File Systems

Chapter 4: Section 4.1-4.2

Why file systems?

- Storage needs exceed virtual address space.
- Virtual address space storage is lost when a process terminates
- Multiple processes may need the same data
  - Can't access other process' address space

Three Essential Requirements

1. It must be possible to store a very large amount of information.
2. The information must survive the termination of the process using it.
3. Multiple processes must be able to access the information at once.

Disks

» Disks are a linear sequence of fixed-size blocks*and supporting two operations:

1. Read block k.
2. Write block k

*For now**

** Also not actually paper

File System Design

1. How do you find information?
2. How do you keep one user from reading another user's data?
3. How do you know which blocks are free?

OS Abstractions

- Operating system abstracted away:
  - Processor to create the abstraction of a process
  - Physical memory to offer processes (virtual) address spaces
- Now we solve this problem with a new abstraction: the file.

Alternate Course Title

CSE 3320: Abstractions

Files

- Files are logical units of information created by processes
  - O/S contains millions
  - An address space
  - Persistent: not be affected by process creation and termination.
  - Deleted only when explicitly removed.

Files

- How they are structured, named, accessed, used, protected, implemented, and managed are major topics in operating system design.
- File System: part of the operating system dealing with files

File System

- User's perspective, the most important:
  - How it appears
  - How files are named and protected
  - Operations are allowed on files

- File system designer perspective:
  - Linked lists or bitmaps to track free storage
  - How many sectors there are in a logical disk block

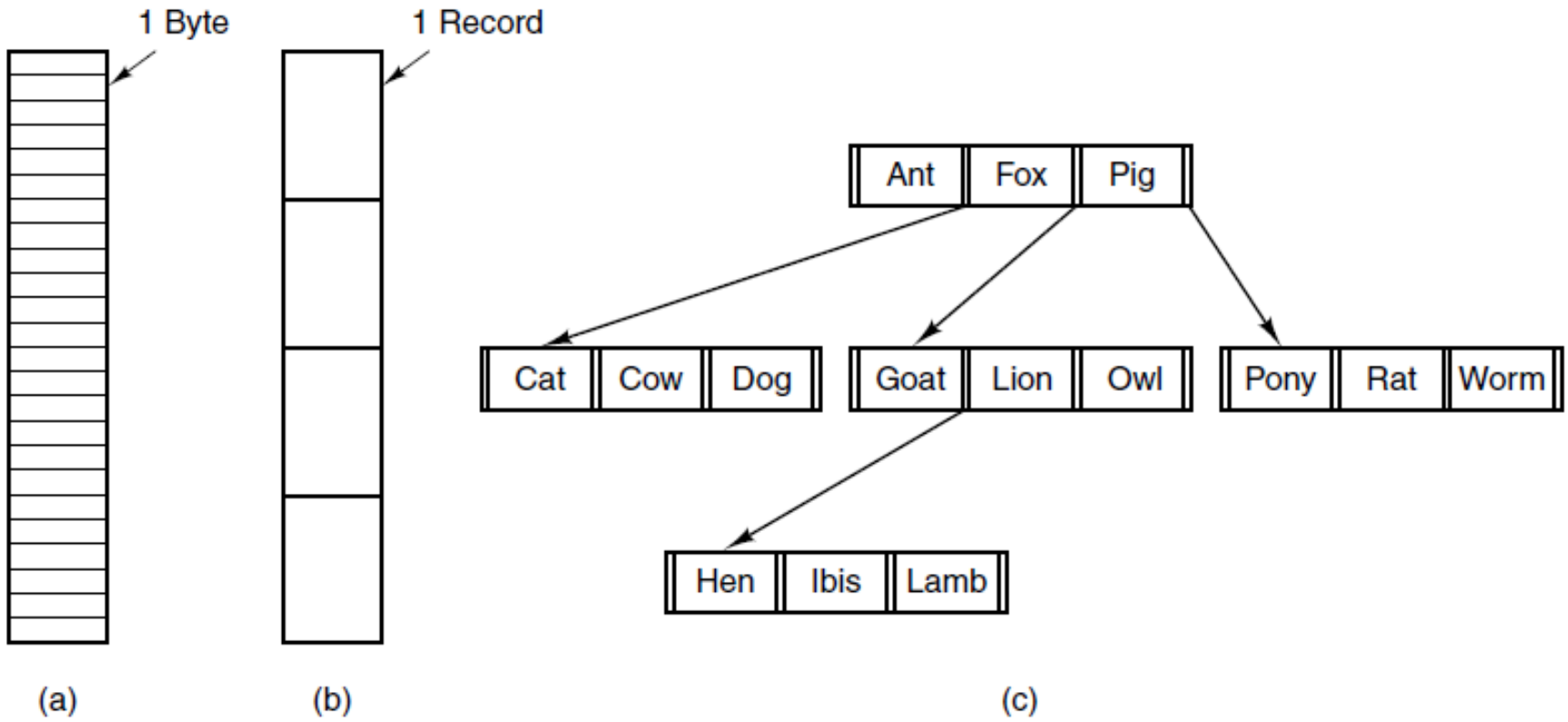
Files

- File is an abstract mechanism that provides a way to store information on the disk and read it back later.
- Shields the user from where the bits are stored and how it's retrieved.

File Naming

- The exact rules for file naming vary
  - Most modern OS allow 255+ characters
  - Case sensitive / Case insensitive
  - Unicode
  - File extensions (8 + 3)
    - Unix: convenience
    - Windows: cares

File Structure



Three kinds of files. (a) Byte sequence. (b) Record sequence. (c) Tree.

File Model

- Regarding files as nothing more than byte sequences provides the maximum amount of flexibility.
  - User programs can put anything they want in their files and name them any way that they find convenient.
  - All versions of UNIX (including Linux and OS X) and Windows use this file model.

Palm OS File Structure

- Data is saved in records
  - Contact information for a person in the address book would be stored in a record
- Records are stored in chunks of memory
- Databases are collections of chunks
  - NOT the usual definition of database
  - Think of each Palm database as a "file"
- To save space, you edit a database in place in memory instead of creating it in RAM and then writing it out to storage

PDB - Palm Database

» Format for databases in the Palm OS file system

PDB Structure
PDB Header
PDB Record Header
PDB Records

As many record headers as noted in

Name	Type	Size	Note
offset	integer	4 bytes	Byte number in the PDB file (counting from zero), where the record is located
attributes	byte	1 byte	Attributes of the record (delete/ dirty/busy/secret/category)
UniqueID	integer	3 bytes	

Offset	Name	Type	Size
0x00	name	char	32 Bytes
0x20	file attributes	int 16	2 Bytes
0x22	version	int 16	2 Bytes
0x24	creation time	int 32	4 Bytes
0x28	modification time	int 32	4 Bytes
0x2c	backup time	int 32	4 Bytes
0x30	modification number	int 32	4 Bytes
0x34	app_info	int 32	4 Bytes
0x38	sort_info	int 32	4 Bytes
0x3c	type	int 32	4 Bytes
0x40	creator	int 32	4 Bytes
0x44	unique_id_seed	int 32	4 Bytes
0x48	next_record_list	int 32	4 Bytes
0x4c	num_records	int 16	2 Bytes

Tree Structure

- File consists of a tree of records:
  - Not necessarily all the same length
  - Each containing a key field in a fixed position in the record.
  - The tree is sorted on the key field, to allow rapid searching for a particular key.

File Types

» Regular files are the ones that contain user information

» Directories are system files for maintaining the structure of the file system

» Character special files are related to input/output and used to model serial I/O devices

» Block special files are used to model disks.

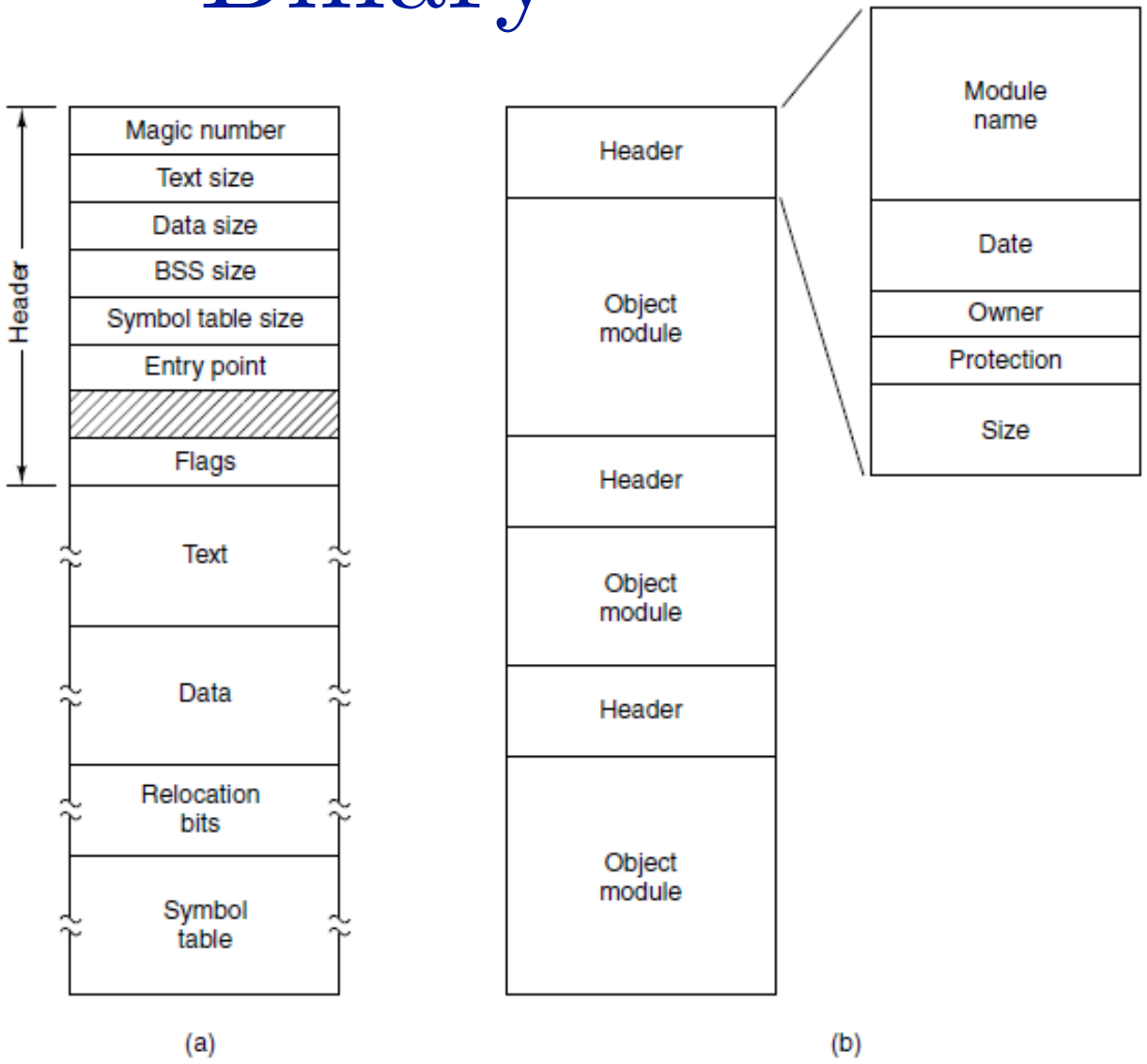
Regular Files

- Regular files are generally either ASCII files or binary files.
- Some systems each line is terminated by a carriage return character.
- Others, the line feed character is used.
  - Windows uses both

ASCII

- Big advantage of ASCII files
  - Can be displayed and printed as is
  - Edited with any text editor.
  - Easy to connect the output of one program to the input of another, as in shell pipelines.

Binary



(a) An executable file. (b) An archive

Binary Files

- Although technically the file is just a sequence of bytes, the operating system will execute a file only if it has the proper format.
  - Five sections: header, text, data, relocation bits, and symbol table.
  - Unix uses the ELF format

Header

- Header
  - Magic Number - Identifies the file as executable*
    - Magic numbers are common in programs across many operating systems.
    - Magic numbers implement strongly typed data
    - Form of in-band signaling to the controlling program.

Header (cont)

- » Sizes of the various pieces of the file
- » Address at which execution starts
- » Flag bits

» Following the header

- » Text segment
- » Data segment

Strongly Typed Files

- Having strongly typed files causes problems whenever the user does anything that the system designers did not expect.
  - user friendliness may help novices
  - drives experienced users up the wall

Access Methods

- Sequential Access

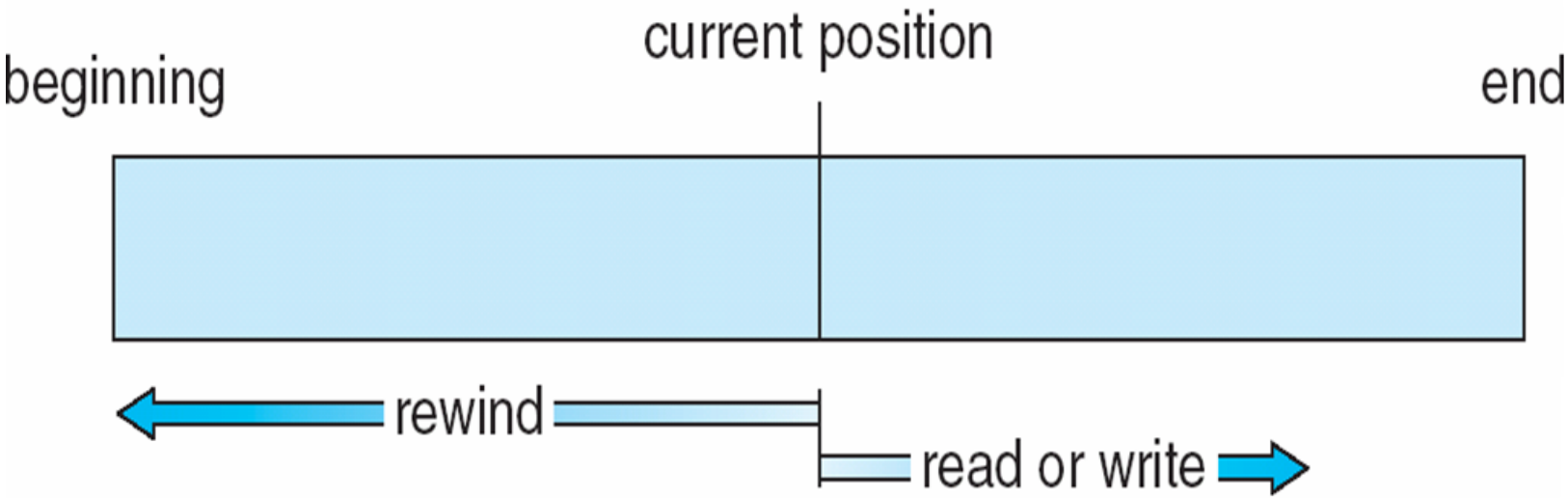
  read next
  write next
  reset
  no read after last write
  (rewrite)

- Direct Access

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

  n = relative block number

Sequential Access



Sequential files were convenient when the storage medium was magnetic tape rather than disk.

Random Access

» When disks came into use for storing files, it became possible to read the bytes or records of a file out of order, or to access records by key rather than by position.

» Operations Read, Write and now Seek

File Metadata

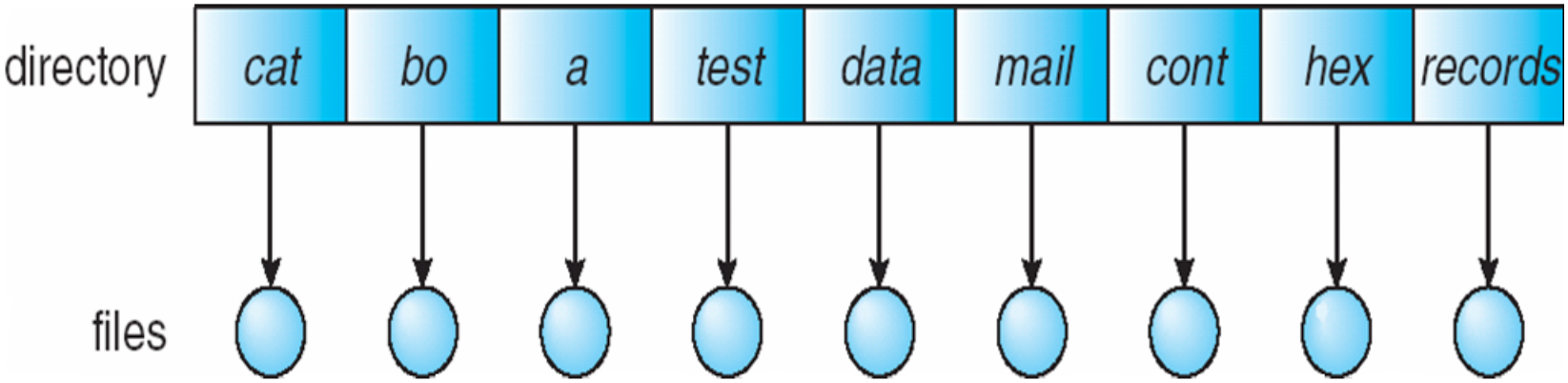
» Varies considerably from system to system

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file was last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

File Operations

1. Create
2. Delete
3. Open
4. Close
5. Read
6. Write
7. Append
8. Seek
9. Get attributes
10. Set attributes
11. Rename

Directories



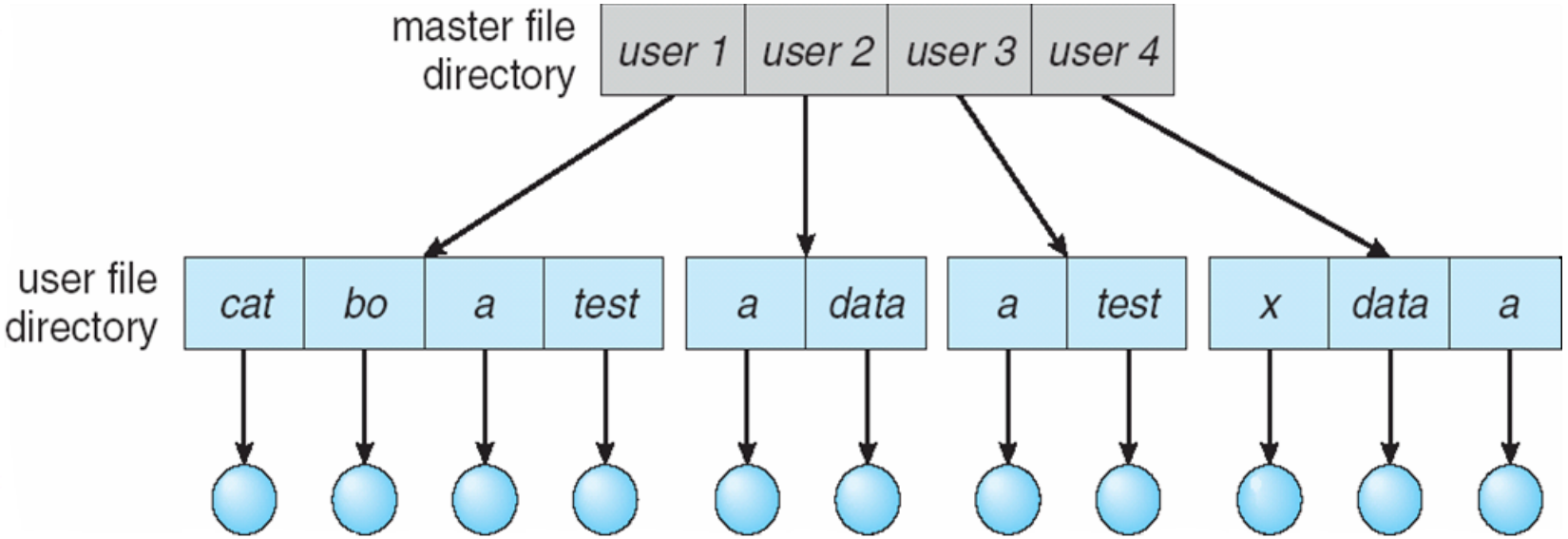
» The simplest form of directory system is having one directory containing all the files.

  » Naming problem

  » Grouping problem

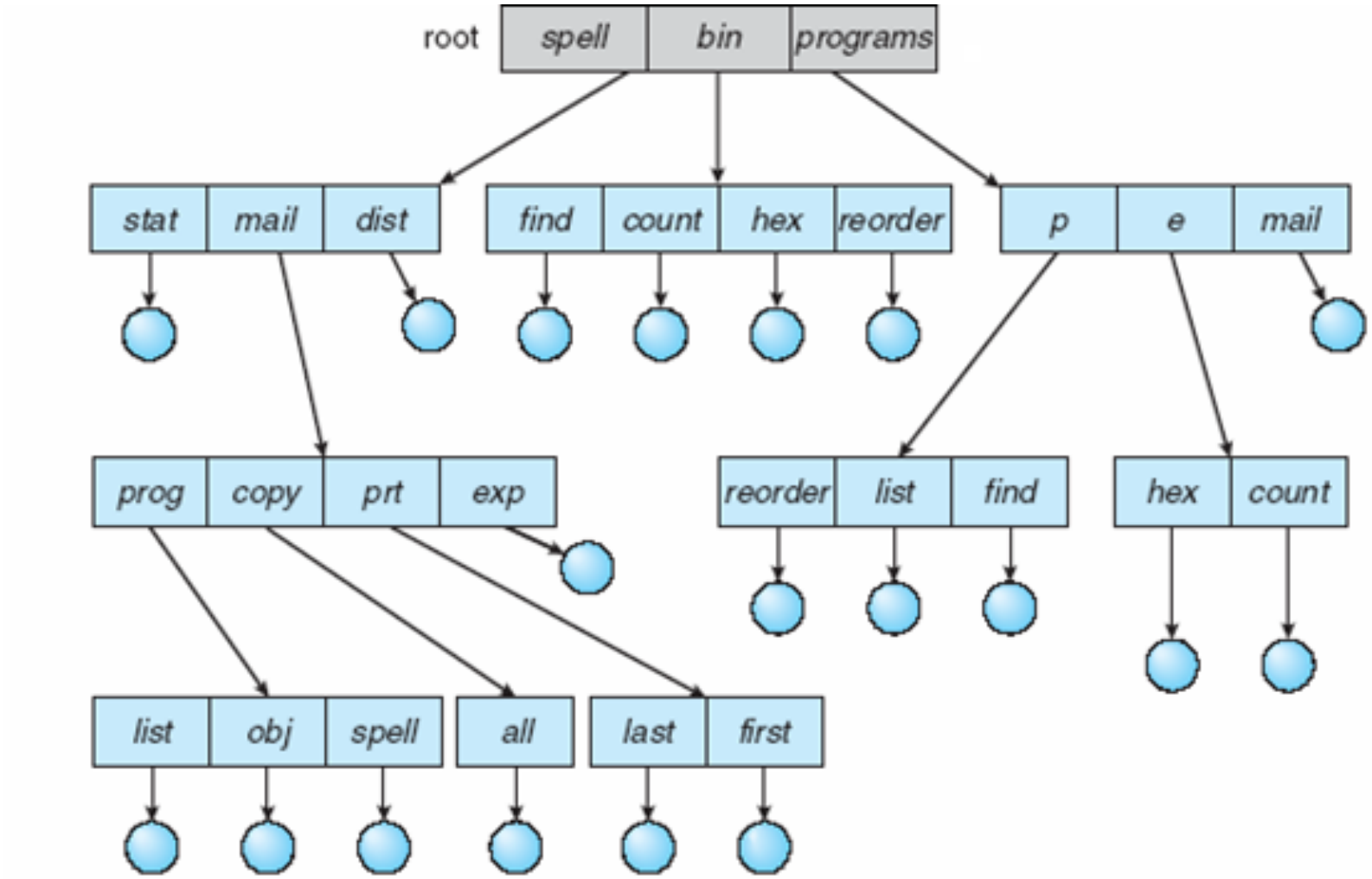
Two-Level Directory

- 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)

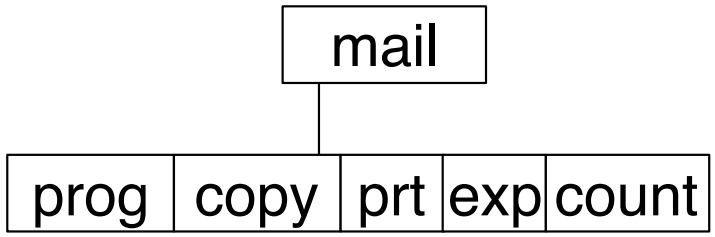
- **Absolute** or **relative** path name
- Creating a new file is done in current directory
- Delete a file

  rm <file-name>
- Creating a new subdirectory is done in current directory

  mkdir <dir-name>

  Example: if in current directory **/mail**

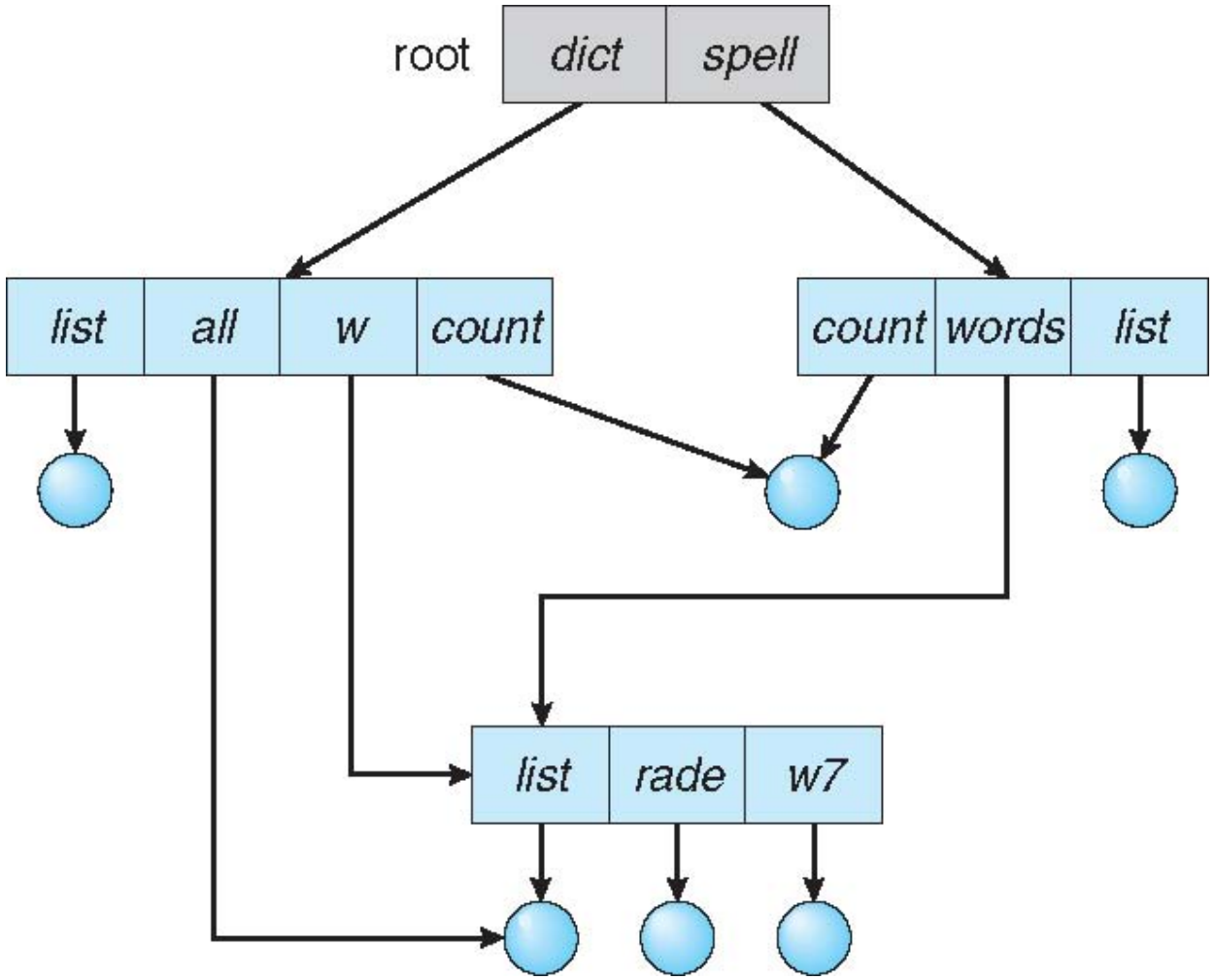
  mkdir count



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

Directed Acyclic-Graph Directories

- Have shared subdirectories and files



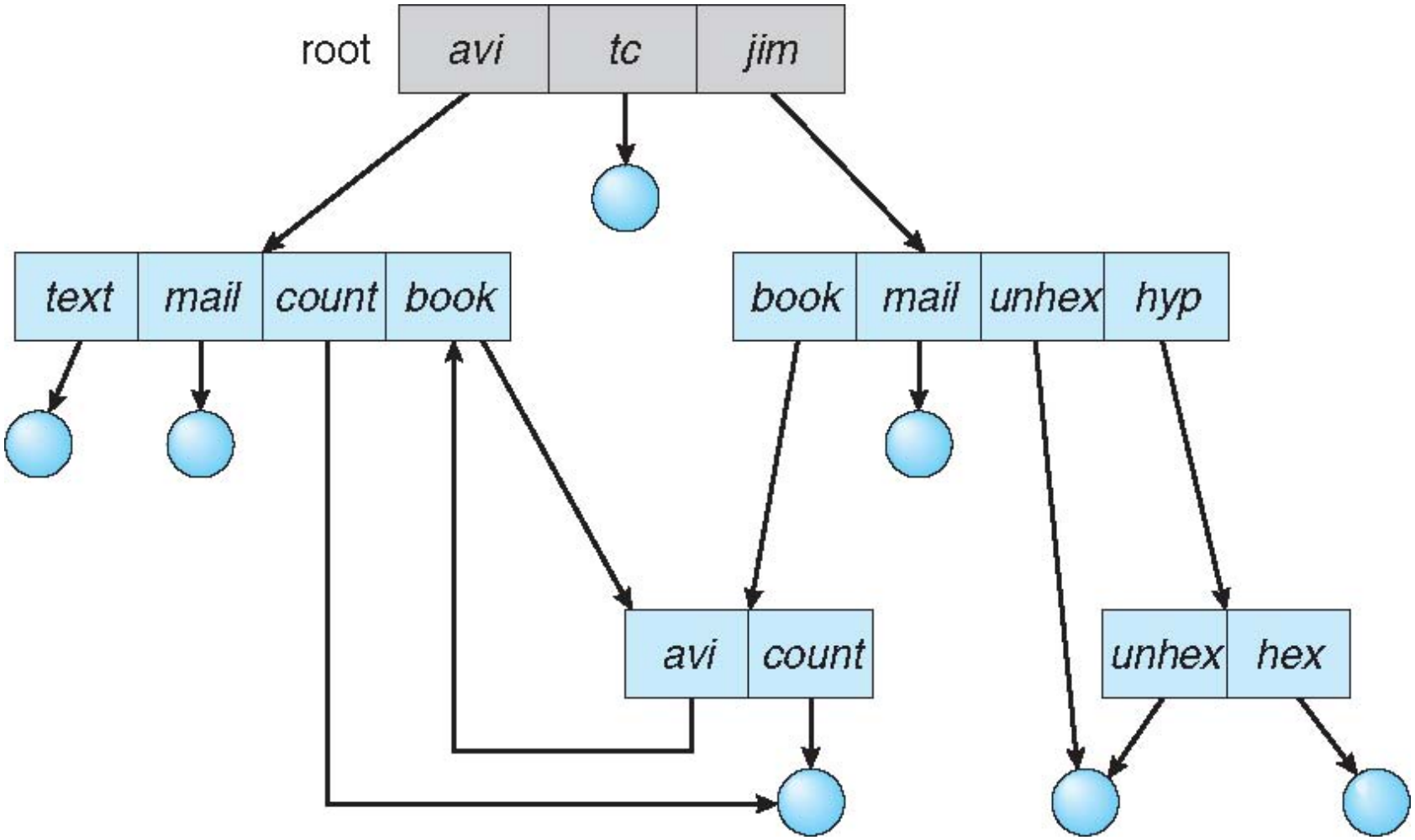
Acyclic-Graph Directories (Cont.)

- Two different names (aliasing)
- If *dict* deletes *list* ⇒ 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