Chapter 10 File-System Interface

LI Wensheng, SCS, BUPT

Strong points:

- **File Concept**
- **Access Methods**
- **Directory Structure**

* Overview of File Management

- File Management
  - File management system is considered part of the operating system.
  - Input to applications is by means of a file.
  - Output is saved in a file for long-term storage.
- File Management System
  - The way a user of application may access files.
  - Programmer does not need to develop file management software.

* Objectives for File Management System

- Meet the data management needs and requirements of the user.
- Guarantee that the data in the file are valid.
- Optimize performance
- Provide I/O support for a variety of storage device types.
- Minimize or eliminate the potential for lost or destroyed data.
- Provide a standardized set of I/O interface routines.
- Provide I/O support for multiple users.

* Minimal Set of Requirements

- Each user should be able to create, delete, read, and change files.
- Each user may have controlled access to other users' files.
- Each user may control what type of accesses are allowed to the users' files.
- Each user should be able to restructure the user's files in a form appropriate to the problem.
- Each user should be able to move data between files.
- Each user should be able to back up and recover the user's files in case of damage.
- Each user should be able to access the user's files by using symbolic names.

* File Management Functions

- Identify and locate a selected file.
- Use a directory to describe the location of all files plus their attributes.
- On a shared system describe user access control.
- Blocking for access to files.
- Allocate files to free blocks.
- Manage free storage for available blocks.

Chapter 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.

Contents

10.1 File Concept

- Contiguous logical address space.
- OS provides a uniform logical view of information storage.
- A file is a named collection of related information that is recorded on secondary storage.
- A file is the smallest allotment of logical secondary storage.
- Commonly, files represent programs and data.
  - Program: source, object
  - Data files: numeric, character, binary
- files may be free form or may be formatted rigidly.
- The information in a file is defined by its creator.
- A file is named, and is referred to by its name, and is independent.

File Attributes

- Name – only information kept in human-readable form.
- Identifier – a unique number that identifies the file within the system, non-human-readable name for the file.
- Type – needed for systems that support different types.
- Location – a pointer to the location of the file on device.
- Size – current size of the file.
- Protection – access-control information, controls who can do reading, writing, executing.
- Time, date, and user identification – data for protection, security, and usage monitoring.
- Information about files is kept in the directory structure, which is maintained on the disk.
- A directory entry consists of the file's name and its unique id. The id in turn locates the other file attributes.

File Operations

- A file is an abstract data type.
- The minimal set of required file operations
  - Create – find space, make an entry in directory for the file.
  - Write – file name, information to be written. *write* pointer.
  - Read – file name, position in memory, *read* pointer.
    A per-process current-file-position pointer.
  - Reposition within file – file seek, not need I/O.
  - Delete – name, release file space, erase directory entry.
  - Truncate – reset length to 0, release file space.
- These primitive operations may be combined to perform other file operations.
  - Copy
- Most file operations involve searching the directory for the entry associated with the named file.

File Operations (Cont.)

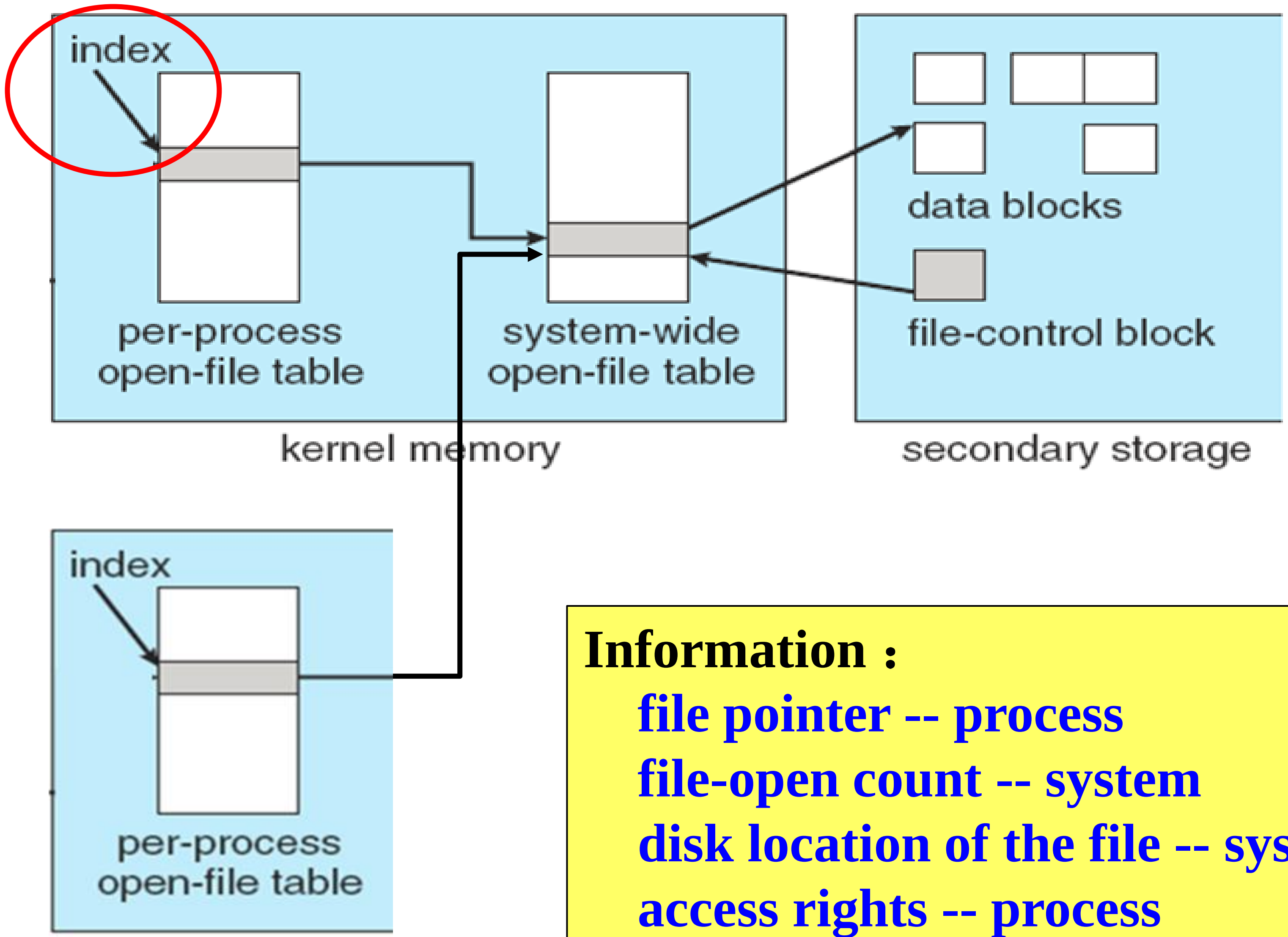
directory structure

- **Open-file table**
  - **A kernel data structure.**
  - **containing information about all open files.**
- **Open(F_i)**
  - **search the directory structure on disk for entry of F_i**
  - **move the content of the entry to open-file table.**
  - **Return a pointer to the entry in the open-file table.**
  - **also accept access mode information, such as create, read-only, read-write, append-only, etc.**
- **Close (F_i)**
  - **move the content of entry F_i in open-file table to directory structure on disk.**

File Operations (Cont.)

- File sharing
  - Several processes may open the file at the same time.
  - Several different applications open the file at the same time.
- Two levels of internal tables:
  - **Process open-file table**, a per-process table
    - tracks all files a process has open.
    - Keeps information regarding the use of the file by the process. e.g. the current file pointer, access rights, accounting information, etc.
  - **System open-file table**, a system-wide table
    - contains process-independent information, such as file location on disk, access dates, file size.
  - **Each entry in the per-process table in turn points to a system-wide open-file table.**

The two levels of open-file tables


index
per-process open-file table
system-wide open-file table
data blocks
file-control block
kernel memory
secondary storage
index
per-process open-file table

Information :

file pointer -- process

file-open count -- system

disk location of the file -- system

access rights -- process

Information associated with an Open File

- Information associated with an open file: file pointer, file-open count, disk location of the file, access rights.
  - 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. Tracks the number of opens and closes, and reaches zero on the last close. The system can then remove the entry from system open-file table.
  - Disk location of the file: cache of data access information, to locate the file on disk, such as the beginning block and length, index block, etc.
  - Access rights: per-process access mode information,

Open File Locking

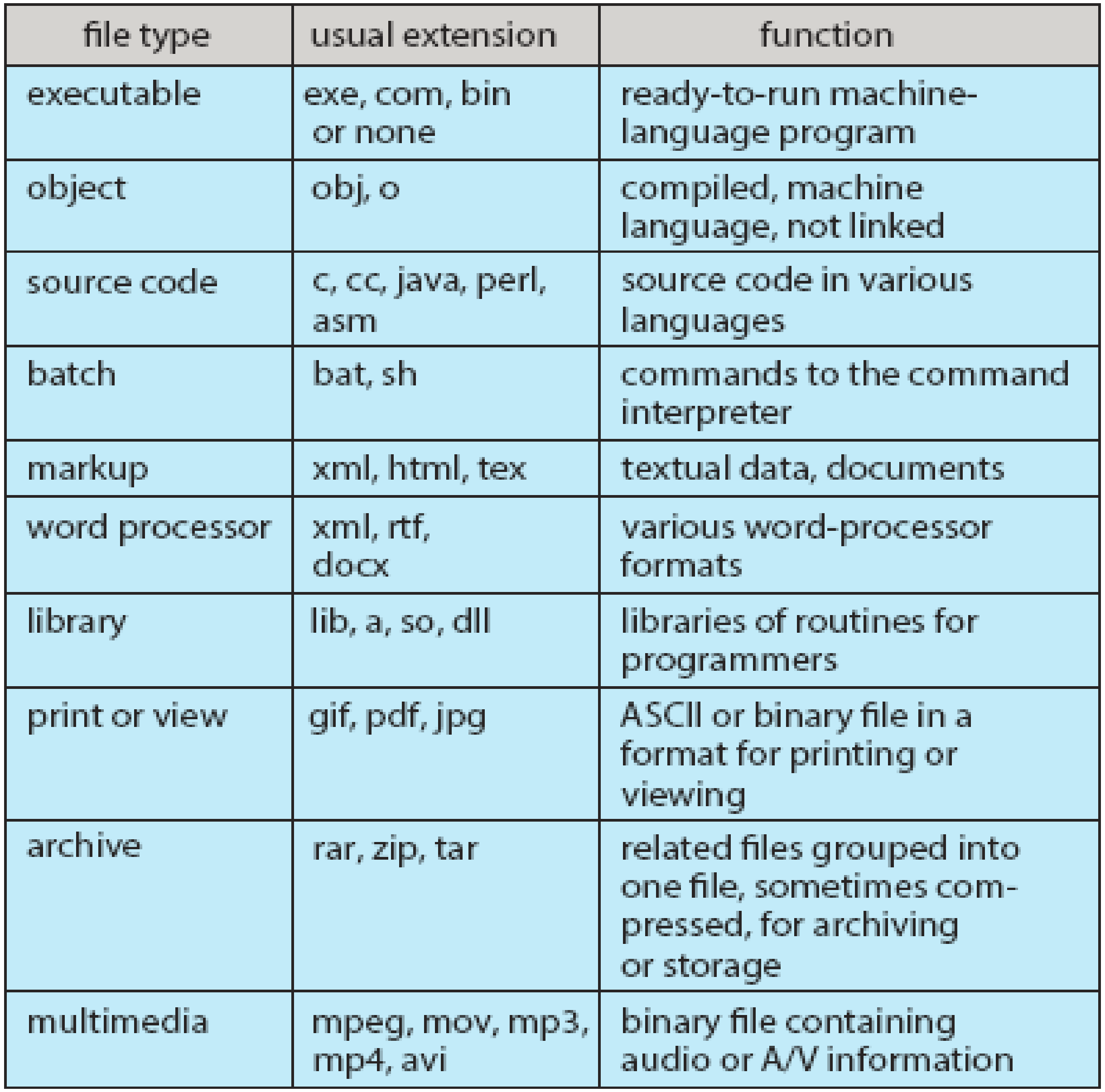
- File locks allow one process to lock a file and prevent other processes from gaining access to it.
- Provided by some OSs and file systems
  - Similar to reader-writer locks
  - Shared lock similar to reader lock – several processes can acquire concurrently.
  - Exclusive lock similar to writer lock.
- Mediates(调停) access to a file
- OS may provide either mandatory or advisory file-locking mechanisms:
  - Mandatory – access is denied depending on locks held and requested. -- Windows OS.
  - Advisory – processes can find status of locks and decide what to do. -- Unix OS.

File types

- Whether the operating system should recognize and support file types?
- A common technique for implementing file types is to include the type as part of the file name.
  - name is split into two parts: a name and an *extension*, separated by a period character.
  - The system uses the extension to indicate the type of the file and the type of operations that can be done on that file.
  - Application programs also use extensions to indicate file types in which they arc interested.
- Some OS, each file has a type, and a creator attribute containing the name of the program that created it.
- UNIX, a magic number stored at the beginning of some files to indicate roughly the type of the file.

.exe   Linux, 0x7f45   AIX, 0x01df

File types (Cont.)



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, perl, asm	source code in various languages
batch	bat, sh	commands to the command interpreter
markup	xml, html, tex	textual data, documents
word processor	xml, rtf, docx	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	rar, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, mp3, mp4, 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

Internal file structure

- **Logical structure**
  - **Two types**
    - **Text file: a sequence of 8-bit bytes**
    - **Record file: a sequence of records with fixed or variable length**
  - **Logical address**
    - **Offset from the beginning of the file**
    - **Logic record no.**
- **Physical structure: A set of disk blocks.**
- **Basic I/O functions operate in terms of blocks.**
- **Mapping logical address to physical address**
  - **Logical block no.**
  - **Physical disk block no.**
  - **Offset in block.**
- **Internal fragmentation**

The logical record size, physical block size, and packing technique determine how many logical records are in each physical block.

disk space is always allocated in blocks

End 1

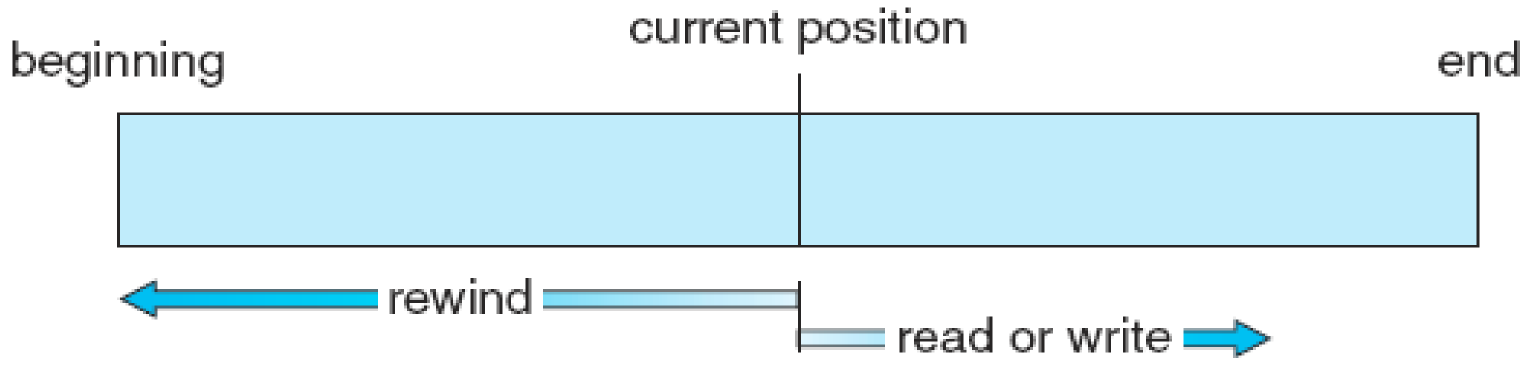
10.2 Access Methods

- Reflect different file structures.
- Different ways to store and process data.
- Criteria for File Organization
  - Rapid access
    - Needed when accessing a single record
    - Not needed for batch mode
  - Ease of update
    - File on CD-ROM will not be updated, so this is not a concern.
  - Economy of storage
    - Should be minimum redundancy in the data.
    - Redundancy can be used to speed access such as an index.
  - Simple maintenance
  - Reliability

The Pile
The Sequential File
Indexed Sequential File
Indexed File
The Direct, or Hashed File

Access Methods

- **Sequential Access**
  - read next
  - write next
  - reset
  - rewrite, no read after last write
- **sequential-access file**

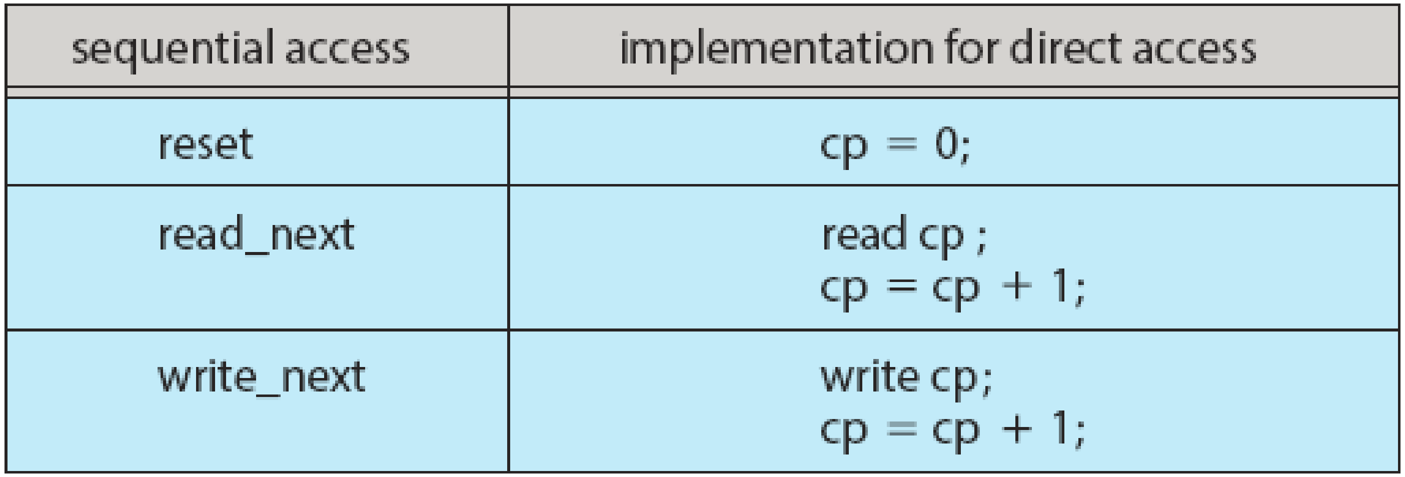


Access Methods (Cont.)

- **Direct Access**
  - **Read(n)**
  - **Write(n)**
  - **Position(n)**
    - **read next, write next**
  - **Rewrite(n)**

  n = relative block number, the first is 0.

- **simulation of sequential access on a direct-access file.**

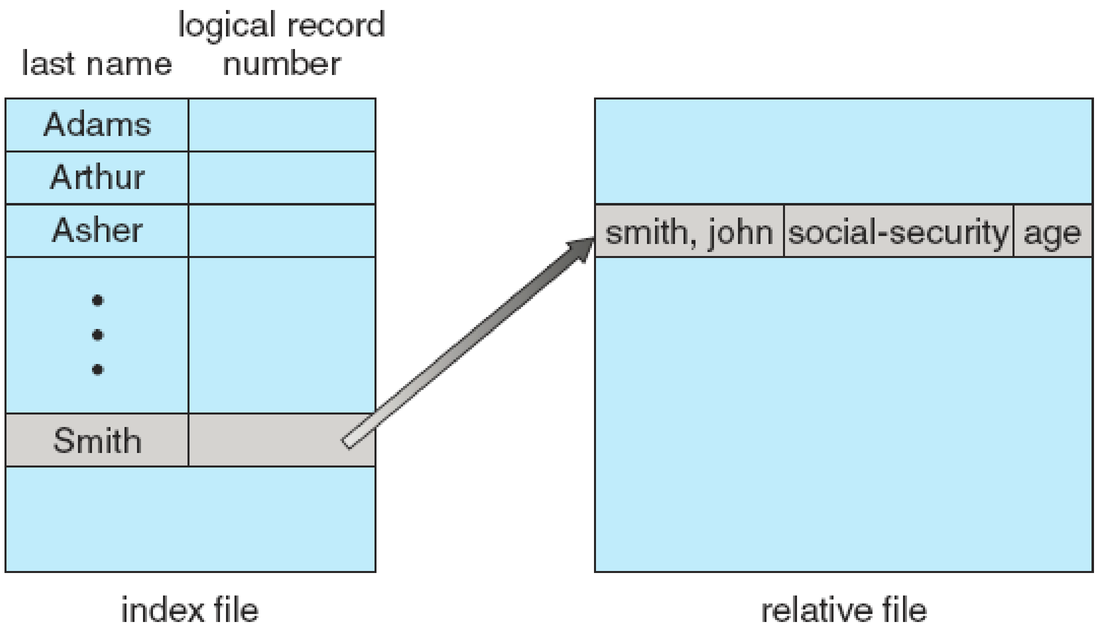


sequential access	implementation for direct access
reset	cp = 0;
read_next	read cp ; cp = cp + 1;

write_next	write cp;
	cp = cp + 1;

Other Access Methods (Cont.)

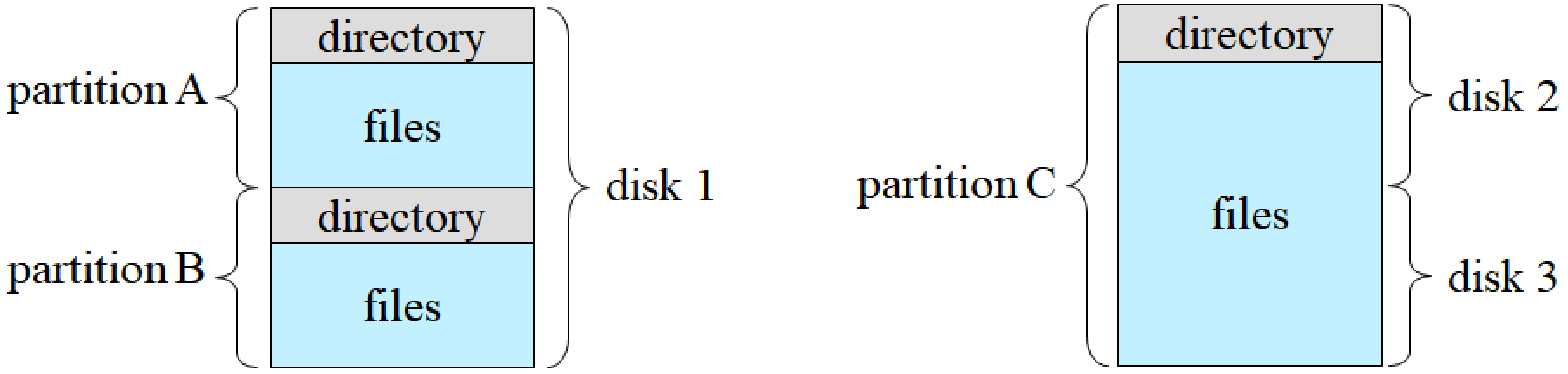
- **Can be built on top of a direct-access method.**
- **General involve creation of an index for the file.**
- **Keep index in memory for fast determination of location of data to be operated on.**
- **If index is too large, two-level or multiple-level index files.**
- **Example of index and relative files.**



End 2

10.3 Directory Structure

- Disk can be subdivided into **partitions**.
  - Each disk on a system contains at least one partition.
  - Some systems allow partitions to be larger than a disk to group disks into one logical structure.



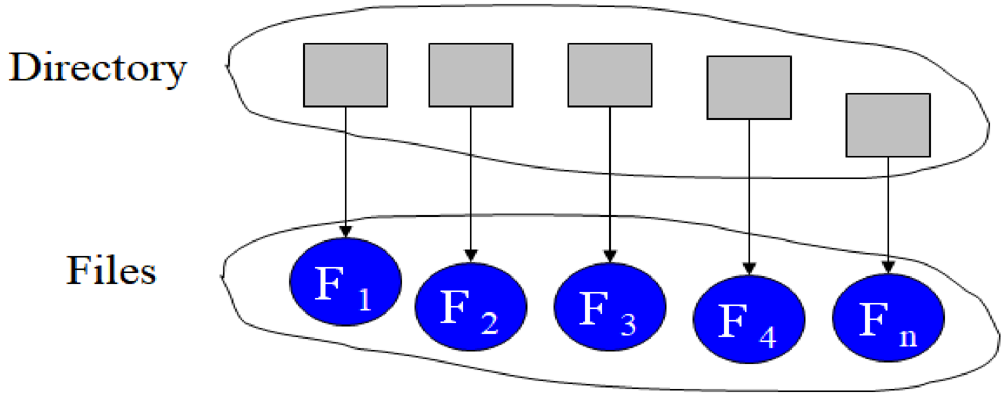
- Disk or partition can be used
  - **raw,** without a file system, e.g. swap space
  - **formatted** with a file system.

Directory Structure (Cont.)

- 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.**
  - Information such as name, location, size, and type is kept in entries in a directory.
- Disks or partitions can be **RAID** protected against failure.
  - **R**edundant **A**rrays of **I**ndependent **D**rives， **RAID**

Directory Overview

- A collection of nodes containing information about all files: Attributes, Location, Ownership.
  - Can be viewed as a symbol table that translates file names into their directory entries.
  - Provides mapping between file names and the files themselves.
  - Directory itself is a file owned by the OS.
- Both the directory structure and the files reside on disk.



* Information in a Directory Entry

- Name
- Type
- Location
- Current length
- Maximum length
- Date last accessed (for archival)
- Date last updated (for dump)
- Owner ID (who pays)
- Protection information

Operations Performed on a Directory

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

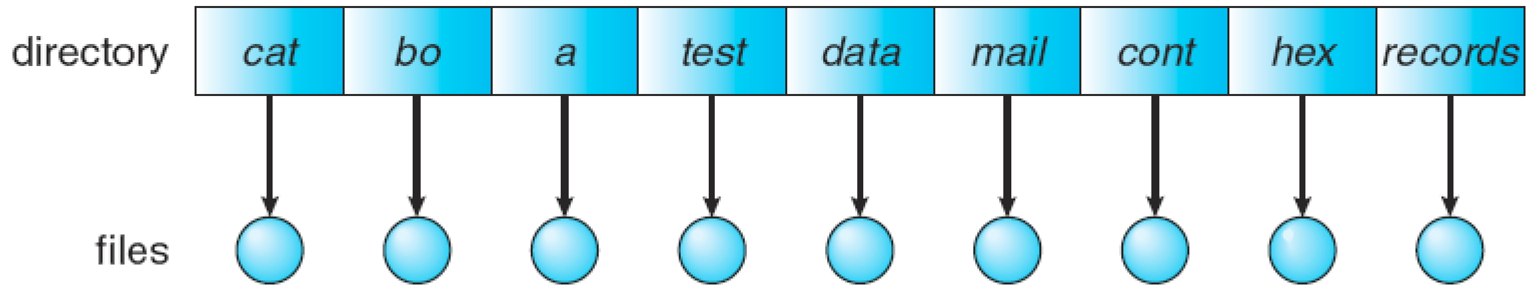
* objects of organizing Directory

- **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, …

科学规划，合理组织，使用高效

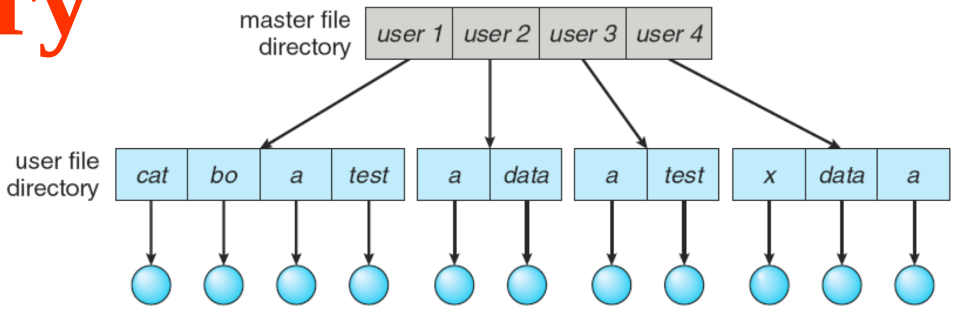
Single-Level Directory

- All files are contained in the same directory.
- List of entries, one for each file.
- Sequential file with the name of the file serving as the key.



- Naming problem
  - Files must have unique names.
  - The length of a file name.
- Grouping problem

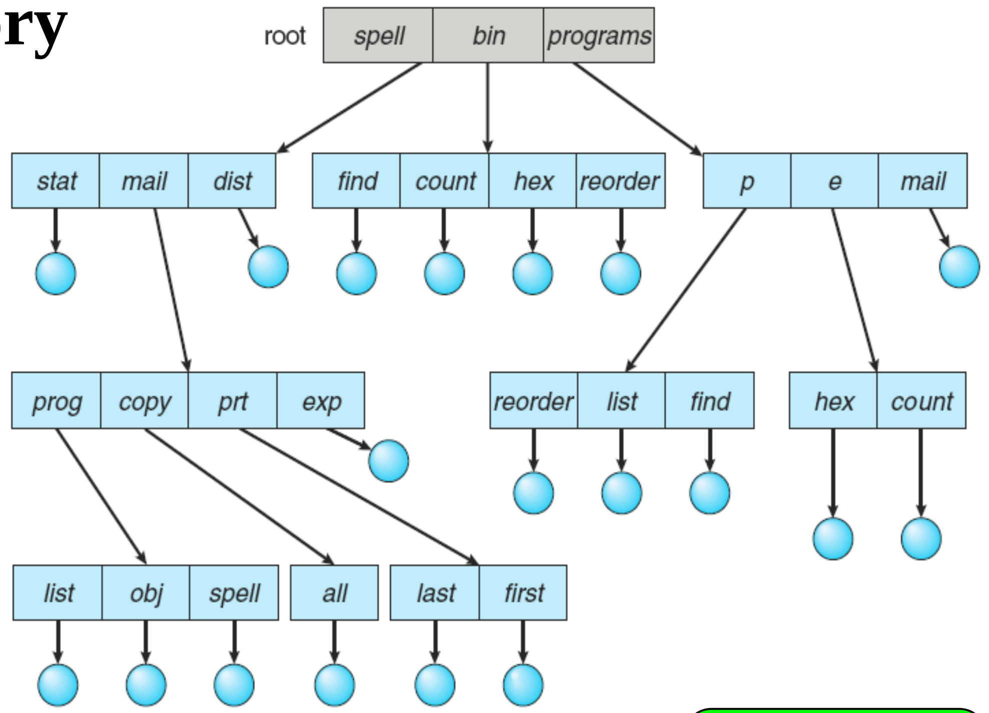
Two-Level Directory



- Create a separate directory for each user.
- A master file directory and one user file directory for each user.
  - Master file directory contains entry for each user, indexed by user name or account number.
    provides address and access control information.
  - Each user file directory is a simple list of files for that user.
- Different users may have files with the same name.
- Efficient searching.
- No grouping capability.
- File sharing?
  - Path name. defined by a user name and a file name.
  - Special user directory, containing system files.
  - Search path, the sequence of directories searched.

Tree-Structured Directories

- the most common directory structure.
- every file has a unique path name.
- Allow users to create their own sub-directories and to organize their files.
- Each user directory may have one or more sub-directories and files as entries.
- All directories have the same internal format. One bit in each entry defines the entry as a file(0) or as a subdirectory(1).



组织合理
使用高效

Tree-Structured Directories (Cont.)

- Efficient searching
- convenient Naming
- Grouping Capability
- Current directory (working directory)
  - In accounting file, a pointer/the name of the user's initial directory.
  - Copied to a local variable for this user.
- Creating a new file is done in current directory.
- Delete a file

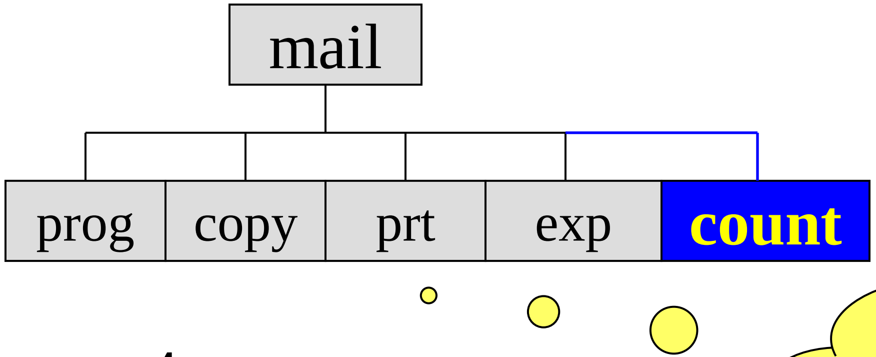
  rm <file-name>
- Change current directory.
  - cd /spell/mail/prog
- Absolute or relative path name

Tree-Structured Directories (Cont.)

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

  mkdir <dir-name>

  e.g. if in current directory /mail



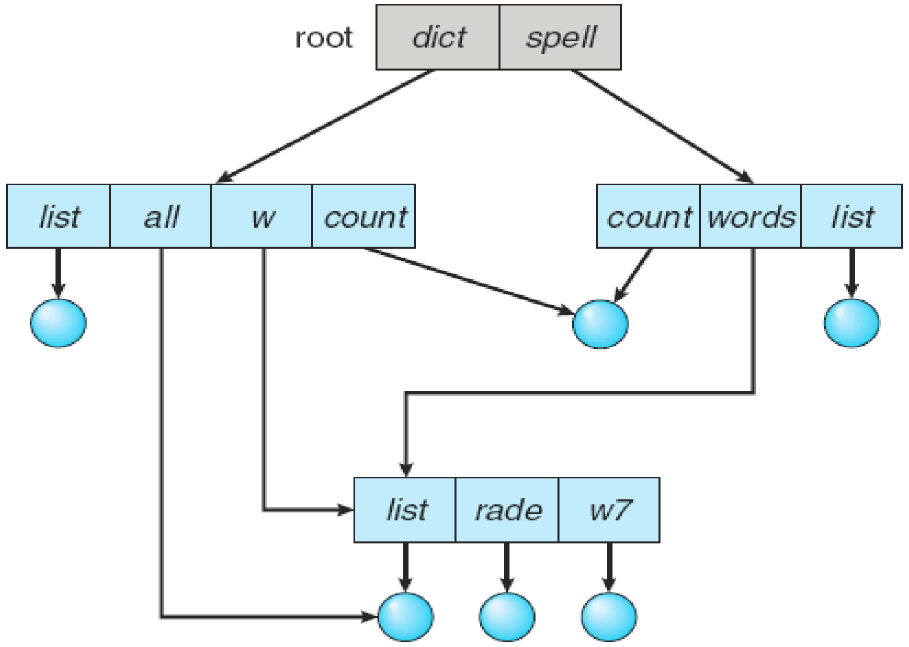
  - **mkdir count**

How to share files? e.g. working on a joint project

- **Deleting a subdirectory**
  - **deleting a empty directory. -- MS-DOS**
  - **deleting all files and subdirectories that it contains. -- UNIX**

方便，但危险

Acyclic-Graph Directories



- **Have shared subdirectories and files.**
  - **The same file or subdirectory may be in two different directories.**
- **Note: a shared file/directory is not the same as two copies of the file/directory.**

Acyclic-Graph Directories (Cont.)

- Ways implementing shared files and subdirectories:
  - Create a new directory entry, called a link. -- UNIX link, a pointer to another file or subdirectory.
    e.g. path name
    - the directory entry is marked as a link.
    - Resolve the link – follow pointer to locate the file.
    - A link is clearly different from the original directory entry.
  - duplicate all information about shared files in all sharing directories.
    - Both entries are identical and equal.
    - the original and the copy indistinguishable.
    - Problem: maintaining consistency when a file is modified?

Acyclic-Graph Directories (Cont.)

- **Problems**
  - **A file may have multiple absolute path names. Distinct file names may refer to the same file. (aliasing problem)**
    - **Traverse the entire file system, accumulate statistics on all files, copy all files to backup storage.**
  - **Deletion of the shared file.**

    ① Remove the file whenever anyone deletes it, dangling pointers.

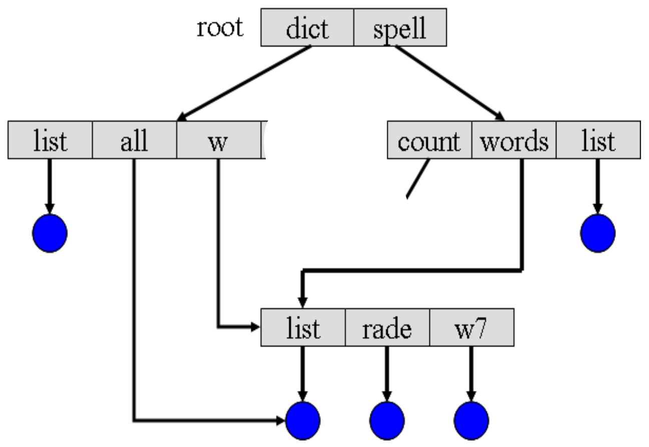
    E.g. If *dict* deletes *count*

    - **Links,**

      Deletion of a link need not affect the original file.

      Deletion of the file entry, leaving the links dangling.
    - **Duplication,**

      File-reference list, Backpointers.



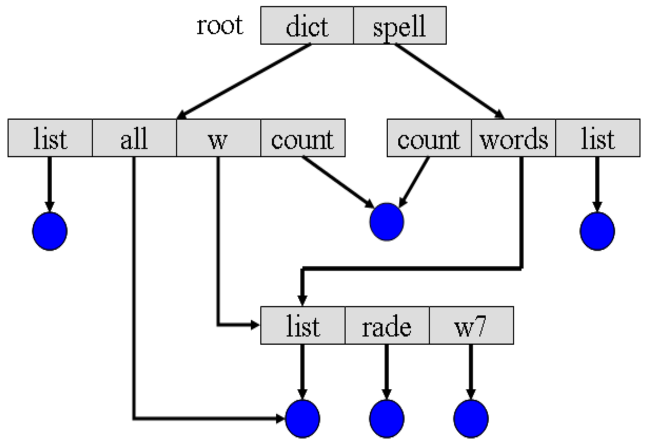
Acyclic-Graph Directories (Cont.)

- **Problems(Cont.)**
  - **Deletion of the shared file (Cont.).**
    - **② Preserve the file until all references to it are deleted.**
    - **Solutions:**

      File-reference list

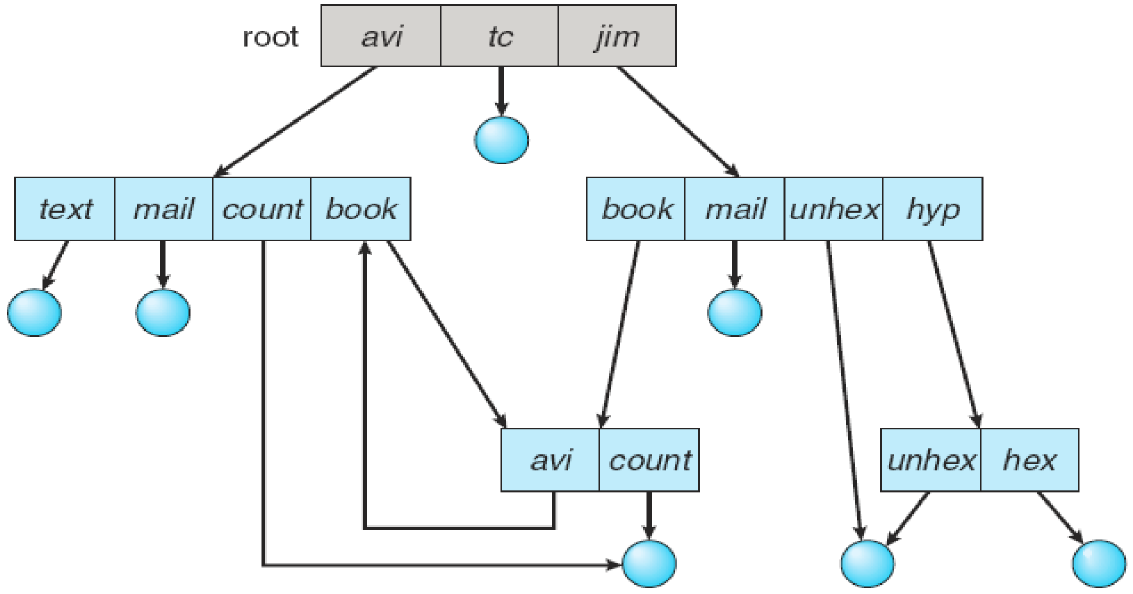
      File-reference count

      Backpointers, so we can delete all pointers. Variable size records a problem.



General Graph Directory

- The primary advantage of an acyclic graph is the relative simplicity of the algorithms
  - to traverse the graph.
    Avoid searching the shared subdirectory twice.
  - to determine when there are no more references to a file.



- A poorly designed algorithm might result in an infinite loop continually searching through the cycle and never terminating.

General Graph Directory (Cont.)

- **How to determine when a file can be deleted?**
  - **With acyclic-graph directory structures, when the value of of the reference count is 0.**
  - **With general graph directory structures, when cycles exist, the reference count may not be 0 even when it is no longer possible to refer to a directory or file.**
    - **This anomaly results from the possibility of self-referencing. (*cycle*)**
    - **Solution, using Garbage collection.**
      - **The first pass, traversing the entire file system, marking everything that can be accessed.**
      - **The second pass, collects everything that is not marked onto a list of free space.**

General Graph Directory (Cont.)

- How to avoid cycles?
  - Allow only links to file not subdirectories. guarantee no cycles.
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK.
    Very expensive!
    especially when the graph is on disk storage.
  - A simpler algorithm, bypass links during directory traversal.

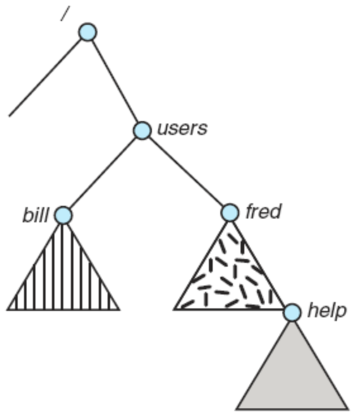
End 3

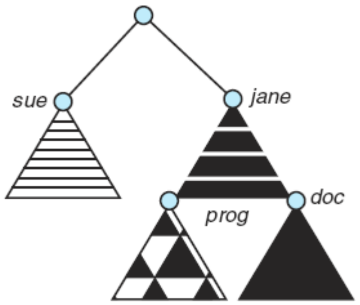
10.4 File System Mounting

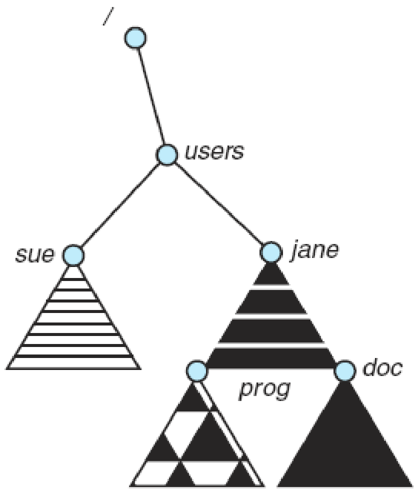
- A file system must be mounted before it can be accessed.
- The mount procedure
  - The operating system is given the name of the device, and the mount point -- the location within the file structure at which to attach the file system.
  - The operating system verifies that the device contains a valid file system by asking the device driver to read the device directory and verifying that the directory has the expected format.
  - The operating system notes in its directory structure that a file system is mounted at the specified mount point.

File System Mounting(Cont.)

Existing **Unmounted Partition** **Mount point: users**







- **Systems impose semantics to clarify functionality.**
  - **may disallow a mount over a directory that contains files.**
  - **may make the mounted file system available at that directory and obscure the directory's existing files until the file system is unmounted, terminating the use of the file system and allowing access to the original files in that directory.**
  - **may allow the same file system to be mounted repeatedly, at different mount points.**
  - **may only allow one mount per file system.**

End 4

10.5 File Sharing

- Sharing of files on multi-user systems is desirable.
- Sharing may be done through a *protection* scheme.
- More file and directory attributes are needed.
  - File / directory owner
  - File / directory user, access rights
  - File / directory user groups, access rights
- If multi-user system
  - User IDs identify users, allowing permissions and protections to be per-user.
    Group IDs allow users to be in groups, permitting group access rights.
  - Owner of a file / directory
  - Group of a file / directory

保护机制，
授权访问。
遵守规则，
安全共享。

Remote File Systems

- Uses networking to allow file system access between systems
  - 1st method: manually transferring files via programs like **FTP.**
    - used for both anonymous and authenticated access.
  - 2nd method: using distributed file systems (**DFS**).
    - remote directories are visible from a local machine.
    - much tighter integration between the machine.
  - 3rd method: automatically via the **WWW.**
    - A browser is needed.
    - uses anonymous file exchange.
  - Cloud computing is being used.

遵纪守法意识
遵守使用规则
尊重知识产权
保护知识产权

Failure Modes

- All file systems have failure modes.
- Local file system fails, reasons:
  - disk failure, disk-controller failure, cable failure, host-adapter failure.
  - corruption of directory structures or other non-user data, called metadata.
  - User or system-administrator failure
- 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.
  - If both server and client maintain knowledge of their current activities and open files, then they can seamlessly recover from a failure.
- Stateless protocols such as NFS V3 include all information in each request, allowing easy recovery but less security.
- In NFS V4, it is made stateful to prove its security, performance, and functionality.

Consistency Semantics

- **Consistency semantics specify how multiple users are to access a shared file simultaneously**
  - **Specify when modifications of data by one user will be observable by other users.**
  - **Similar to process synchronization algorithms of Chapter 6**
    - **Tend to be less complex due to disk I/O and network latency (for remote file systems).**
  - **UFS (Unix file system) implements:**
    - **A file is associated with a single physical image.**
    - **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.**

Consistency Semantics(Cont.)

- Consistency semantics specify how multiple users are to access a shared file simultaneously (Cont.)
  - AFS (Andrew File System) implemented complex remote file sharing semantics
    - The server records what its clients do and keeps the file information they are caching.
    - When a client changes a file, the server notifies other clients with a call back promise technique
    - Writes only visible to sessions starting after the file is closed.
- Immutable shared files, declared by its creator. Read-only.
  - Name may not be reused.
  - Contents may not be altered.

文件管理
版本控制

end 5

10.6 Protection

可靠性问题
保护问题

- Reliability -- duplicate copies of files
- Protection -- controlled access.
- File owner/creator should be able to control:
  - what can be done
  - by whom
- Types of access
  - Read -- read from the file.
  - Write -- write or rewrite the file.
  - Execute -- load the file into memory and execute it.
  - Append -- write new information at the end of the file.
  - Delete -- delete the file and free its space for possible reuse.
  - List -- list the name and attributes of the file.
  - Other operations, such as renaming, copying, or editing the file, may also be controlled.

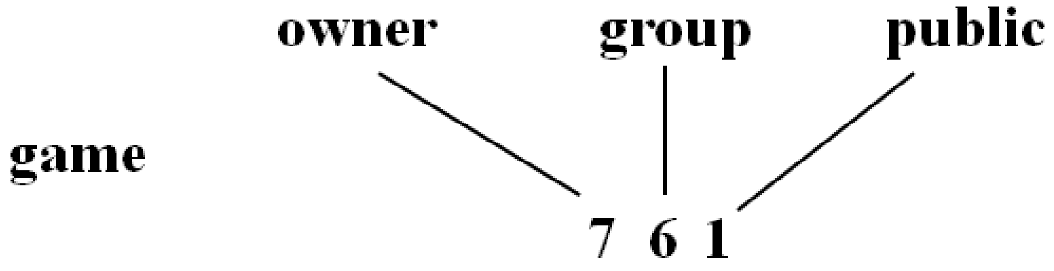
Access control

- Mode of access: read, write, execute.
- Dependent on the identity of the user.
- Access-control list (ACL): associates with each file and specifies the user name and the types of access allowed for each user.
- Three classes of users
  - Owner -- the user who created the file.
  - Group -- a set of users who are sharing the file and need similar access.
  - Universe (Public) -- all other users in the system.
- Access-control information
  - User access: RWX (111 – 7)
  - Group access: RW – (110 – 6)
  - Public access: – – X (001 – 1)

Advantage?
Problem?

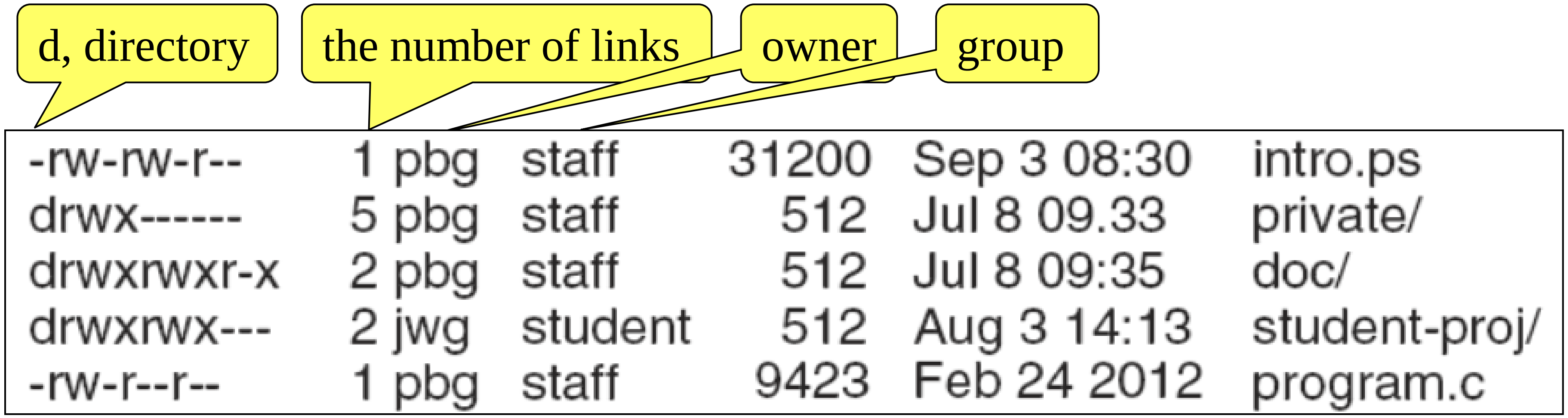
Access control (Cont.)

- groups can be created and modified only by the manager or superuser.
- 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.



- Change user's access methods, e.g rwxr--r--
  chmod G +W o=x file rwxrw---x
- Attach a group to a file
  chgrp G game.

A Sample UNIX Directory Listing



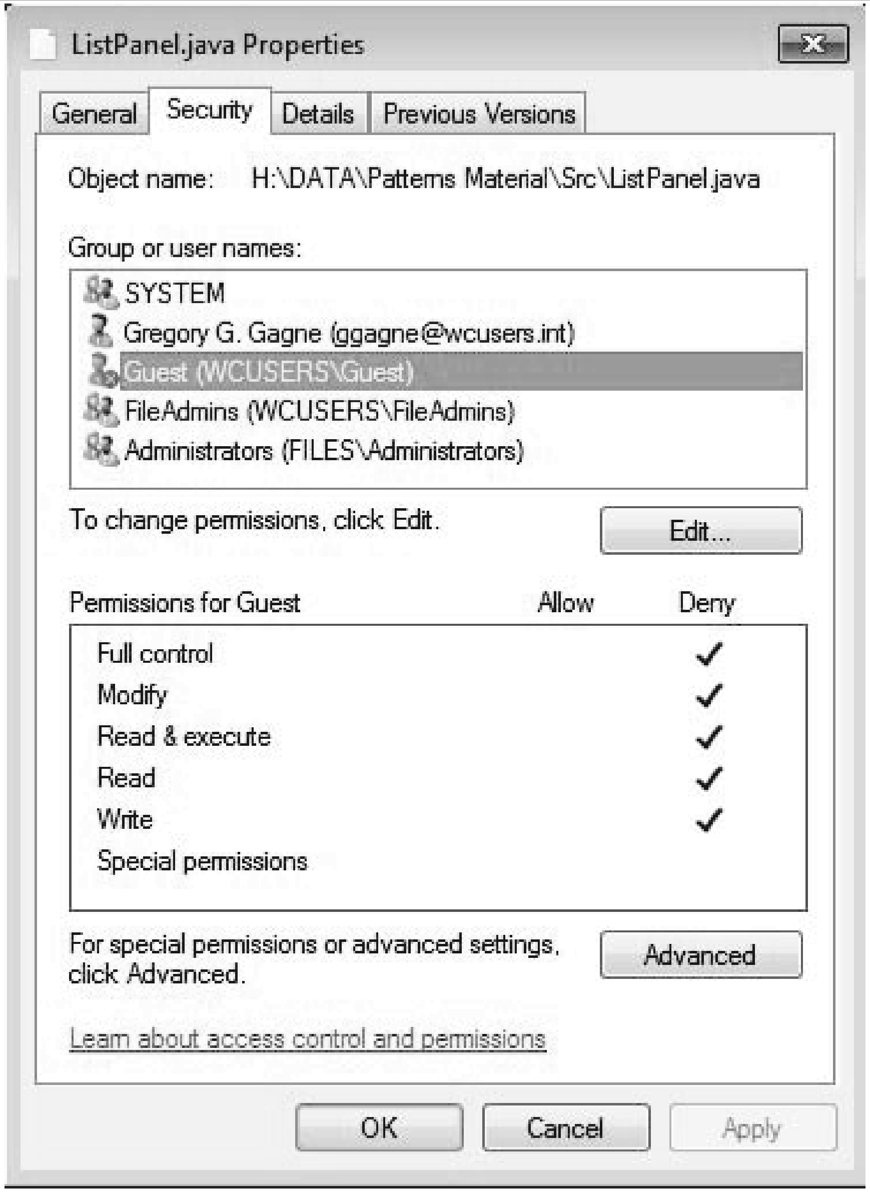
```
-rw-rw-r--    1 pbg   staff     31200  Sep 3 08:30   intro.ps
drwx------    5 pbg   staff       512  Jul 8 09.33   private/
drwxrwxr-x    2 pbg   staff       512  Jul 8 09:35   doc/
drwxrwx---    2 jwg   student     512  Aug 3 14:13   student-proj/
-rw-r--r--    1 pbg   staff      9423  Feb 24 2012   program.c
```

- combining approaches, when the optional ACL permissions are set on a file. E.g. Solaris

```
19 -rw-r--r--+ 1 jim staff 130 May 25 22:13 file1
```

- commands, setfacl and getfacl, is used to manage the ACLs.
- when permission and ACLs conflict, ACLs precedence.
  - general rule -- specificity should have priority.

Windows Access-control List Management



ListPanel.java Properties

General | Security | Details | Previous Versions

Object name: H:\DATA\Patterns Material\Src\ListPanel.java

Group or user names:

- SYSTEM
- Gregory G. Gagne (ggagne@wcusers.int)
- Guest (WCUSERS\Guest)
- FileAdmins (WCUSERS\FileAdmins)
- Administrators (FILES\Administrators)

To change permissions, click Edit.

Permissions for Guest	Allow	Deny
Full control		✓
Modify		✓
Read & execute		✓
Read		✓
Write		✓
Special permissions		

For special permissions or advanced settings, click Advanced.

Learn about access control and permissions

End 6

Homework (page 408)

10.2

10.9