OPERATING SYSTEMS

Chapter 03a

File-System Interface

Khoa Công Nghệ Thông Tin
Trường Đại Học Khoa Học Tự Nhiên
ĐHQG-HCM

GV: Thái Hùng Văn

Outline

- File Concept
- Access Methods
- Disk and Directory Structure
- File-System (FS) Mounting
- File Sharing
- Protection

Outline

- To explain the function of FSs
- To describe the interfaces to FSs
- To discuss FS design tradeoffs, including access methods, file sharing, file locking, and directory structures
- To explore FS protection

File Concept

- Computer store information in storage media (*disk, disc, flash card,..*). The OS provides a logical view of them, and this logical storage unit is a file.
- The information stored in files are not lost during power failures. A file is named collection of related information that is stored on physical storage.
- A file is a sequence of bits, bytes, lines, or records defined by its owner or creator.

File Types

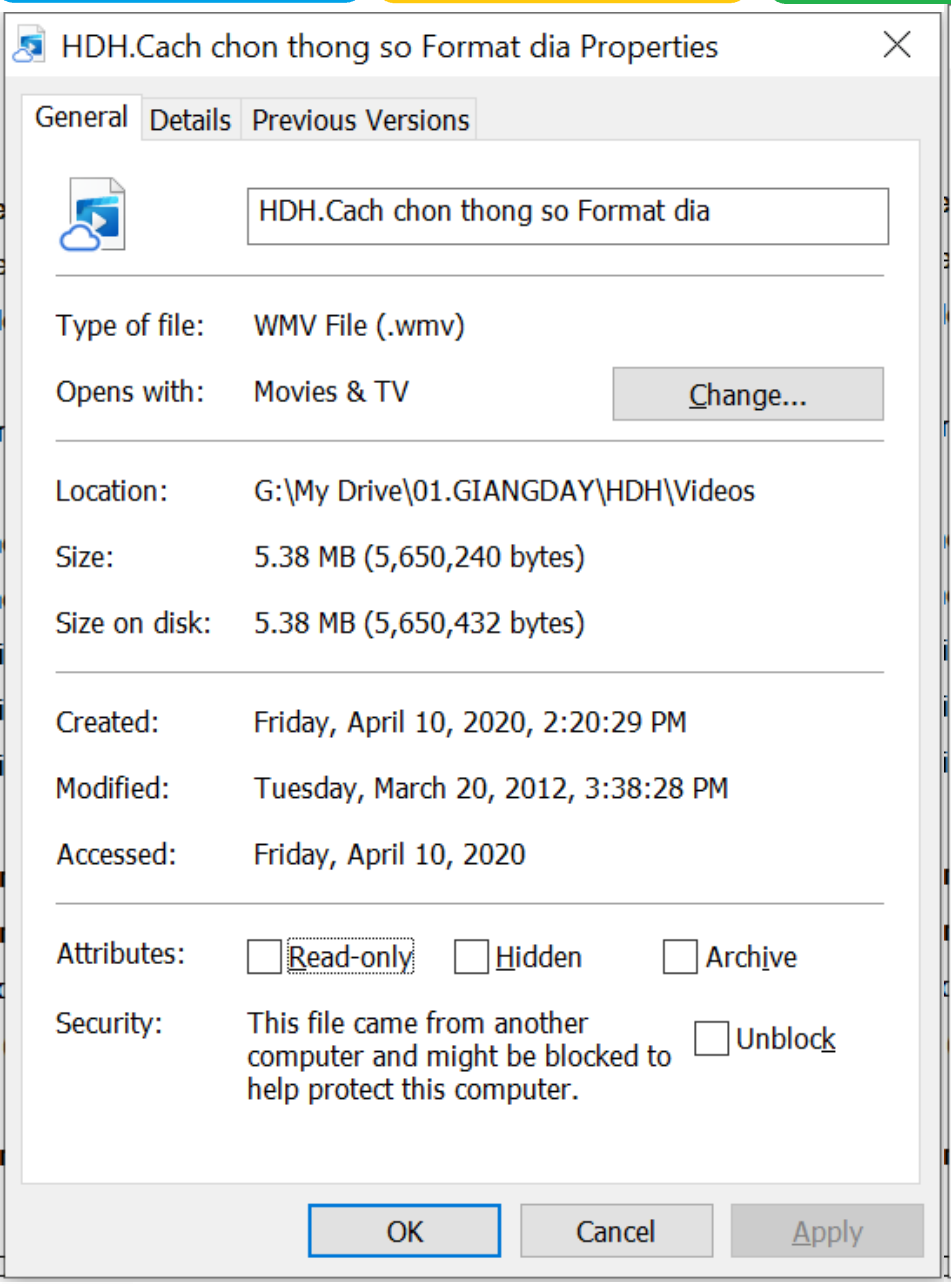
- Text file – It has a sequence of characters.
- Source file – It has subroutines and function that are compiled later *(to object / executable file)*
- Executable file – The binary code that the loader brings to memory for execution *(is stored in an exe file)*
- Image file – It has visual information such as photographs, vectors art and so on.
- Audio /Video file

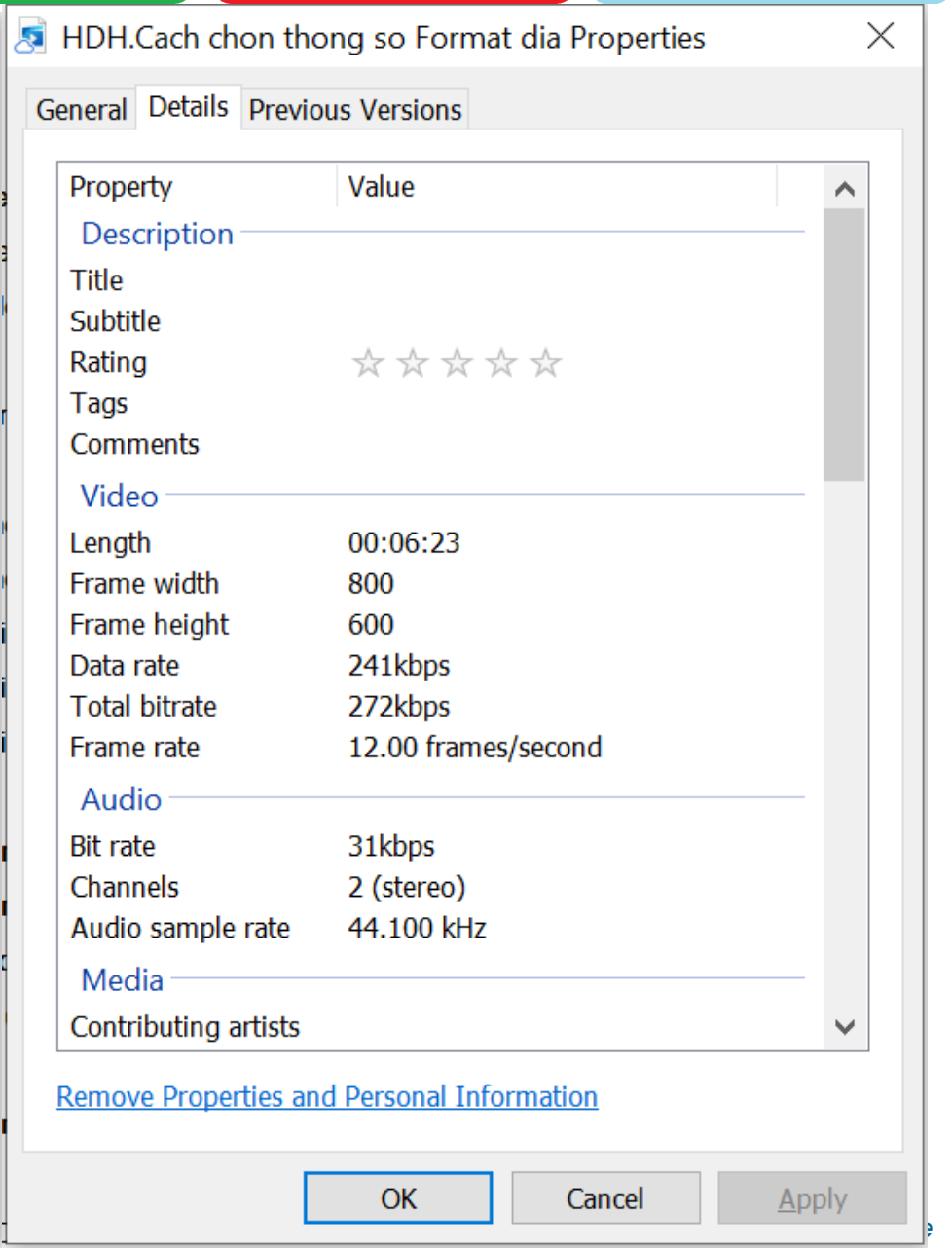
File Attributes

They can be different on many OSs. Common properties:

- **Name** – unique name for a human to recognize the file.
- **Identifier** – A unique number tag that identifies the file in the FS.
- **Type** – Information about the file to get support from the system.
- **Size** – The current size of the file in bytes, kilobytes, or words.
- **Access Control** – Defines who could read, write, execute, change, or delete the file in the system.
- **Time, Date**, and **User identification** –kept for date created, last modified, or accessed and so on.

An Video file properties on Windows OS





File Operations

- **Create**
- **Write** – at write pointer location
- **Read** – at read pointer location
- **Reposition within file** - seek
- **Delete**
- **Truncate**
- ***Open*** **(*F_i*)** – search the directory structure on disk for entry *F_i*, and move the content of entry to memory

Open Files

Several pieces of data are needed to manage open files:

- Open-file table: tracks 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

- Provided by some operating systems 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 access to a file
- Mandatory or advisory:
  - Mandatory – access is denied depending on locks held and requested
  - Advisory – processes can find status of locks and decide what to do

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

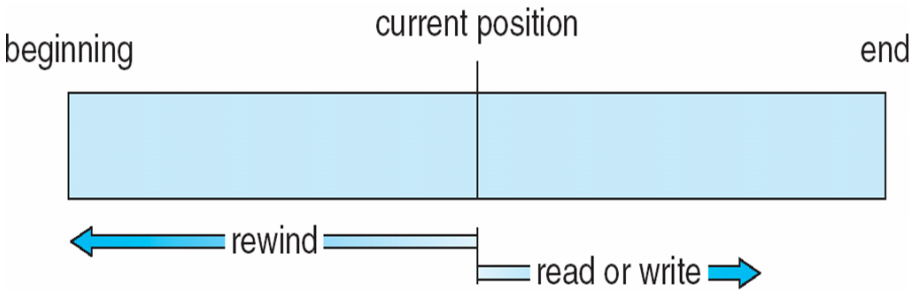
Access Methods

- A file is fixed length logical records
- **Sequential Access**
- **Direct Access**
- **Other Access Methods**

Sequential Access

- Operations
  - **`read next`**
  - **`write next`**
  - **`Reset`**
  - no read after last write (rewrite)

- Figure



Direct Access

- Operations
  - **`read n`**
  - **`write n`**
  - **`position to n`**
    - **`read next`**
    - **`write next`**
    - **`rewrite n`**

  n = relative block number

- Relative block numbers allow OS to decide where file should be placed

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;

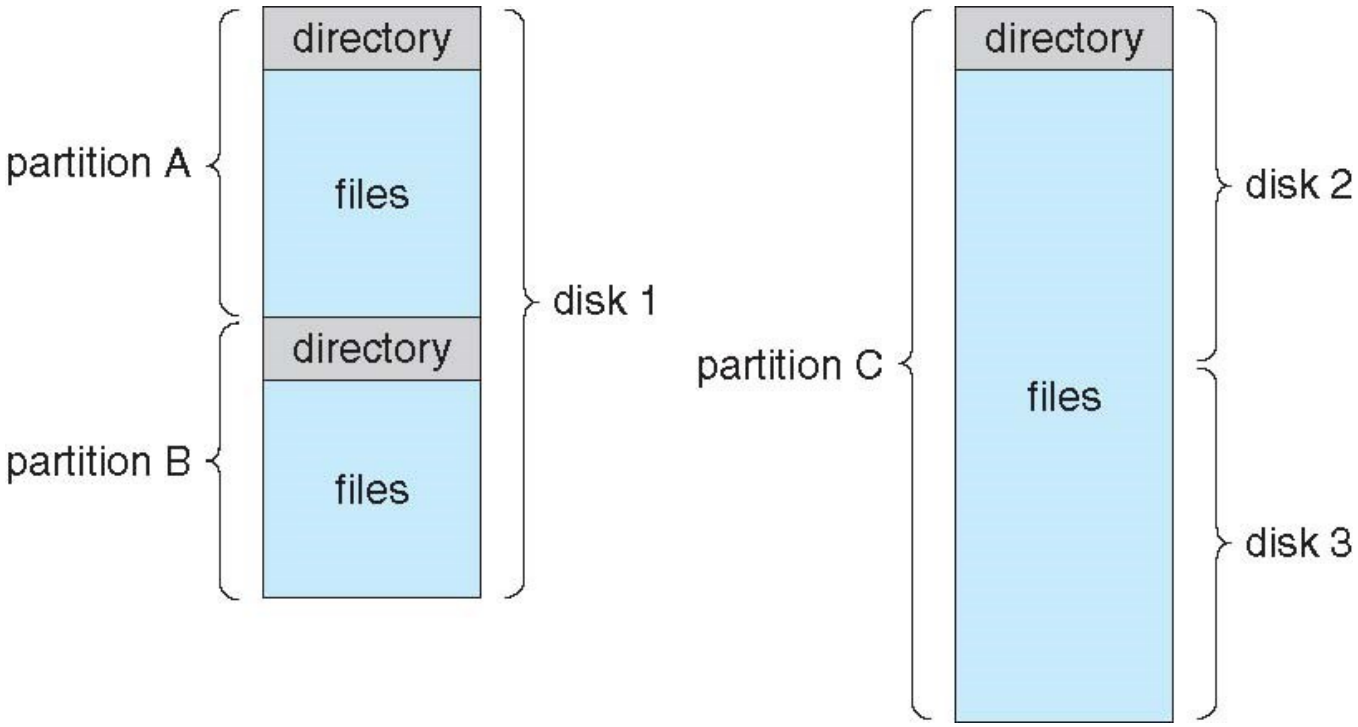
Other Access Methods

- Can be other access methods built on top of base methods
- General involve creation of an index for the file
- Keep index in memory for fast determination of location of data to be operated on (consider Universal Produce Code plus record of data about that item)
- If the index is too large, create an in-memory index, which an index of a disk index
- IBM indexed sequential-access method (ISAM)
  - Small master index, points to disk blocks of secondary index
  - File kept sorted on a defined key
  - All done by the OS
- VMS OS provides index & relative files as another example

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 FS, or formatted with a FS
- Partitions also known as minidisks, slices
- Entity containing FS is known as a volume
- Each volume containing a FS also tracks that FS's info in device directory or volume table of contents
- In addition to general-purpose FSs there are many special-purpose FSs, frequently all within the same OS or computer

A Typical File-system Organization

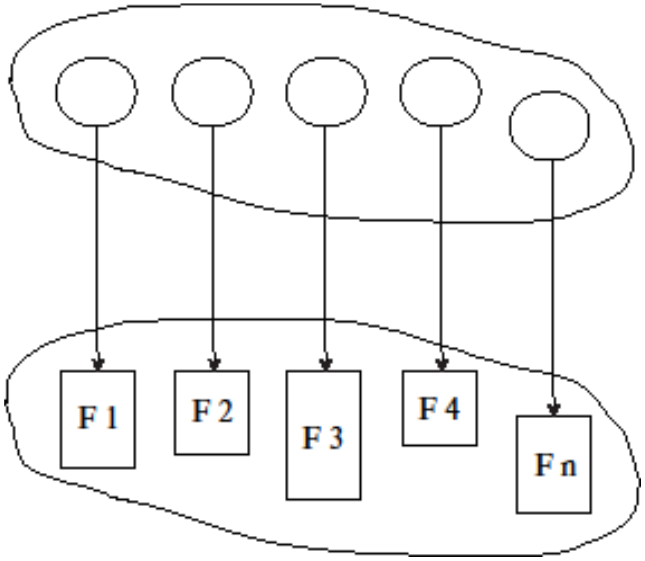


Types of File Systems

- We mostly talk of general-purpose file systems
- But systems frequently have may file systems, some general- and some special- purpose
- Consider Solaris has
  - tmpfs – memory-based volatile FS for fast, temporary I/O
  - objfs – interface into kernel memory to get kernel symbols for debugging
  - ctfs – contract file system for managing daemons
  - lofs – loopback file system allows one FS to be accessed in place of another
  - procfs – kernel interface to process structures
  - ufs, zfs – general purpose file systems

Directory Structure

- A collection of nodes containing information about all files



- Both the directory structure and the files reside on disk

Operations Performed on Directory

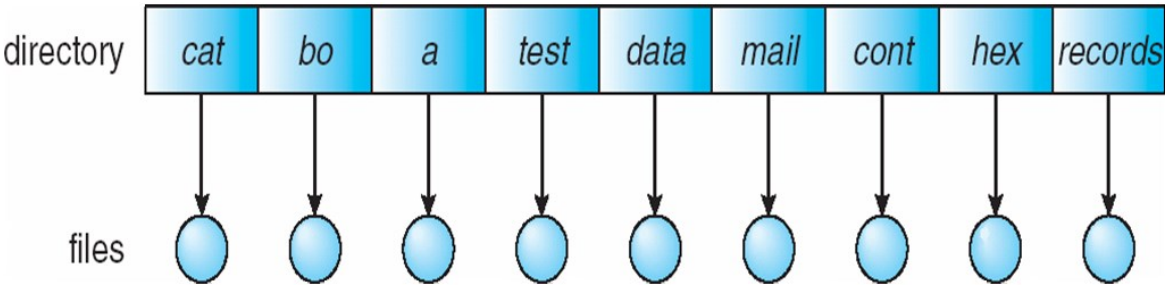
- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

Directory Organization

- 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

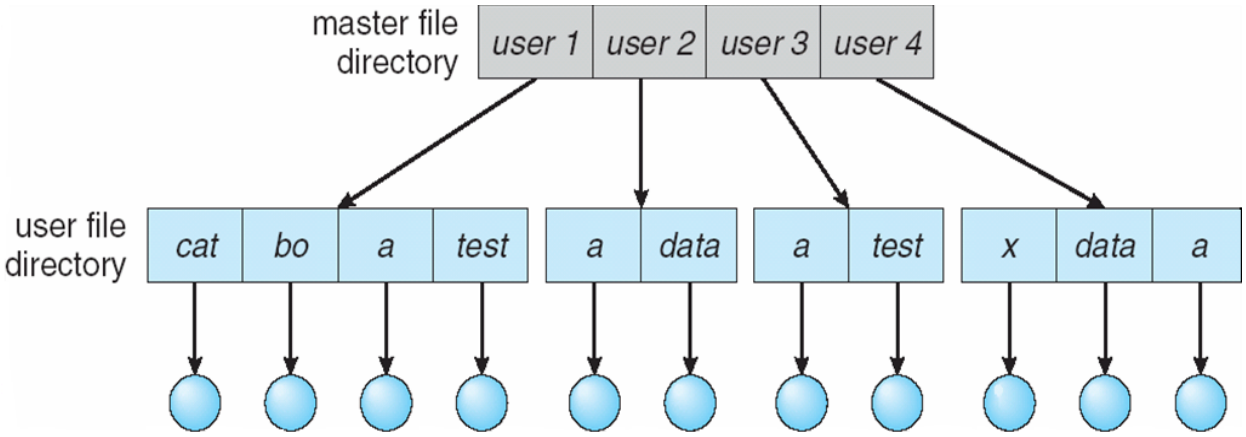
- A single directory for all users



- Naming problem
- Grouping problem

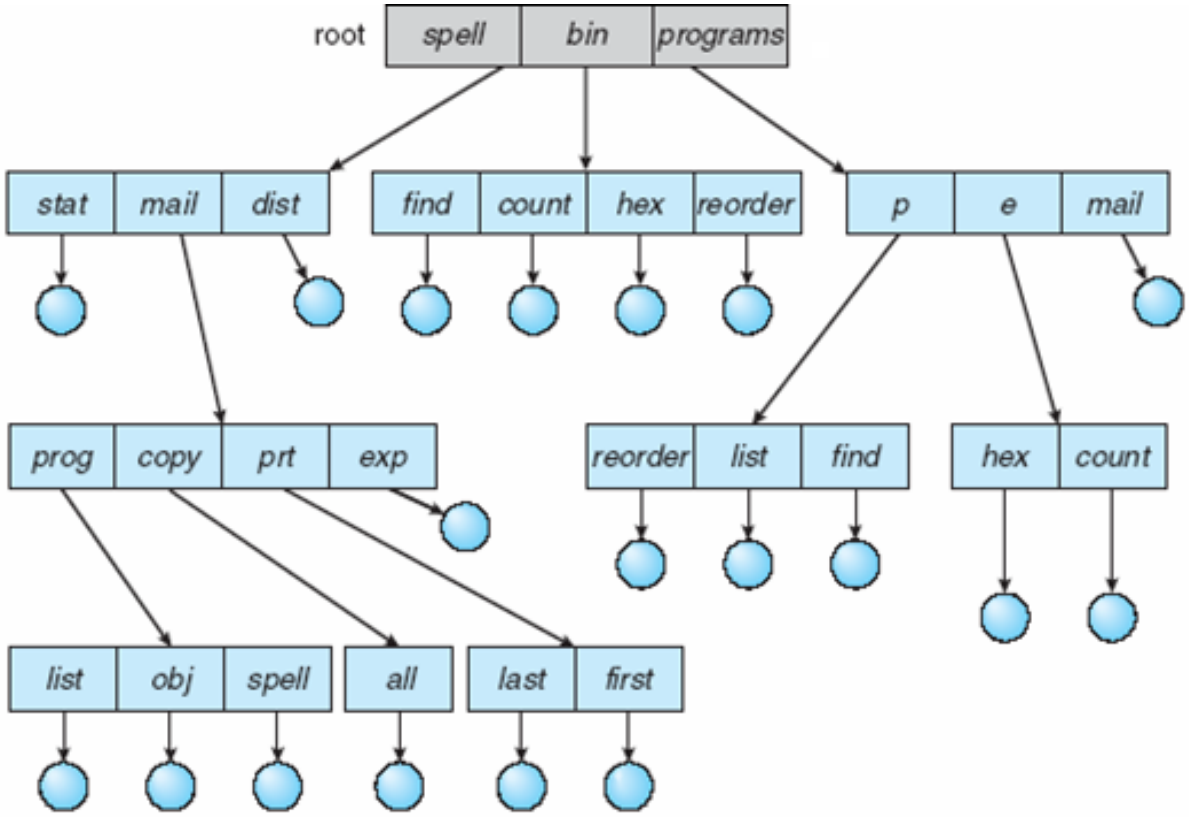
Two-Level Directory

- Separate directory for each user



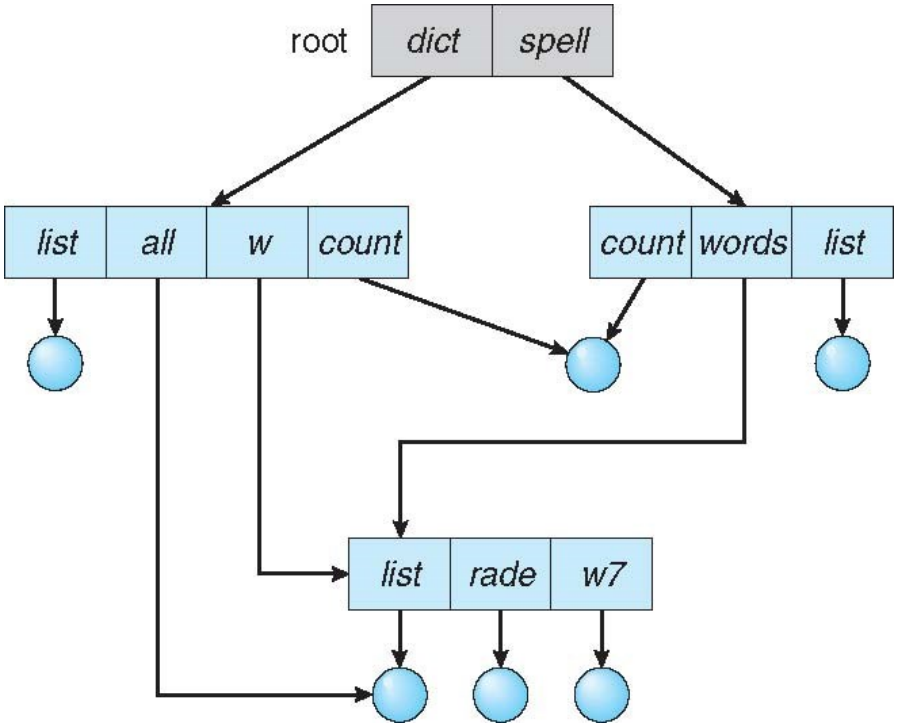
  - Path name
  - Can have the same file name for different user
  - Efficient searching
  - No grouping capability

Tree-Structured Directories



Acyclic-Graph Directories

- Have shared subdirectories and files
- Example

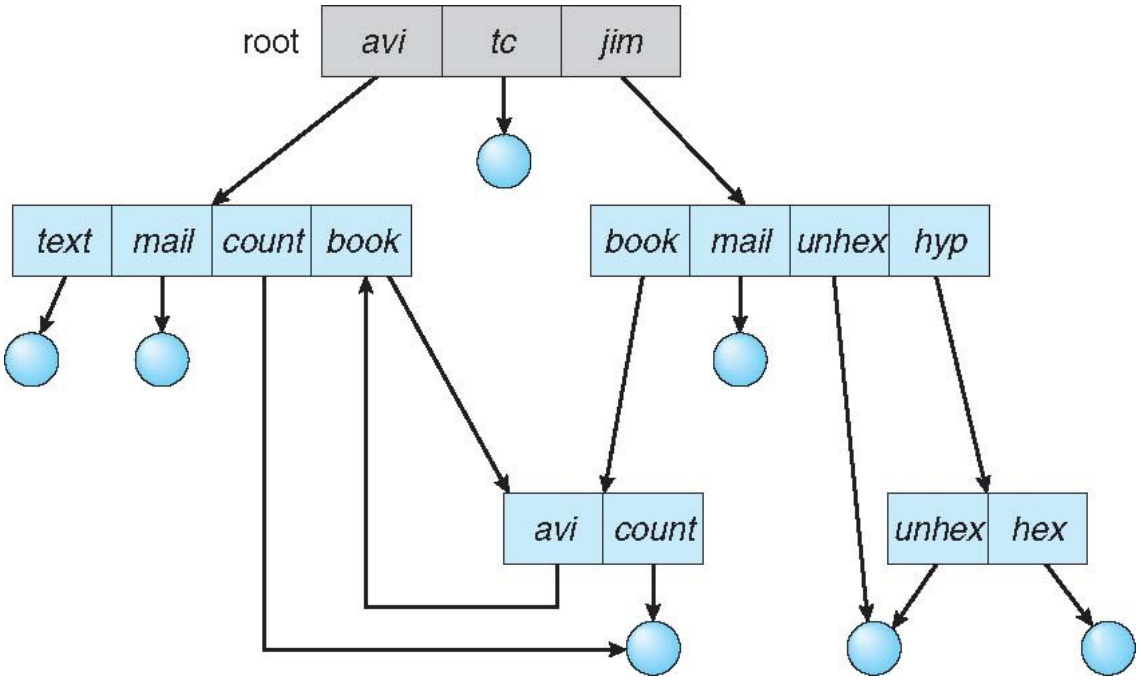


Acyclic-Graph Directories (Cont.)

- Two different names (aliasing)
- If ***dict*** deletes **w/*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

General Graph Directory

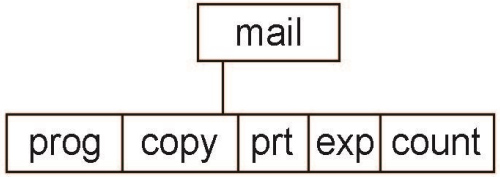


- How do we guarantee no cycles?
  - Allow only links to files not subdirectories
  - Garbage collection
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK

Current Directory

- Can designate one of the directories as the current directory
  - **`cd /spell/mail/prog`**
  - **`type list`**
- Creating and deleting a file is done in current directory
- Example of creating a new file
  - If in current directory is **`/mail`**
  - The command

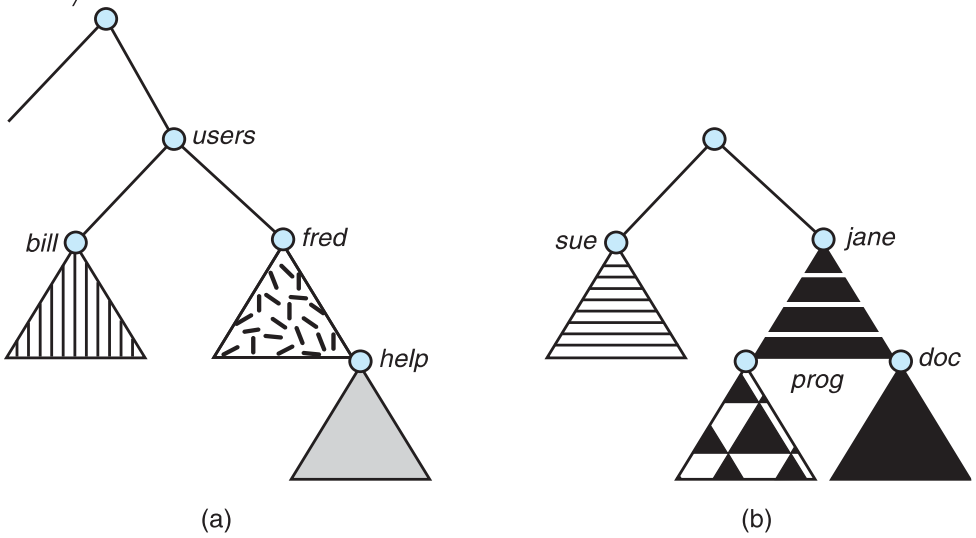
    mkdir <dir-name>
  - Results in:



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

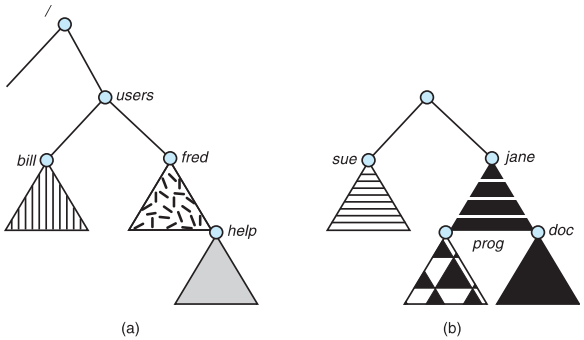
File System Mounting

- A file system must be mounted before it can be accessed
- Figure (a) is a mounted file system that can be accessed by users.
- Figure (b) is an unmounted files system that cannot be accessed by users

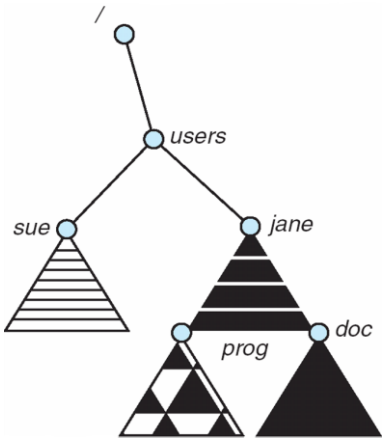


Mount Point

- Consider the file system of previous slide:



- Mounting (b) over "users" results in



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
- 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

File Sharing – Remote File Systems

- Uses networking to allow FS access between computing OSs
  - Manually via programs like FTP
  - Automatically, seamlessly using distributed file systems
  - Semi automatically via the world wide web
- Client-server model allows clients to mount remote FSs 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 is standard Windows protocol
  - Standard OS file calls are translated into remote calls
- Distributed Information Systems such as LDAP, DNS, NIS, Active Directory implement unified access to information needed for remote computing

File Sharing – Failure Modes

- All file systems have failure modes
  - For example, corruption of directory structures or other non-user data, called metadata
- 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 v3 include all information in each request, allowing easy recovery but less security

File Sharing – Consistency Semantics

- Specify how multiple users are to access a shared file simultaneously
  - Similar to Ch 5 process synchronization algorithms
    - Tend to be less complex due to disk I/O and network latency (for remote file systems
  - Andrew File System (AFS) implemented complex remote file sharing semantics
  - 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

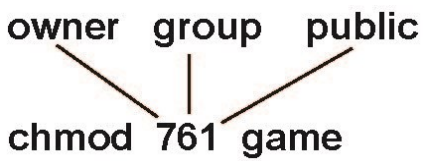
- File owner/creator should be able to control:
  - What can be done
  - By whom
- Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

Access Lists and Groups in Unix

- Mode of access: read, write, execute
- Three classes of users on Unix / Linux

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



- 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 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/
```

Windows Access-Control List Management