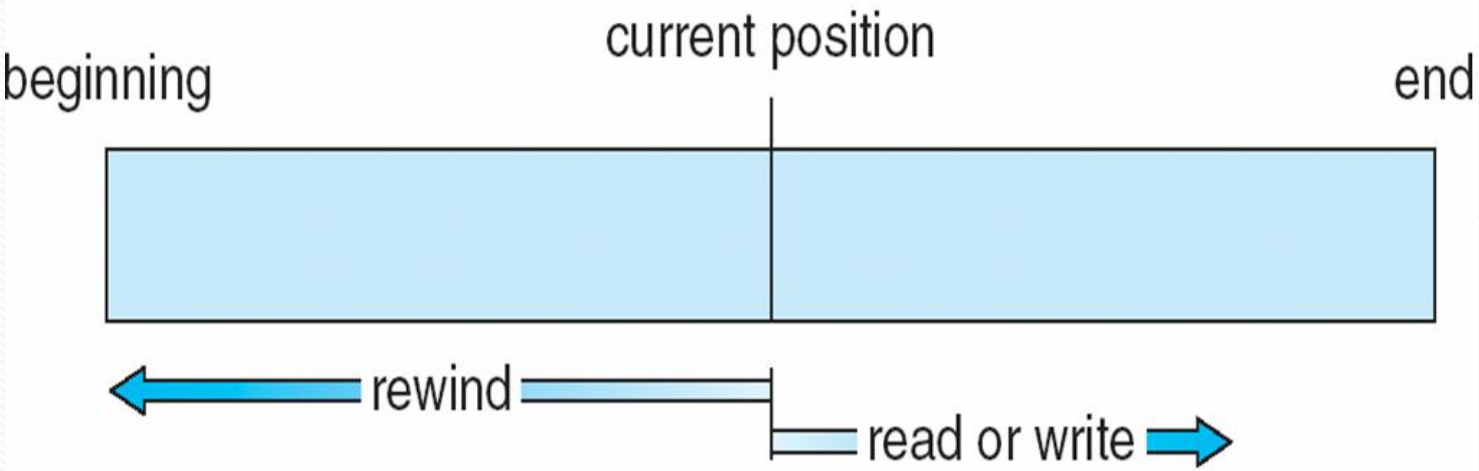
Access Methods

Information stored in files must be accessed & read into memory.
So how can this information be accessed?

- **Sequential Access**
  - information in the file is processed in order- 1 record after another
  - Most common, Editors & compilers access files in this fashion.

- **Direct Access / Relative Access**
  - File is made up of fixed-length logical records, allow programs to read/write records rapidly in no particular order.
  - file is viewed as a numbered sequence of records.
  - Records are organized in a manner that allows direct access to a particular record without having to read any of the preceding records.
  - Allows blocks to be read or written in any order- no restriction on the order of reading/writing for a direct-access file.
  - Useful for immediate access to large amounts of info (DB).

Sequential-access File

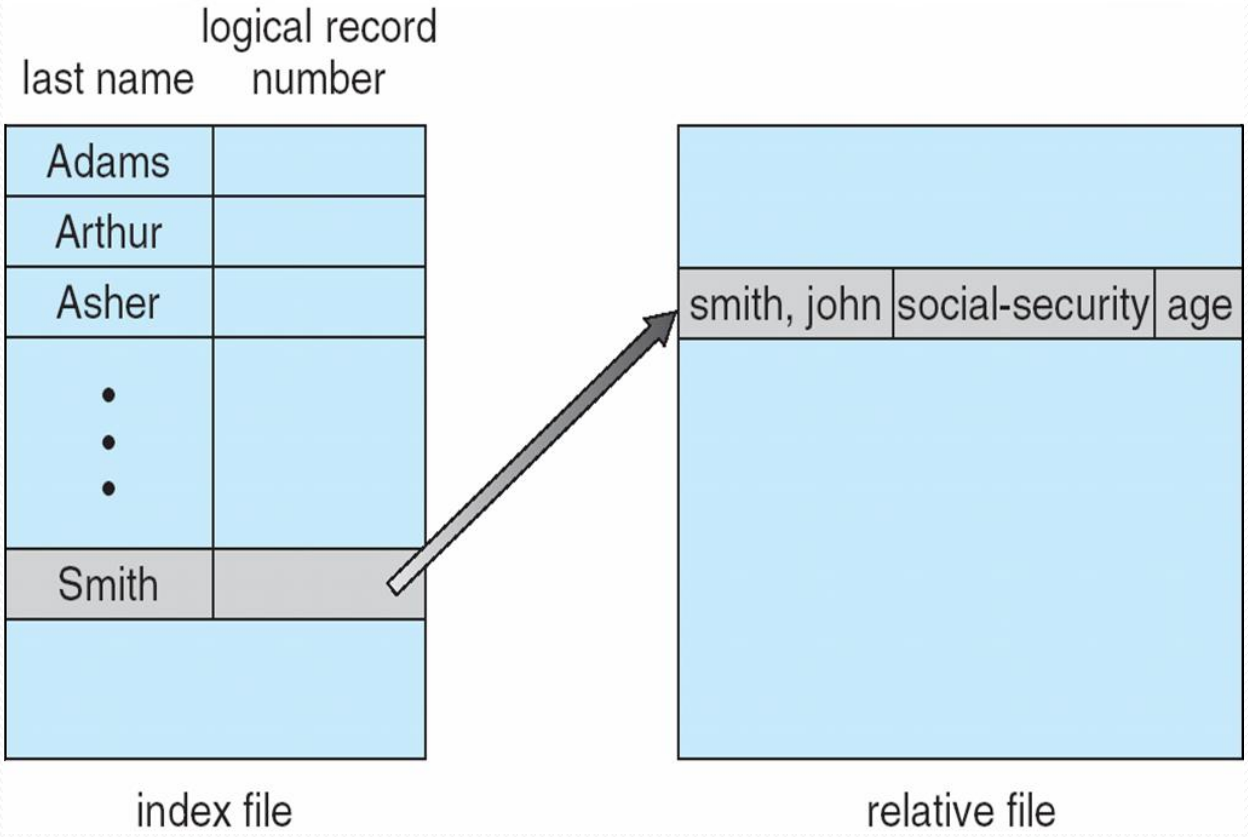


Other Access Methods

- Can be built on top of direct Access
- General involve creation of an index for the file
- ***Index file-*** stores an index into another file
- With large files, the index file become too large --------→ solution
- ***Primary, secondary index file-*** primary index file contains pointers to the secondary one, which points to the actual data items.

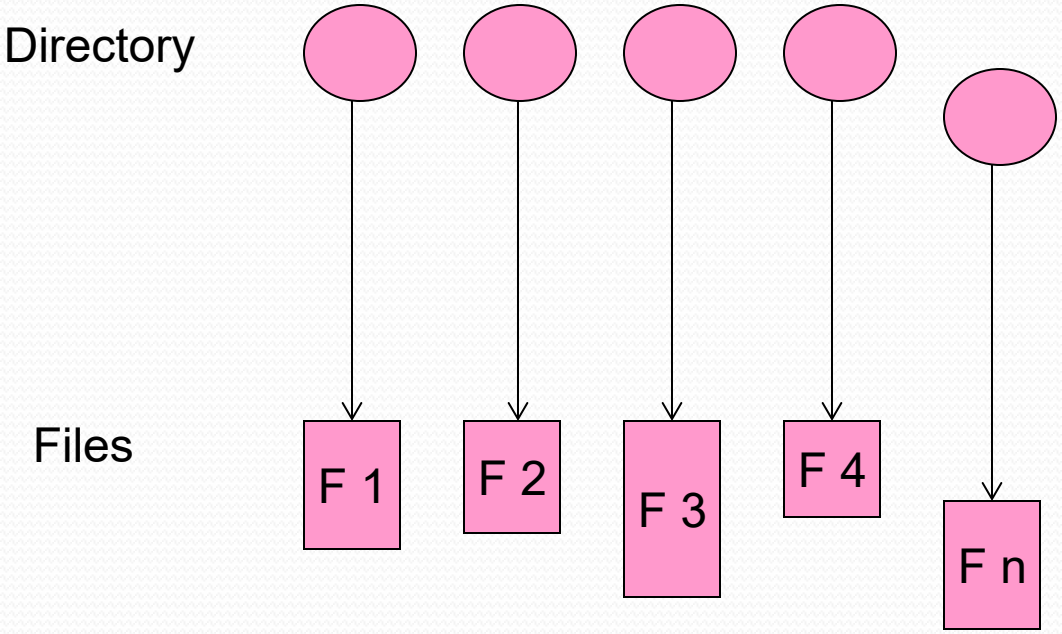
- IBM indexed sequential-access method (ISAM)
  - Small master index, points to disk blocks of secondary index
  - File kept sorted on a defined key
  - All done by the OS
- VMS operating system provides index and relative files as another example (see next slide)

Example of Index and Relative Files



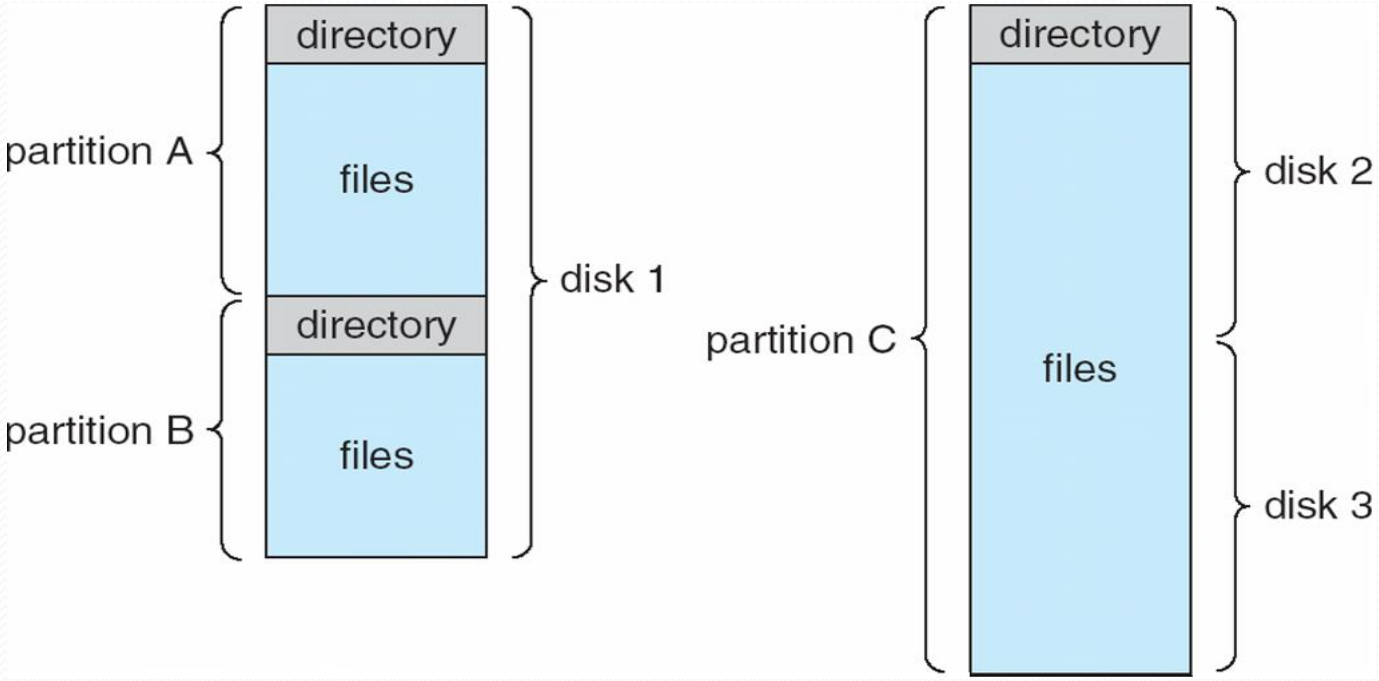
Directory Structure

- A collection of nodes containing information about all files.



- Both the directory structure and the files reside on disk.

A Typical File-system Organization



- Disks are split into 1 or more partitions, each treated as a separate storage device.
- Each partition contains info about files within it.
- Directory records info. i.e. name, location, size & type for all file on the partition.

Operations Performed on Directory

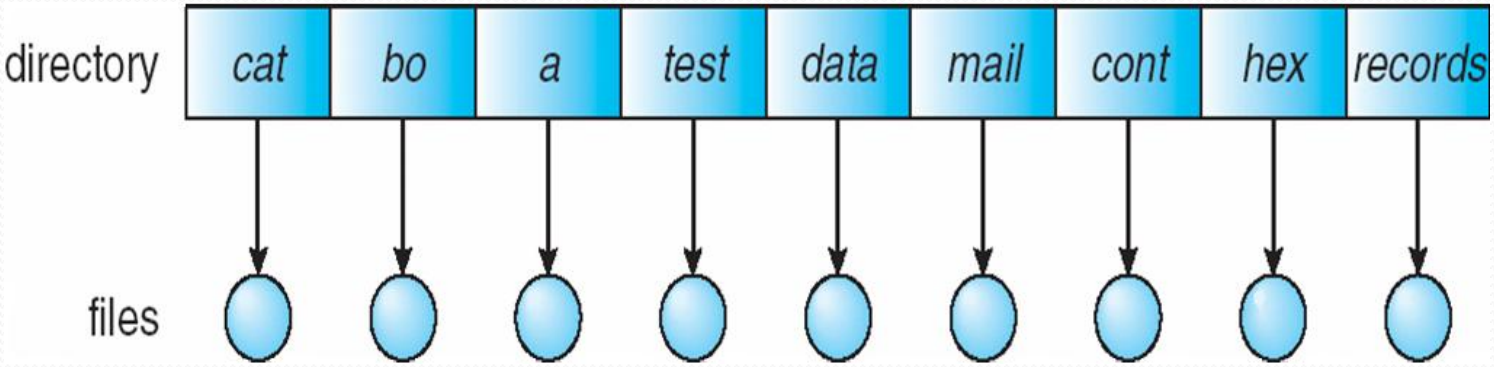
- *Search for a file*- be able to search a directory structure to find particular file
- *Create a file*- new files need to be created & added to the directory.
- *Delete a file*- be able to remove a file from the directory.
- *List a directory*- be able to list the files in a directory.
- *Rename a file*- be able to change the name of a file when the contents or use of the file changes.
- *Traverse the file system*- be able to access every directory & every file within a directory structure.

Directory Organization

- Efficiency – locating a file quickly.
- 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

- All files are in the same directory- easy to support & understand.

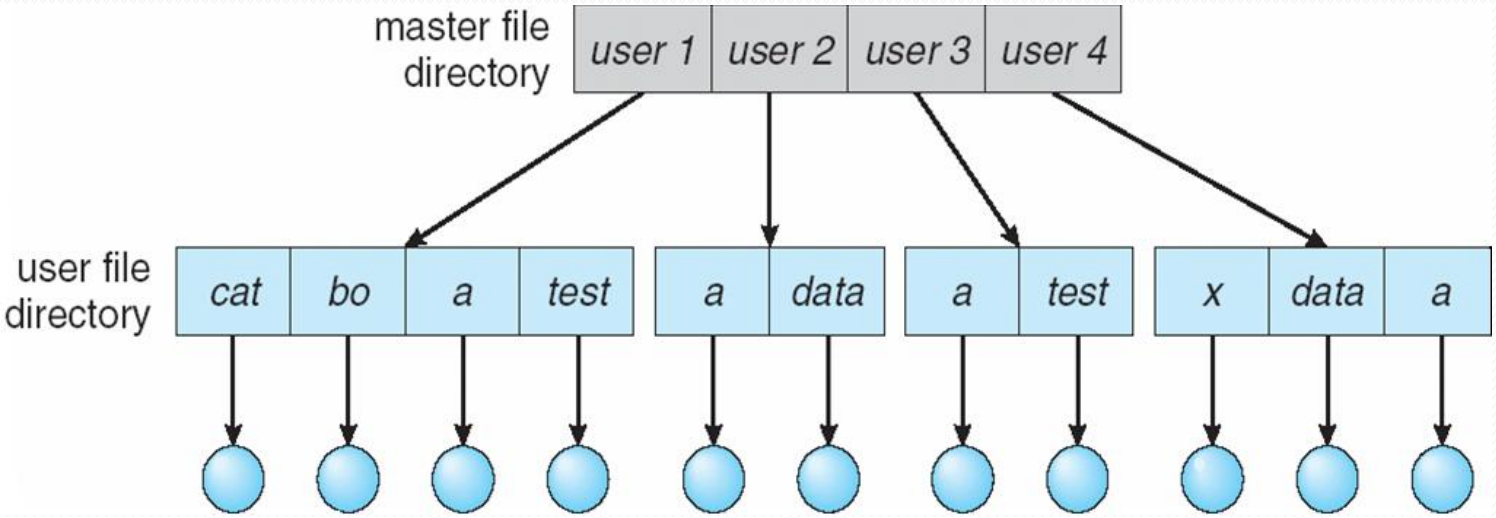


Limitations:

- *Naming problem*- all files need to have a unique names, as they are in the same directory.
- *Grouping problem*-

Two-Level Directory

- Separate directory for each user.
- Each user has her own *user file directory* (UFD).
- *Master file directory* (MFD) is searched when a user logs in.



- Can have the same file name for different user
- Isolates one user from another- problem when users want to cooperate on some tasks & to access one another's files.
- Efficient searching
- No grouping capability

Tree-Structured Directories

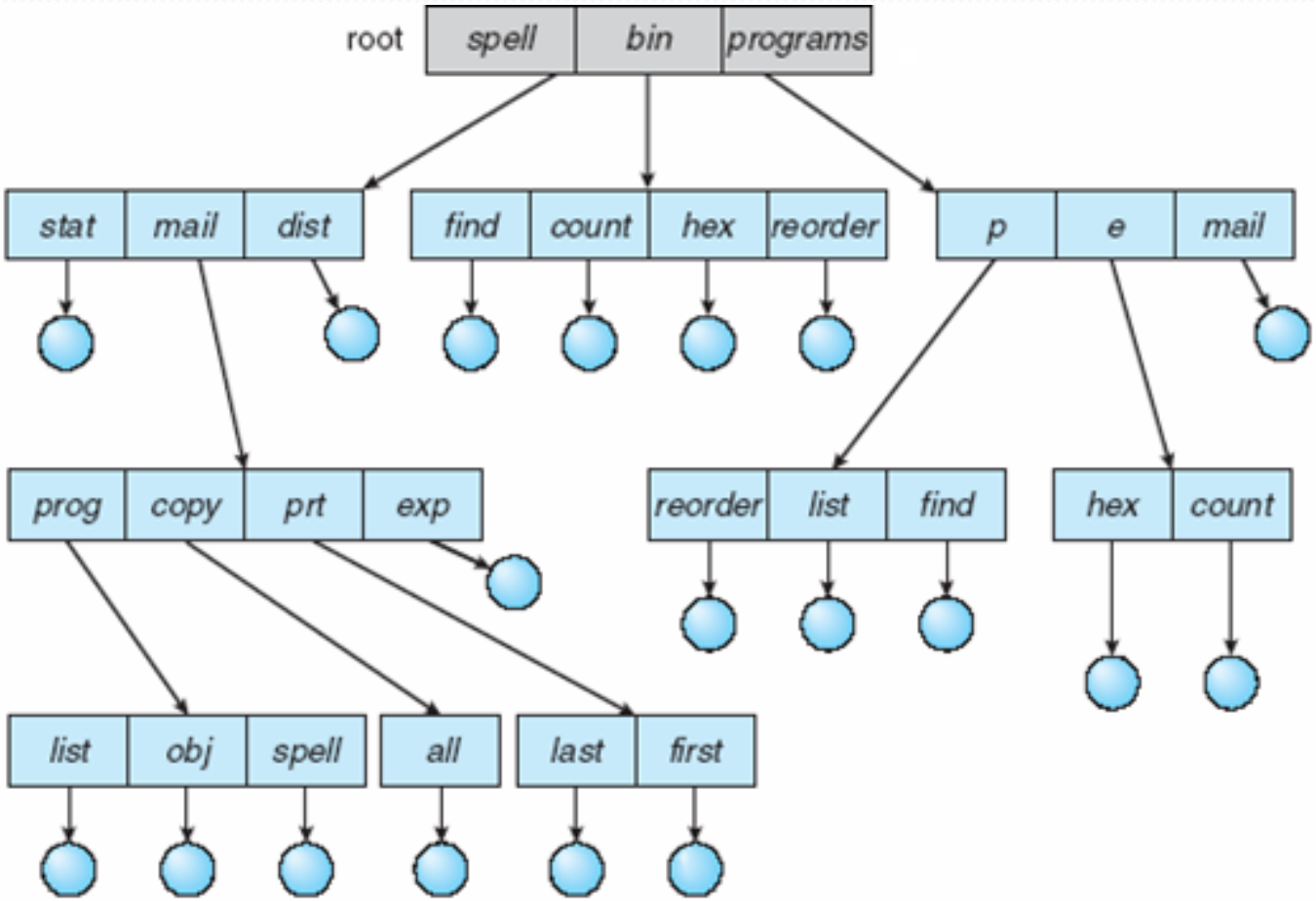
- A tree has a *root directory*, & every file in the system has a unique *path name*.
- *path name*- root ---→subdirectories ----→specified file.
- Efficient searching
- Grouping Capability: Each directory contains a set of files or Subdirectories.
- ***Current directory***- contains most the files that are of current interest to the user.
- ***Absolute path name:*** begins at the root --→ a path down--→ specified file.
- ***Relative path name:*** defines path from the current directory.
- Deletion of a directory
  - If empty- can be deleted.
  - If not empty- some systems, will not delete a directory. e.g: MS-DOS.

    - other systems, remove the entire dir structure. e.g. UNIX

      rm <file-name>

      - But, if the command is issued in error-→ large no of files & directories will need to be restored.

Tree-Structured Directories (Example)

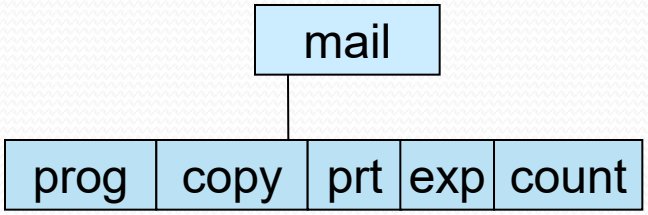


Tree-Structured Directories (Cont.)

- Creating a new file is done in current directory.
- Creating a new subdirectory is done in current directory.

  Ex: if in current directory /spell/mail

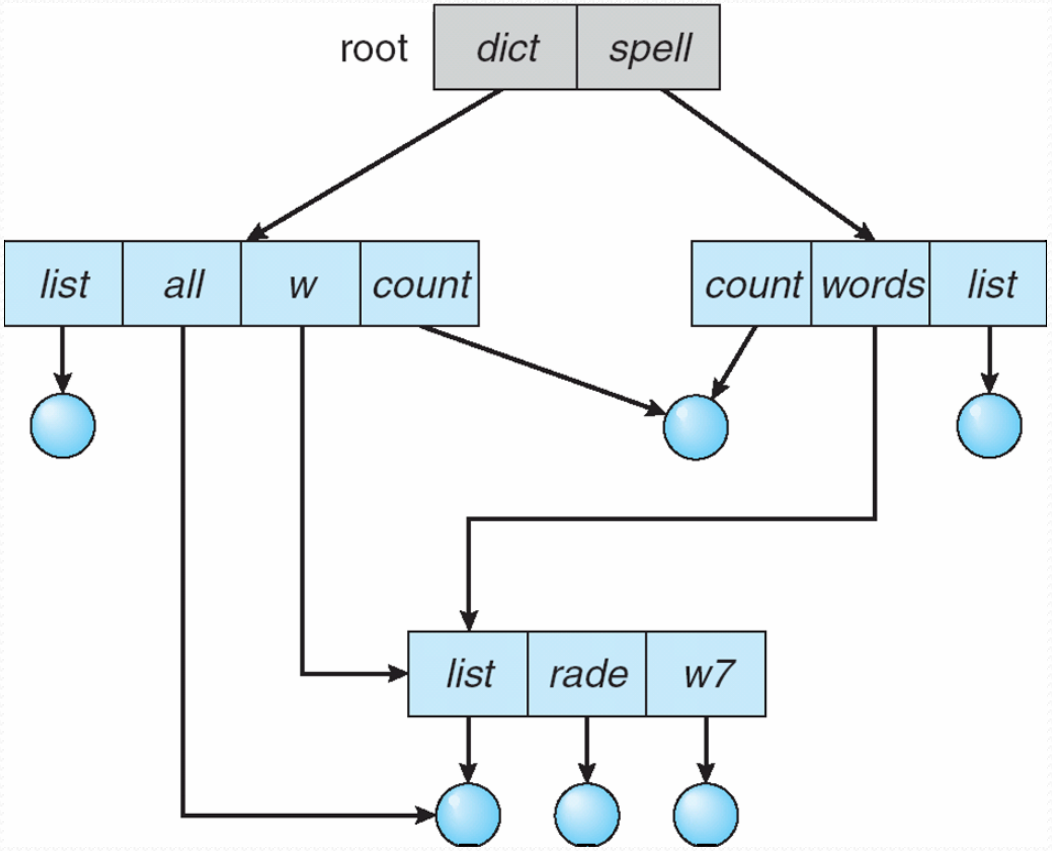
  mkdir count



- Deleting "mail" $\Rightarrow$ deleting the entire subtree rooted by "mail"

Acyclic-Graph Directories

- Allows directories to share subdirectories and files- any changes made by one user are immediately available to the other.



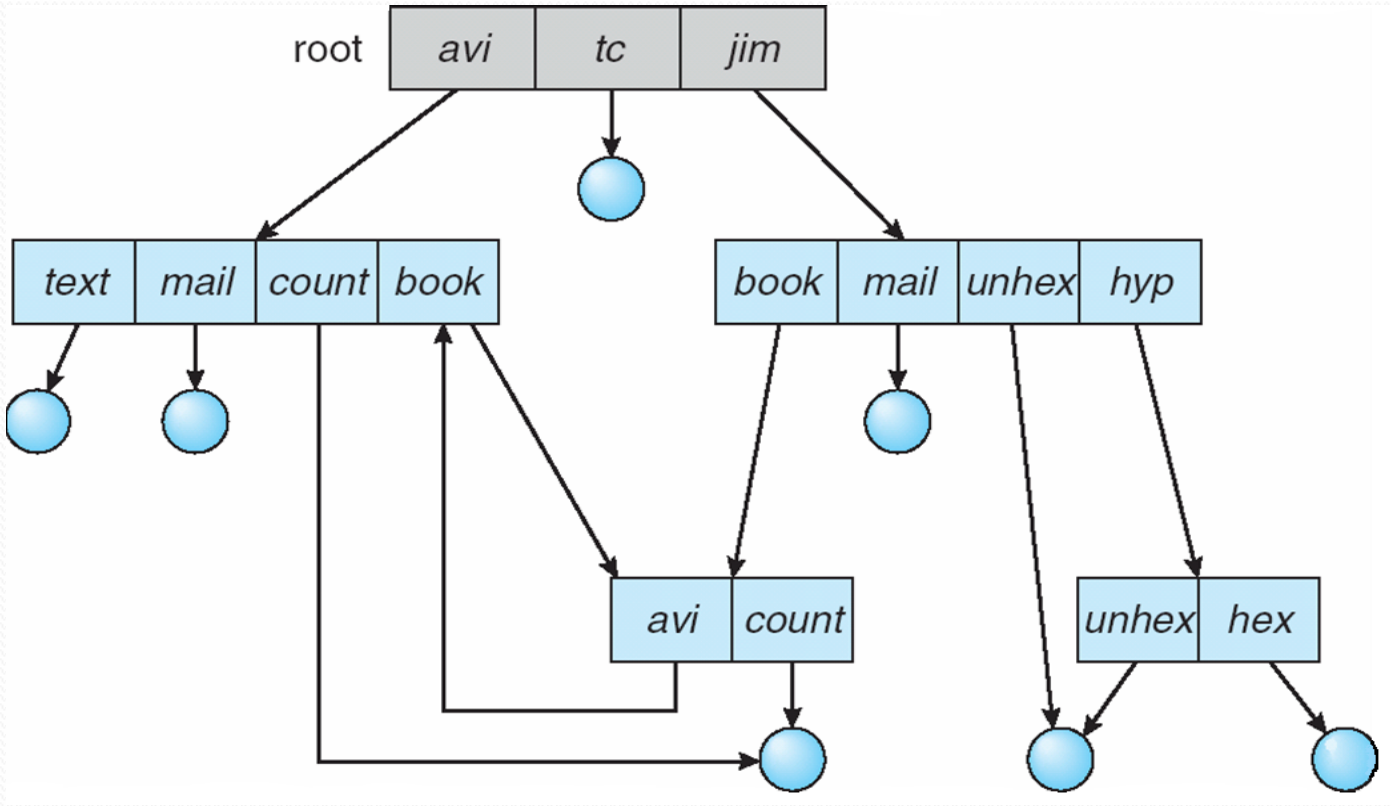
Acyclic-Graph Directories (Cont.)

- There are several ways to implement shared files and directories:
  - ***Link-*** pointer to another file or subdirectory (maybe implemented as relative/absolute path name).
  - **Resolve the link** – using path name to locate the real file
  - ***Duplicate File Information*** - Duplicating all information about the shared files in both sharing directories.

- **Aliasing problem** - A file can have multiple file absolute path names

- **Deletion Problem**: dangling pointers- remove the file whenever anyone deletes it
  - **Solution**
    - Delete only the link.
    - Preserve file until all references to it are deleted- when the file-reference list is empty, the file is deleted.

General Graph Directory



- If we start with a two-level dir & allow users to create subdirectories, what would result?
- By adding links to an existing tree structure dir, the tree structure is destroyed--→simple graph structure.

General Graph Directory (Cont.)

- When a file can be deleted?
- A value of 0 in the reference count→ no reference → delete in acyclic graph
- When cycles exist, reference count may not be 0 even when it is no longer possible to reference to a file/directory, due to self-referencing/ cycle in the directory
  - Allow only links to file not subdirectories.
  - Garbage collection to determine when the last reference has been deleted & the disk space can be reallocated.
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK. Will the new link complete a cycle or not?

Protection

- Protection is needed so that users do not accidentally/deliberately destroy someone else's information.
- Information is private & should not be read by other users.
- OS must provide means to protect the info in the file system.

- Many types of operations could be controlled:
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

Access Lists and Groups

- Different users may need different types of access to a file or directory: read, write, execute
- Access-control list (ACL) to be associated with each file & directory specifying: username & type of access allowed for each user.
  - **problems:**
    - *Length*- must list all users with read access to allow read a file.
    - *List construction* could be a boring task- list of users in the system not known in advance.
    - *Variable size directory entry*- complicated space management.

- A more general scheme with three classes of users
  Example: UNIX

		RWX
a) owner access	⇒	1 1 1
b) groups access	⇒	RWX 1 1 0
c) universe access	⇒	RWX 0 0 1

- Other protection approaches: Associate a password with each file or dir.

Windows 7 Access-Control List Management