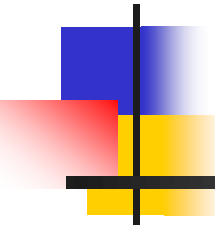
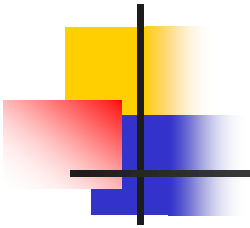
# 操作系统原理及应用

李　伟
xchlw@seu.edu.cn

计算机科学与工程学院、软件学院
江苏省网络与信息安全重点实验室

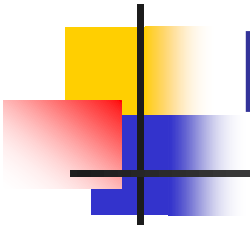


# Chapter 10 File System Interface



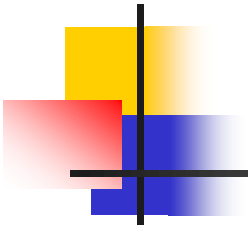
# Outline

- File Concept
- Access Methods
- Directory Structure
- File System Mounting
- File Sharing
- Protection



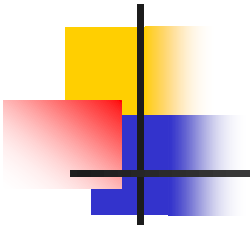
# File Concept

- A file is a named collection of related information that is recorded on secondary storage.
- Contiguous logical address space
- Types
  - Data: numeric, character, binary
  - Program



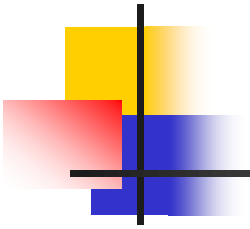
# File Structure

- None - sequence of words, bytes
- Simple record structure
  - Lines
  - Fixed length
  - Variable length
- Complex Structures
  - Formatted document
  - Relocatable load file



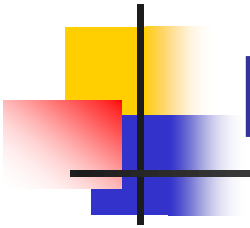
# File Attributes

- **Name – only information kept in human-readable form.**
- **Identifier – the non-human-readable name for the file.**
- **Type – needed for systems that support different types.**
- **Location – pointer to file location on device.**
- **Size – current file size.**



# File Attributes

- **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

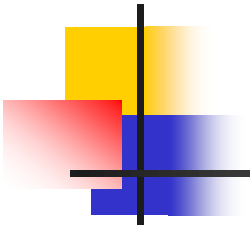
- Create
- Write
- Read
- Reposition within file – file seek
- Delete
- Truncate（截短）— erase the contents of a file but keep its attributes.

# File Operations (Cont.)

- Open($F_i$) – search the directory structure on disk for entry $F_i$ and copy the content of entry into the open file table.
- Close ($F_i$) – remove the content of entry $F_i$ from the open file table.
- Two levels of internal tables
  - A per-process open file table
  - A system-wide open file table

# File Types

| 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 |

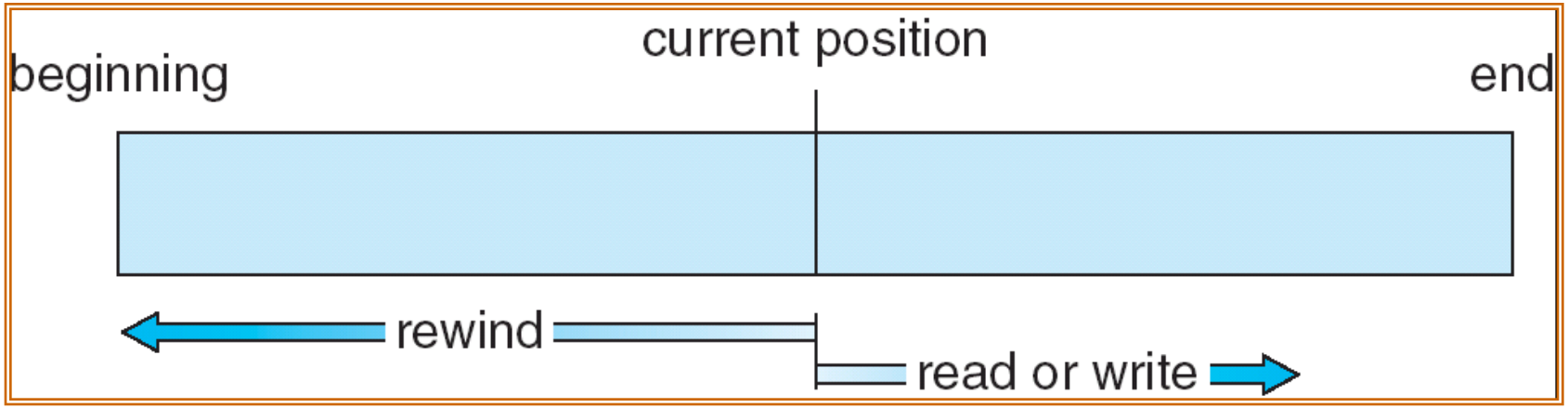


# Outline

- File Concept
- Access Methods
- Directory Structure
- File System Mounting
- File Sharing
- Protection

# Access Methods

- **Sequential Access**

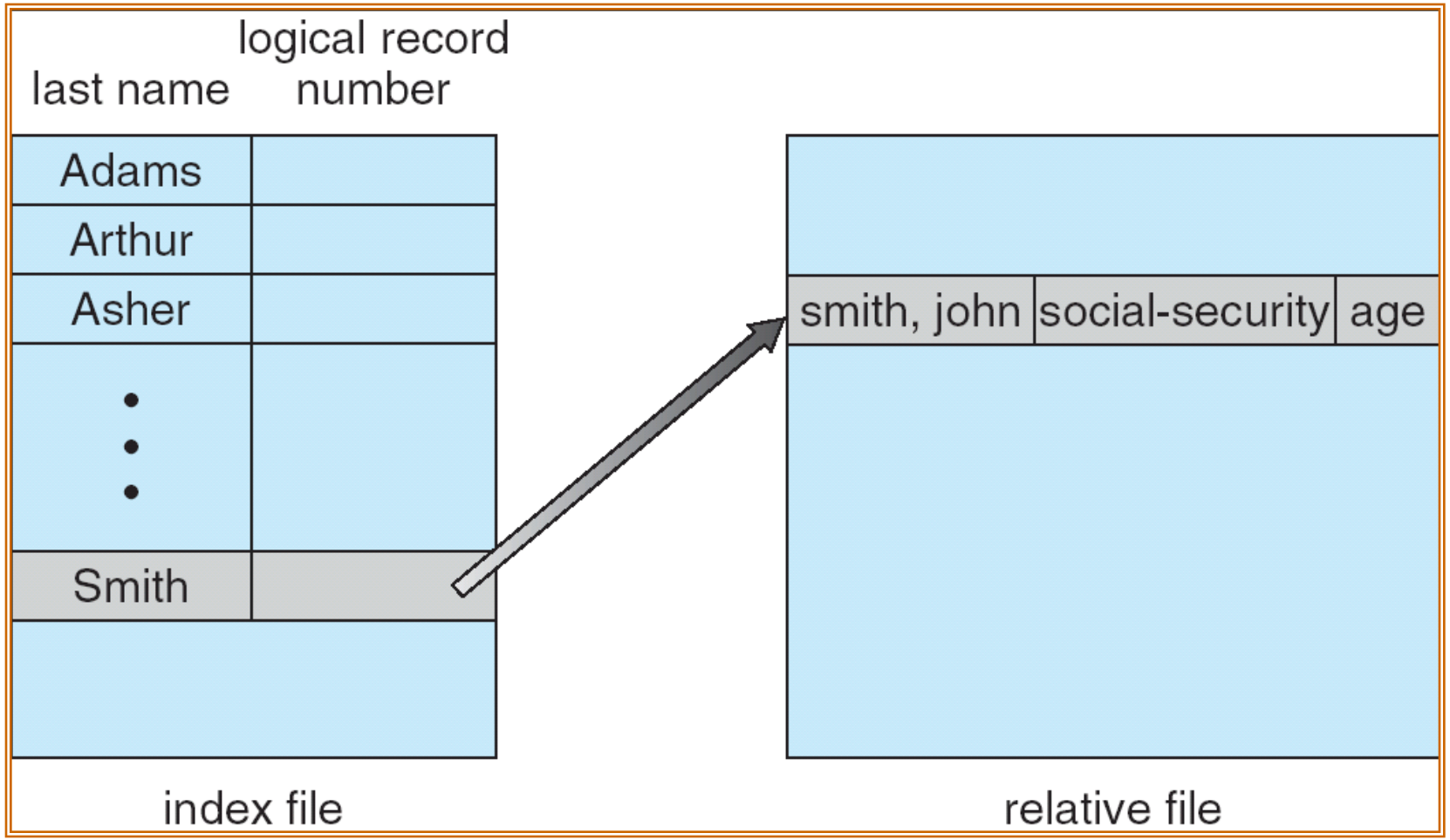


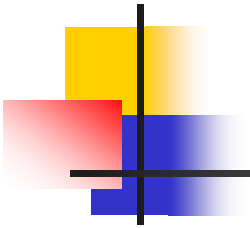

- **Direct Access**

# Simulation of Sequential Access on a Direct-access File

| sequential access | implementation for direct access |
|---|---|
| *reset* | $cp = 0;$ |
| *read next* | *read cp*;<br>$cp = cp + 1;$ |
| *write next* | *write cp*;<br>$cp = cp + 1;$ |

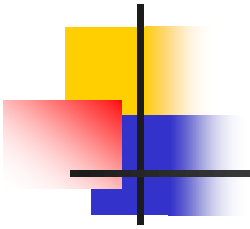
# Example of Index and Relative Files





# Outline

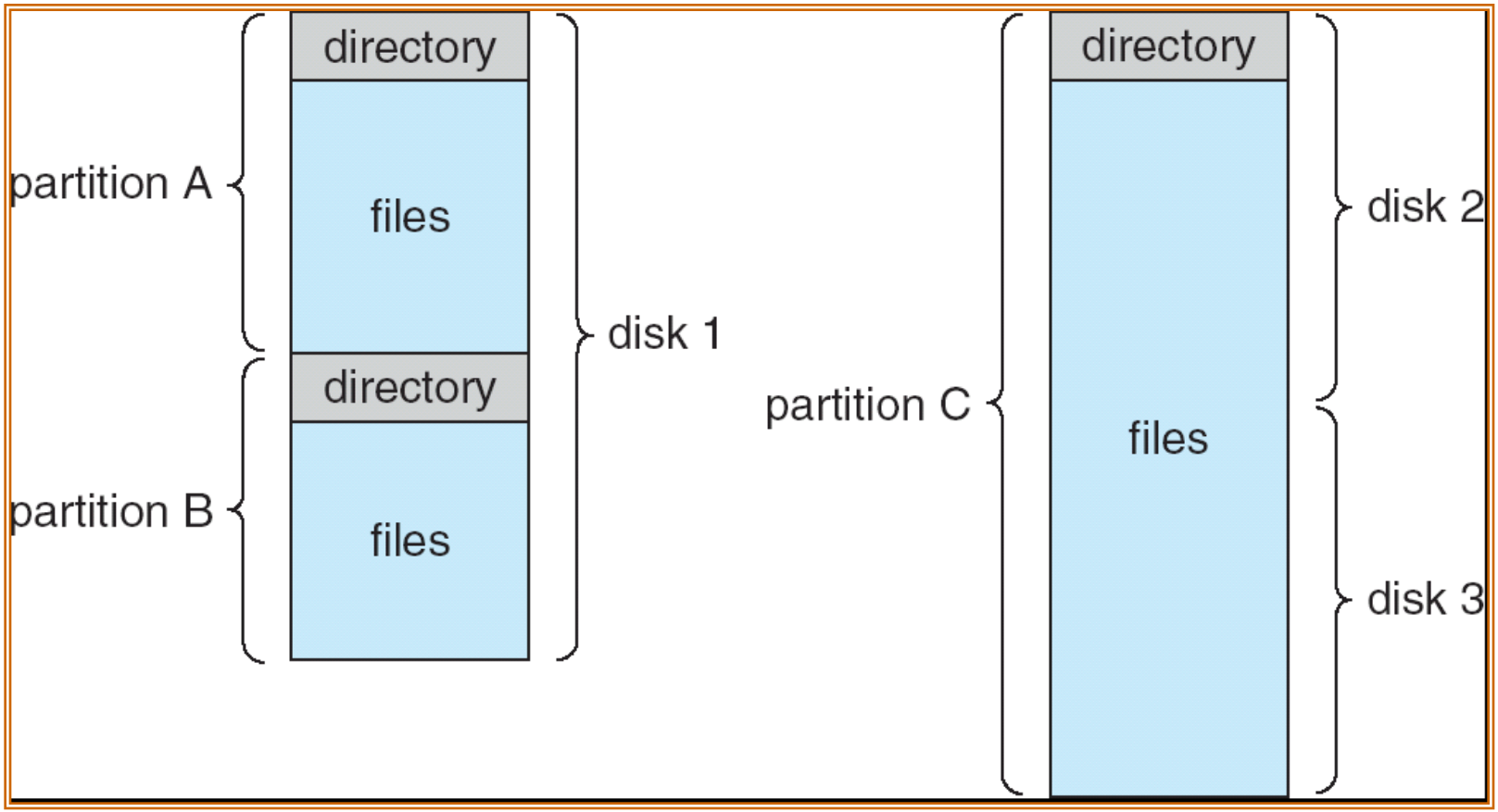
- File Concept
- Access Methods
- Directory Structure
- File System Mounting
- File Sharing
- Protection



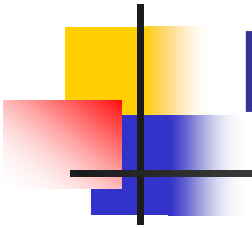
# Directory Structure

- Disks are split into one or more partitions.
- Each partition contains information about files within it.
- The information is kept in entries in a device directory or volume table of contents

# A Typical File-system Organization

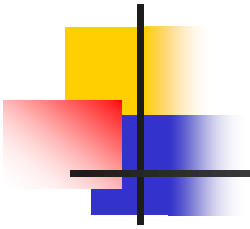


# Operations Performed on Directory



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

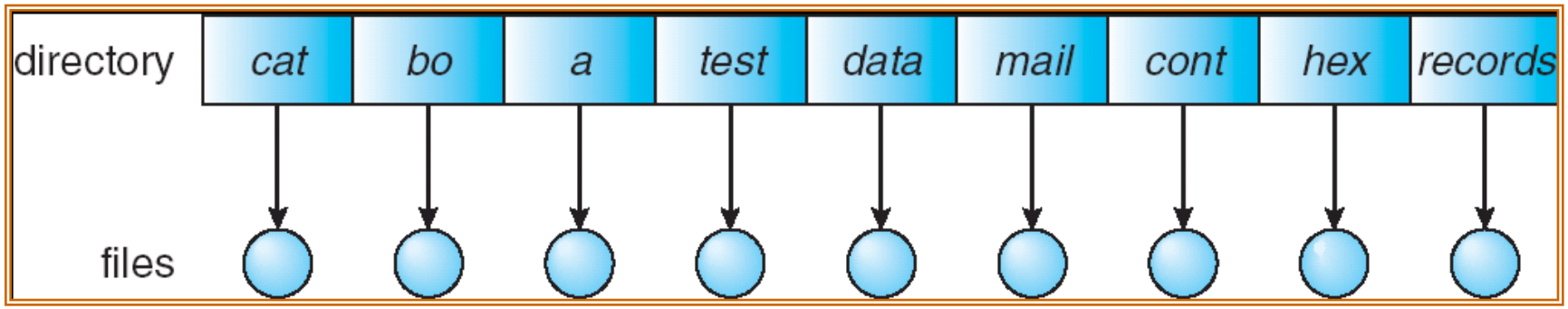
# Organize the Directory (Logically) to Obtain



- **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

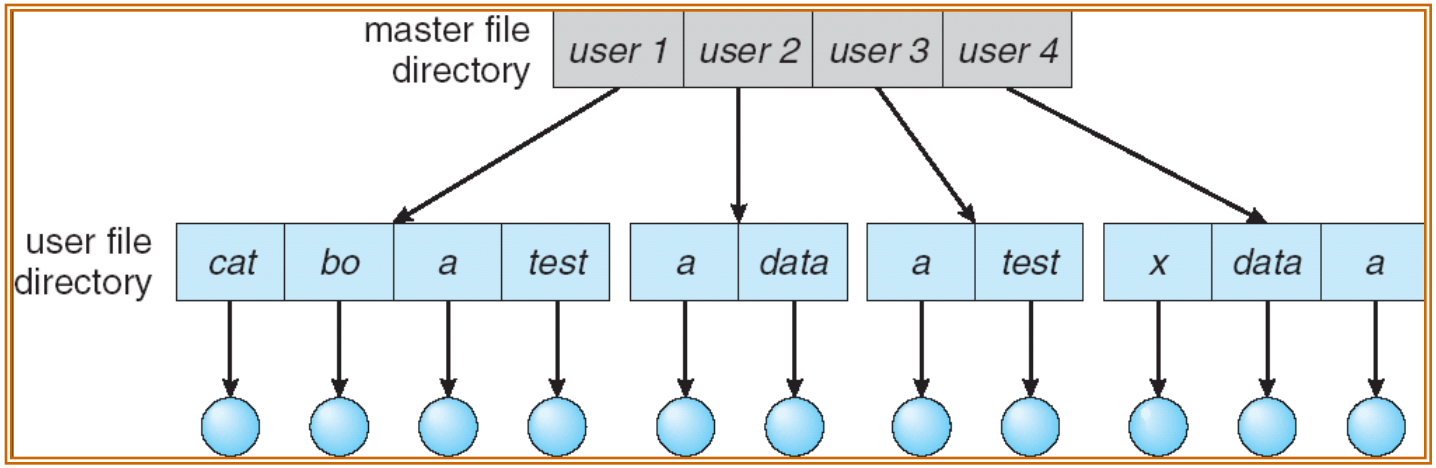



Naming problem

Grouping problem

# Two-Level Directory

- **Separate directory for each user**



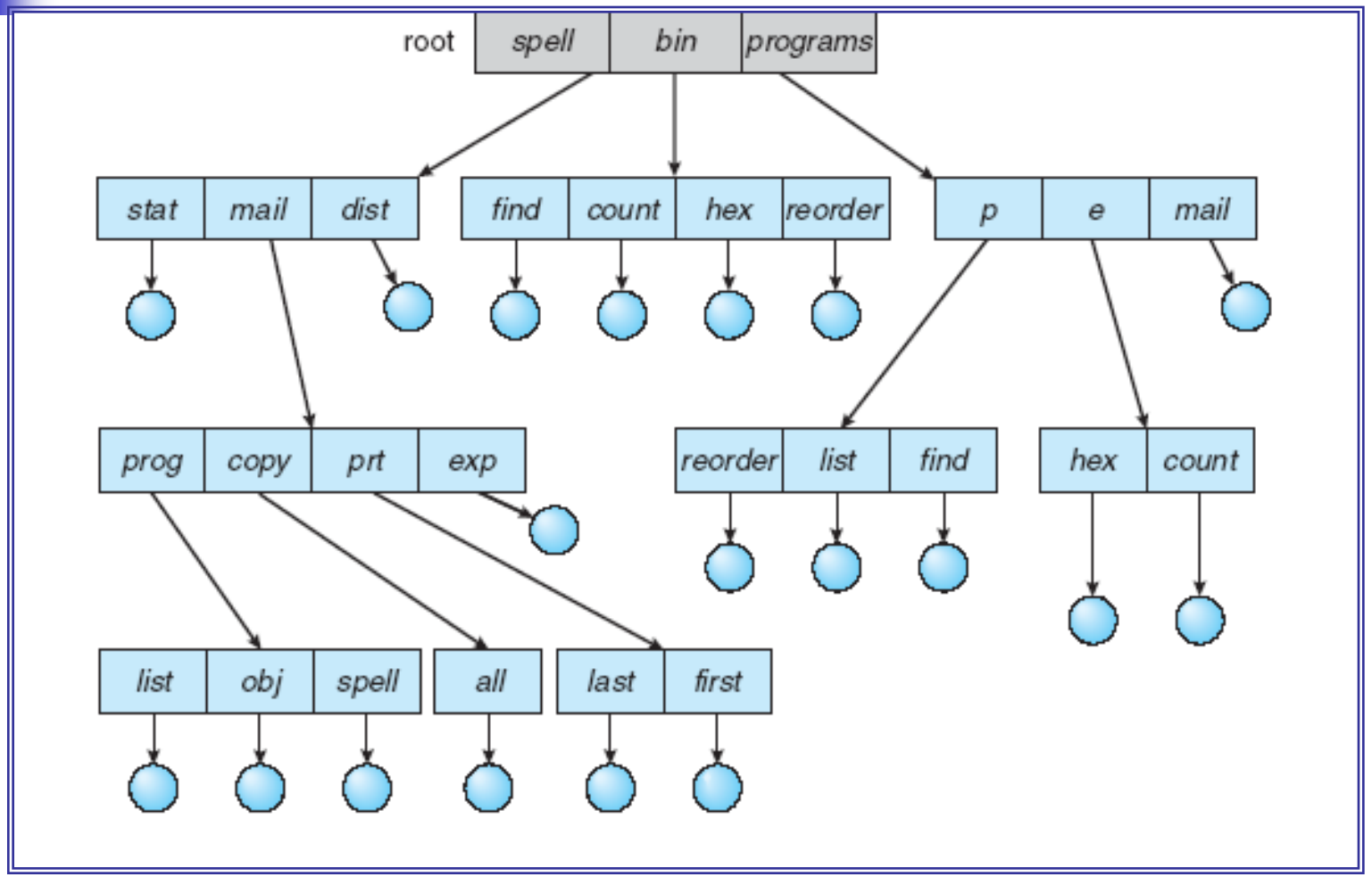

**Path name**

**Can have the same file name for different user**

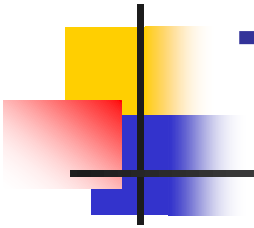
**Efficient searching**

**No grouping capability**

# Tree-Structured Directories



# Tree-Structured Directories



- Efficient searching
- Grouping Capability
- Current directory (working directory)
  - cd /spell/mail/prog
- Absolute or relative path name
- Creating a new file is done in current directory
- Delete a file
  - rm <file-name>

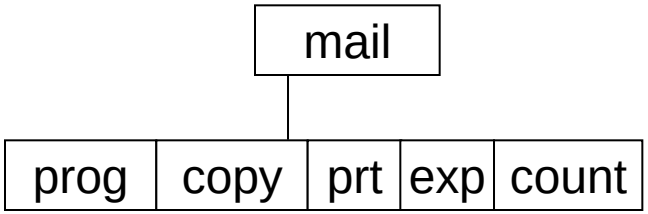
# Tree-Structured Directories

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

  **mkdir <dir-name>**

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

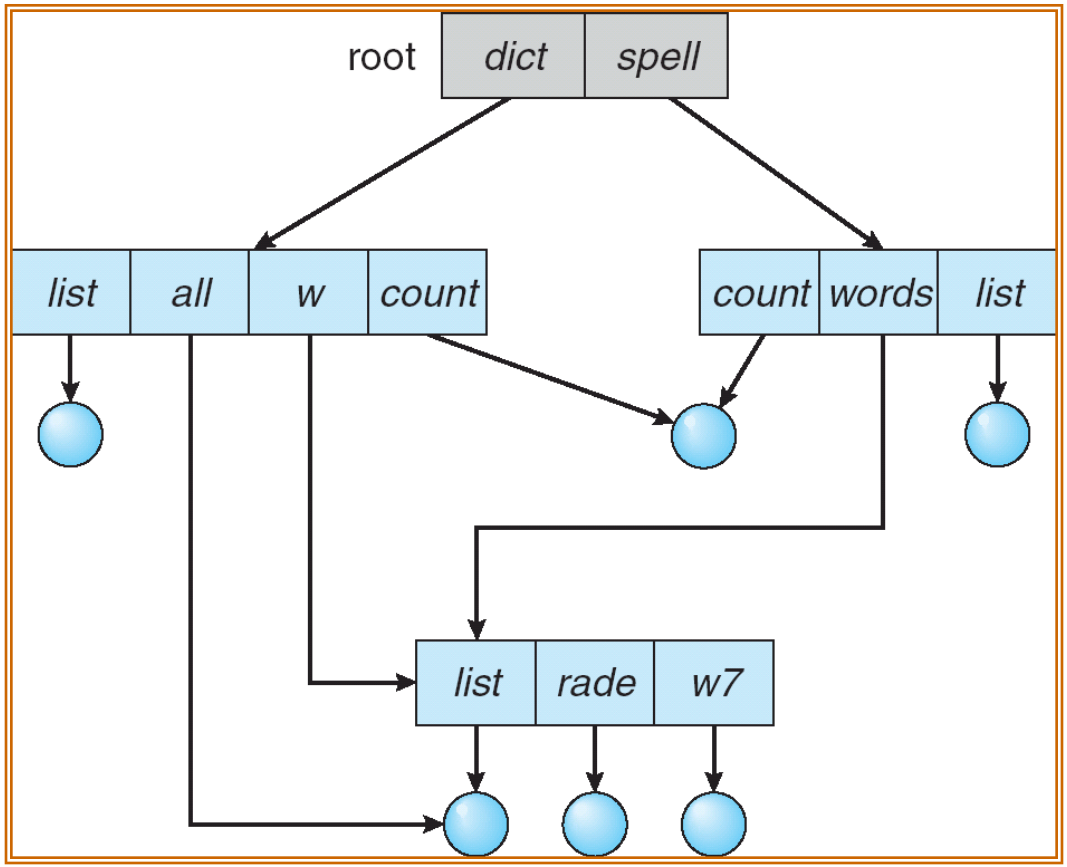
  **mkdir count**

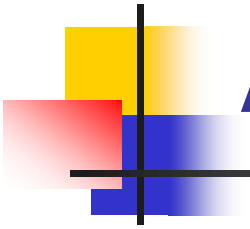



**Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”.**

# Acyclic-Graph Directories

- Have shared subdirectories and files.

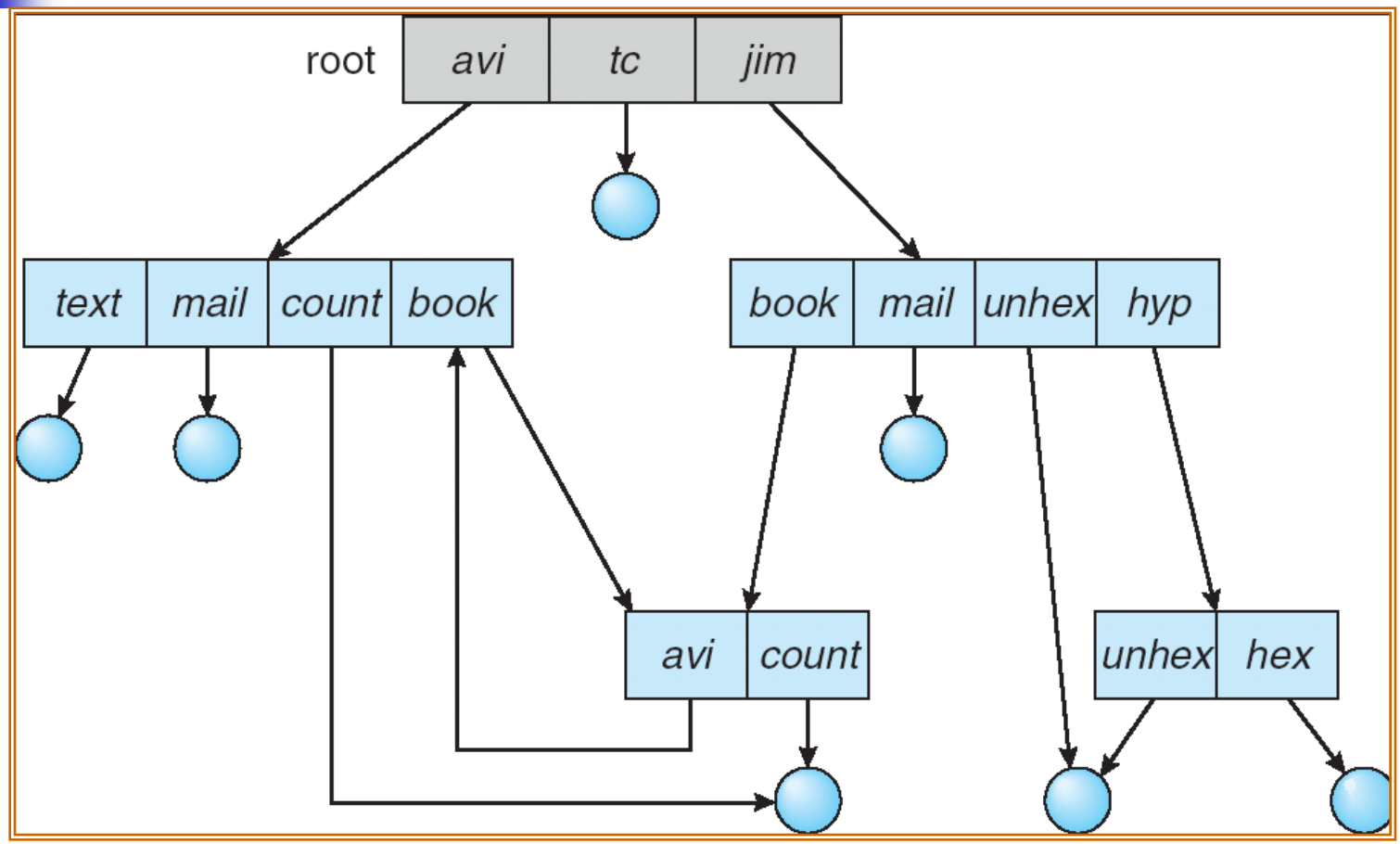


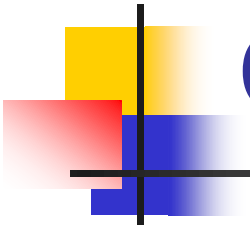


# Acyclic-Graph Directories

- Two different names (aliasing)
- If *dict* deletes *count* $\Rightarrow$ dangling pointer.
- Solutions:
  - Backpointers, so we can delete all pointers.
  - Entry-hold-count solution.

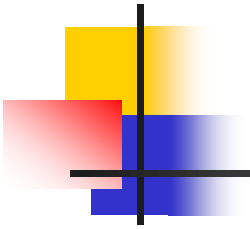
# General Graph Directory





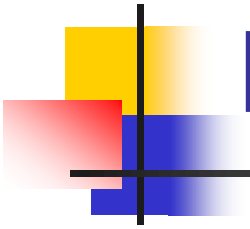
# General Graph Directory

- 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.



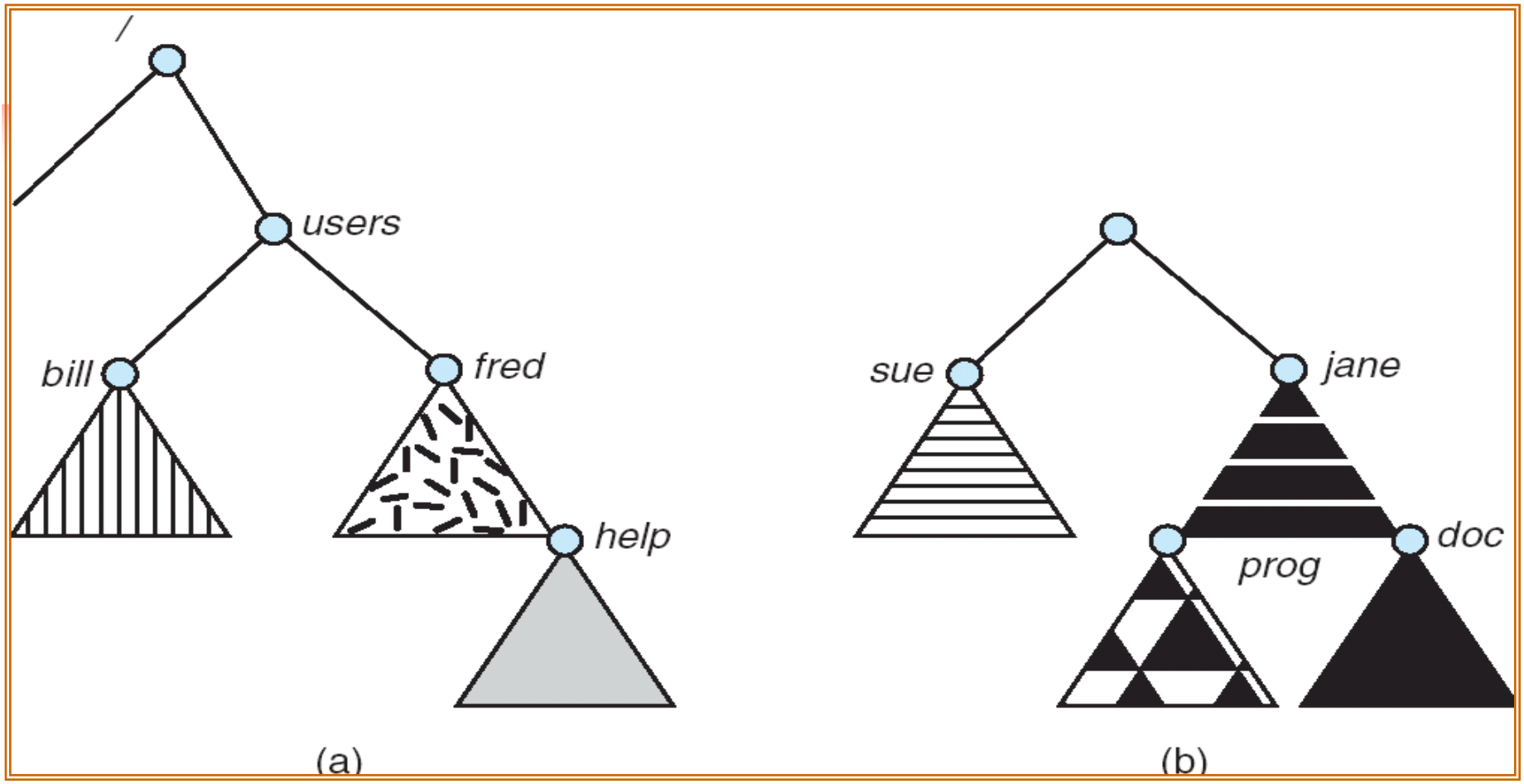
# Outline

- File Concept
- Access Methods
- Directory Structure
- File System Mounting
- File Sharing
- Protection



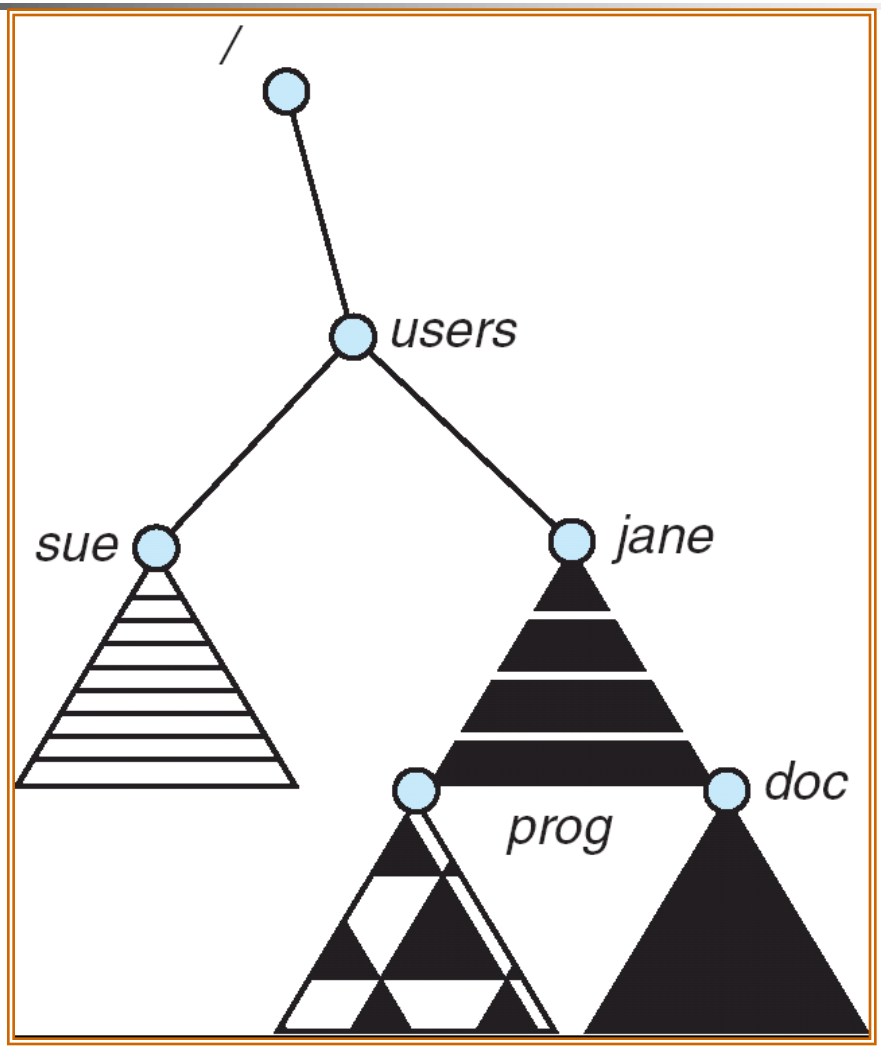
# File System Mounting

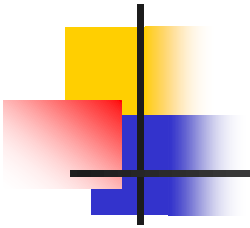
- A file system must be mounted before it can be accessed.
- An unmounted file system is mounted at a mount point.




**(a) Existing** **(b) Unmounted Partition**

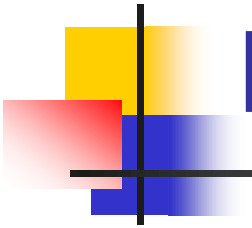
# Mount Point





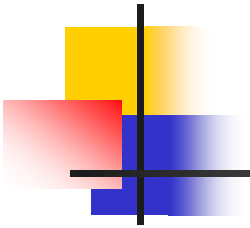
# Outline

- File Concept
- Access Methods
- Directory Structure
- File System Mounting
- File Sharing
- Protection



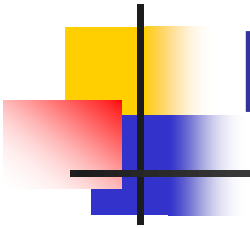
# File Sharing

- **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.**



# Outline

- File Concept
- Access Methods
- Directory Structure
- File System Mounting
- File Sharing
- Protection



# 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**

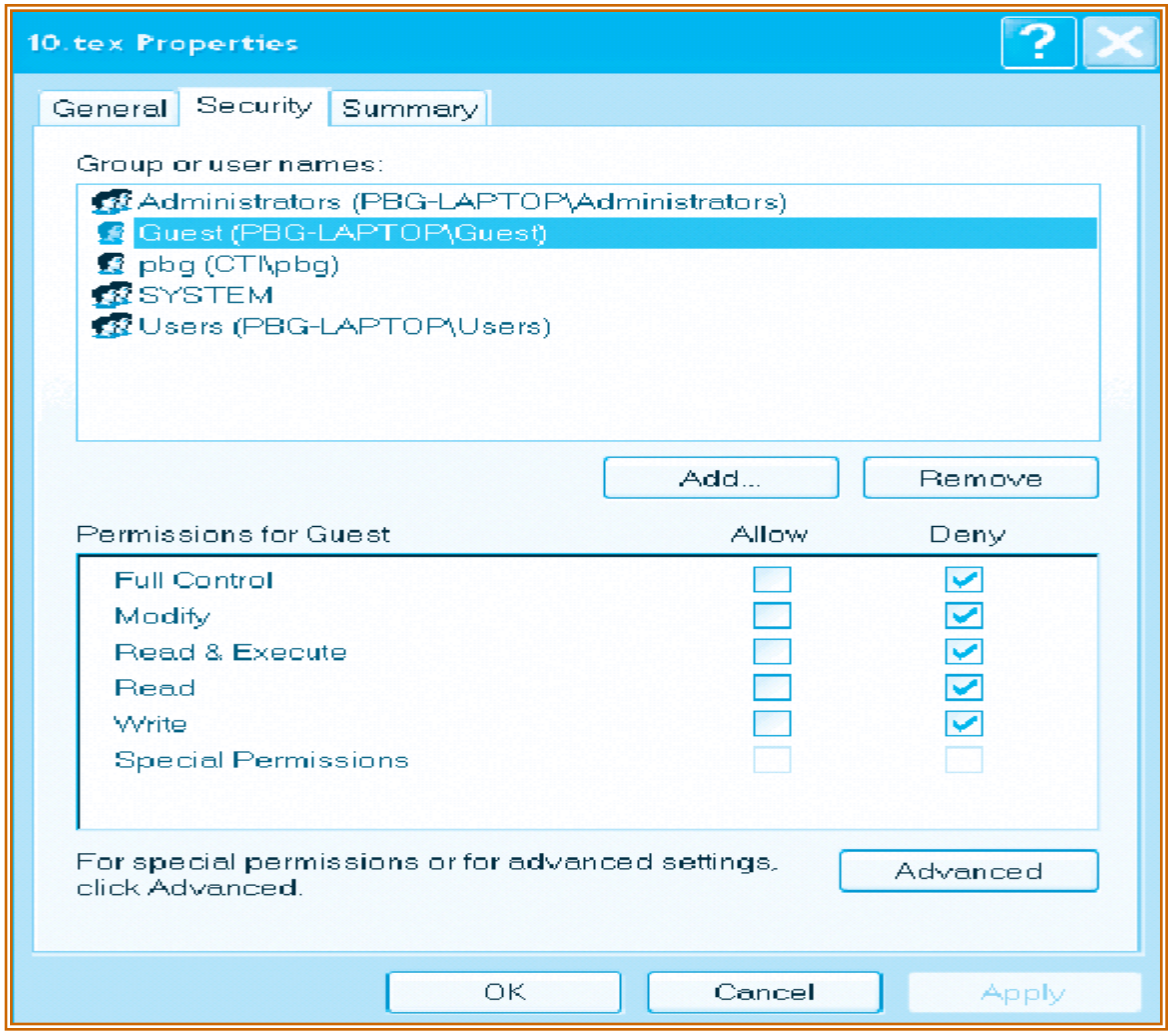
  **a) owner access 7 ⇒ 1 1 1**

  **b) group access 6 ⇒ 1 1 0**

  **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 XP 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/ |