EECS 211:

Advanced System Software

Lecture: File-System Interface

Prof. Mohammad Al Faruque

The Henry Samueli School of Engineering
Electrical Engineering & Computer Science
University of California Irvine (UCI)

File-System Interface

- File Concept
- Access Methods
- Disk and Directory Structure
- File-System Mounting
- Protection

File Concept

- A file is a named collection of related information that is recorded on secondary storage.
  - OS provides a uniform logical view of the stored information
  - OS abstracts from the physical properties of its storage devices to define a logical storage units → file
- Types:
  - Data Files: numeric, character, binary
  - Program (both source and object forms)
- Files may be free form, e.g. text file or may be formatted rigidly
- Contents defined by file's creator
  - A file has a certain defined structure depending on its type
    - Consider text file (characters organized onto lines), source file (sequence of functions), executable file (sequence of code sections)

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
- Many variations, including extended file attributes such as file checksum
- Information kept in the directory structure: typically a directory entry consists' of the file's name and its unique identifier

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

- **It may take more than a KB to record this information for each file**
- **In a system with many files, the size of the directory itself may be megabytes**

- Information about files are kept in the directory structure, which is maintained on the disk
- Many variations, including extended file attributes such as file checksum
- Information kept in the directory structure: typically a directory entry consists' of the file's name and its unique identifier

File info Window on Mac OS X

File Operations

- File is an abstract data type
- OS provides necessary system calls

- These are the 6 basic operations → minimal set of required file operations
- Basic Operations may form other functions, e.g. copy a file to another IO device
- Most of the file operation involves searching the directory

6. 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 Operations

- File is an abstract data type
- OS provides necessary system calls

1. Create: Two step process
   1. Check for available space
   2. An entry for the new file in the directory
2. Write – at write pointer location → input file name and information
3. Read – at read pointer location → input file name and memory address to write the block
4. Reposition within file - seek
5. Delete
6. 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:**
  - **Open-file table: tracks open files**
    - Two levels of internal tables
      - **Per-process Table:** process's use of the file→ current file pointer, access rights
      - **System-wide Table:** process independent information → location of the file to the disk, access dates, file size. Each entry in per-process table points to a system-wide open-file table
- **Information associated with the open file**
  - **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**
  - **Access rights: per-process access mode information**
  - **Disk location of the file: cache of data access information**

Open File Locking

- **Provided by some operating systems and file systems**
  - **Similar to reader-writer locks**
  - **Shared lock similar to reader lock – several processes can acquire concurrently**
  - **Exclusive lock similar to writer lock**

- **Mandatory or advisory:**
  - **Mandatory – access is denied depending on locks held and requested (Windows OS)**
    - OS responsibility to ensure integrity
  - **Advisory – processes can find status of locks and decide what to do (Linux OS)**
    - Software programmer responsibility to ensure locking → acquire and release

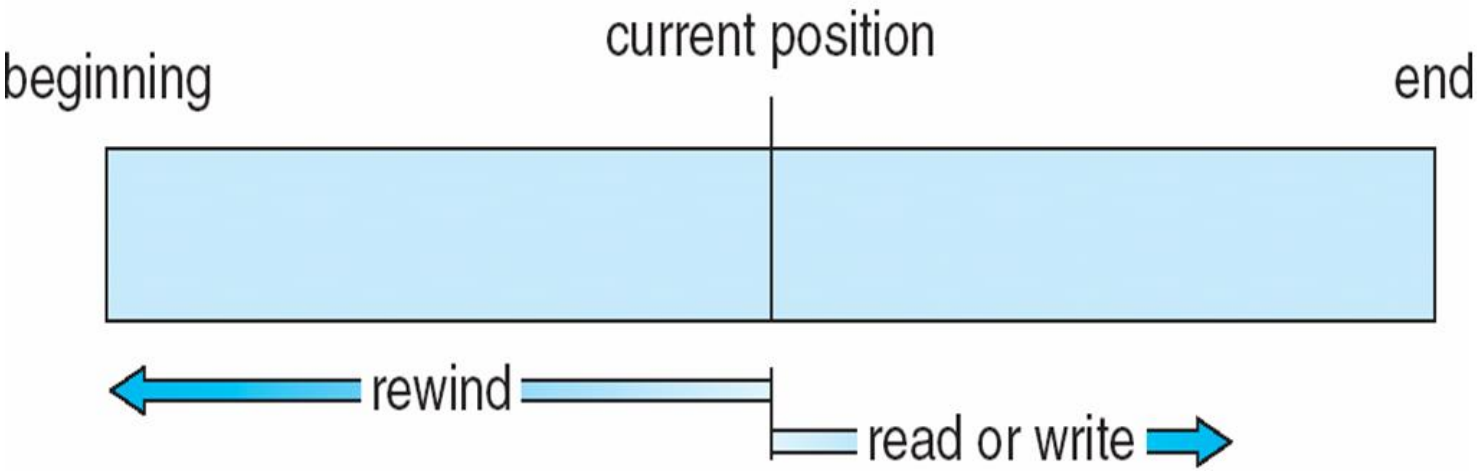
File Types – Name, Extension

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 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

Sequential-access File



- The logical record size, physical block size, and packing techniques determine how many logical records are in each physical block → may suffer internal fragmentation
  - The packing job can be done by OS or by application program
  - All basic I/O operations are done in terms of blocks.

Access Methods

- **Sequential Access**

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

- **Direct Access – file is fixed length logical records → based on a disk model of a file**

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

n = relative block number

- **Relative block numbers allow OS to decide where file should be placed → Allocation problem → will study next lecture**

Simulation of Sequential Access on Direct-access File

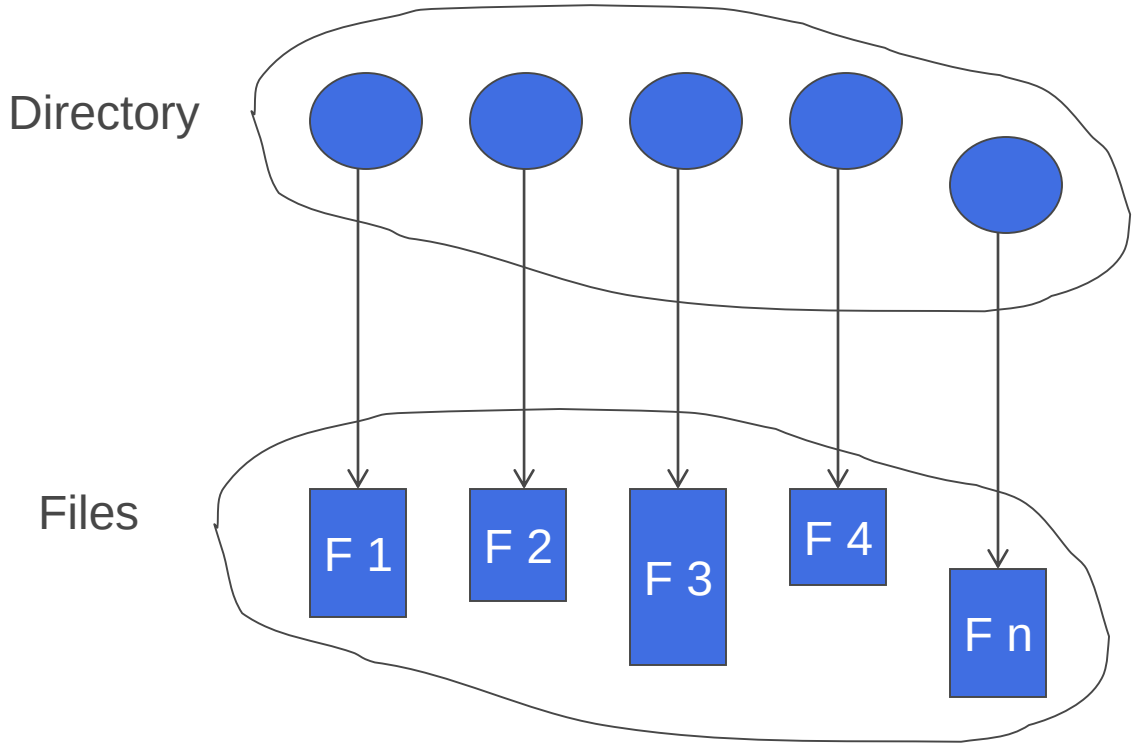
sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

Cp defines the current position

- **Not all OS supports both sequential and direct access → Design decision.**

Directory Structure

- **A collection of nodes containing information about all files**

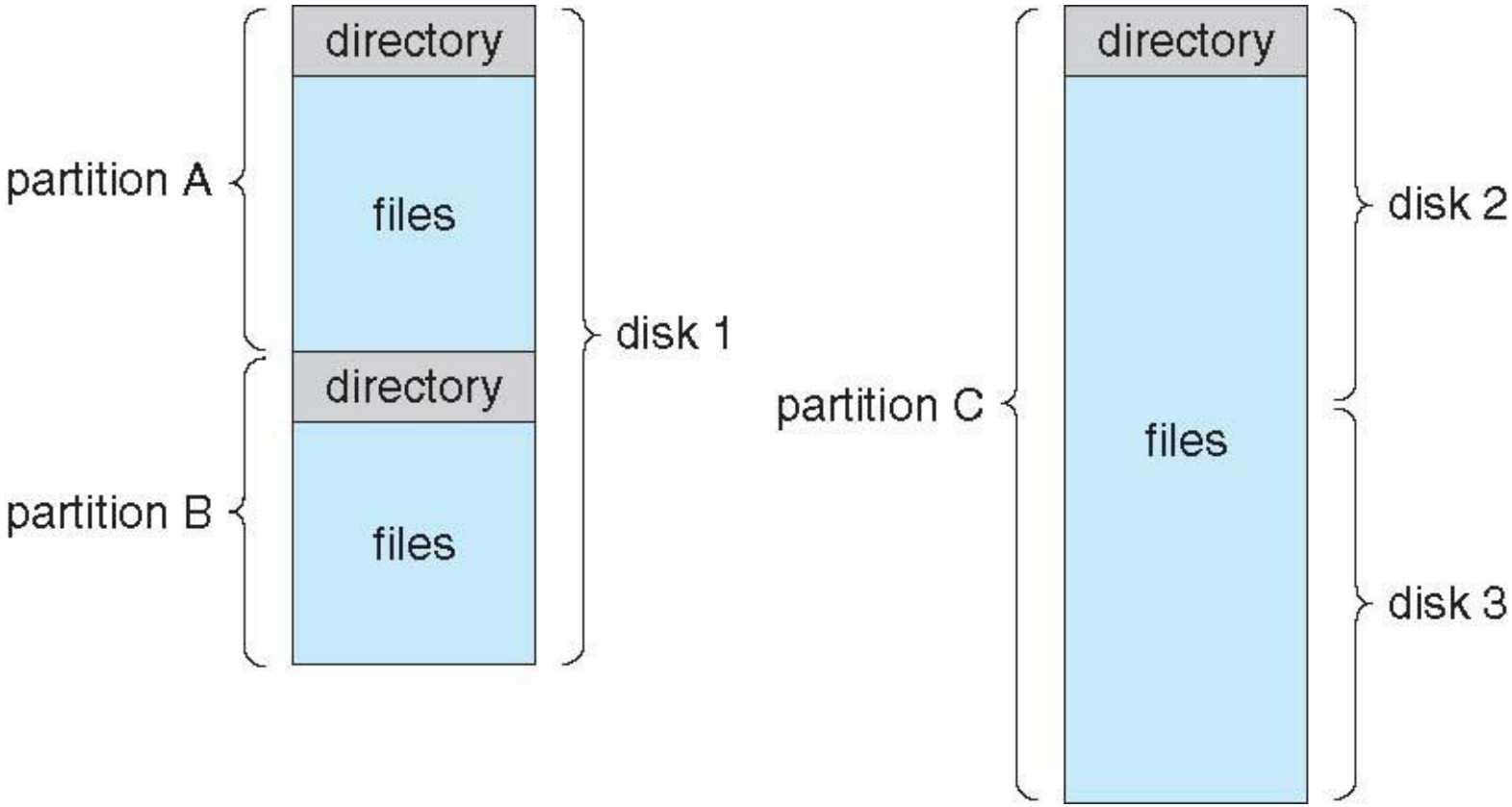


Both the directory structure and the files reside on disk

Disk Structure

- Disk can be subdivided into partitions
- Disks or partitions can be RAID protected against failure
- Disk or partition can be used raw – without a file system, or formatted with a file system
- Partitions also known as minidisks, slices
- Entity containing file system known as a volume
- Each volume containing file system also tracks that file system's info in device directory or volume table of contents
- As well as general-purpose file systems there are many special-purpose file systems, frequently all within the same operating system or computer

A Typical File-system Organization



Operations Performed on Directory

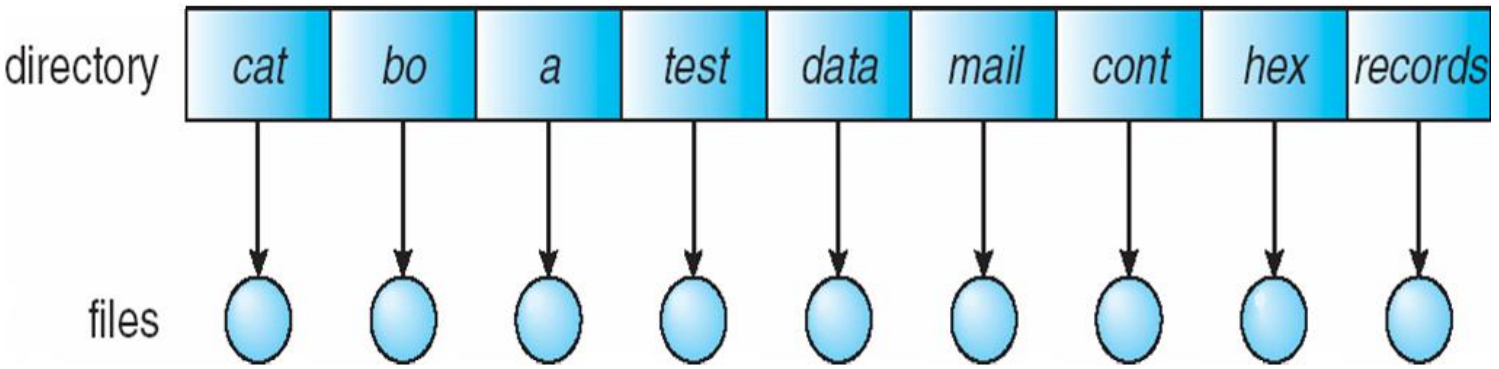
- A directory may be viewed as a symbol table that translates file names into their directory entries
- Operations to be performed on a directory
  - Search for a file
  - Create a file
  - Delete a file
  - List a directory
  - Rename a file
  - Traverse the file system

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, …)**

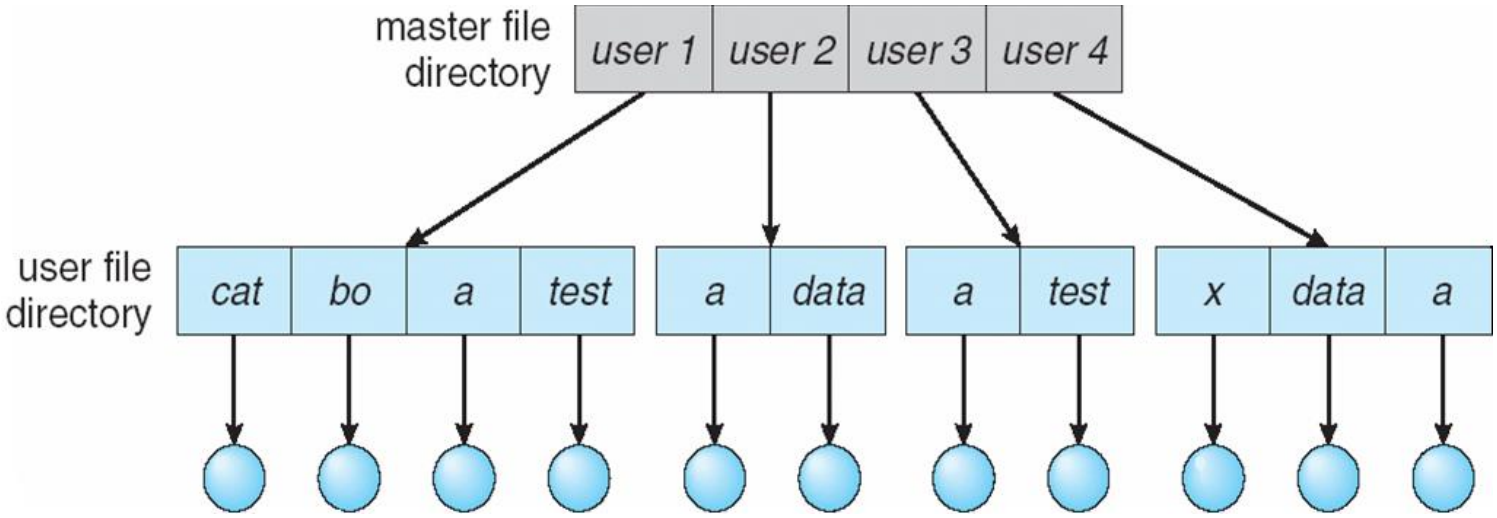
Single-Level Directory

- A single directory for all users



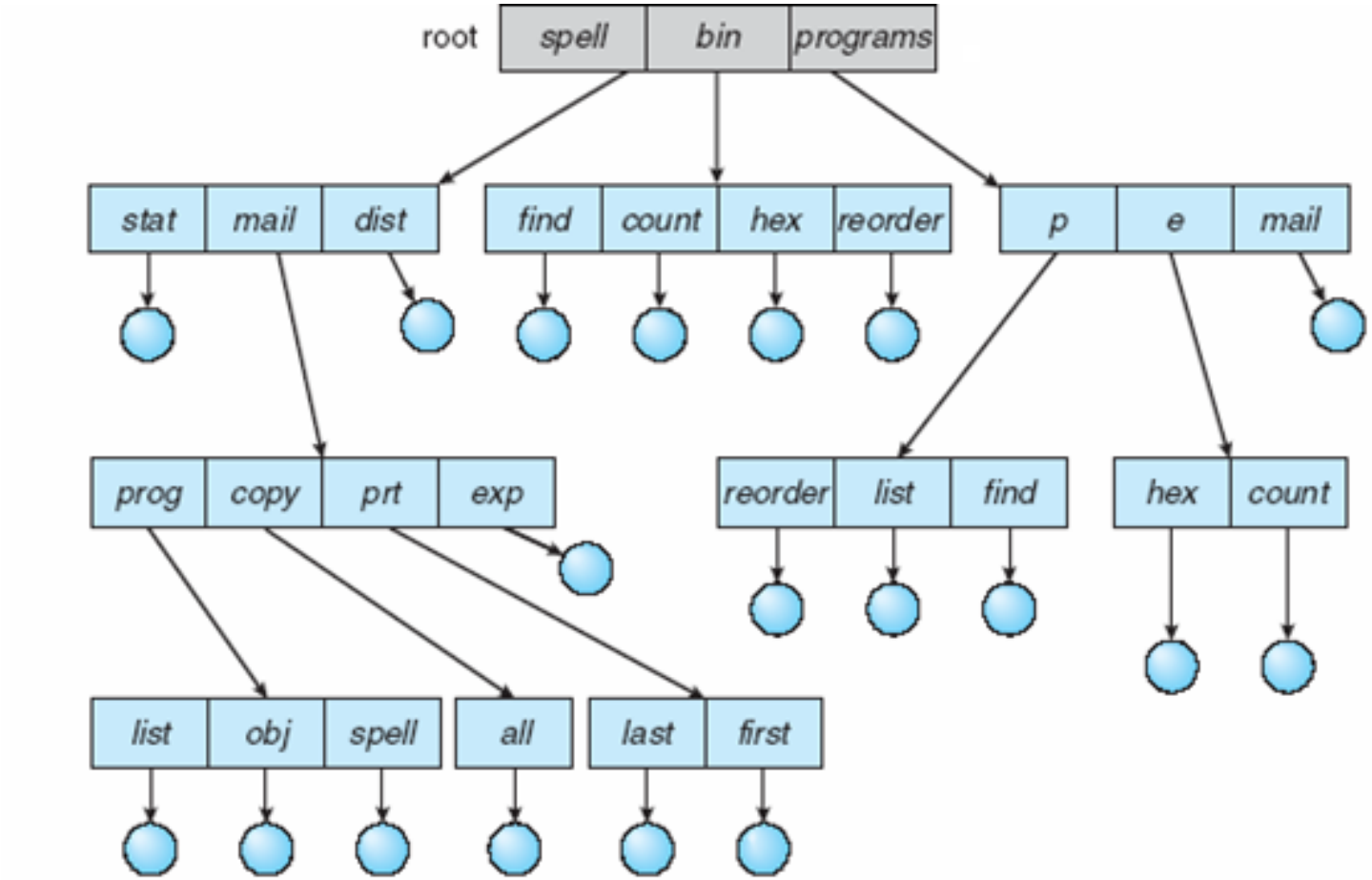
Two-Level Directory

- 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**

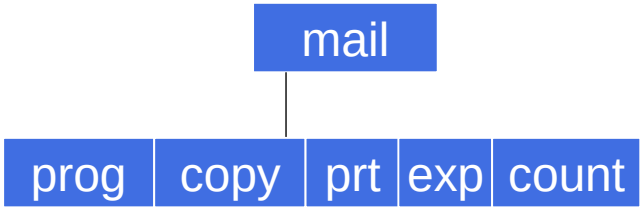
```
rm <file-name>
```

- **Creating a new subdirectory is done in current directory**

```
mkdir <dir-name>
```

Example: if in current directory /mail

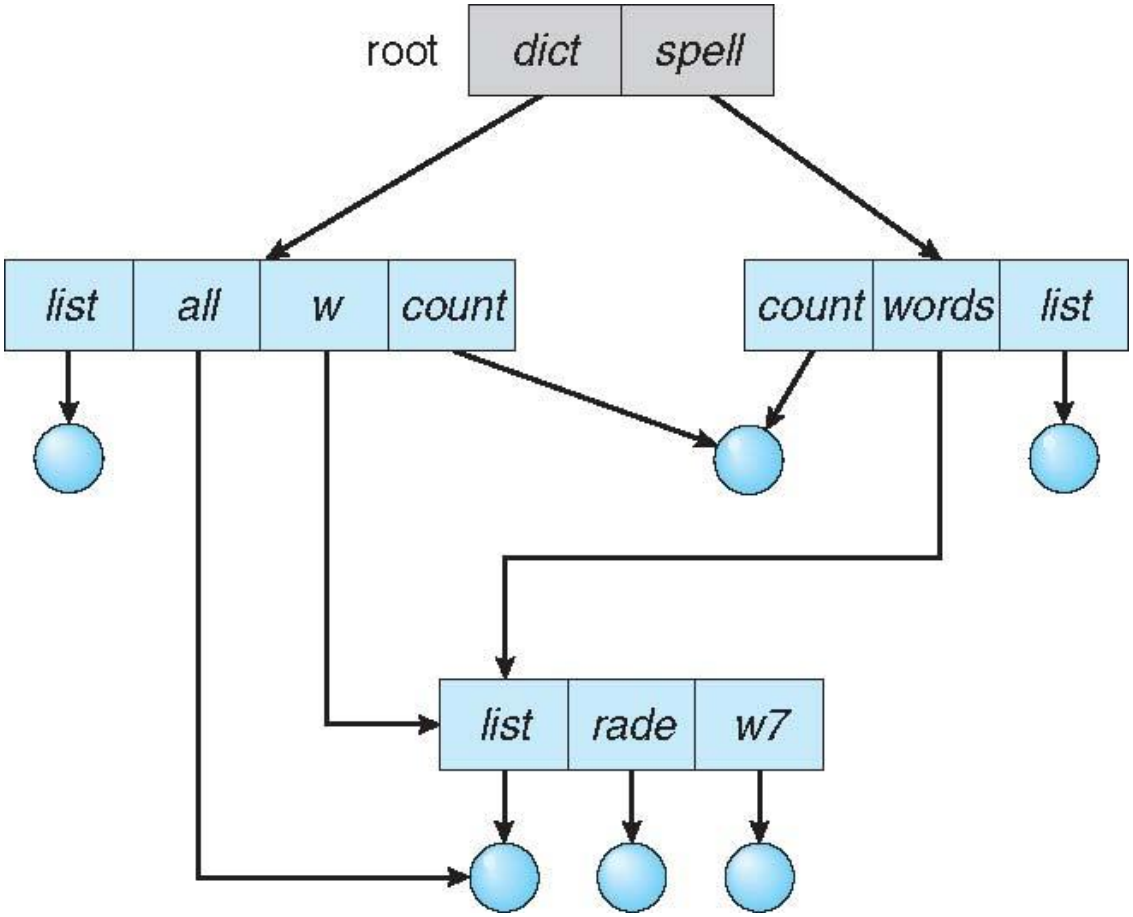
```
mkdir count
```



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

Acyclic-Graph Directories

- Have shared subdirectories and files



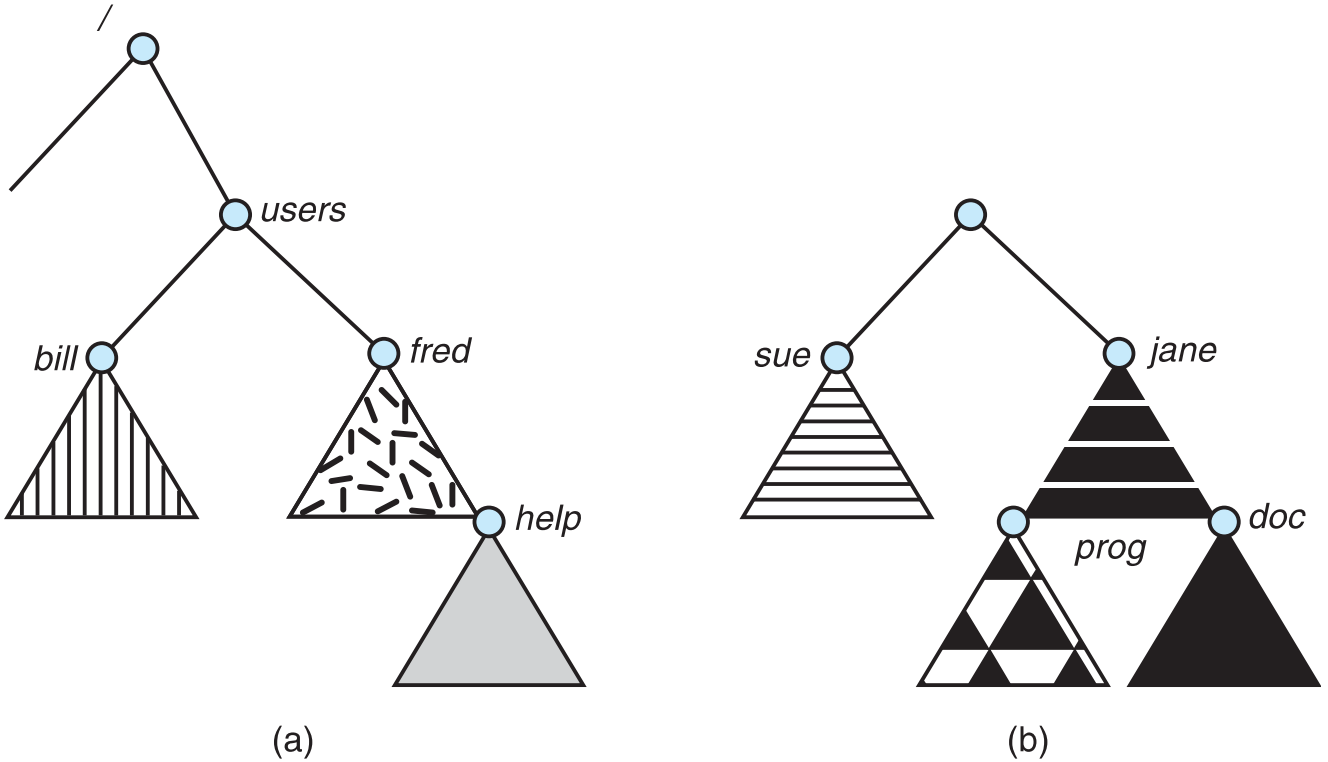
Acyclic-Graph Directories (Cont.)

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

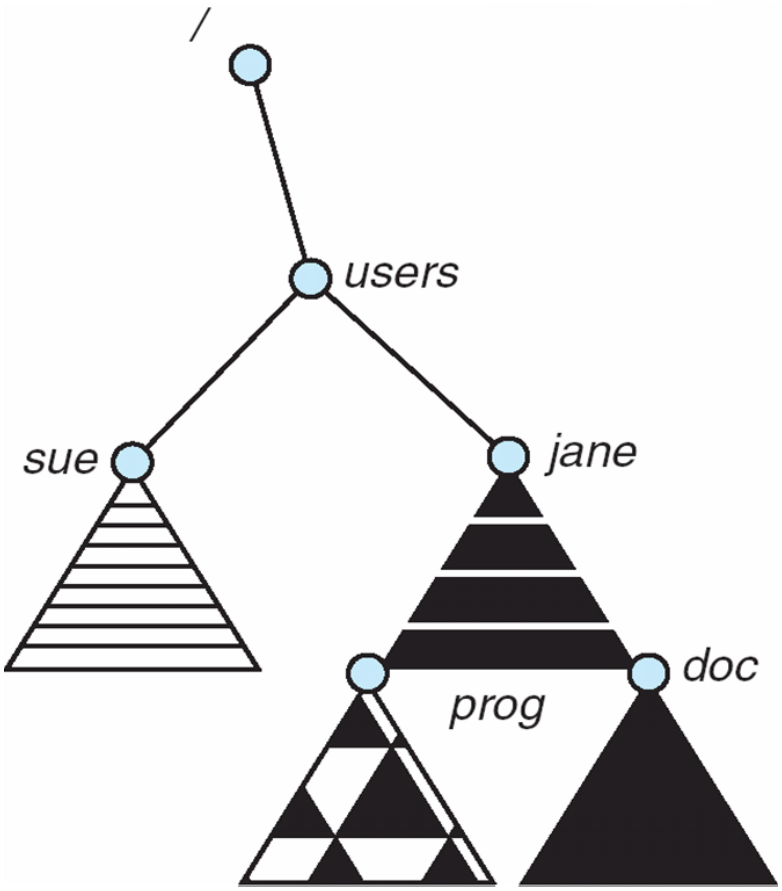
  Solutions:
  - Backpointers, so we can delete all pointers
    Variable size records a problem
  - Backpointers using a daisy chain organization
  - Entry-hold-count solution
- New directory entry type
  - Link – another name (pointer) to an existing file
  - Resolve the link – follow pointer to locate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

File System Mounting

- A file system must be mounted before it can be accessed
- A unmounted file system (i.e., Fig. 11-11(b)) is mounted at a mount point



Mount Point



Protection

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

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

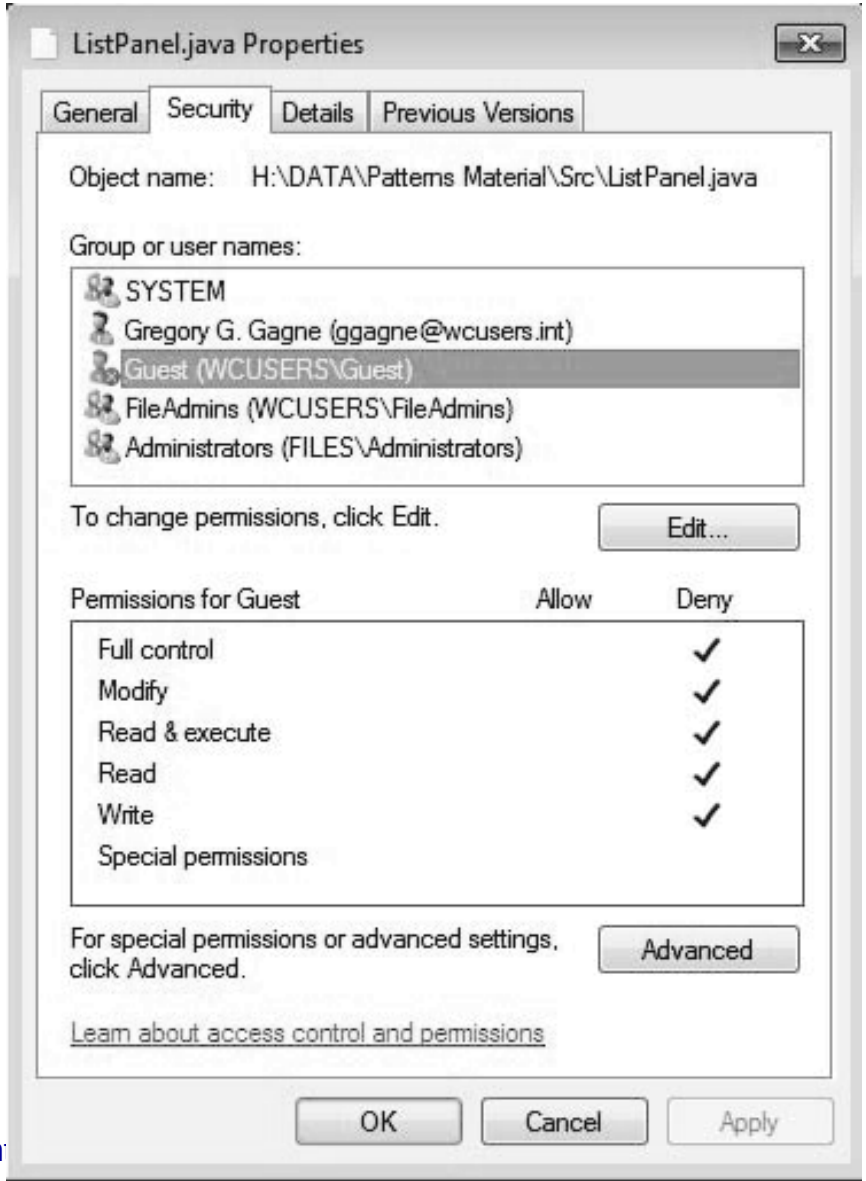
Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users on Unix / Linux

			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.

Windows 7 Access-Control List Management



A Sample UNIX Directory Listing

-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4 pbg	faculty	512	Jul 31 10:31	lib/
drwx------	3 pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3 pbg	staff	512	Jul 8 09:35	test/

Summary

- The major task of OS is to map the logical file concept onto physical storage device
- 6 basic operations on a file and some other optional and helpful operations
- Directory structure and various operations
- OS provides file system mounting
- File protection and access rights

References

Part of the contents of this lecture has been adapted from the book Abraham Silberschatz, Peter B. Galvin, Greg Gagne: "Operating System Concept ", Publisher : Wiley; 9 edition (December 17, 2012), ISBN-13: 978-1118063330

Slides also contain lecture materials from John Kubiatowicz (Berkeley), John Ousterhout (Stanford), Nalini (UCI), Rainer (UCI), and others

Some slides adapted from http://www-inst.eecs.berkeley.edu/~cs162/ Copyright © 2010 UCB

Thank you for your attention