Chapter 10: File-System Interface

Chapter 10: File-System Interface

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

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 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
- 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
- **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

File Operations

- File is an **abstract data type**
- **Create**
- **Write**
- **Read**
- **Reposition within file**
- **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

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

扩展知识

1. **文件扩展名是一个文件的必要构成部分。**

   任何显示文件扩展名一个文件可以有或没有扩展名。对于打开文件操作，没有扩展名的文件需要选择程序去打开它，有扩展名的文件会自动用设置好的程序（如有）去尝试打开。

2. **文件扩展名表明了该文件是何种类型。**

   文件扩展名可以人为设定，扩展名为TXT的文件有可能是一张图片，同样，扩展名为MP3的文件，依然可能是一个视频。

扩展知识

3. 文件扩展名大小写的区别

在一个文件中的扩展名的大小写，系统的大小写是不予区别的。

4.文件扩展名的漏洞

有些木马文件(可运行的,扩展名为exe)会伪装成图片文件或其他的文件.

hack.jpg.exe.它的图标也是jpg图片的图标,如果你选择了隐藏文件扩展名,那显示为hack.jpg,且图标是图片的,那你就很容易上当,双击它的话,就是运行了一个木马程序.有些更毒的,它还绑定了图片,双击这类文件时,会出现一个图片,但木马程序已悄悄地在后台运行安装了,而你又以为真的是一张图片而已.

扩展知识

1. **扩展名对文件类别的知识仅仅是指示性的，并不具有强制性。**

   UNIX下，扩展名仅仅提醒用户，系统并不遵守，只要是一个文件是可执行文件，即使其扩展名不是.exe，该文件也能在UNIX下运行。但有些系统却对扩展名进行了强制服从，否则无法使用。例如Windows

Access Methods

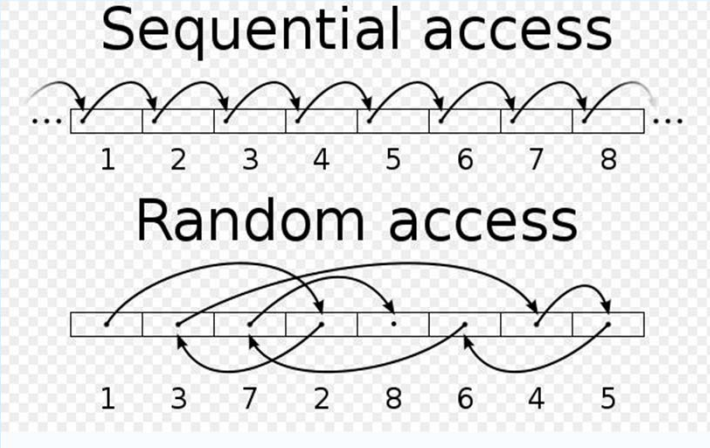
- **Sequential Access**

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

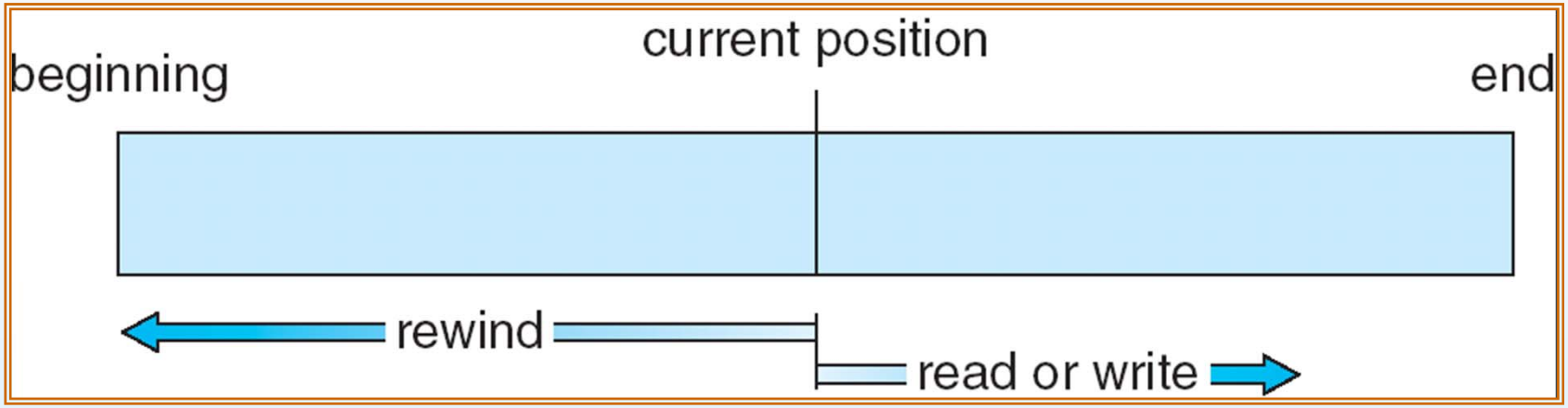
- **Direct Access**

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

  n = relative block number



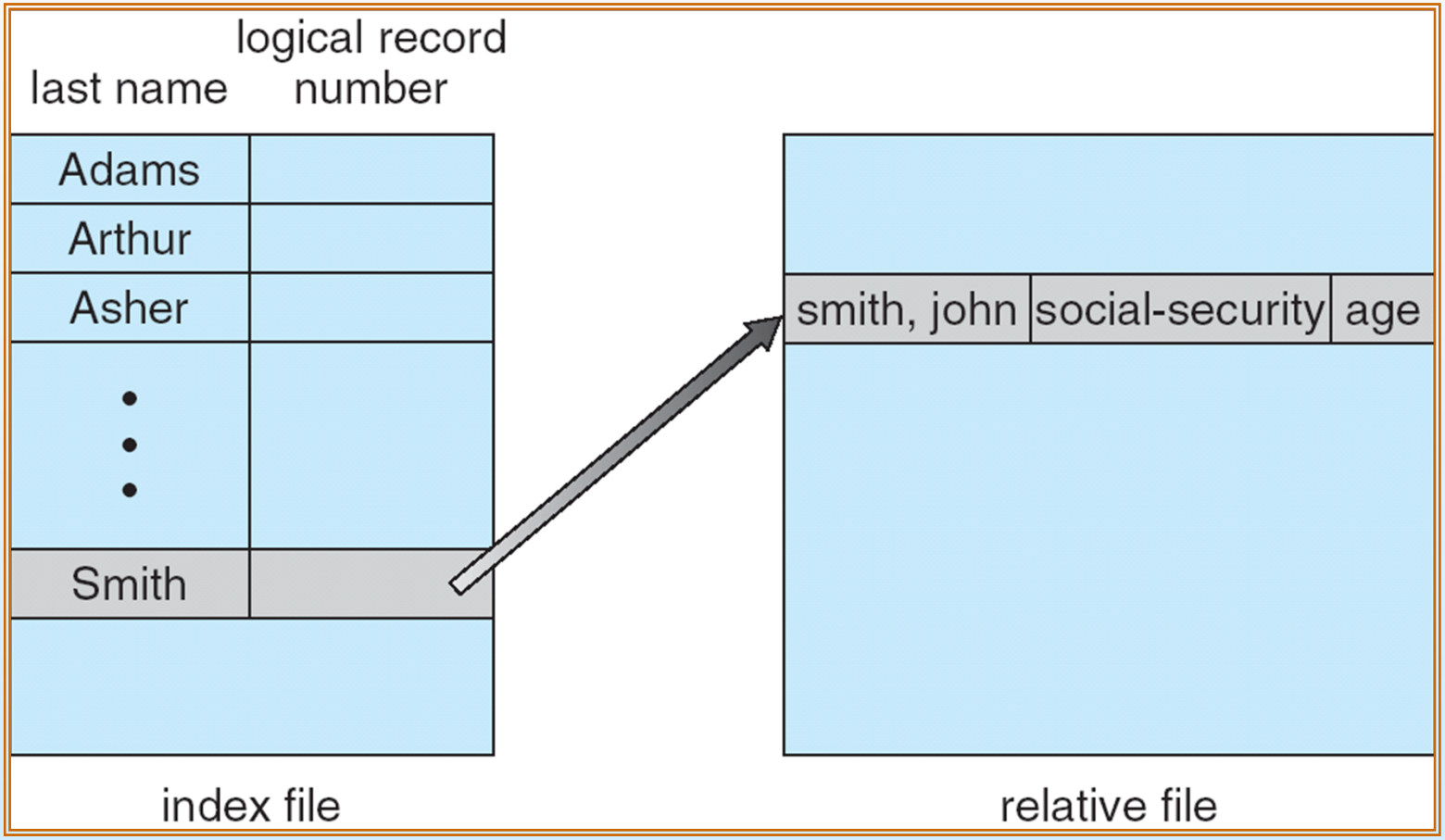
Sequential-access File



Simulation of Sequential Access on a Direct-access File

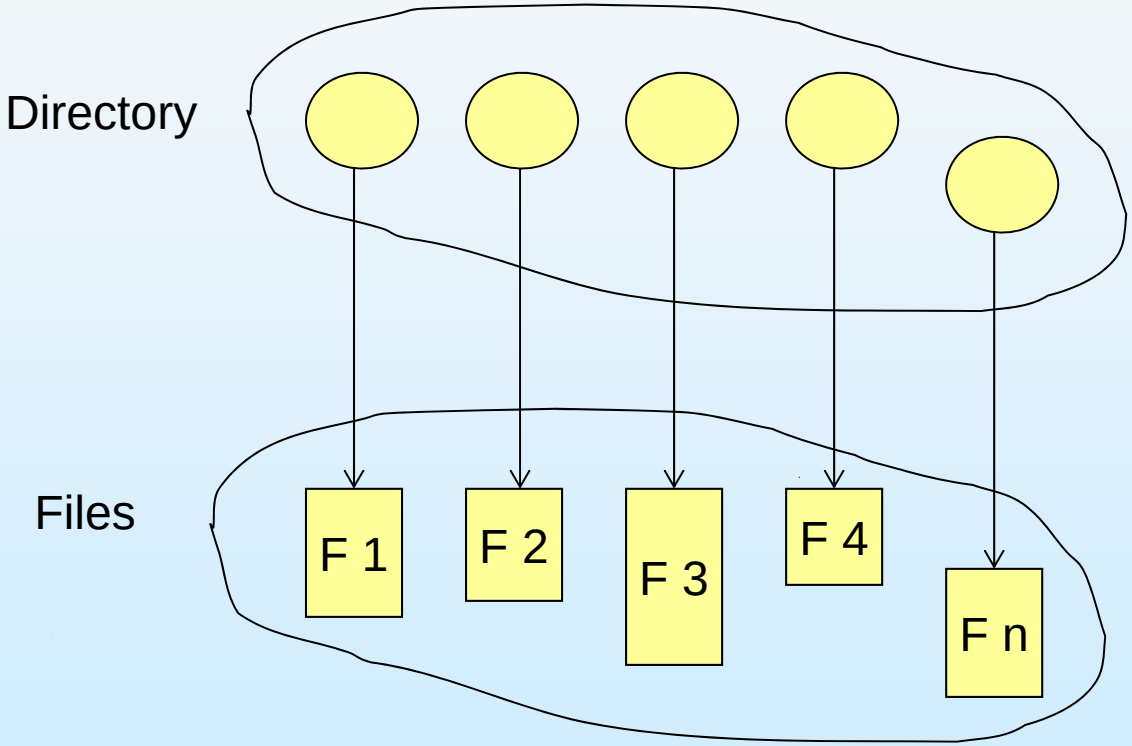
sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

Example of Index and Relative Files



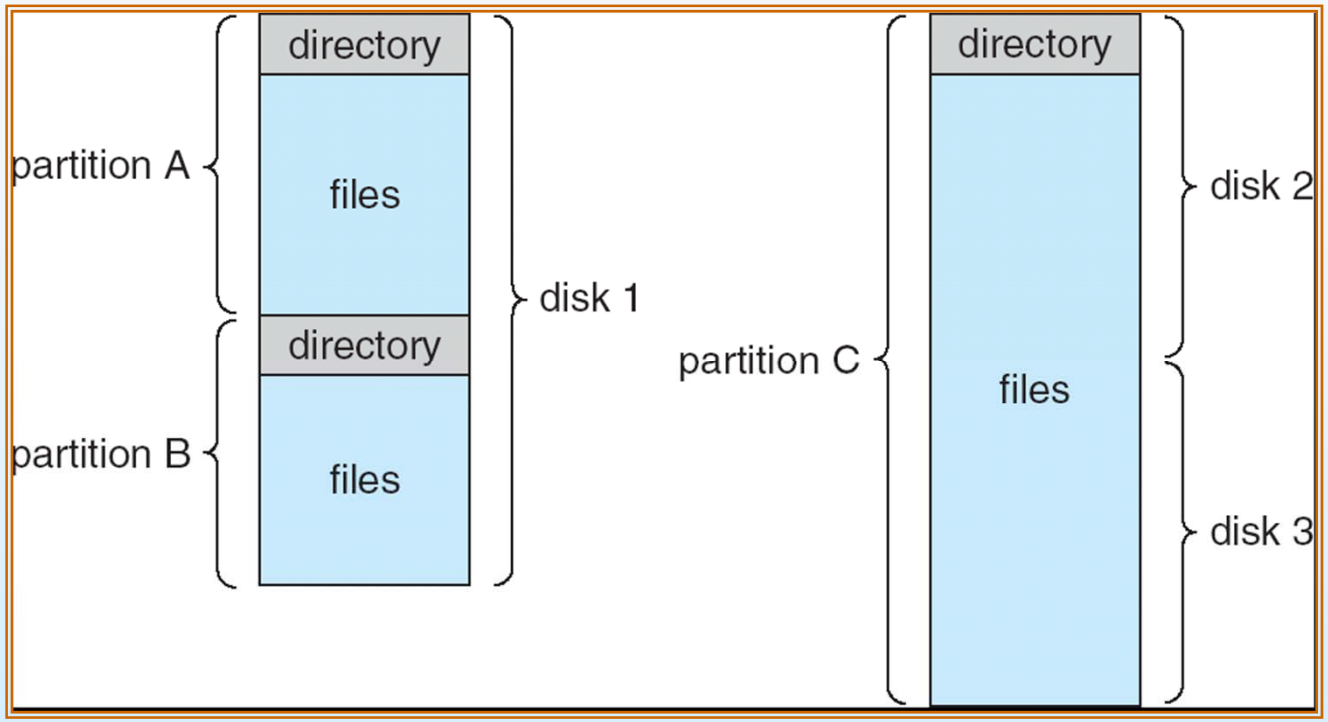
Directory Structure

- 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

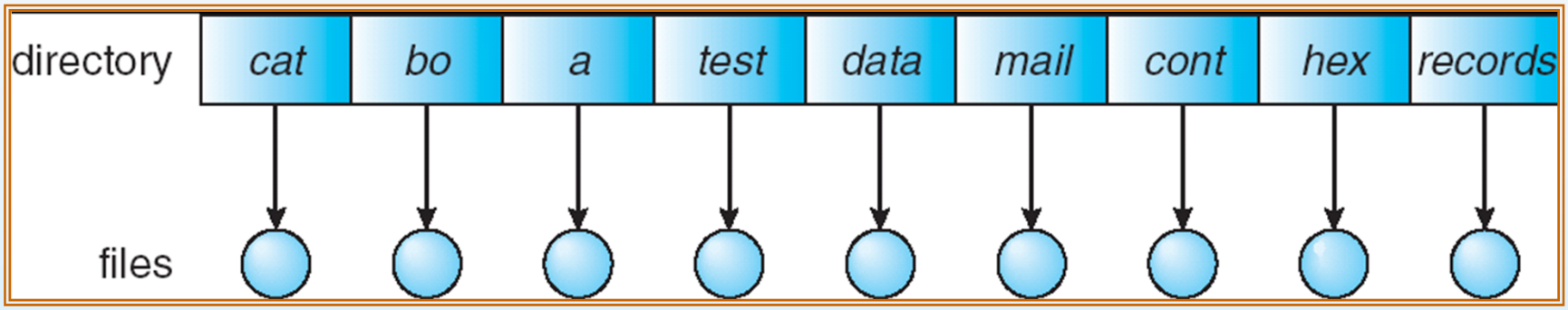
Organize the Directory (Logically) to Obtain

- 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

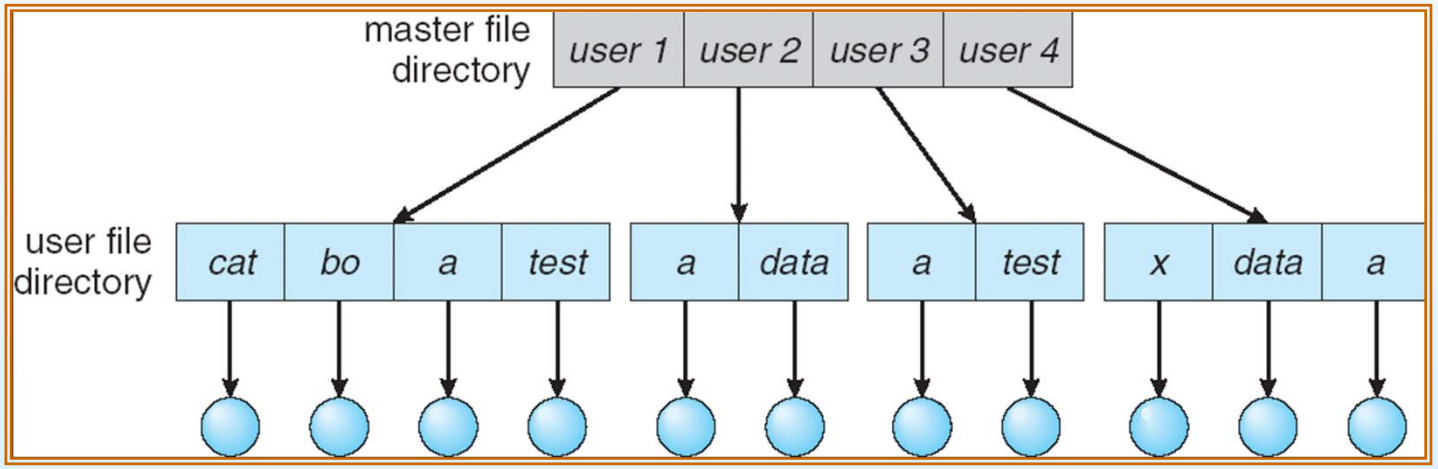


Naming problem

Grouping problem

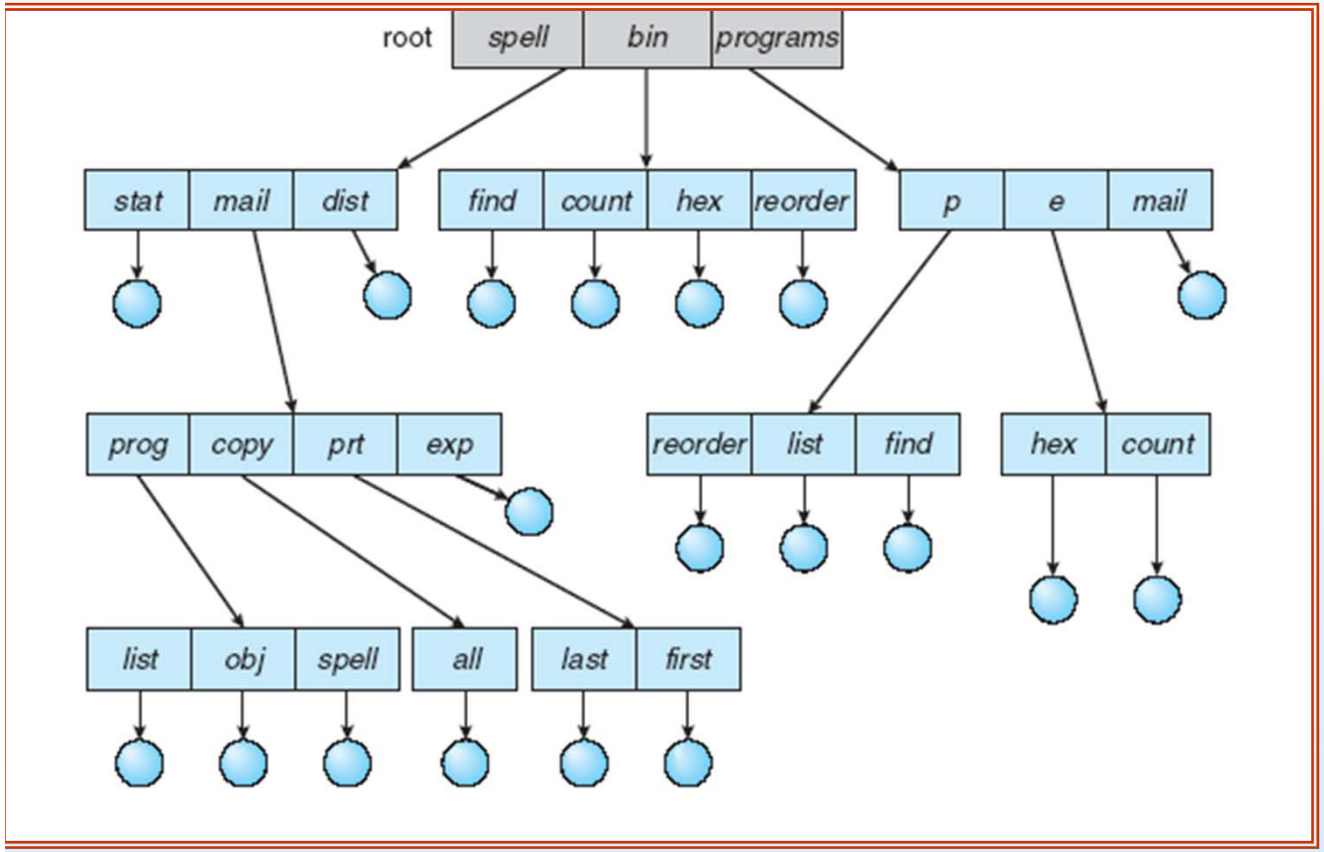
Two-Level Directory

- **Separate directory for each user**



- Path name
- Can have the same file name for different user
- Efficient searching

Tree-Structured Directories



Tree-Structured Directories (Cont)

- Efficient searching
- Grouping Capability
- Current directory (working directory)

  cd /spell/mail/prog

  type list

http://disk.baidu.com/

http://desktop.google.com/zh/

Tree-Structured Directories (Cont)

Absolute or **relative** path name

Creating a new file is done in current directory

Delete a file

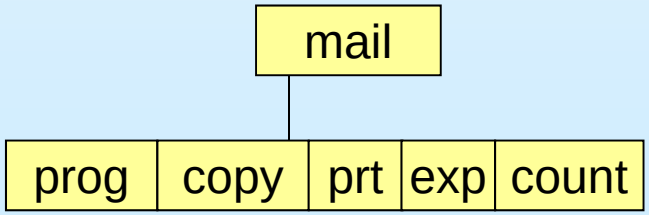
rm <file-name>

Creating a new subdirectory is done in current directory

mkdir <dir-name>

Example: if in current directory /mail

mkdir count

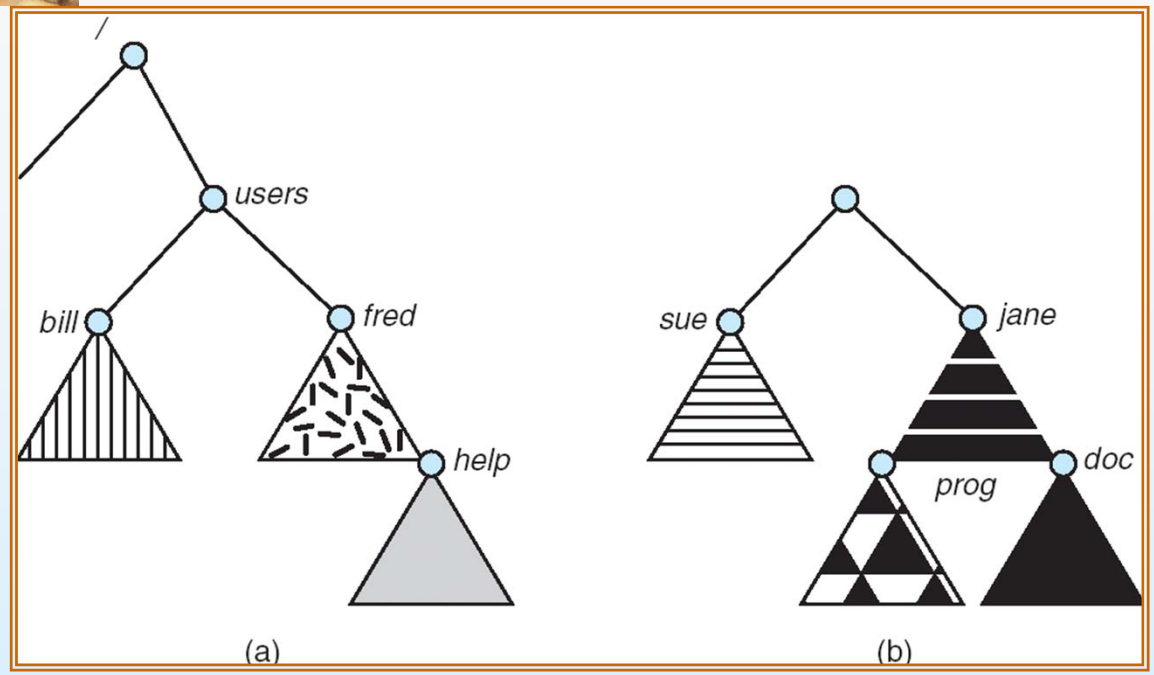


Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”

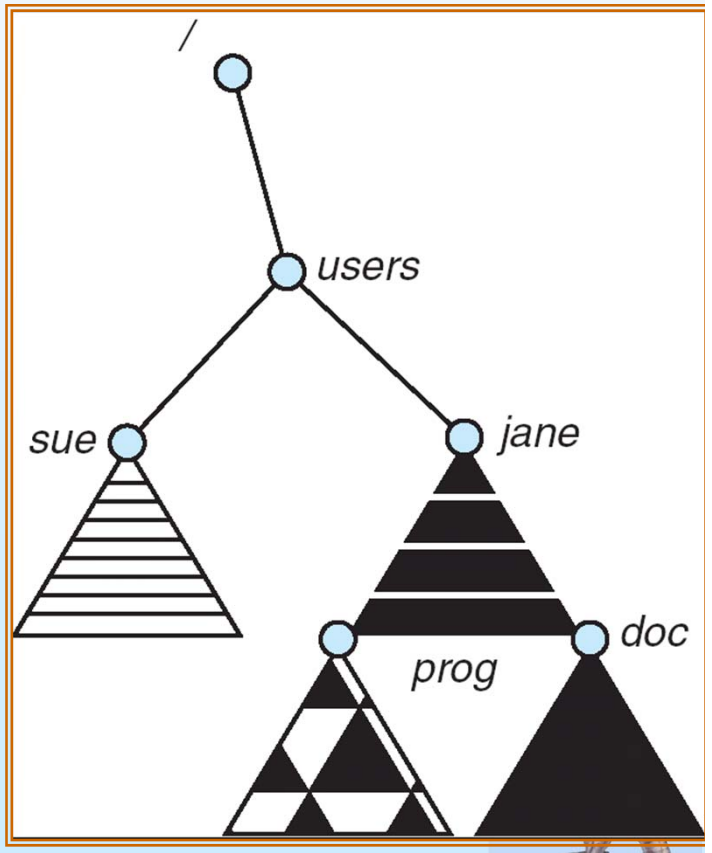
File System Mounting

- 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**

(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

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

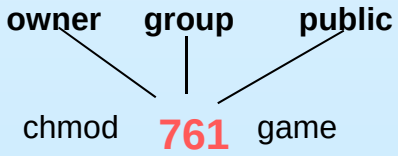
Mode of access: read, write, execute

Three classes of users

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

Ask manager to create a group (unique name), say G, and add some users to the group.

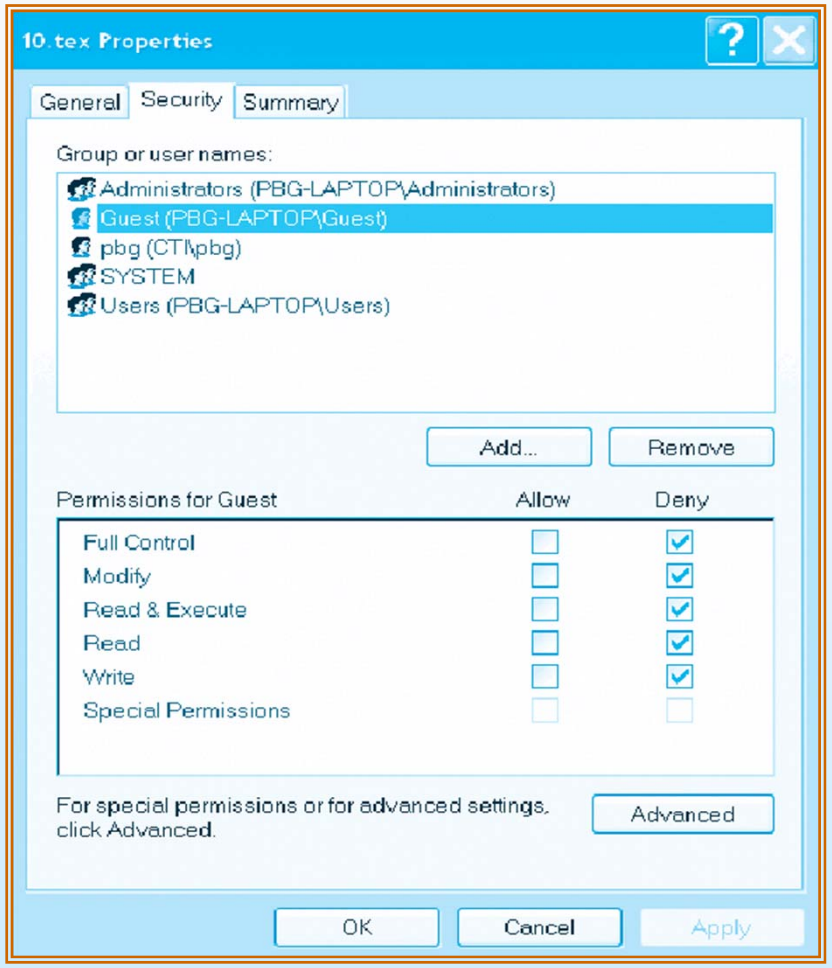
For a particular file (say *game*) or subdirectory, define an appropriate access.



Attach a group to a file

chgrp G game

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/

End of Chapter 10