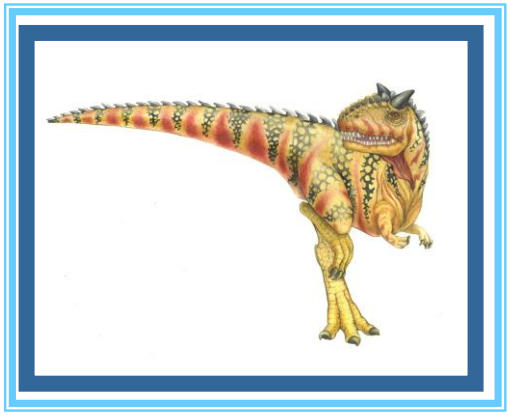
Chapter 10
File-System



Chapter 10: File System

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 methods with first method by inserting appropriate control characters (CR, LF)

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 process closes it
- **Disk location of the file**: cache of data access information
- **Access rights**: per-process access mode information

Open File Locking

Provided by some operating systems and file systems

Shared Lock: several processes can acquire the lock concurrently (like a reader lock)

Exclusive Lock: Only one process at a time can acquire such a lock (like a writer lock)

Mandatory or advisory file locking mechanisms:

Mandatory – Once a process acquires an exclusive lock, the OS will prevent any other process from accessing the locked file. (Windows)

Advisory – The OS will not prevent a process from acquiring access to a locked file. Rather, the process must be written so that it manually acquiring the lock before accessing the file. (UNIX)

File Locking Example – Java API

```
import java.io.*;
import java.nio.channels.*;
public class LockingExample {
    public static final boolean EXCLUSIVE = false;
    public static final boolean SHARED = true;
    public static void main(String arsg[]) throws IOException {
        FileLock sharedLock = null;
        FileLock exclusiveLock = null;
        try {
            RandomAccessFile raf = new RandomAccessFile("file.txt", "rw");
            // get the channel for the file
            FileChannel ch = raf.getChannel();
            // this locks the first half of the file - exclusive
            exclusiveLock = ch.lock(0, raf.length()/2, EXCLUSIVE);

            /** Now modify the data . . . */

            // release the lock
            exclusiveLock.release();
```

File Locking Example – Java API (cont)

```
            // this locks the second half of the file - shared
            sharedLock = ch.lock(raf.length()/2+1, raf.length(), SHARED);

            /** Now read the data . . . */

            // release the lock
            sharedLock.release();
        } catch (java.io.IOException ioe) {
            System.err.println(ioe);
        }finally {
            if (exclusiveLock != null)
            exclusiveLock.release();
            if (sharedLock != null)
            sharedLock.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

Access Methods

Sequential Access

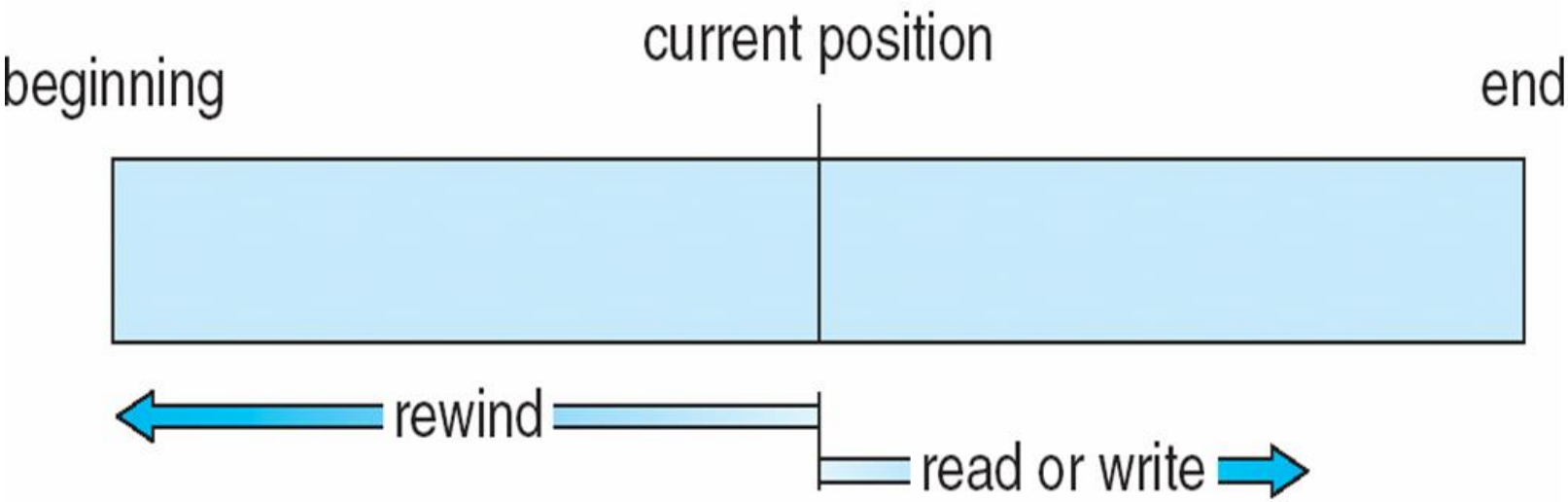
read next
write next
reset

Direct Access

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

n = relative block number

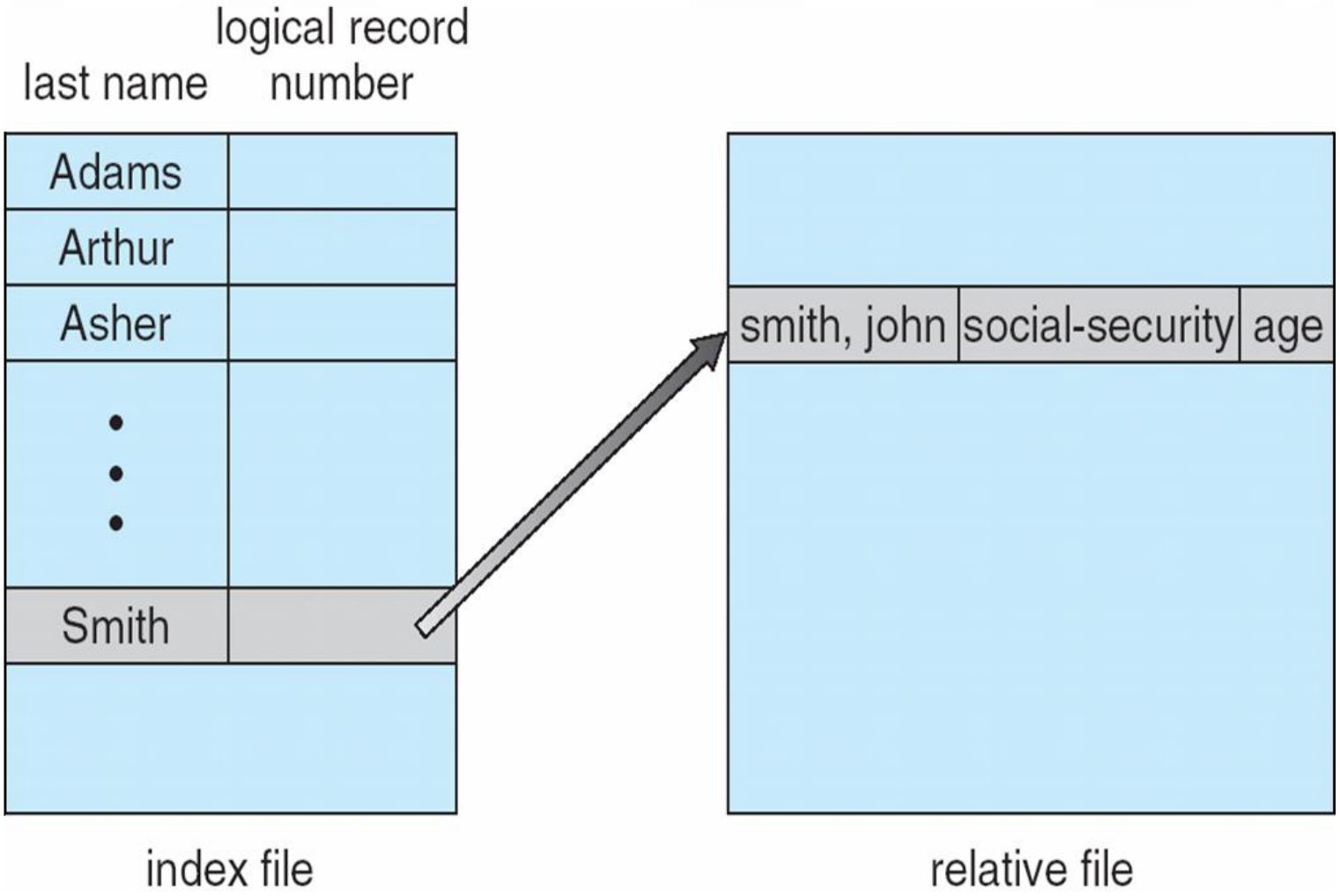
Sequential-access File



Simulation of Sequential Access on Direct-access File

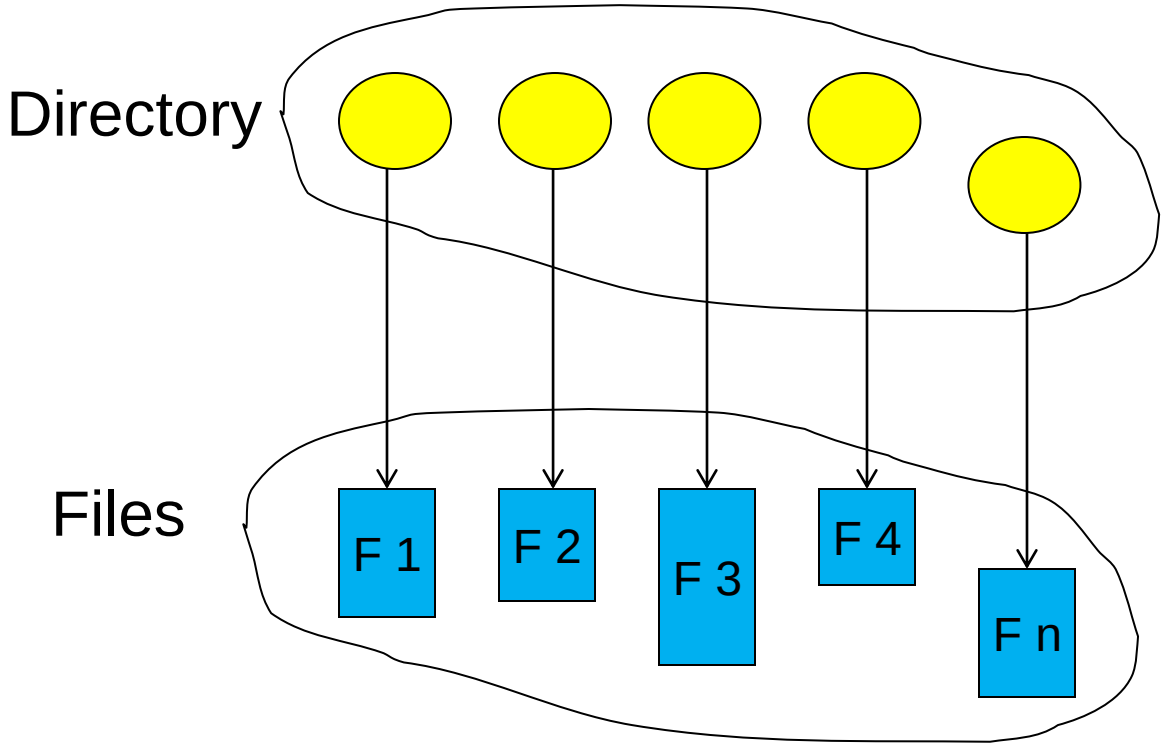
sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

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

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

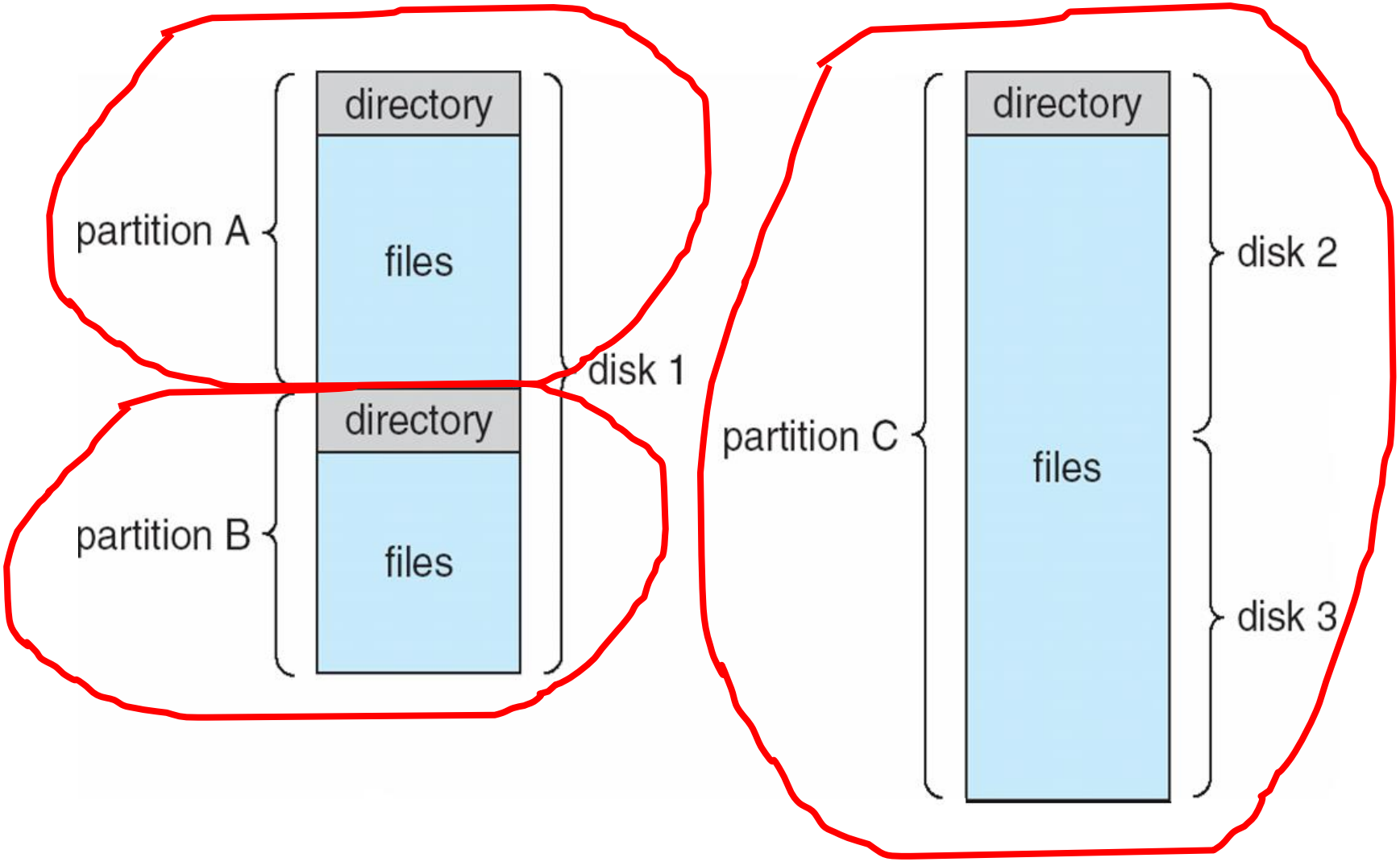
Partitions also known as minidisks, slices

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 (Solaris)

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

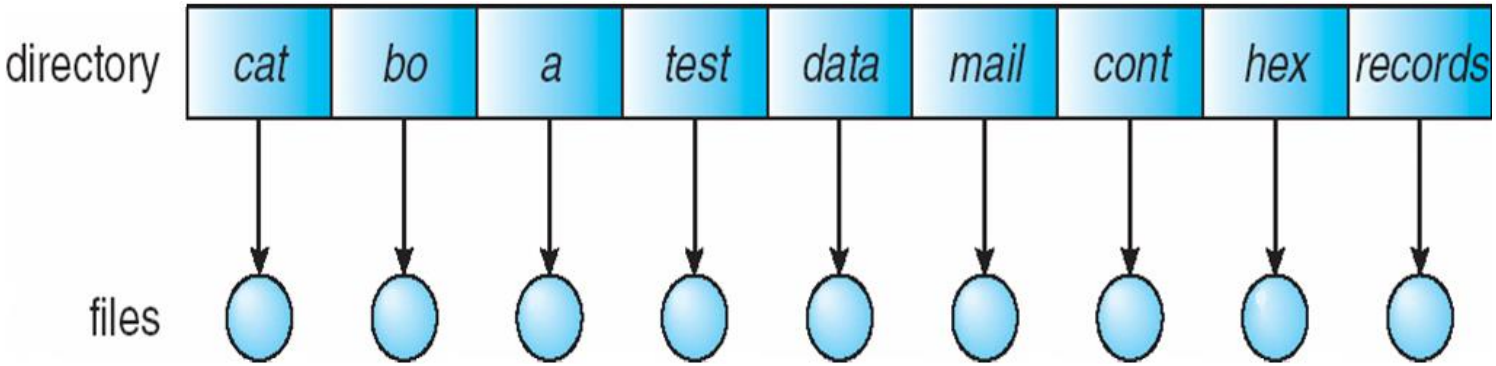
Naming – convenient to users

- 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

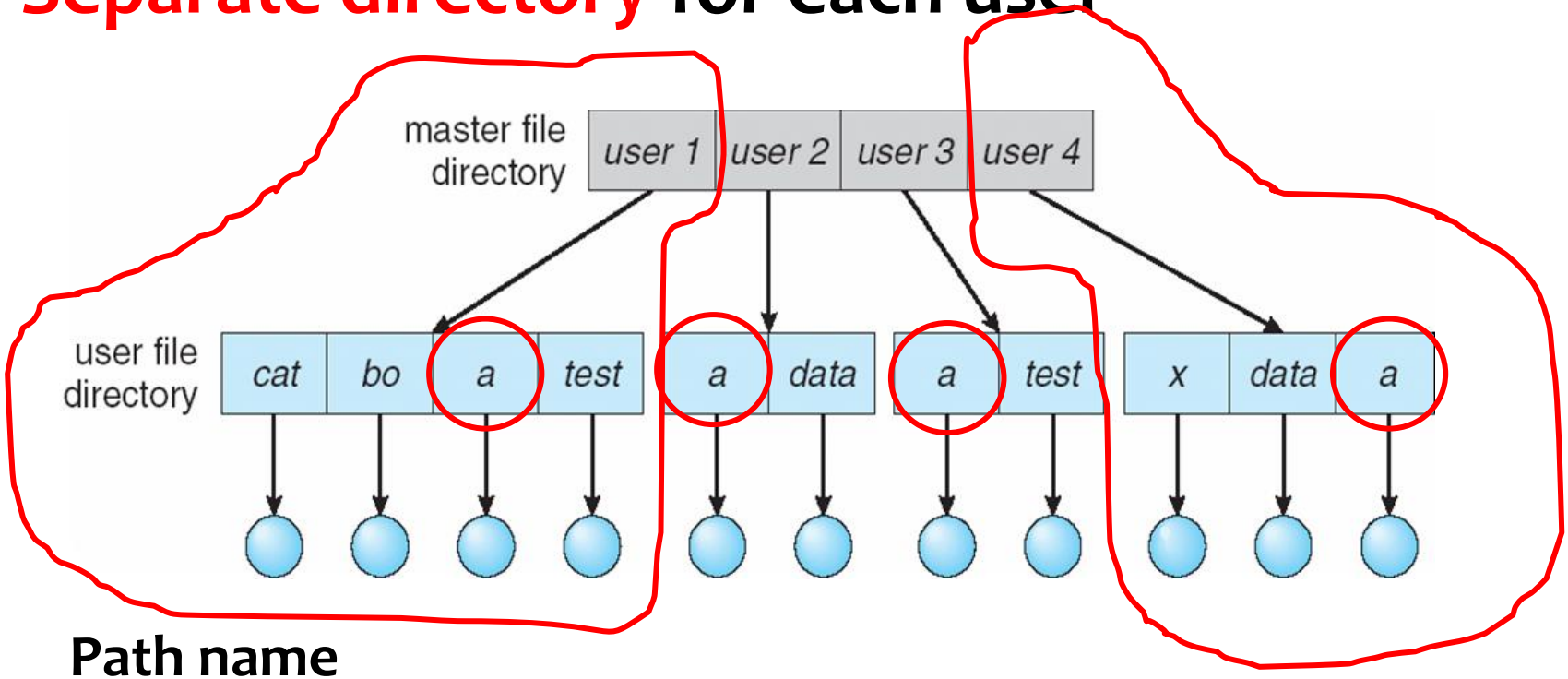
A single directory for all users



- Naming problem
- Grouping problem

Two-Level Directory

Separate directory for each user



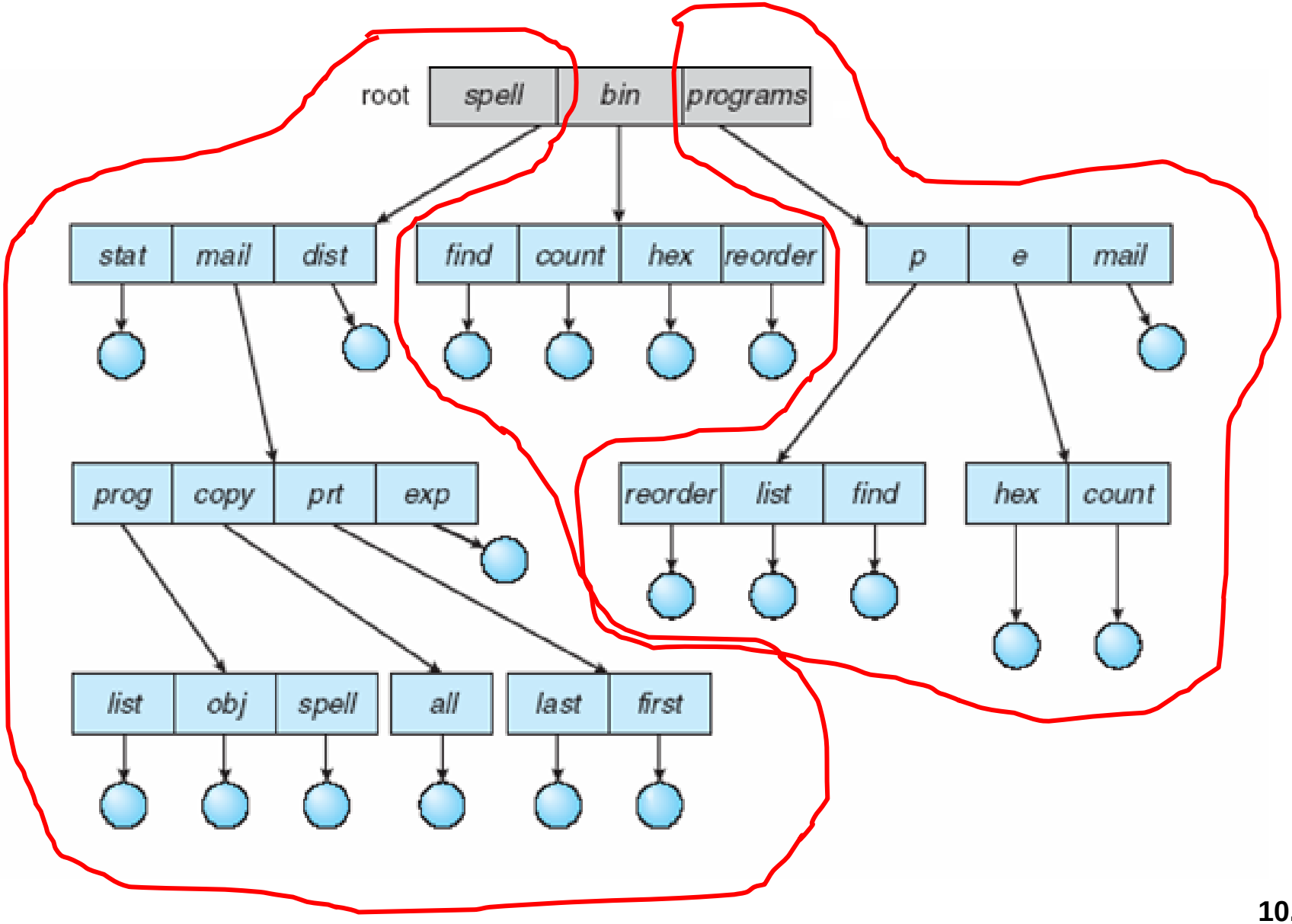
Path name

Can have the same file name for different user

Efficient searching

No grouping capability

Tree-Structured Directories



Tree-Structured Directories (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

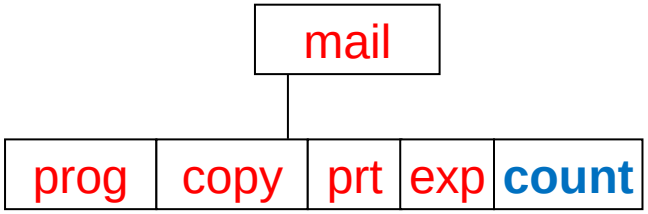
rm <file-name>

Creating a new subdirectory is done in current directory

mkdir <dir-name>

Example: if in current directory /mail

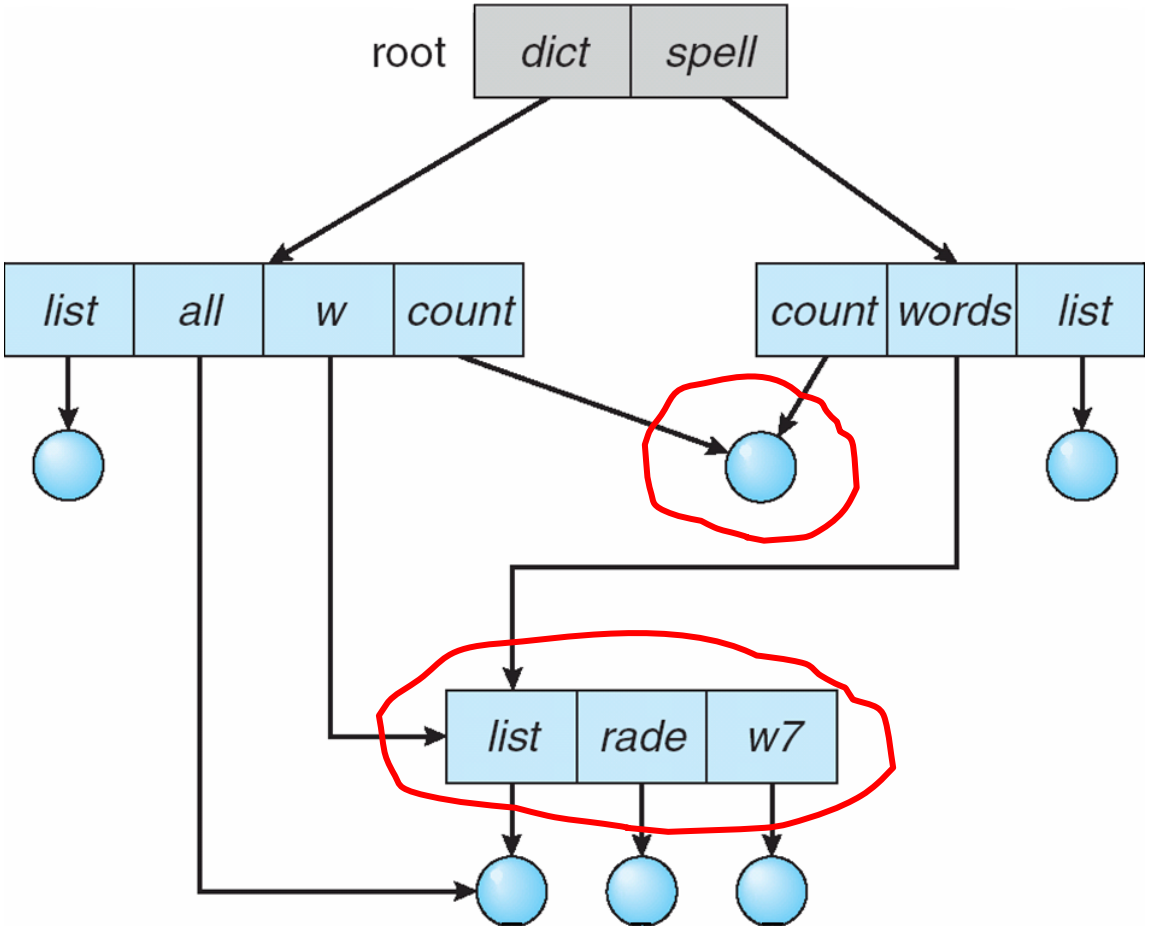
mkdir count



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

Acyclic-Graph Directories

Have shared subdirectories and files, for joined project, for example



Acyclic-Graph Directories (Cont.)

Allows directories to share subdirectories and files. The same file or subdirectory may be in two different directories

Shared files and subdirectories can be implemented in several ways.

Create a new directory entry - Link

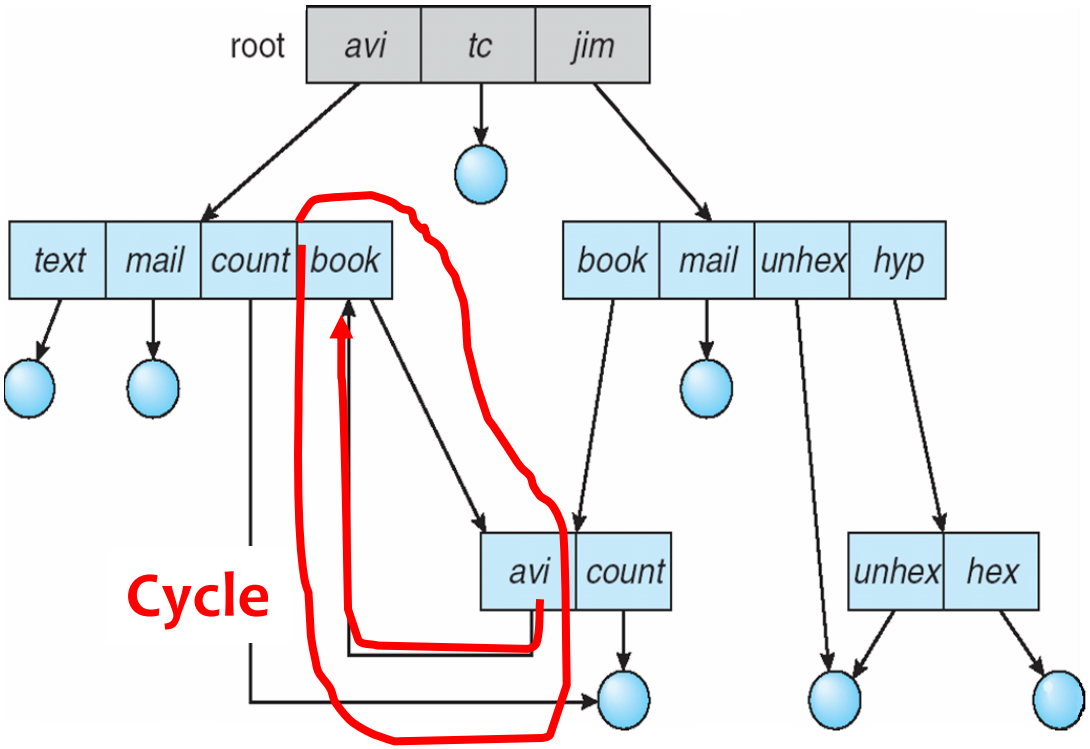
Link – a pointer to another file or subdirectory. A link may be implemented as an absolute or a relative path name.

Resolve the link – using that path name to locate the real file. Links are easily identified by their format in the directory entry and are effectively indirect pointers.

General Graph Directory

A serious problem with using acyclic-graph structure is ensuring that there is no cycles.

However, when we add links, the tree structure is destroyed, resulting in a simple graph structure.



General Graph Directory (Cont.)

If cycles are allowed to exist in the directory

An infinite loop continually searching through the cycle

When a file can be deleted ?

- A Garbage collection scheme is used to determine when the last reference has been deleted and the disk space can be reallocated.

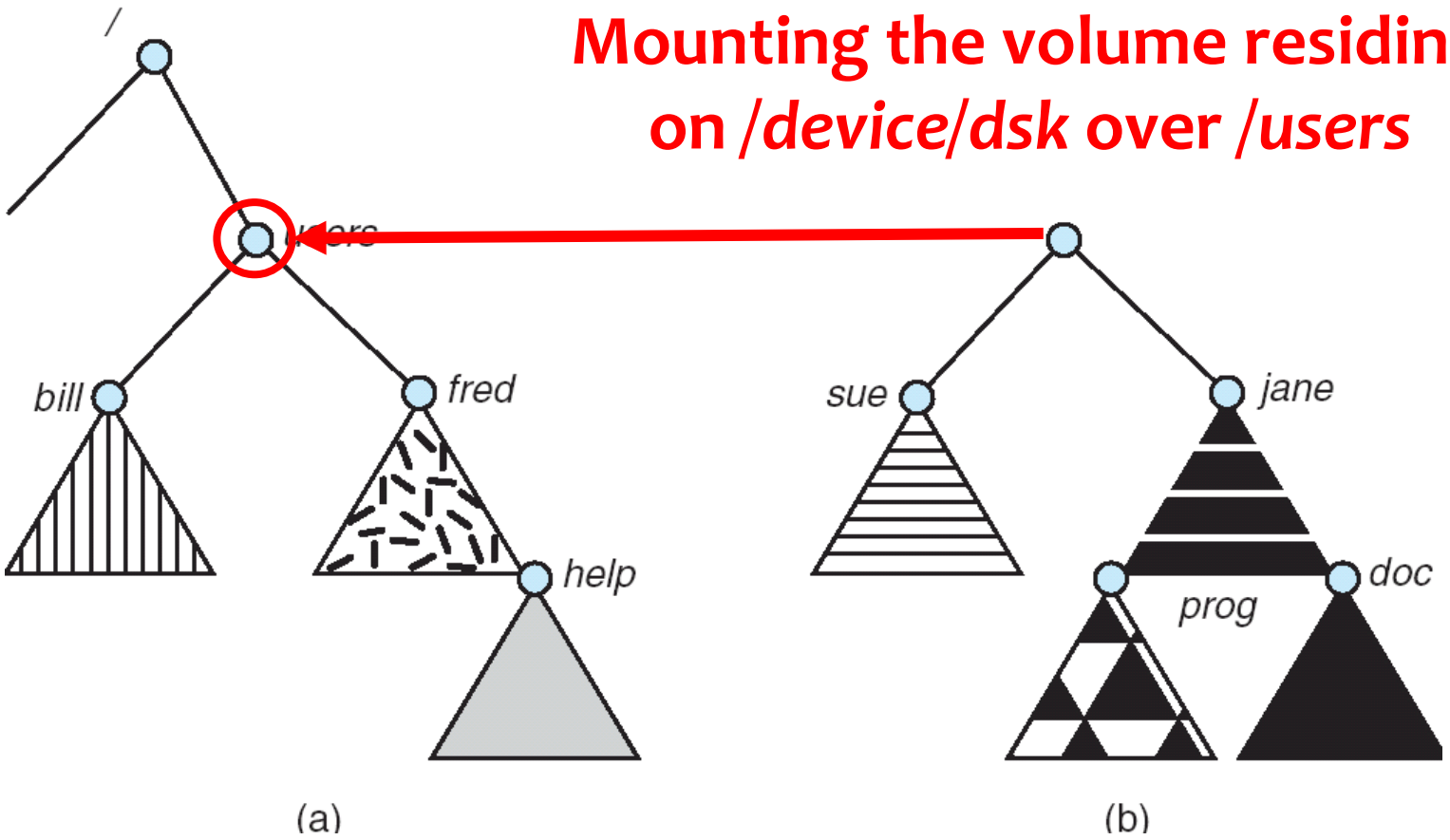
Every time a new link is added use a cycle detection algorithm to determine whether it is OK

File System Mounting

A file system must be mounted before it can be accessed

A unmounted file system is mounted at a mount point

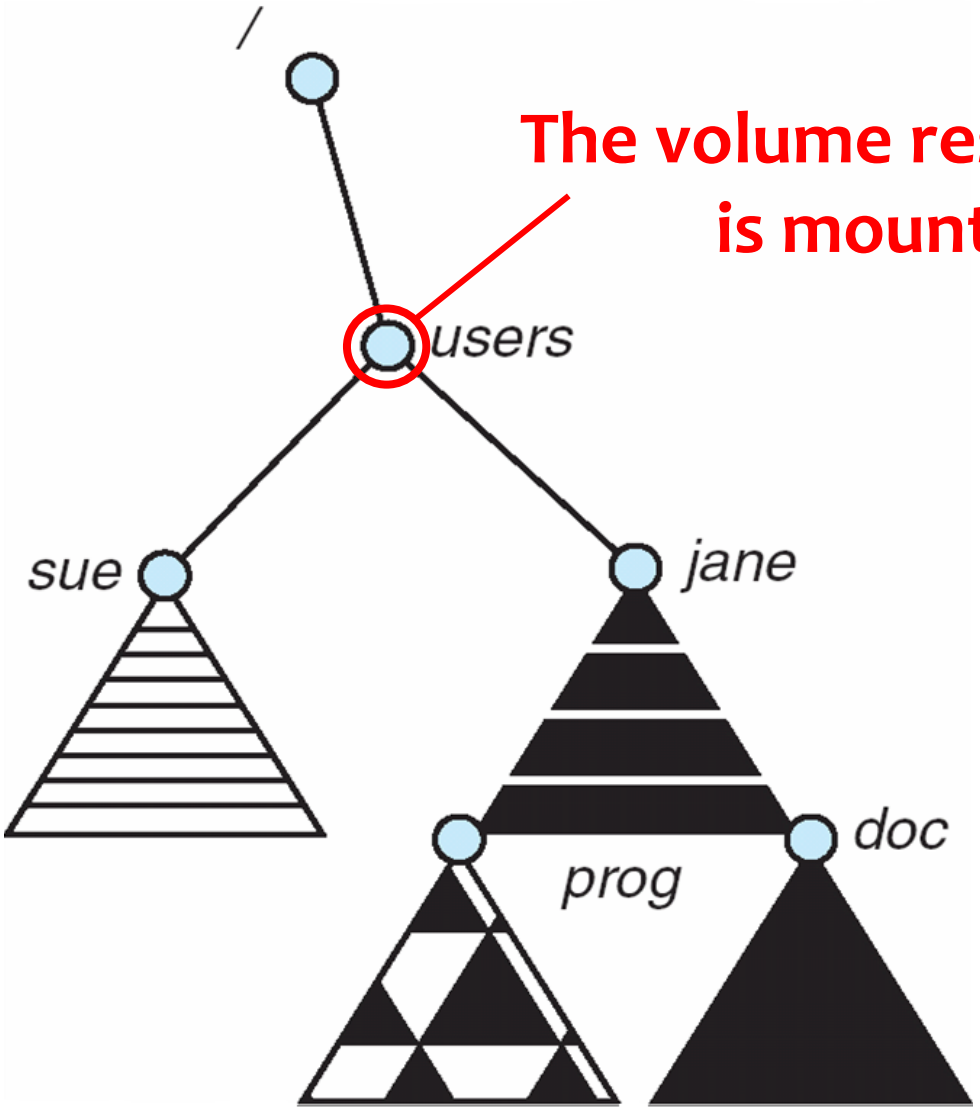
File System Mounting



(a) Existing.

(b) Unmounted Partition

Mount Point



The volume residing on */device/dsk* is mounted over */users*

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

Uses networking to allow file system access between systems

- **Manually** via programs like FTP
- **Automatically,** seamlessly using distributed file systems
- **Semi automatically** via the WWW

File Sharing – Remote File Systems

Client-server model allows clients to mount remote file systems from servers

- Server can serve multiple clients
- Client and user-on-client identification is insecure or complicated
- NFS is standard UNIX client-server file sharing protocol
- CIFS (Common Internet File System) is standard Windows protocol
- Standard OS file calls are translated into remote calls

Distributed Information Systems (distributed naming services) such as LDAP (lightweight directory access protocol), DNS, NIS, Active Directory (Windows XP and Windows 2000) implement unified access to information needed for remote computing

File Sharing – Failure Modes

Remote file systems add new failure modes, due to network failure, server failure

Recovery from failure can involve state information about status of each remote request

Stateless protocols such as NFS include all information in each request, allowing easy recovery but less security

File Sharing – Consistency Semantics

Consistency semantics specify how multiple users are to access a shared file simultaneously

Similar to Ch 6 process synchronization algorithms

- Tend to be less complex due to disk I/O and network latency (for remote file systems)

Andrew File System (AFS, Chapter 17) implemented complex remote file sharing semantics

Unix file system (UFS, Chapter 17) implements:

- Writes to an open file visible immediately to other users of the same open file
- Sharing file pointer to allow multiple users to read and write concurrently

AFS has session semantics

- Writes only visible to sessions starting after the file is closed

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

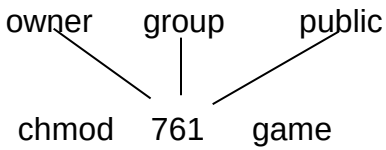
Mode of access: read, write, execute

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.

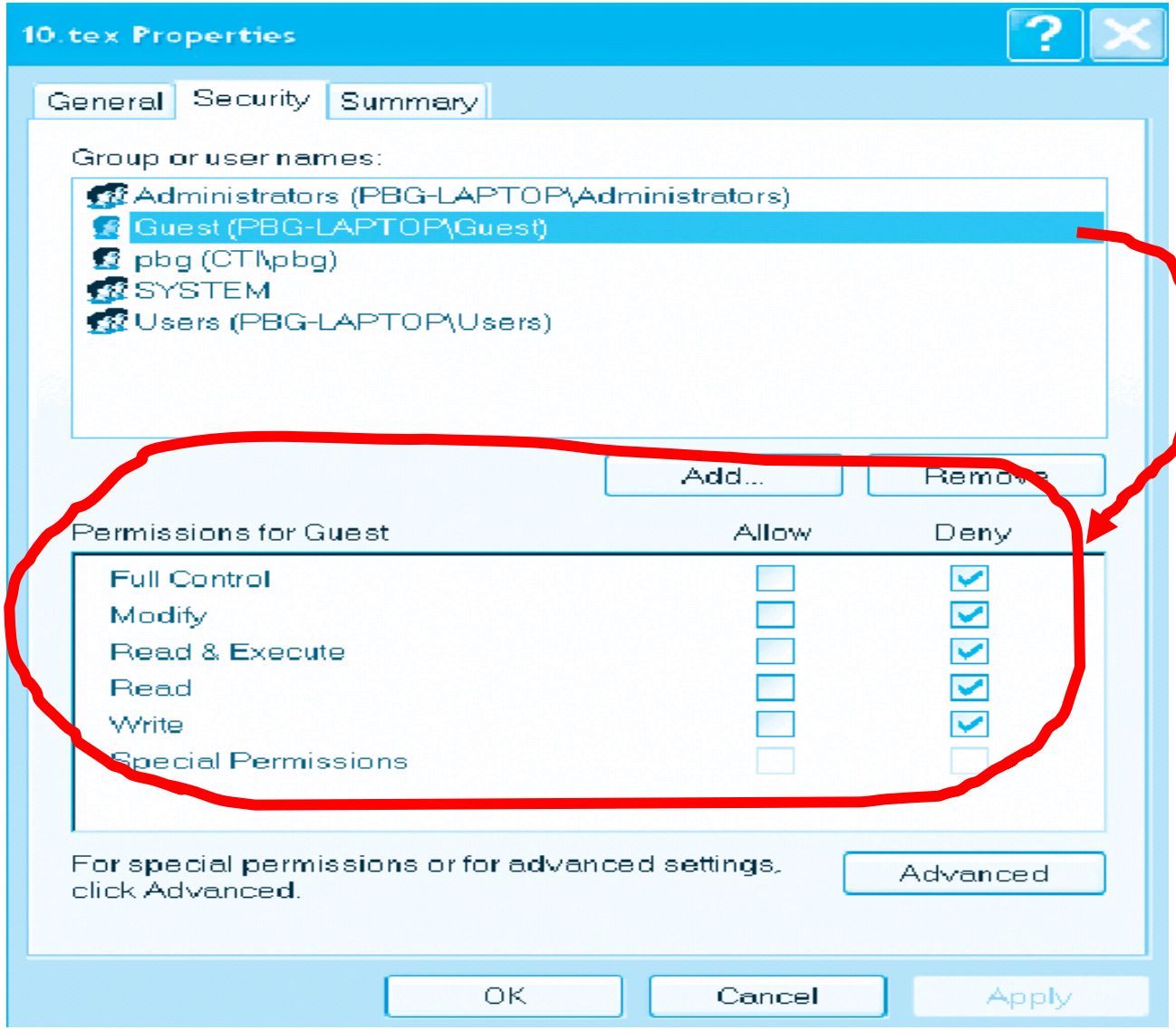
For a particular file (say *game*) or subdirectory, define an appropriate access.



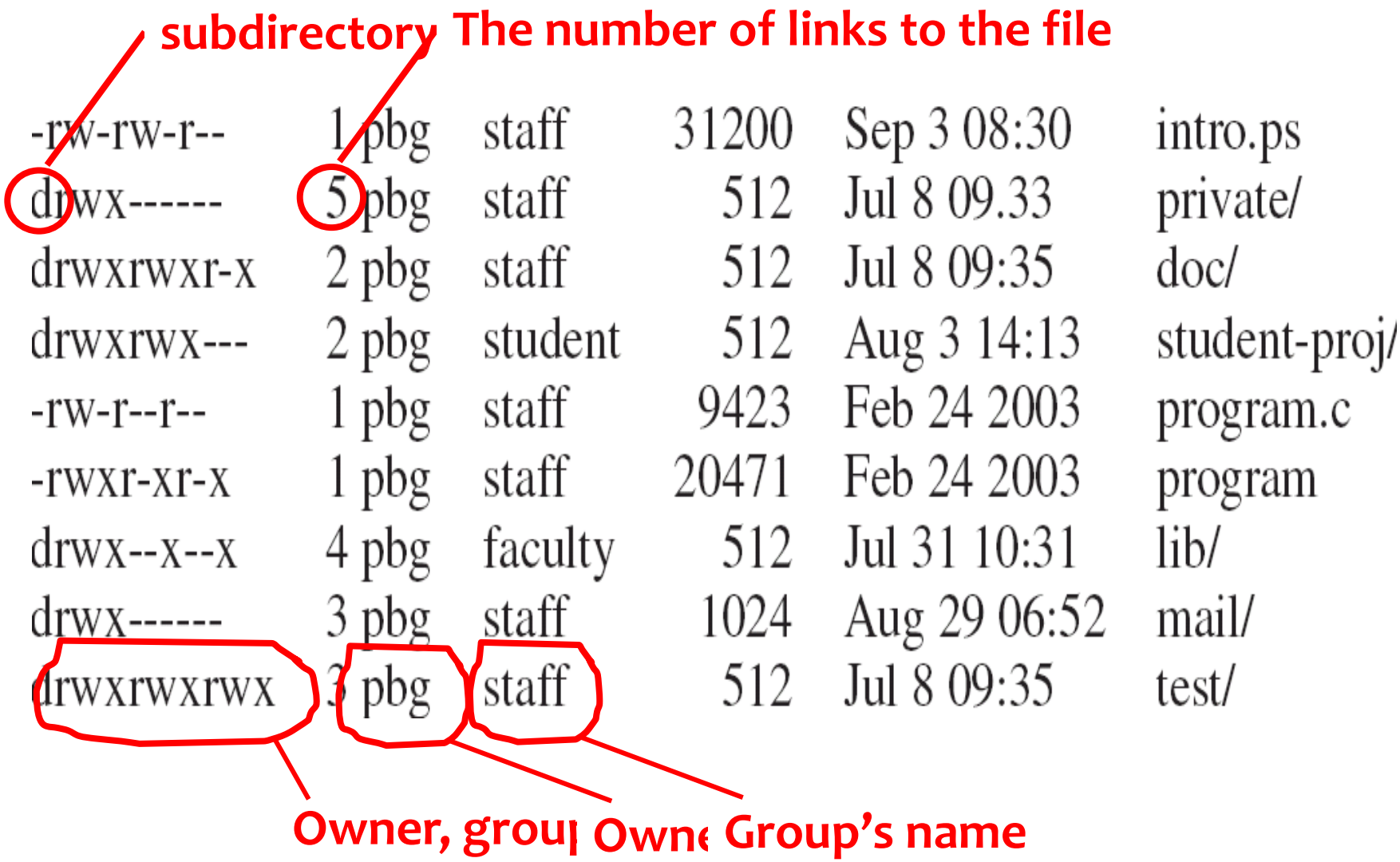
Attach a group to a file

chgrp G game

Windows XP Access-control List Management



A Sample UNIX Directory Listing



End of Chapter 10