FILE SYSTEM

CHAPTER 10

- The slides do not contain all the information and cannot be treated as a study material for Operating System. Please refer the text book for exams.

Topics

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

File Concept

- The OS abstracts from the physical properties of its storage to define a logical storage unit – file
- File is a named collection of related information recorded on secondary storage
- According to user file is smallest allotment of logical secondary storage
- Files represent different 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** – 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**
- **1. Create –** 2 steps – space in the file system must be found – entry for the new file made in directory
- **2. Write -** system call specifying name and information to be written – write pointers points to where to write next
- **3. Read –** system call specifying the name of the file and where in memory to put
- Directory is searched for associated entry and keeps a read pointer
- Process is reading/writing at a time – so maintain per-process *current file position pointer*
- **4. Reposition within file -** Directory is searched for an entry and current file position pointer is repositioned to a value - seek

File Operations

- **5. Delete –** Search directory for file, release space and erase directory entry
- **6. Truncate -** user wants to erase contents but keep its attributes
- ***Other operations***
- Append new information
- Rename existing file
- Create a copy of file for other devices like display and printer

- To avoid constant searching OS maintains ***open-file table***
- It has open count associated with each file indicate how many processes have file open
- close() decreases the open count, when zero remove entry from table

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 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();
```

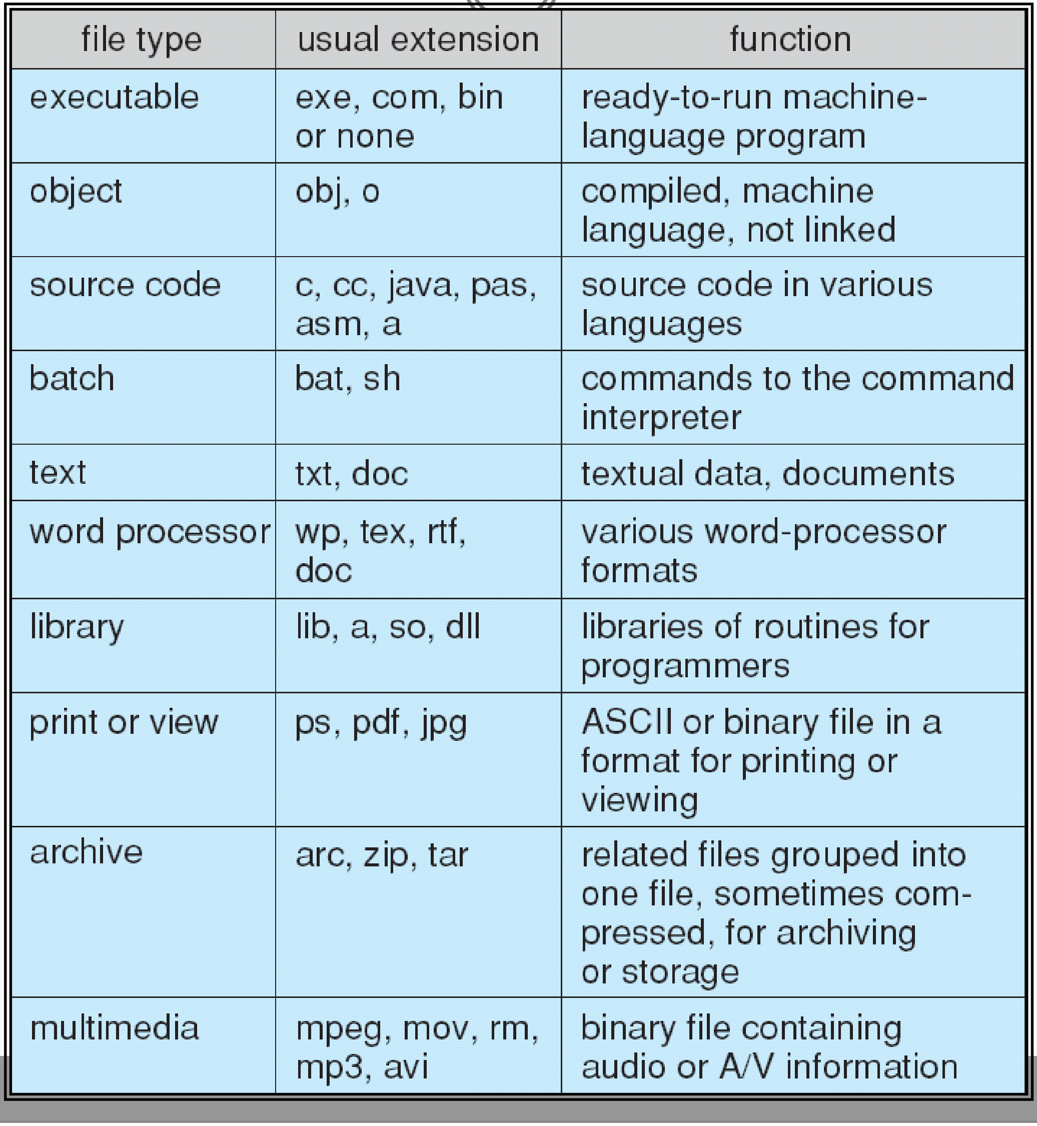
File Locking Example – Java API

```
                // 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();
        }
    }
  }
```

Open File Locking

- Shared lock similar to reader lock and exclusive to writer lock in reader-writer process
- Mandatory or advisory:
    - **Mandatory** – once a process gets exclusive locks OS prevents any other process from accessing the locked file – Windows systems -
    - **Advisory** – once a process gains exclusive locks OS does not prevent other process from getting the lock – rather the process or application must be written to manually acquire lock before accessing file

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 Types – Name, Extension

- If an OS can recognize type of file it can operate on file.
- Mistake of printing binary object form of file
- Include type as part of file name for user and OS
- Name.extension
- TOP-20 OS – if user executes a object program whose source is modified , source will be recompiled automatically by timestamp
- APPL and TEXT files in Mac OS – appends the creator's name along with file(word processor) – so that application is invoked automatically

File Structure

- Certain files must conform to required structure that is understood by the OS
- Executable file has a specific structure that it can determine where in memory it can load the file
- Disadv of OS support multiple file structures – resulting size of OS is cumbersome
- OS supports ASCII and binary files but if users want define an encrypted file to protect contents then both wont suit
- Some OS impose minimal number of file structures
- Unix considers each file to be a sequence of 8 bits bytes ; no interpretation are made
- Each application must include its own code to interpret an input file
- Macintosh has resource fork(interest of user) and data fork (program code or data)

Access Methods

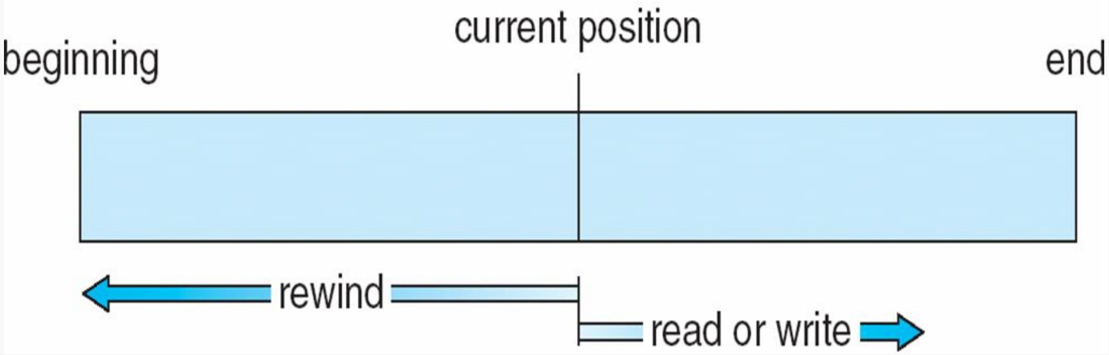
- **Sequential Access**
- Information is processed in order one after the other- editors and compilers use
- Reads and write make up the bulk of operations

read next – reads the next portion of file and advances the pointer

write next – appends to the end of the file and advances the end of the file to new end

reset – can be reset to beginning

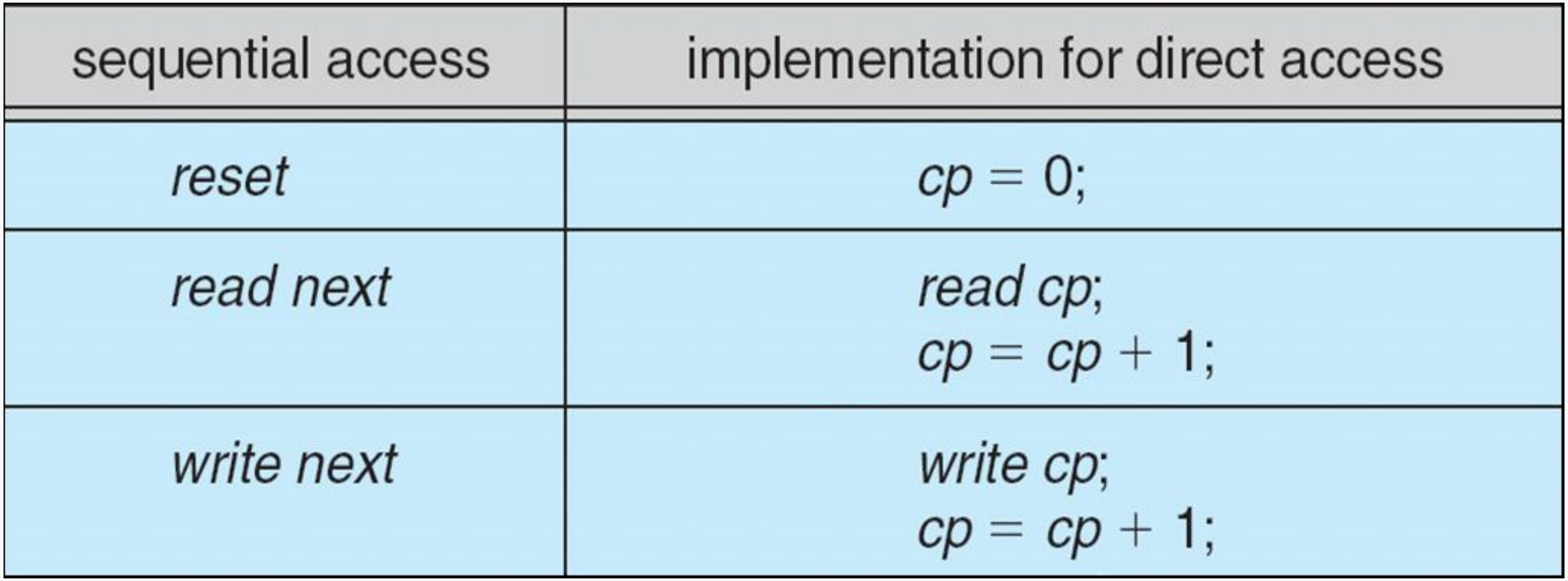
The diagram shows the tape model of a file and works on sequential access



Access Methods

- **Direct Access –** A file is made up of fixed length logical records that allow programs to read , write records rapidly in no particular order
- Based on disk model – great use immediate access to large amounts of information – Databases
- File operations must be modified to include block number as a parameter – read n reads the nth block rather than read next
- Block number provided by the user is ***relative block number*** to the beginning of the file
- First relative block is 0 and second is 1 and so on even though absolute disk address is 14703 and 3192 respectively

Simulation of Sequential Access on Direct-access File

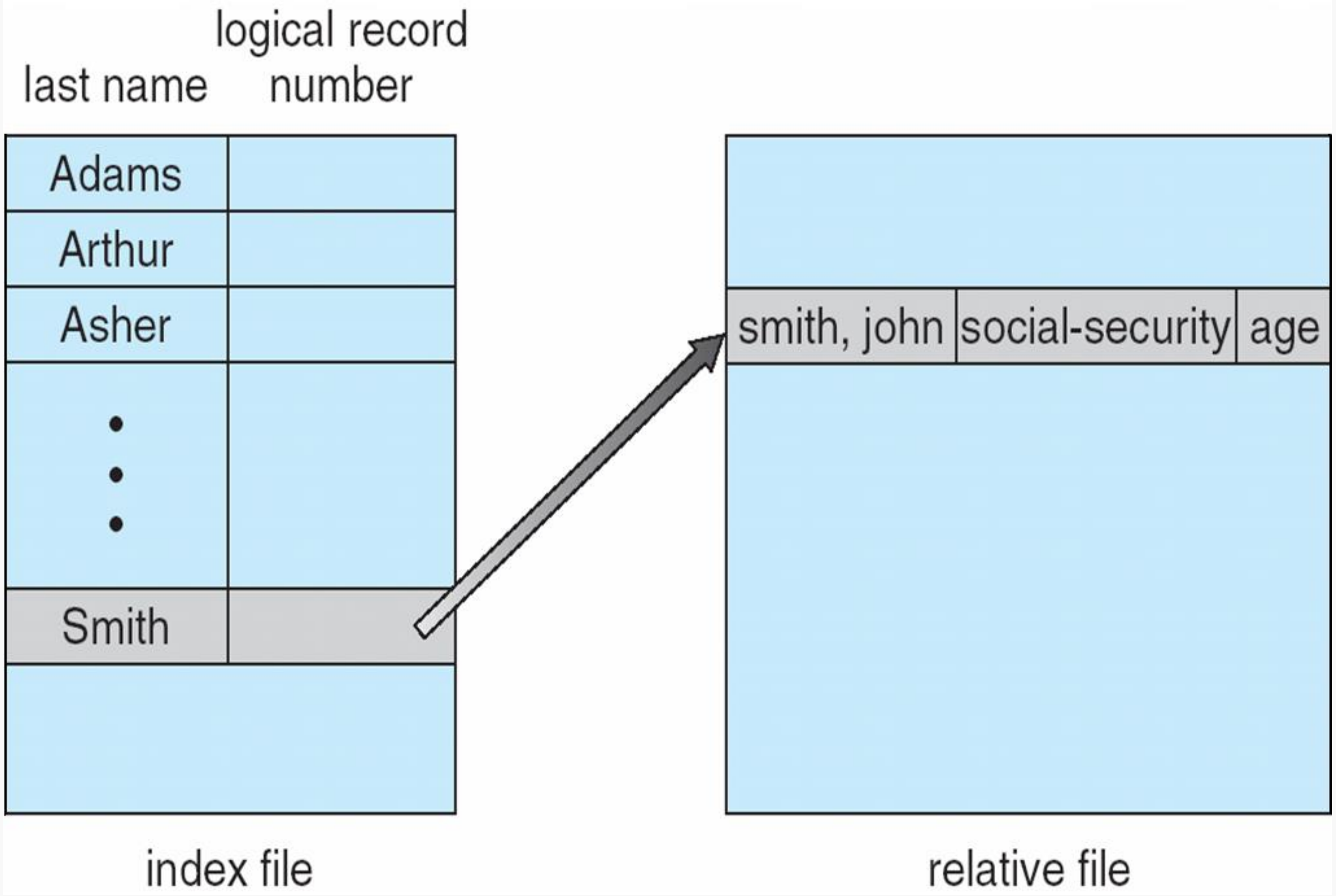


sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

Example of Index and Relative Files

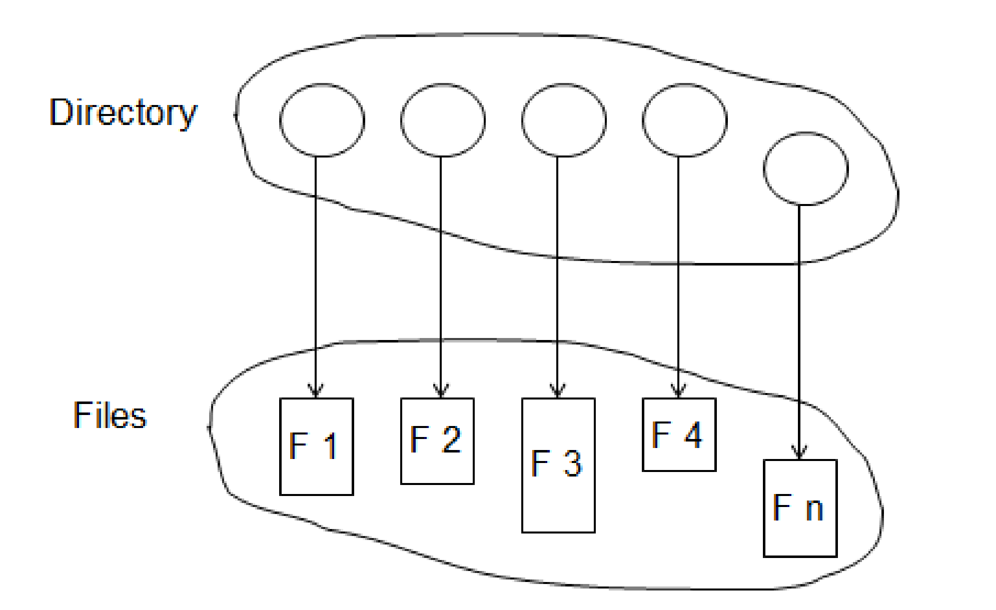
- Index is like back of the book – containing pointers to the various blocks
- To find a record we first search the index and then use the pointer to access the file directly
- Eg. Retail price file would have 1 million records sorted based on product codes
- Index file just has the product codes
- Search involves searching the index file to get the record and then find the desired block to get the record

Example of Index and Relative Files



Directory Structure

- A collection of nodes containing information about all files

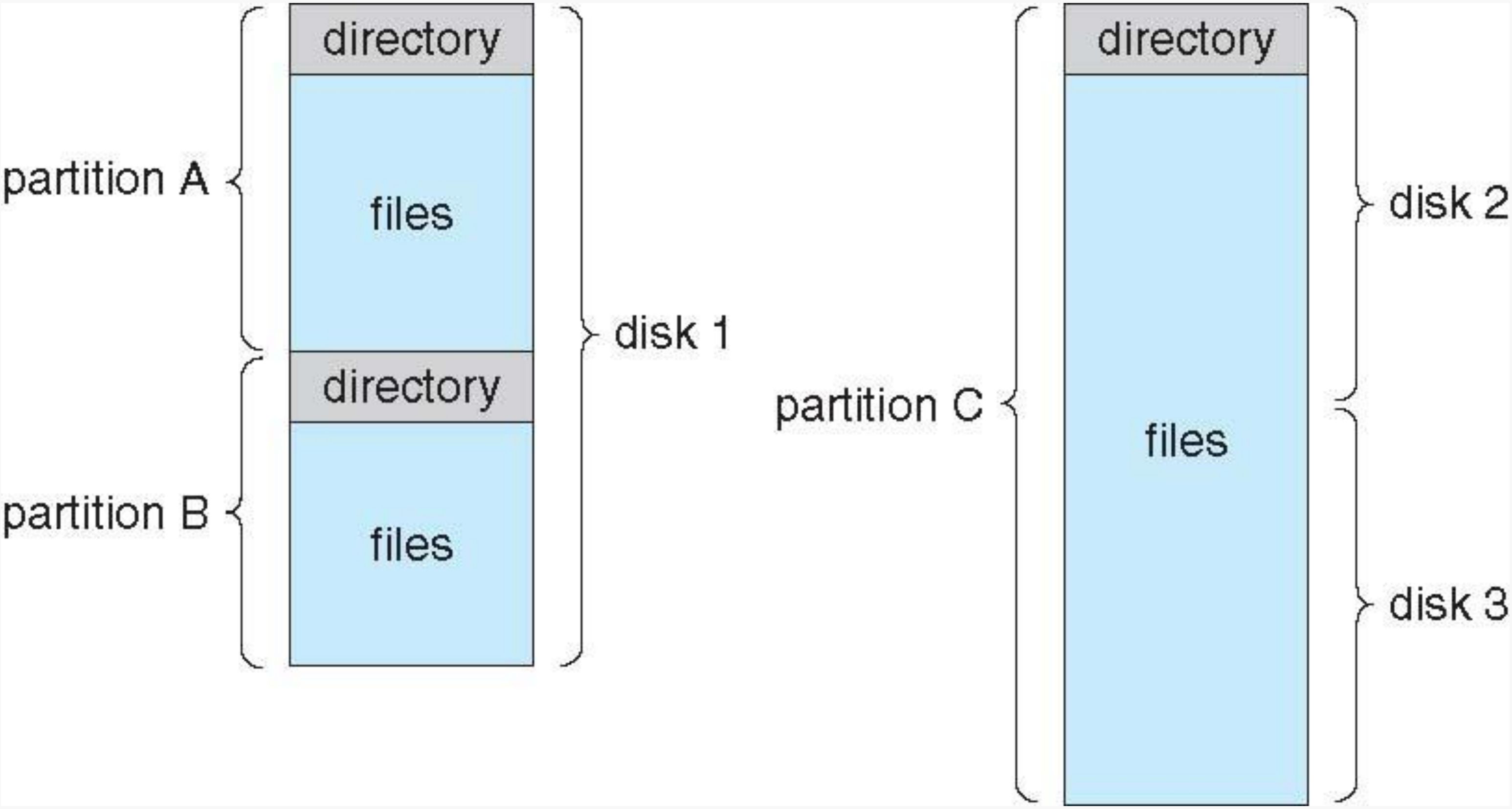


- Both the directory structure and the files reside on disk

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

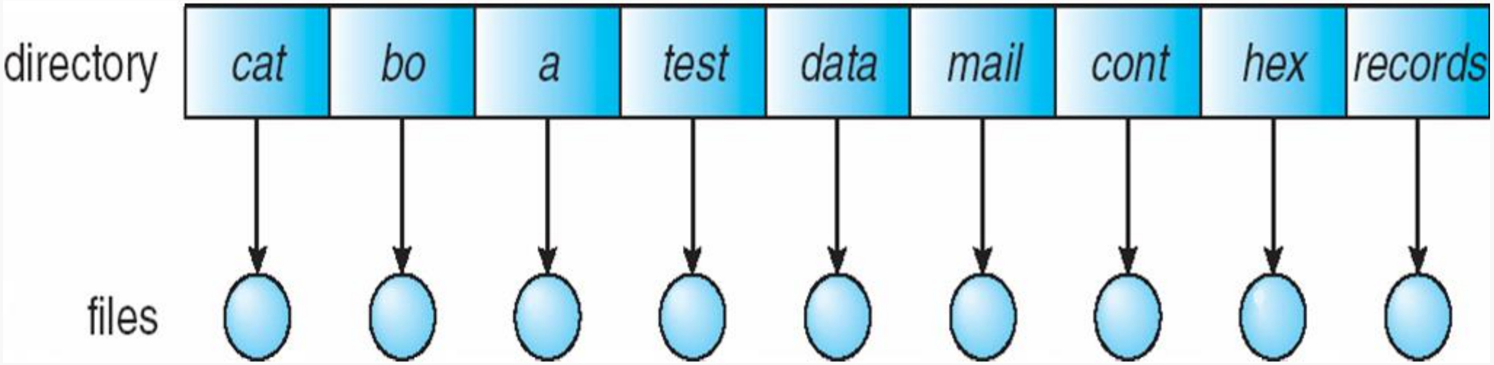
- Directory is viewed as a symbol table that translates file names into directory entries
- Search for a file – Find the entry for a particular file, files with particular pattern
- Create a file – create new files and add it to directory
- Delete a file – remove from the directory
- List a directory – List the contents of the directory
- Rename a file – change the name when the contents change – may also change the position within the directory structure
- Traverse the file system – access every directory and every file

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, …)

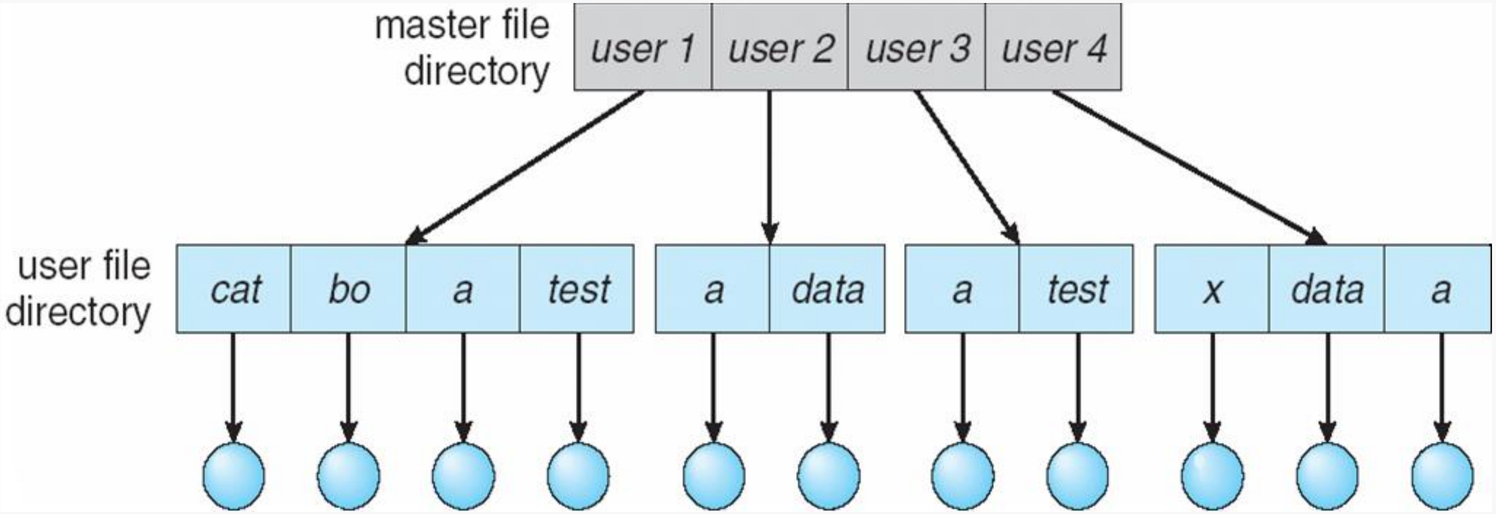
Single-Level Directory

- A single directory for all users
- Naming problem – all users use the same directories
- Grouping problem
- Remembering files is difficult

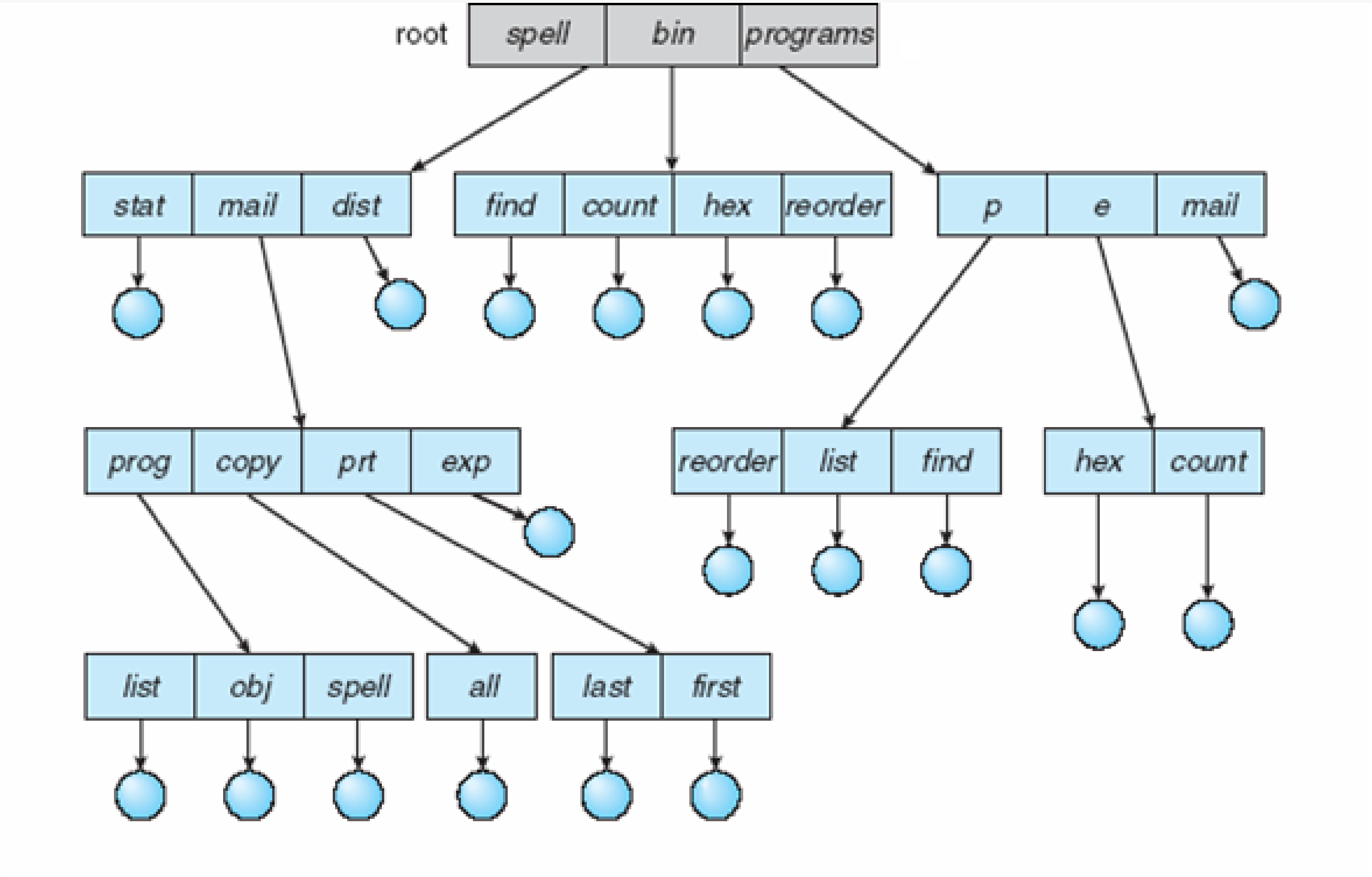


Two-Level Directory

- 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**
- **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

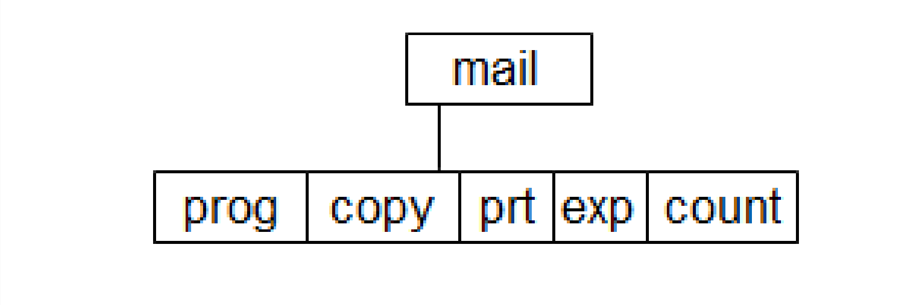
  mkdir <dir-name>

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

  mkdir count

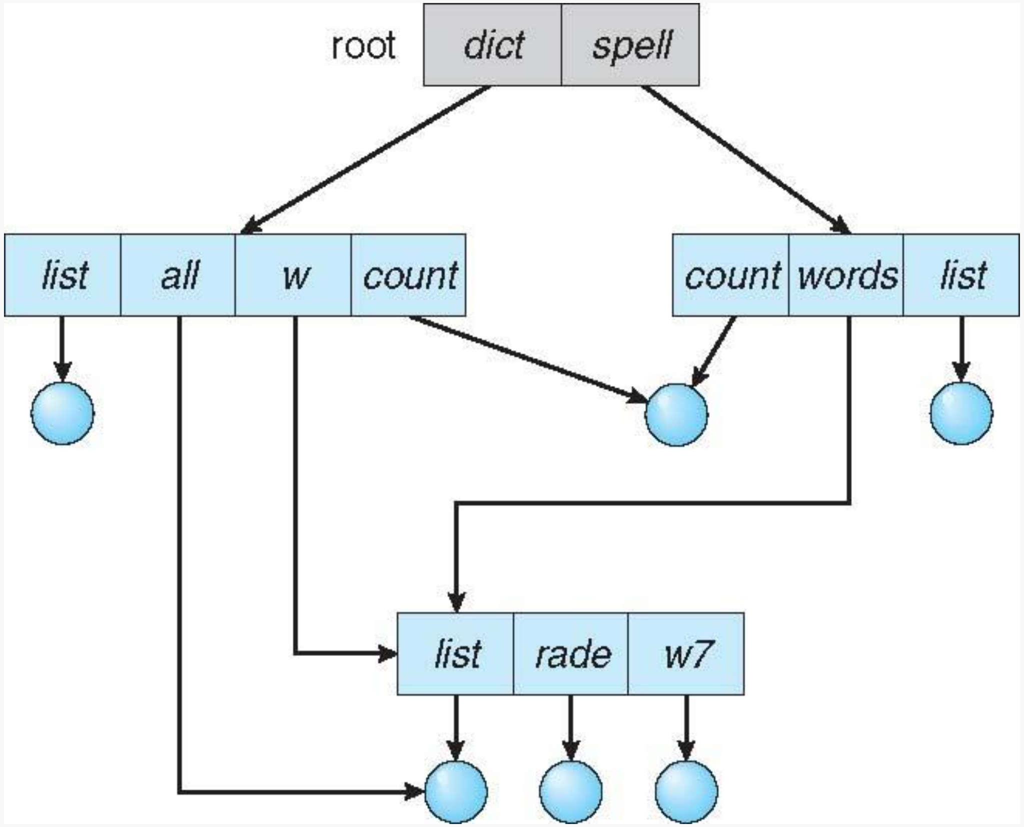
Tree-Structured Directories (Cont.)

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



Acyclic-Graph Directories

- Have shared subdirectories and files

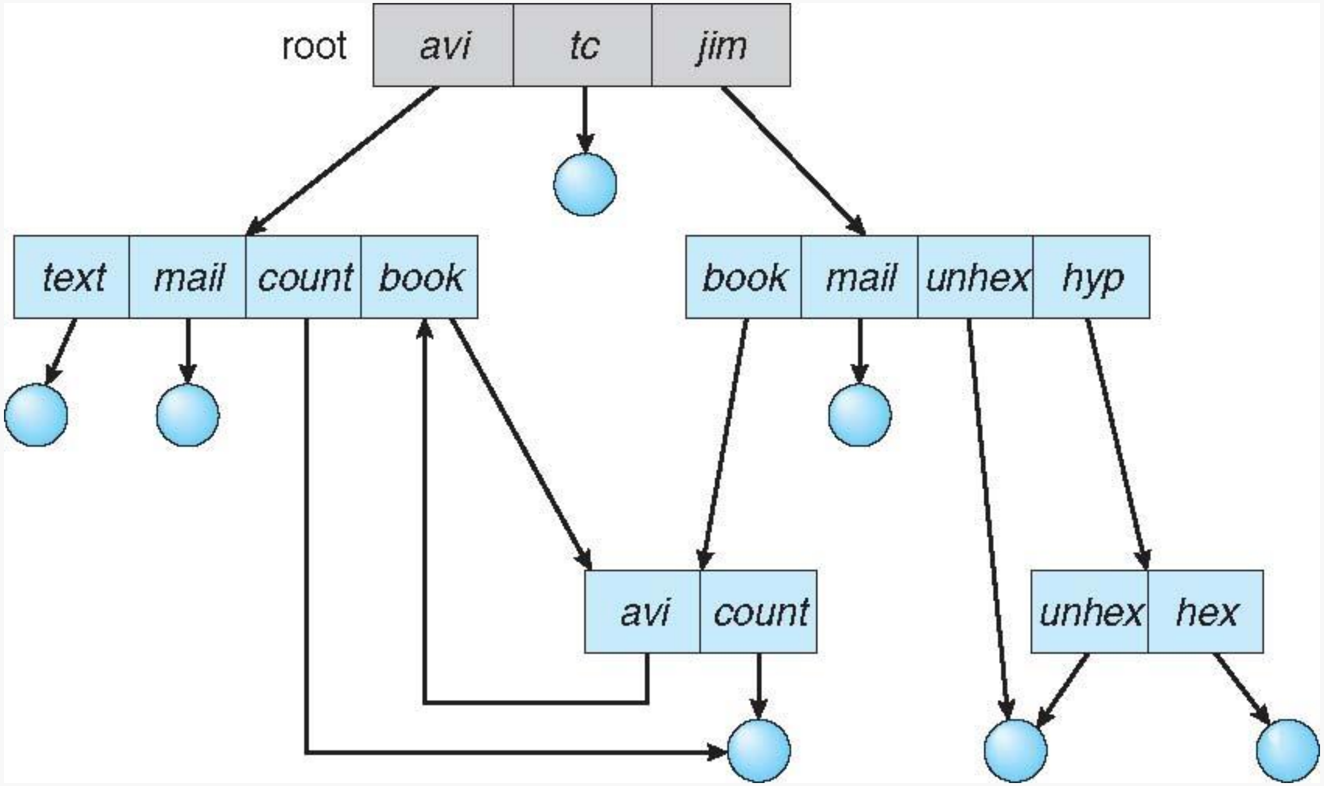


Acyclic-Graph Directories (Cont.)

- 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
  - **Symbolic link vs Hard link**

General Graph Directory



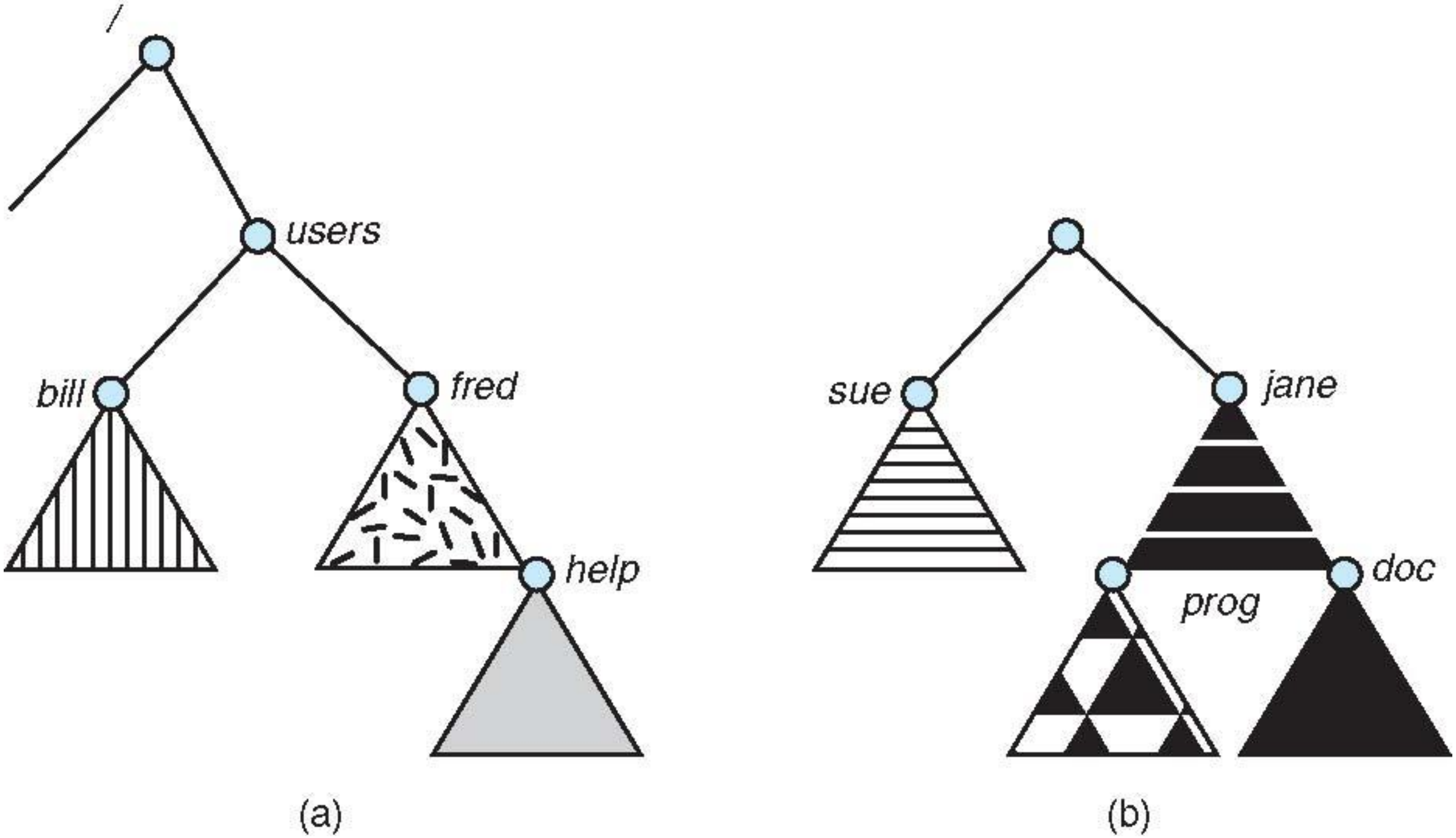
General Graph Directory (Cont.)

- 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

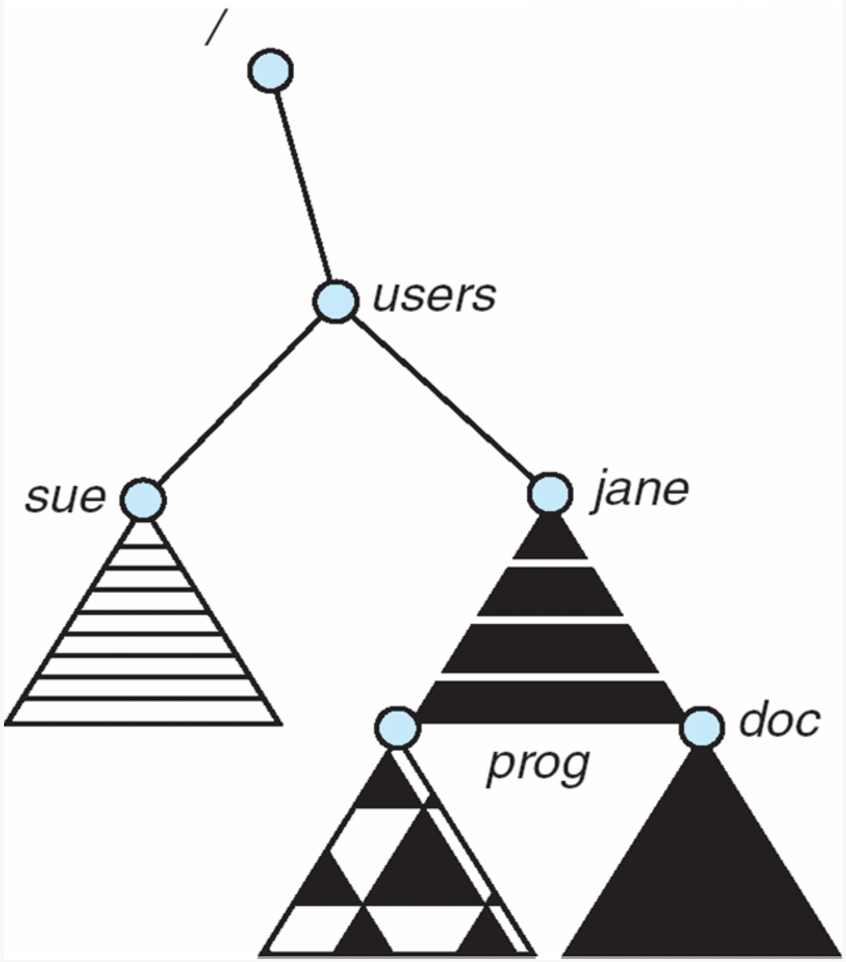
File System Mounting

- 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 -** is an empty directory
- Some systems mount whenever they are connected other require commands
- OS is given the name of the device and the mount point – location within the file structure where the file system is to be attached
- Eg: Unix - /home for every user
- OS verifies that the device contains a valid file system by asking the device driver and verifying the format
- Finally OS makes a note in its directory structure that a file system is mounted

(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

File Sharing – Remote File Systems

- Uses networking to allow file system access between systems
  - Manually via programs like FTP
  - Remote directories are visible from local machine **distributed file systems**
  - Semi automatically via the- reverse of ftp – browser access remote files
  - **world wide web**
- **Client-server** model allows clients to mount remote file systems from servers
  - Server can serve multiple clients – which clients and what resource
  - Client and user-on-client identification is insecure or complicated
  - **NFS** is standard UNIX client-server file sharing protocol
  - Standard operating system file calls are translated into remote calls

File Sharing – Remote File Systems

- Distributed Information Systems **(distributed naming services)** such as LDAP, DNS, NIS, Active Directory implement unified access to information needed for remote computing
- 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
    - Tend to be less complex due to disk I/O and network latency (for remote file systems
  - Unix file system (UFS) 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

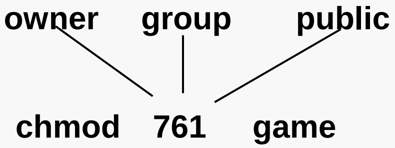
- Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users

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

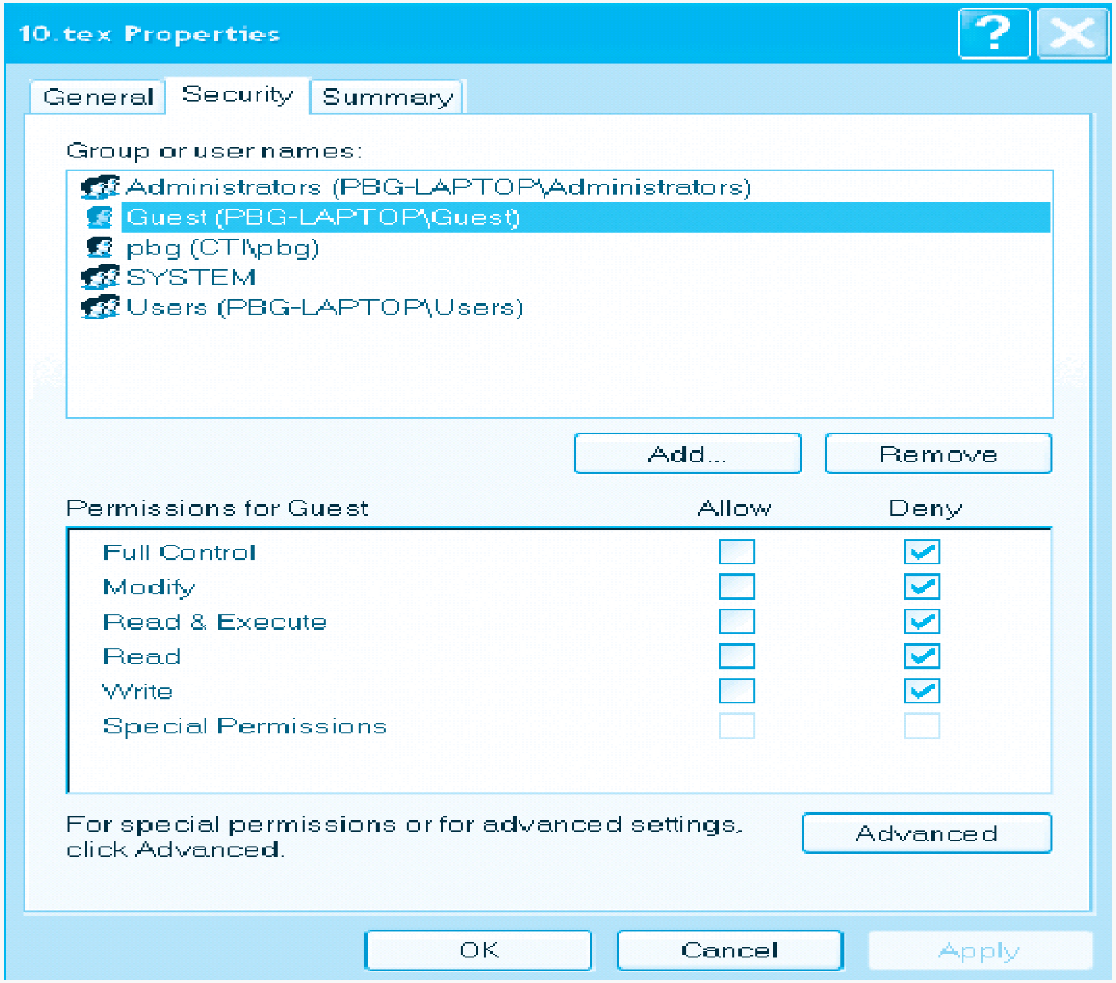
- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say *game*) or subdirectory, define an appropriate access.



Attach a group to a file

chgrp G game

Windows XP Access-Control List Management



10.tex Properties

General | Security | Summary

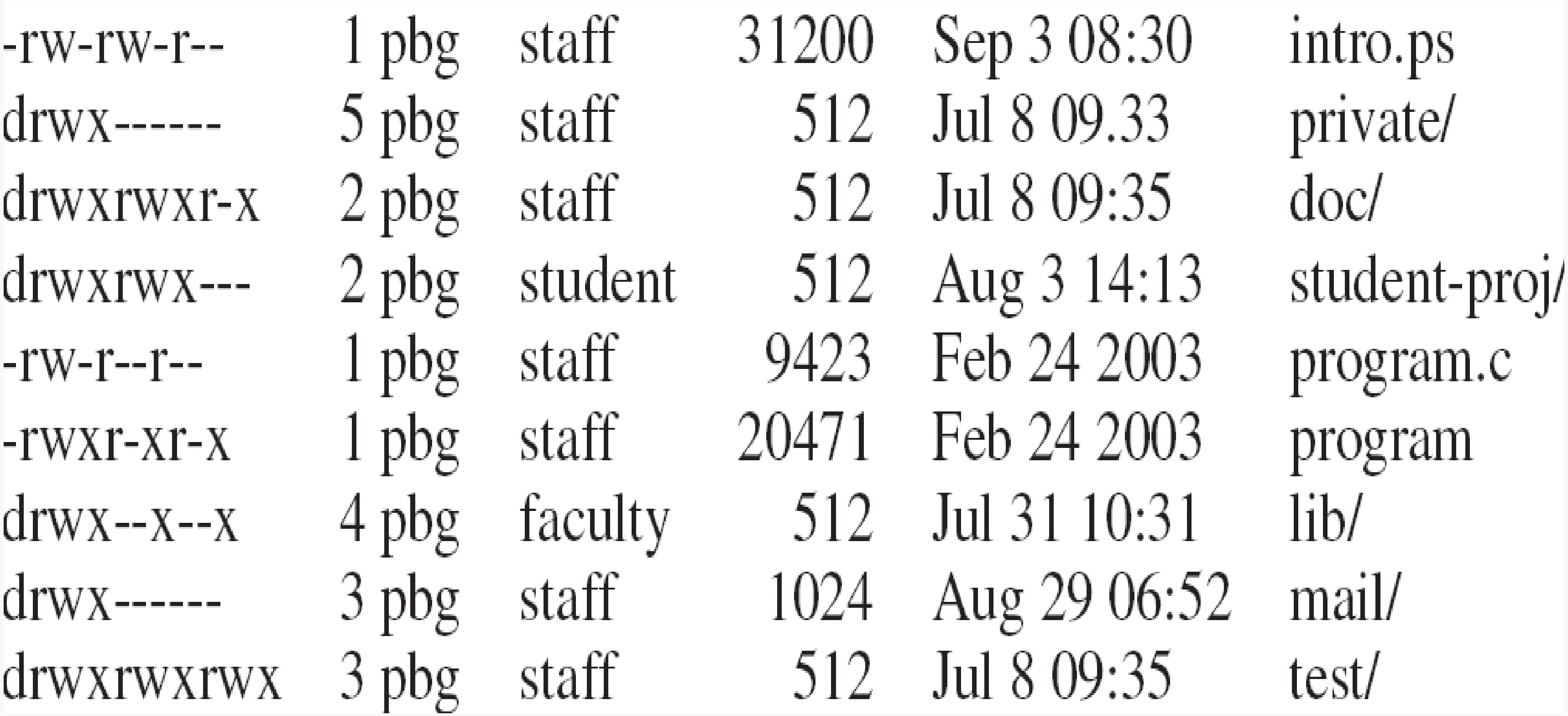
Group or user names:

- Administrators (PBG-LAPTOP\Administrators)
- Guest (PBG-LAPTOP\Guest)
- pbg (CTI\pbg)
- SYSTEM
- Users (PBG-LAPTOP\Users)

Permissions for Guest	Allow	Deny
Full Control	□	☑
Modify	□	☑
Read & Execute	□	☑
Read	□	☑
Write	□	☑
Special Permissions	□	□

For special permissions or for advanced settings, click Advanced.

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/
```