Module 10: File-System Interface

- File Concept
- Access Methods
- Directory Structure
- Protection
- Consistency Semantics

File Concept

- Contiguous logical address space
- Types:
  - Data
    - ∗ numeric
    - ∗ character
    - ∗ binary
  - Program
    - ∗ source
    - ∗ object (load image)
  - Documents

File Structure

- 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

File Attributes

- **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** – data for protection, security, and usage monitoring.
- Information about files are kept in the directory structure, which is maintained on the disk.

File Operations

- create
- write
- read
- reposition within file – 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.

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, p, pas, f77, 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, rrf, etc	various word-processor formats
Library	lib, a	libraries of routines
Print or view	ps, dvi, gif	ASCII or binary file
Archive	arc, zip, tar	related files grouped into one file, sometimes compressed

Access Methods

- Sequential Access

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

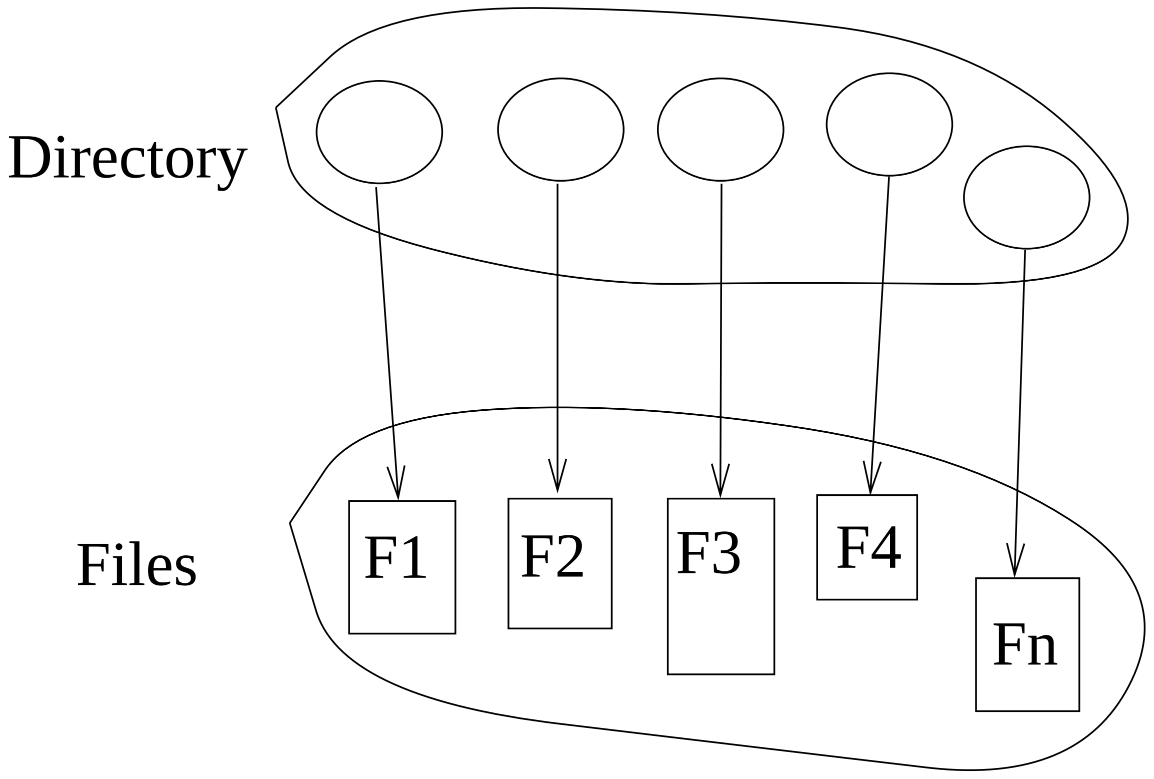
- Direct Access

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

n = relative block number

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.

Information in a Device Directory

- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed (for archival)
- Date last updated (for dump)
- Owner ID (who pays)
- Protection information (discuss later)

Operations Performed on Directory

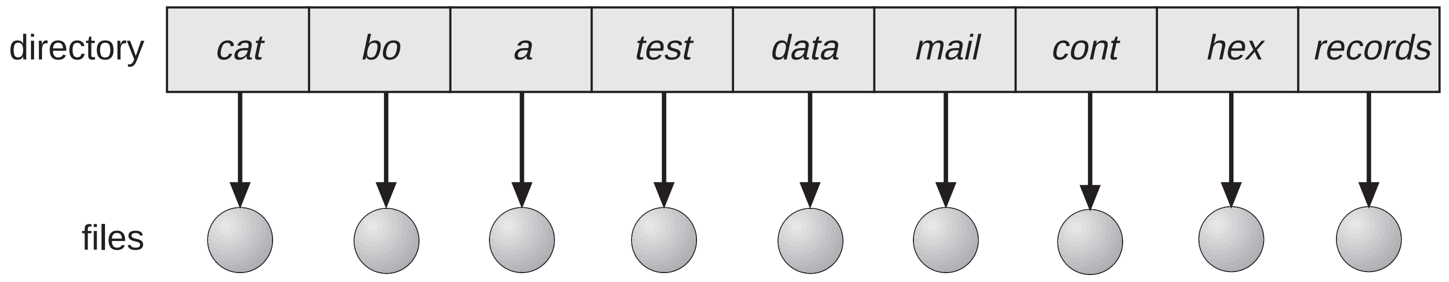
- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

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 Pascal programs, all games, ...)

Single-Level Directory

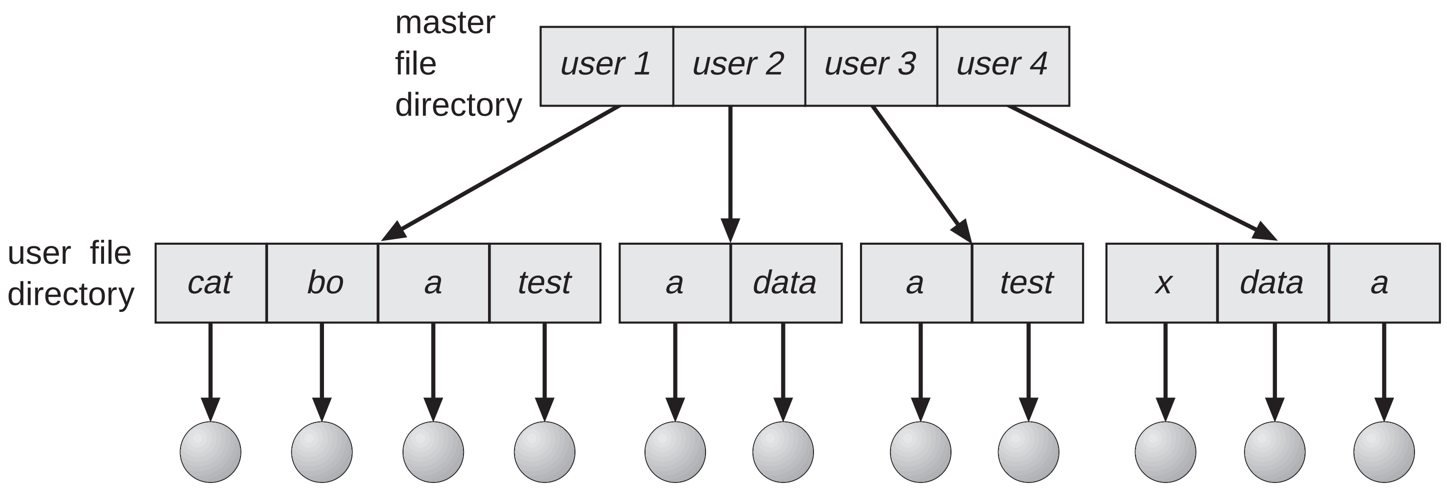
- A single directory for all users.



- Naming problem
- Grouping problem

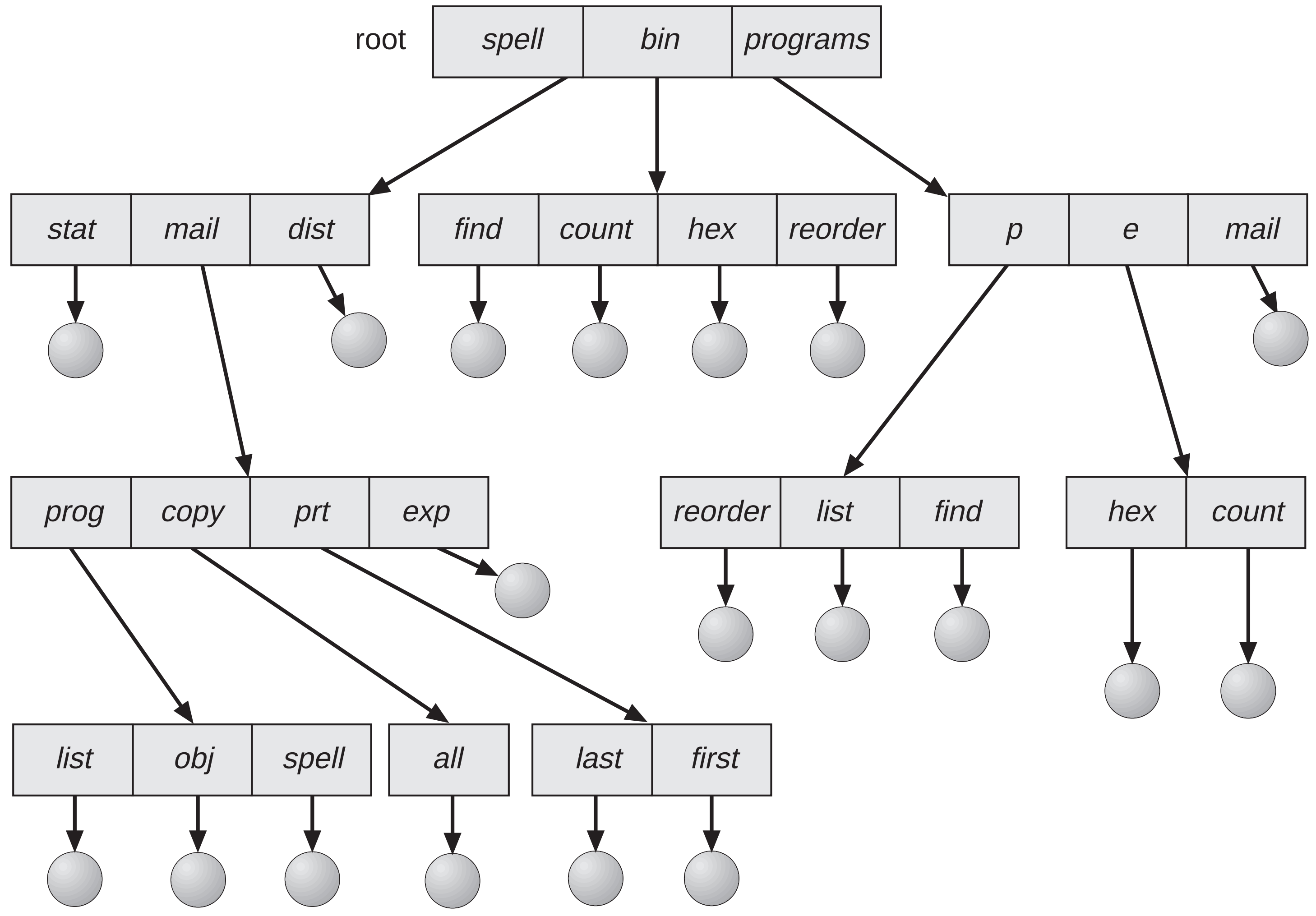
Two-Level Directory

- 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’d)

- 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

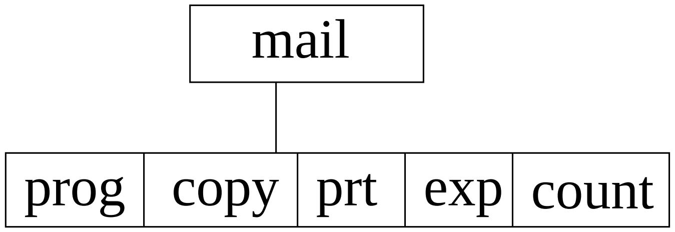
  rm <file-name>

- Creating a new subdirectory is done in current directory.

  mkdir <dir-name>

  Example: if in current directory /spell/mail

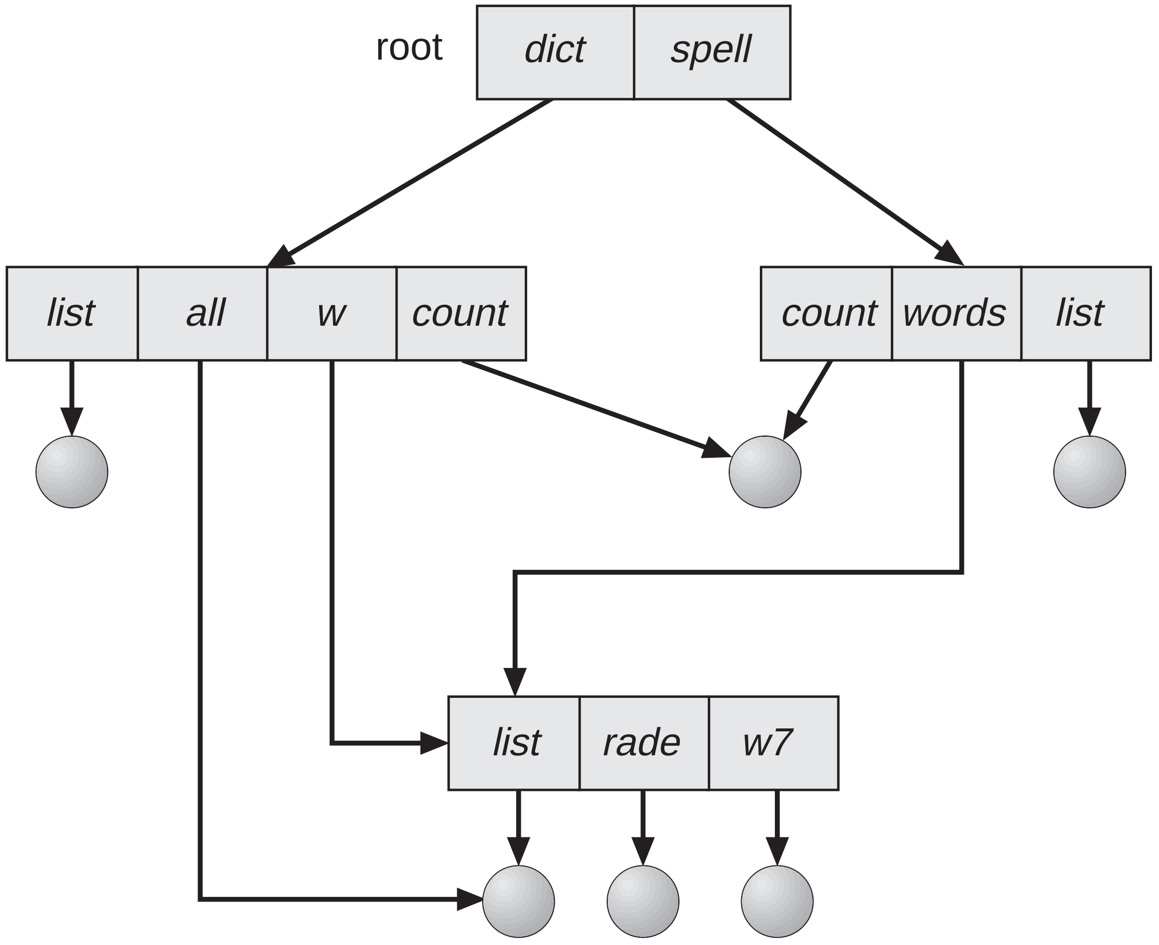
  mkdir count



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

Acyclic-Graph Directories

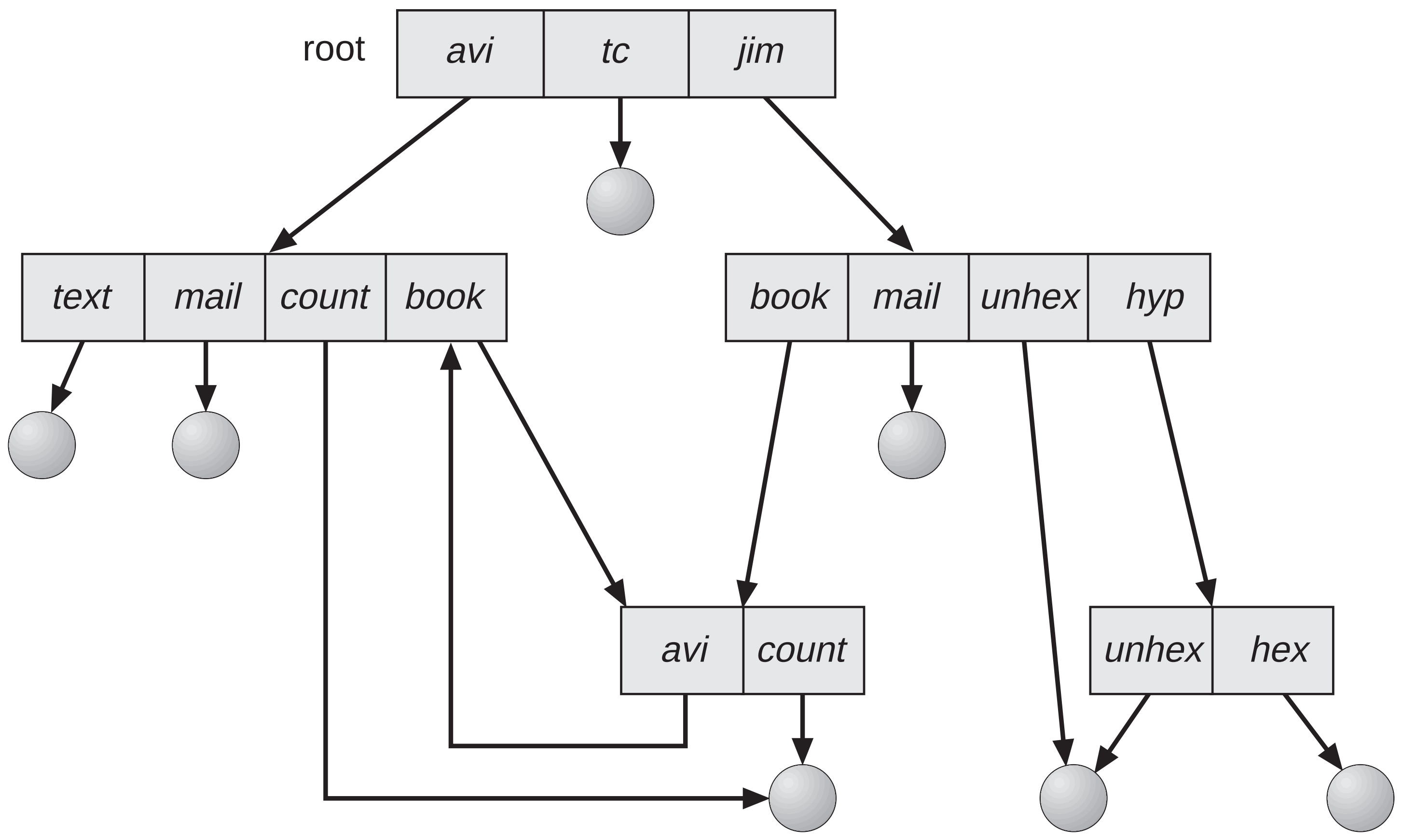
- Have shared subdirectories and files.



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.

General Graph Directory



General Graph Directory (Cont'd)

- 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.

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	$\Rightarrow$	1 1 1
			RWX
b) groups access	6	$\Rightarrow$	1 1 0
			RWX
c) public access	1	$\Rightarrow$	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.

owner group public

chmod 761 *game*

- Attach a group to a file

chgrp *G* *game*