

Chapter 10: File-System Interface

► Objectives:

- To explain the function of file systems
- To describe the interfaces to file systems
- To discuss file-system design tradeoffs, including access methods, file sharing, file locking, and directory structures
- To explore file-system protection



File Concept

- ▶ Contiguous persistent logical address space, can be storing data or programs
- ▶ File Structure:
 - None - sequence of words, bytes
 - Simple record structure: Lines, fixed length, variable length
 - Complex Structures: formatted document, relocatable object
- ▶ Can simulate last two with first method by inserting appropriate control characters
- ▶ Who decides:
 - Operating system
 - Program



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



File Attributes

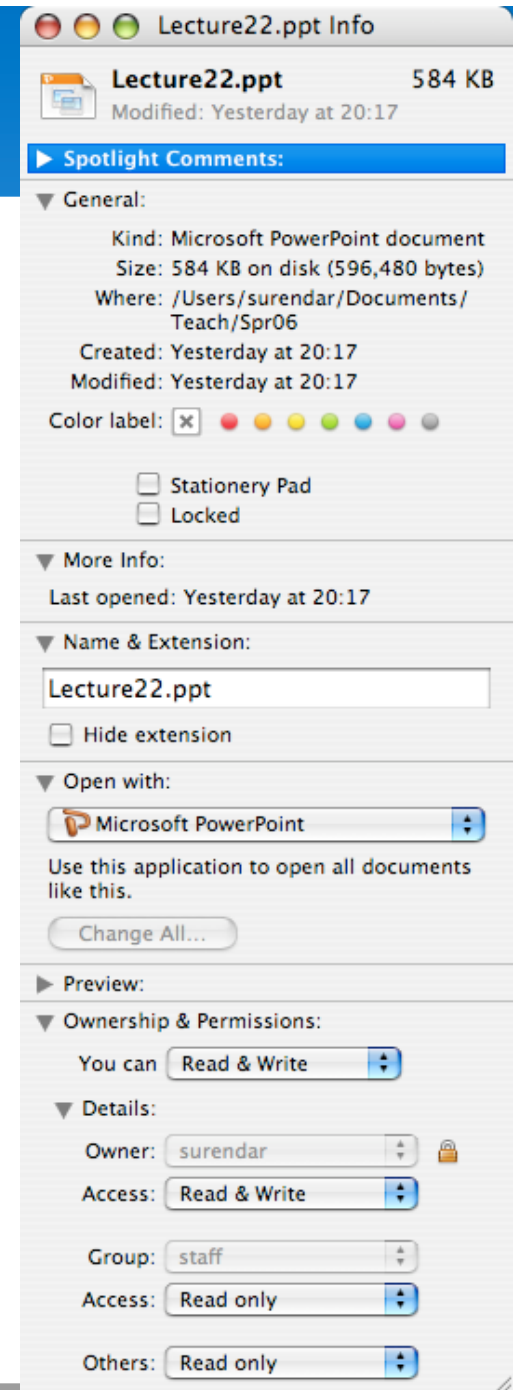
- ▶ **Name** – only information kept in human-readable form
- ▶ **Identifier** – unique tag (number) identifies file within file system
- ▶ **Type** – needed for systems that support different types
- ▶ **Location** – pointer to file location on device
- ▶ **Size** – current file size
- ▶ **Protection** – 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



Examples

► UNIX: ls -li

```
26047823 -rw-r--r--  1 surendar staff
596480 Mar 16 20:17 Lecture22.ppt
```



File Operations

- ▶ File is an **abstract data type**
- ▶ **File operations:**
 - Create
 - Write
 - Read
 - Reposition within file (seek)
 - 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



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:
 - **Mandatory** – access is denied depending on locks held and requested
 - **Advisory** – processes can find status of locks and decide what to do



Access Methods

► Sequential Access

read next
write next
reset
no read after last write
(rewrite)

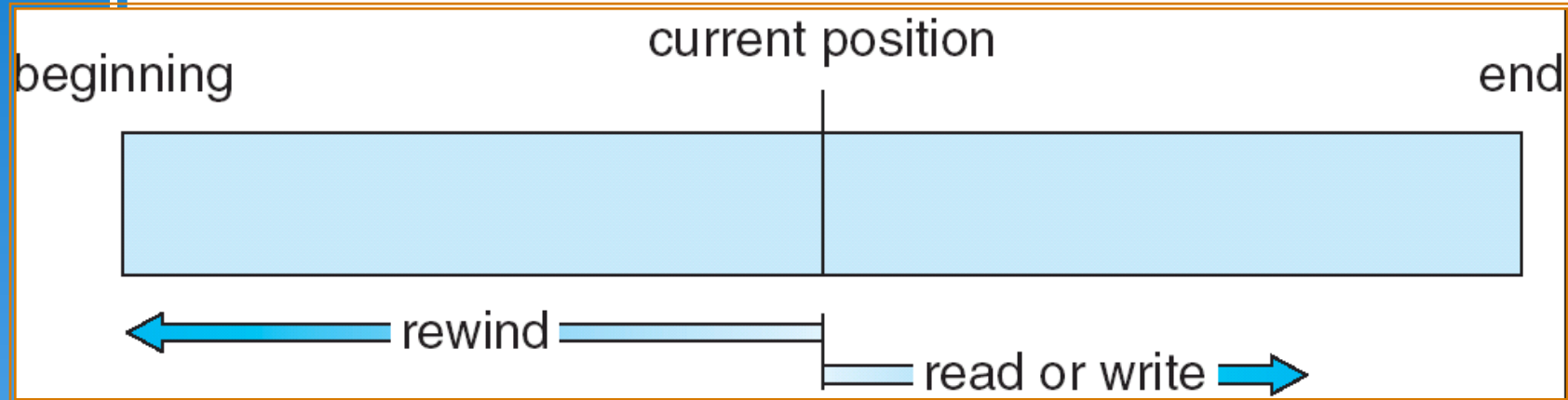
► Direct Access

read n
write n
position to n
 read next
 write next
rewrite n

n = relative block number



Sequential-access File



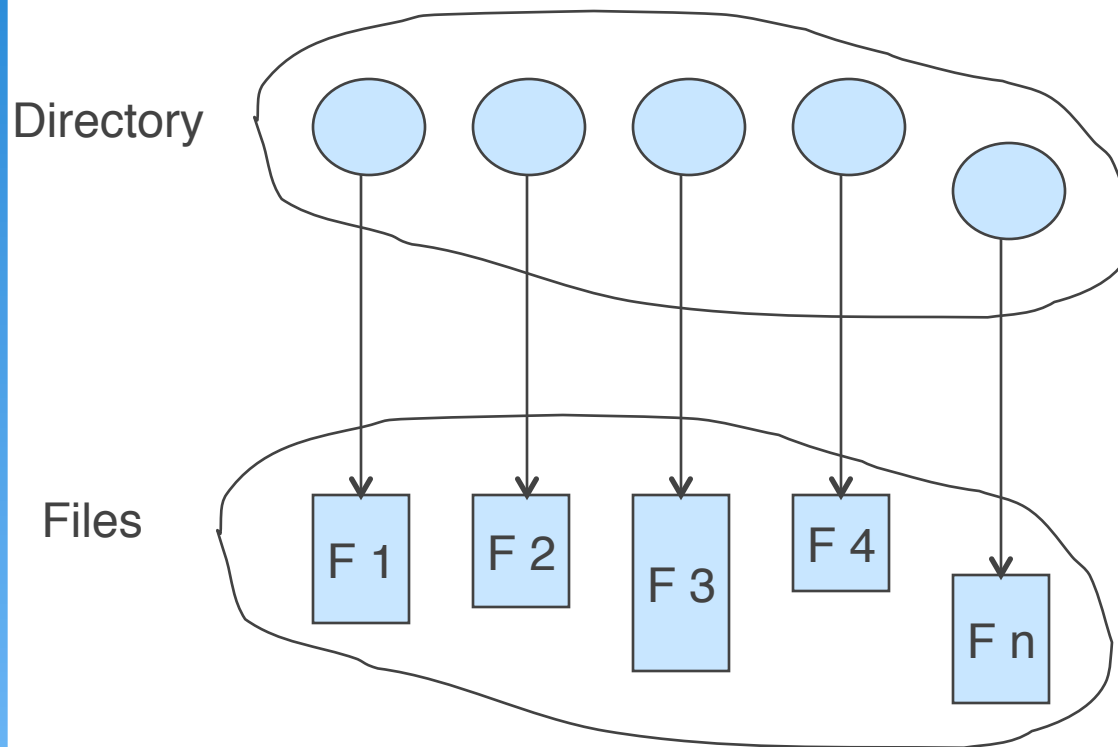
Simulation of Sequential Access on a Direct-access File

sequential access	implementation for direct access
<i>reset</i>	<i>cp</i> = 0;
<i>read next</i>	<i>read cp</i> ; <i>cp</i> = <i>cp</i> + 1;
<i>write next</i>	<i>write cp</i> ; <i>cp</i> = <i>cp</i> + 1;



Directory Structure

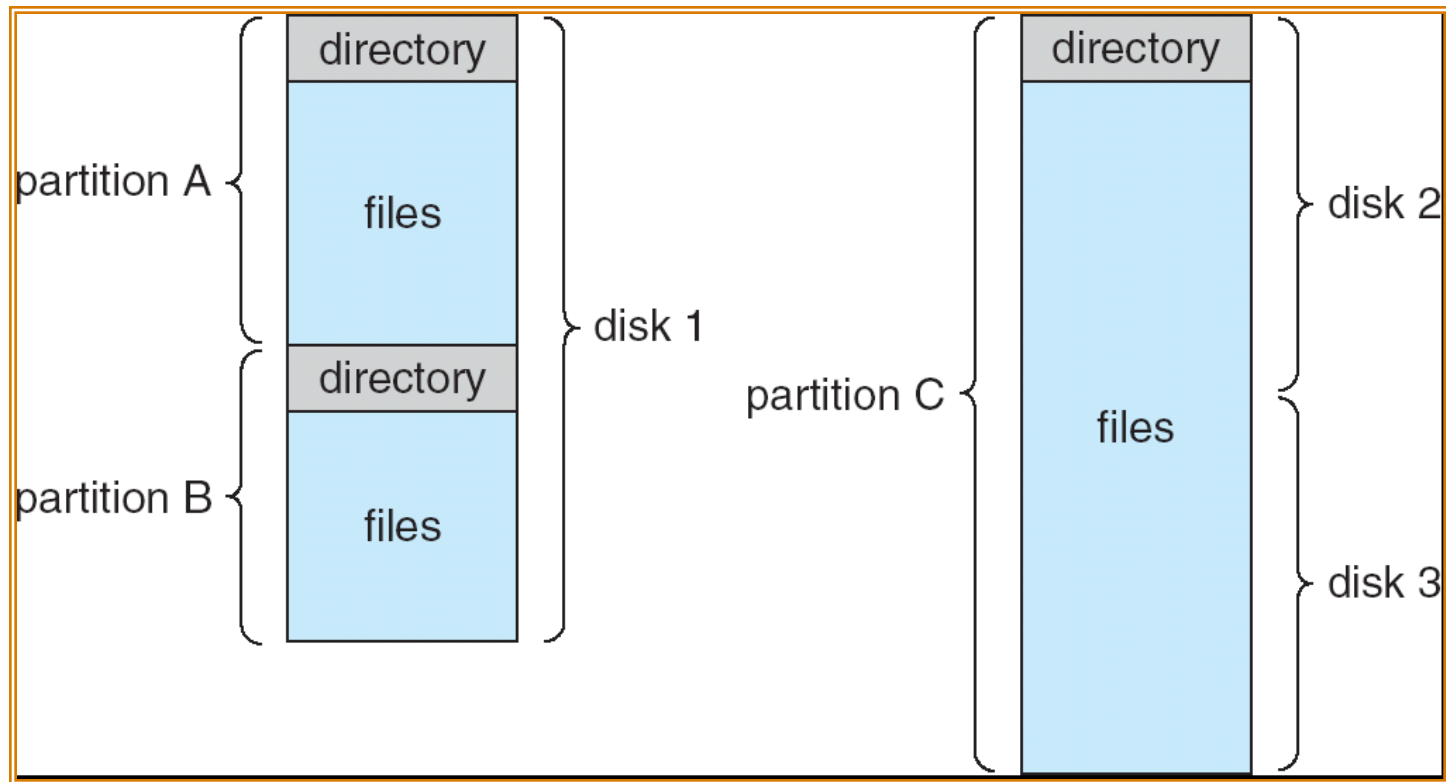
- ▶ A collection of nodes containing information about all files



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
- ▶ Create a file
- ▶ Delete a file
- ▶ List a directory
- ▶ Rename a file
- ▶ Traverse the file system



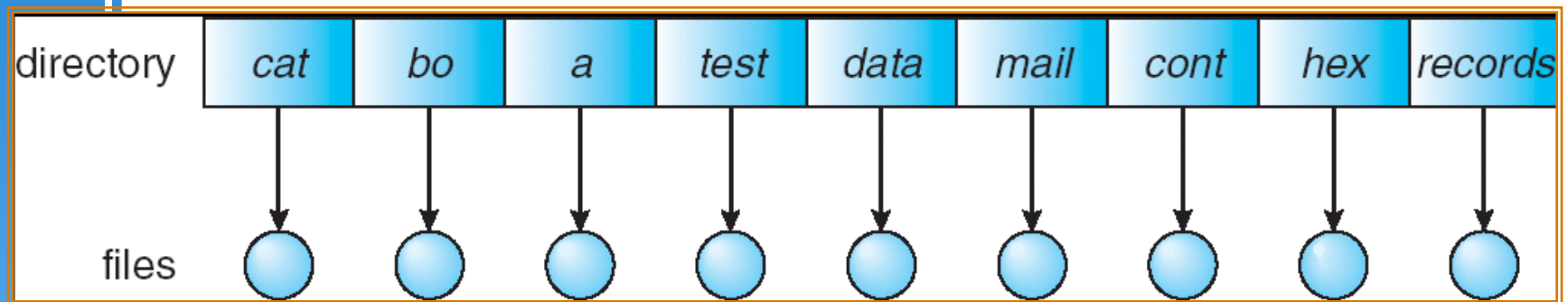
Organize the Directory (Logically) to Obtain

- ▶ 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

- ▶ A single directory for all users



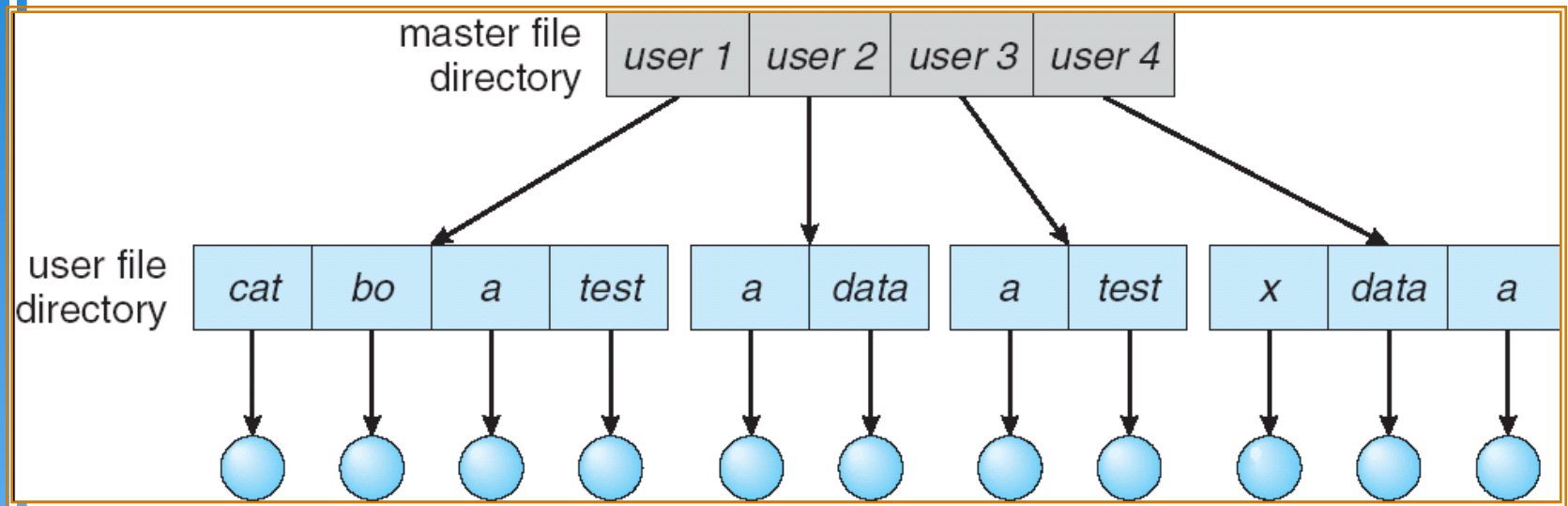
Naming problem

Grouping problem



Two-Level Directory

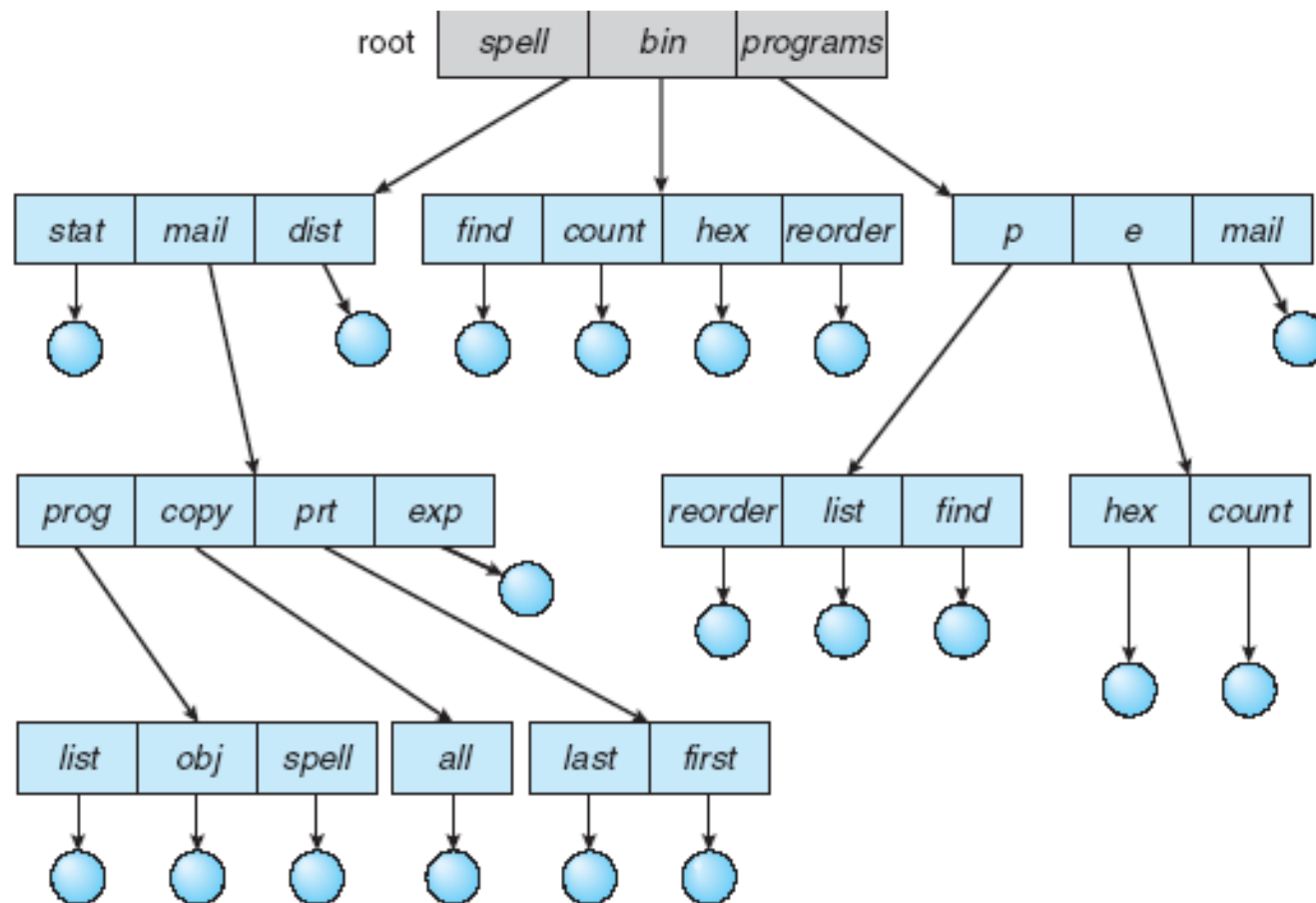
- ▶ Separate directory for each user



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



Tree-Structured Directories



Tree-Structured Directories (Cont)

- ▶ Efficient searching
- ▶ Grouping Capability
- ▶ Current directory (working directory)
 - `cd /spell/mail/prog`
 - `type list`



Tree-Structured Directories (Cont)

- ▶ **Absolute** or **relative** path name
- ▶ Creating a new file is done in current directory
- ▶ Delete a file

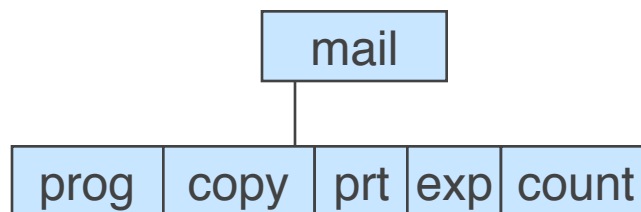
`rm <file-name>`

- ▶ Creating a new subdirectory is done in current directory

`mkdir <dir-name>`

Example: if in current directory `/mail`

`mkdir count`

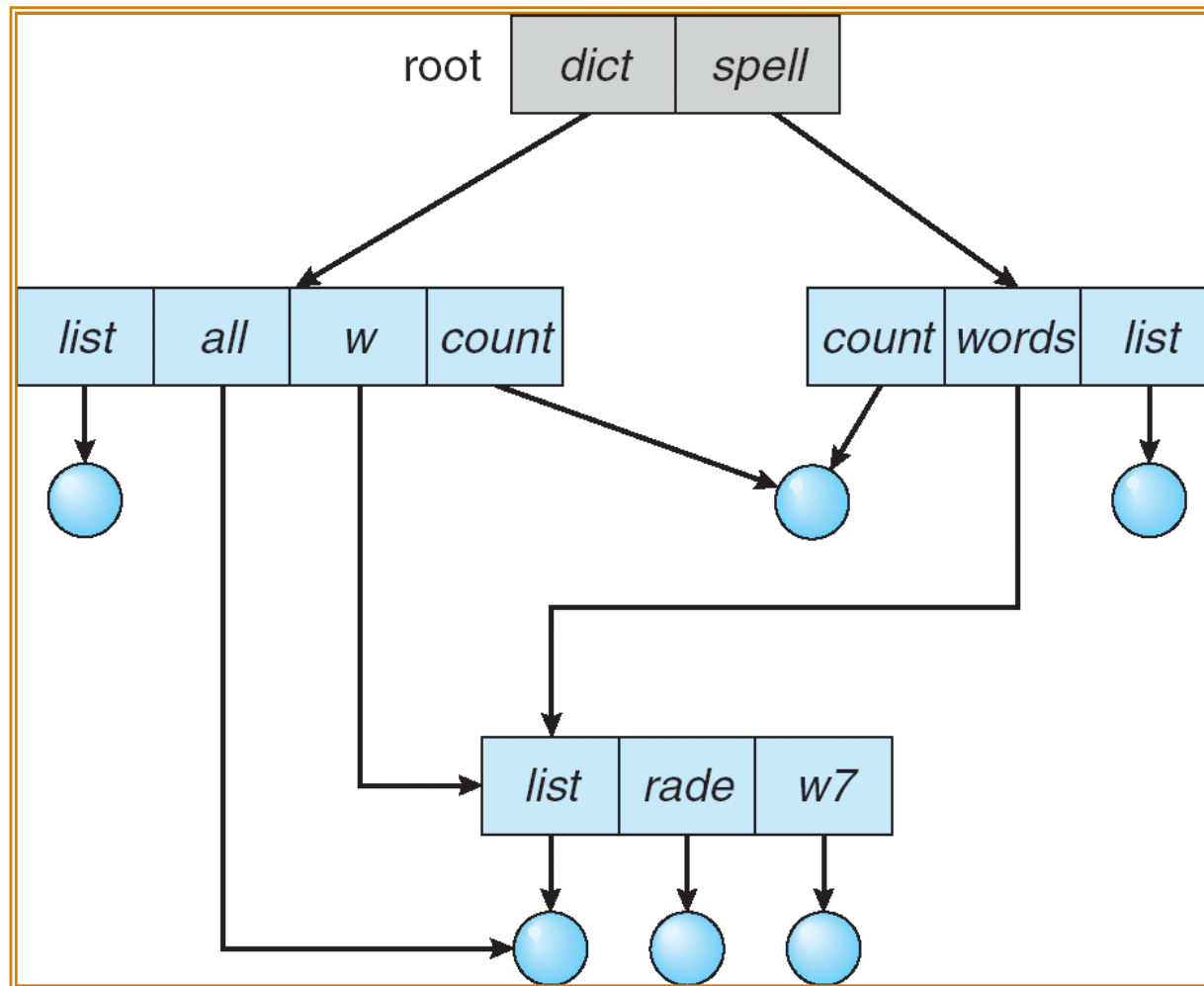


Deleting “mail” \Rightarrow deleting the entire subtree rooted by “mail”



Acyclic-Graph Directories

- ▶ Have shared subdirectories and files



Acyclic-Graph Directories (Cont.)

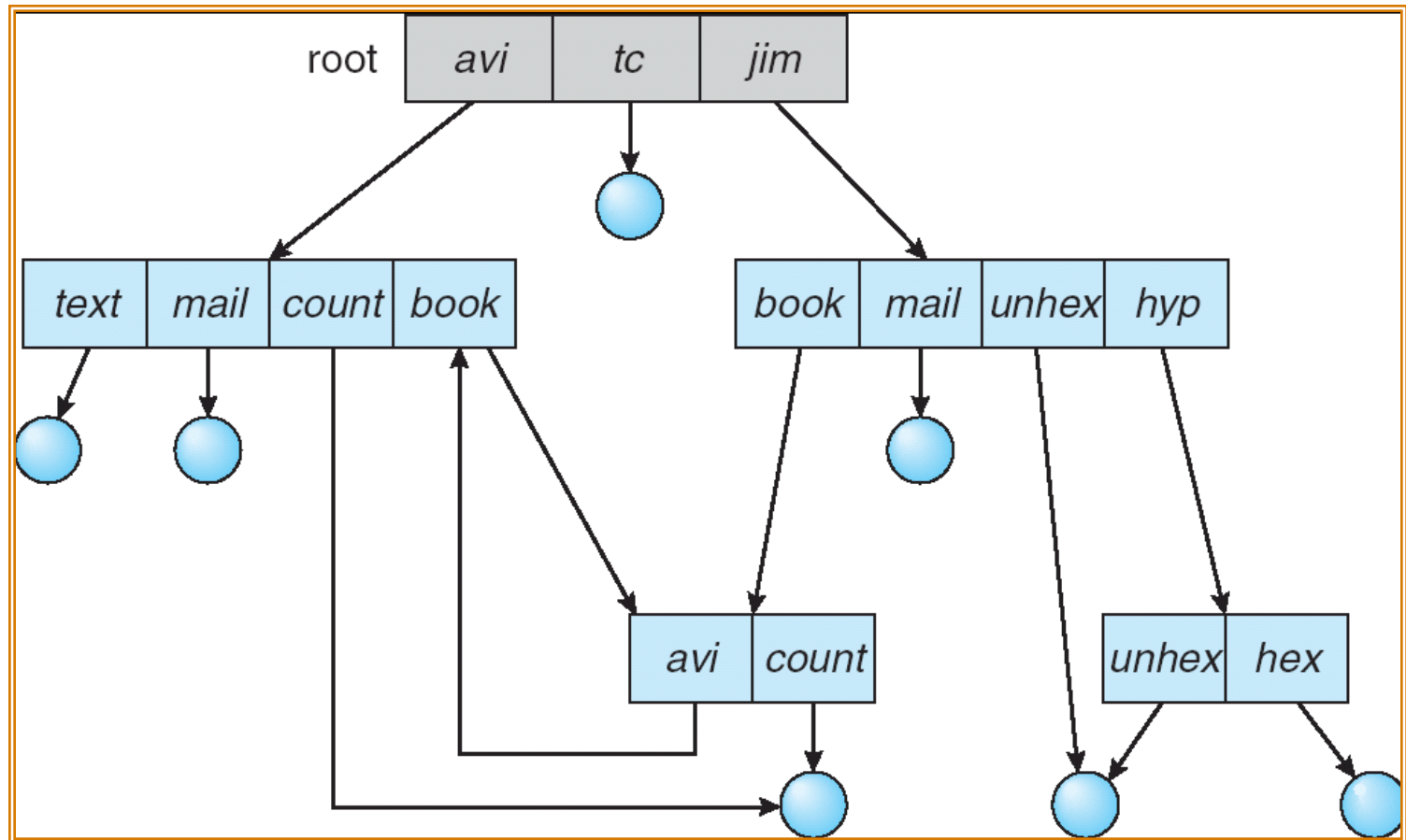
- ▶ Two different names (aliasing)
- ▶ If *dict* deletes *list* \Rightarrow dangling pointer

Solutions:

- Backpointers, so we can delete all pointers
Variable size records a problem
- Backpointers using a daisy chain organization
- Entry-hold-count solution
- ▶ New directory entry type
 - **Link** – another name (pointer) to an existing file
 - **Resolve the link** – follow pointer to locate the file



General Graph Directory



General Graph Directory (Cont.)

- ▶ How do we guarantee no cycles?
 - Allow only links to file not subdirectories
 - Garbage collection
 - Every time a new link is added use a cycle detection algorithm to determine whether it is OK

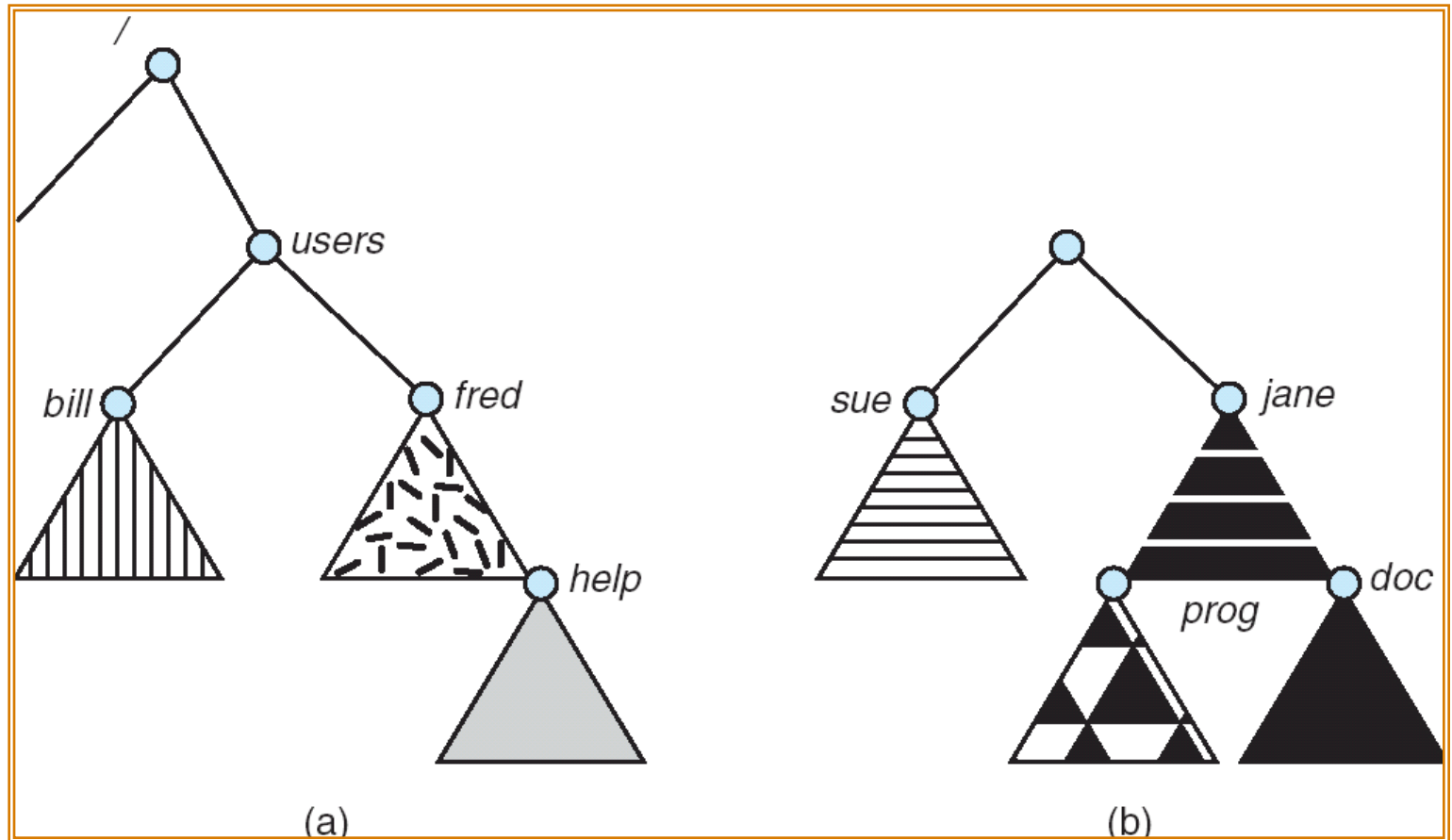


File System Mounting

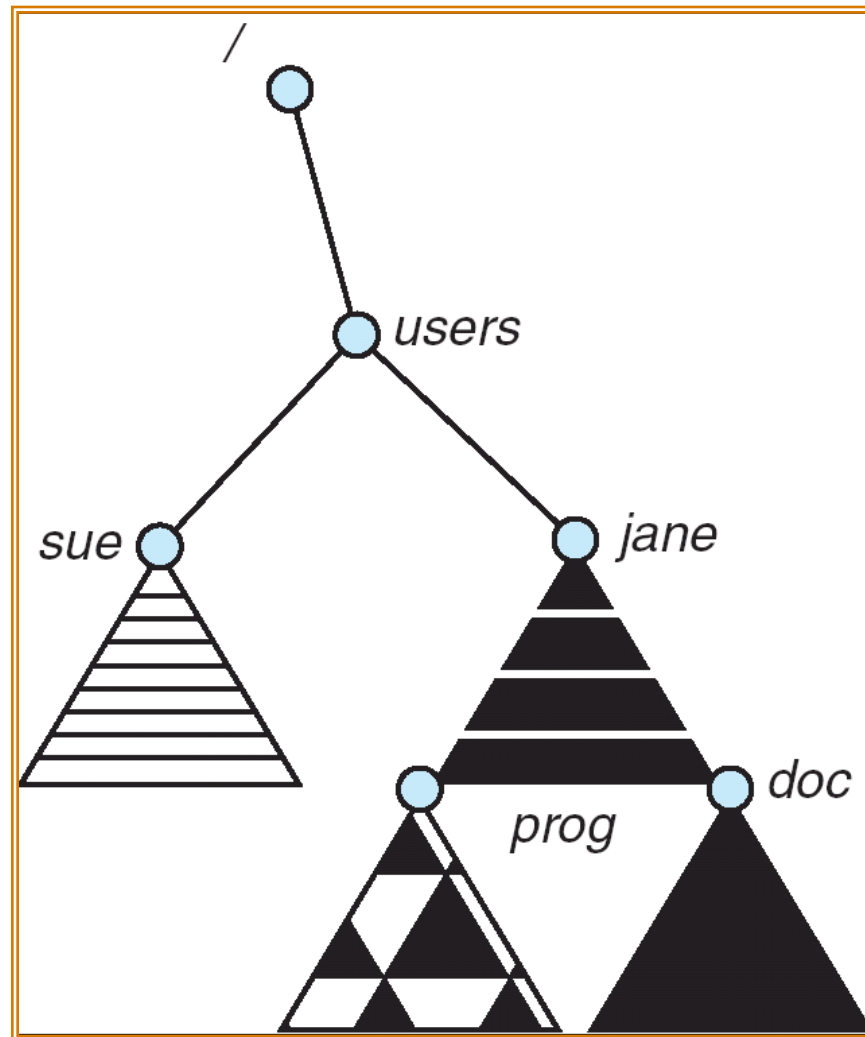
- ▶ A file system must be mounted before it can be accessed
- ▶ A unmounted file system is mounted at a mount point



(a) Existing. (b) Unmounted Partition



Mount Point



File Sharing

- ▶ Sharing of files on multi-user systems is desirable
- ▶ Sharing may be done through a **protection** scheme
- ▶ On distributed systems, files may be shared across a network
- ▶ Network File System (NFS) is a common distributed file-sharing method



File Sharing – Multiple Users

- ▶ **User IDs** identify users, allowing permissions and protections to be per-user
- ▶ **Group IDs** allow users to be in groups, permitting group access rights



File Sharing – Consistency Semantics

- ▶ **Consistency semantics** specify how multiple users are to access a shared file simultaneously
 - Similar to Ch 7 process synchronization algorithms
 - Tend to be less complex due to disk I/O and network latency (for remote file systems)
 - Andrew File System (AFS) implemented complex remote file sharing semantics
 - Unix file system (UFS) implements:
 - Writes to an open file visible immediately to other users of the same open file
 - Sharing file pointer to allow multiple users to read and write concurrently
 - AFS has session semantics
 - Writes only visible to sessions starting after the file is closed



Protection

- ▶ File owner/creator should be able to control:
 - what can be done
 - by whom

- ▶ Types of access
 - Read
 - Write
 - Execute
 - Append
 - Delete
 - List

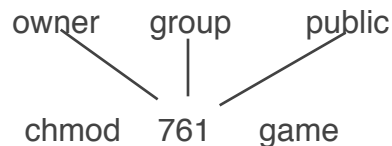


Access Lists and Groups

- ▶ Mode of access: read, write, execute
- ▶ Three classes of users

			RWX
a) owner access	7	⇒	1 1 1
			RWX
b) group access	6	⇒	1 1 0
			RWX
c) public access	1	⇒	0 0 1

- ▶ Ask manager to create a group (unique name), say G, and add some users to the group.
- ▶ For a particular file (say *game*) or subdirectory, define an appropriate access.



Attach a group to a file

chgrp G game

