Amirkabir University of Technology
(Tehran Polytechnic)
Department of Computer Engineering

File System Interface

Hamid R. Zarandi
h_zarandi@aut.ac.ir

Why file system is important?

- For most users, File System (FS) is the most visible aspect of an OS
- Provides mechanism to access data/programs on storage
- Any FS consists of two distinct parts
  - A collection of Files
  - A directory structure that organizes and provides information about all files in the system

What is a file?

- **File**: A contiguous logical address space, logical storage unit
- Types
  - Data
    - Numeric, text, data, photo, music, etc
  - Program
- Contents defined by file's creator, many types are
  - Text file
    - A sequence of characters
  - Source file
    - A sequence of functions
  - Executable file
    - A series of code sections that loader can bring into memory and execute

File types; name and 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 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
- Many variations, including extended file attributes such as file checksum

File operations

- File is an abstract data type
- 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
- *Close (F_i)*
  - Move the content of entry F_i in memory to directory structure on disk

Open files

- **Open (Fi): move the content of a file to memory**
- **Search the directory structure on disk for the file**
- **To avoid constant searching: open() system should be called before a file is first used**
  - **Open-file table: tracks all open files**
    - **Per-process table**
    - **System-wide table**
- **When the file is no longer being actively used, it is closed by the process, and OS removes its entry from open-file table using *Open Count***

Other information for an open file

- **File pointer**
  - Pointer to last read/write location, per process that has the file open
- **File-open count**
  - Counter of the 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**
  - Moving data access information to memory
- **Access rights**
  - Per-process access mode information

Locking in open file

- File locks allows one process to lock a file, prevent other process from gaining access to it

- Similar to reader-writer locks
  - Shared lock similar to reader lock
    - Several processes can acquire concurrently
  - Exclusive lock similar to writer lock
    - Only on process can acquire it

Other locking mechanisms

- **Mandatory**
  - OS will prevent access until the exclusive lock is released
    - Windows®
- **Advisory**
  - OS will not prevent applications from acquiring access to a file, and the application must be developed to manually acquire the lock before accessing the file
    - UNIX®

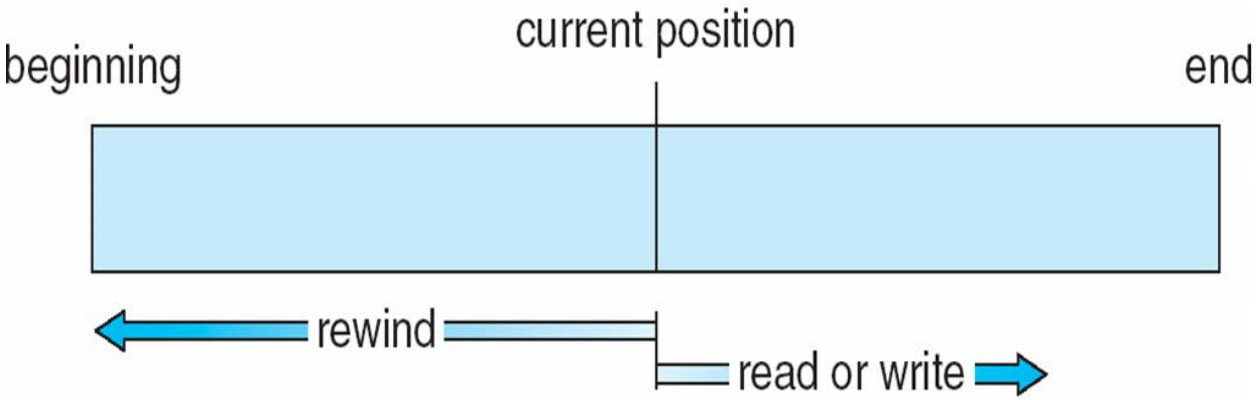
File structure

- Files must conform to structures that are understood by OS
  - OS requires an executable file has a specific structure; it can determine where in memory to load the file, what the location of first instruction is

- Support of multiple file structures?
  - Size of OS could be big; it needs to contain codes to support these file structures
  - Severe problems may result if OS does not support some file structures

File access methods: 1) Sequential access

- Simplest and most common
- Based on tape model of a file
- Processing information in a file is in order: one record after the other
- A read operation, read_next()
  - Reads the next portion of the file and automatically advances a file pointer
- A write operation, write_next()
  - Appends to the end of the file and advances to the end of the newly written information



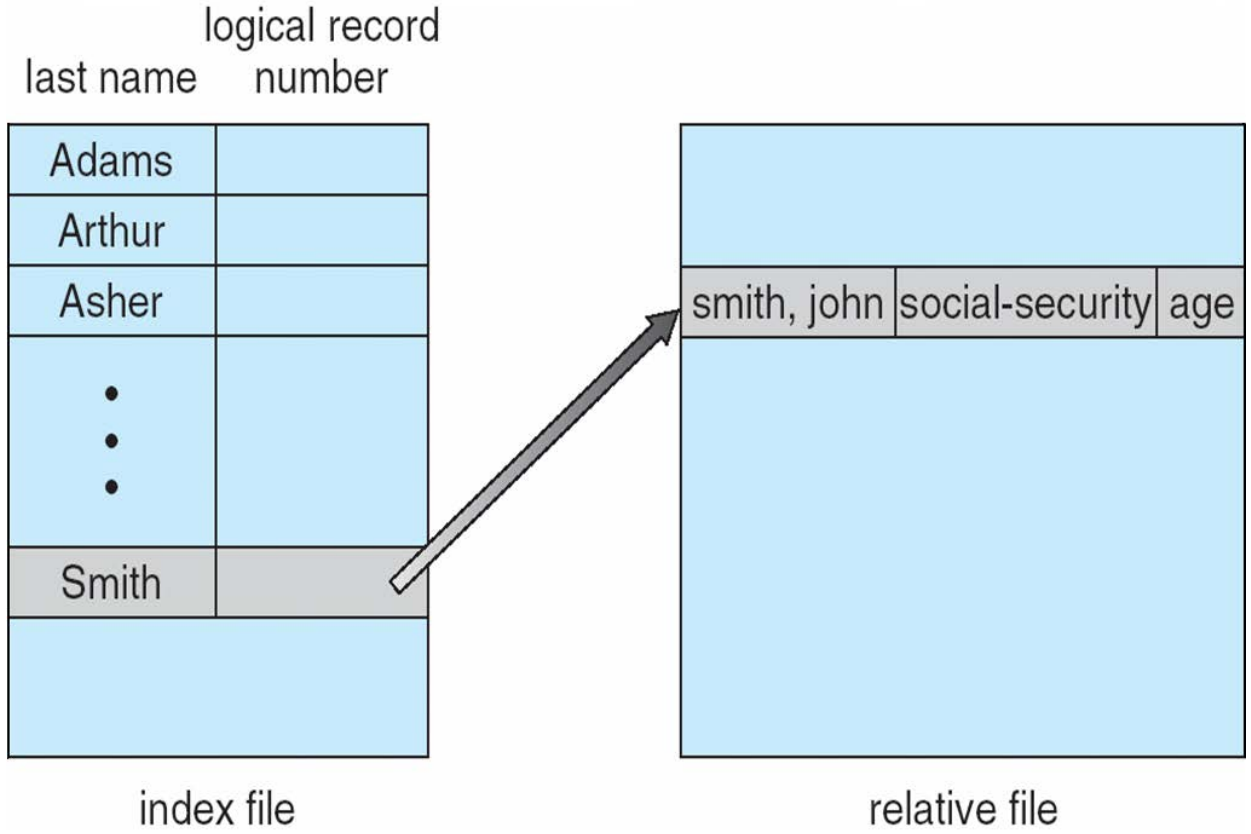
File access methods: 2) Direct access

- Based on a disk model of a file
- A file is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order
- Immediate access to large amount of information
  - Databases are often of this type
- read(*n*) rather than read_next()
  - *n* is block number
- write(*n*) rather than write_next()

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 built on top of base methods
- Creation of an index for a file
  - Having pointers to various blocks
- Keep index in memory for fast determination of location of data



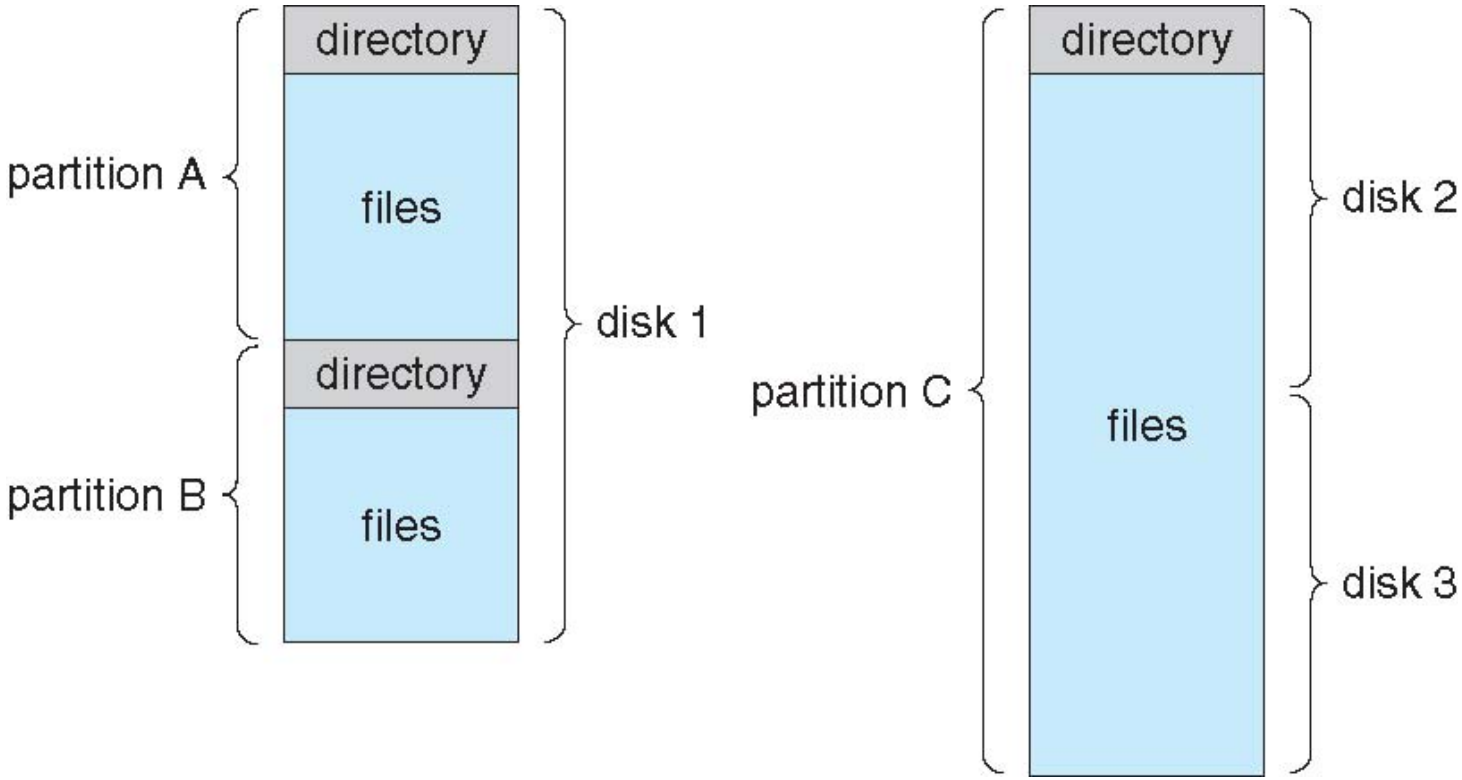
IBM's indexed sequential access method (ISAM)

Directory & Disk Structure

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
- Partitions also known as minidisks, slices
- Entity containing file system known as a volume
- Each volume contains information about the files in the system
  - This information is kept in entries in a device directory or volume table of contents
- The device directory, (known as the directory), records information such as name, location, size, and type for all files on that volume.

A typical file-system organization



Types of file systems

- Systems frequently have many 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

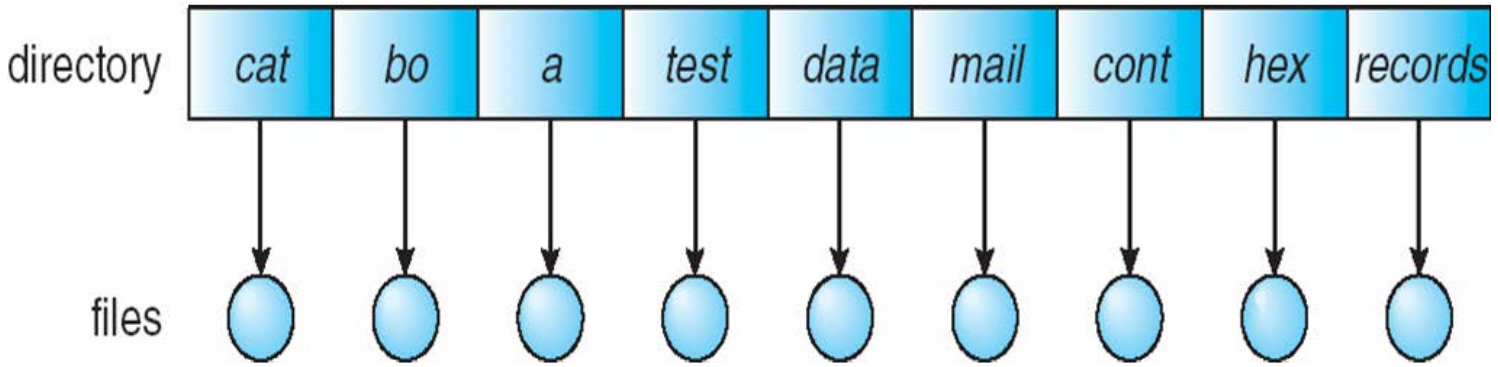
- **Directory**
  - Can be viewed as a symbol table that translates file names into their directory entries
- **Both the directory structure and the files reside on disk**
- **Operations on directories**
  - Search for a file
  - Create a file
  - Delete a file
  - List a directory
  - Rename a file
  - Traverse the file system

Directory organization

- 1) Single-level directories
- 2) Two-level directories
- 3) Tree-structure directories
- 4) Acyclic-graph directories
- 5) General graph directories

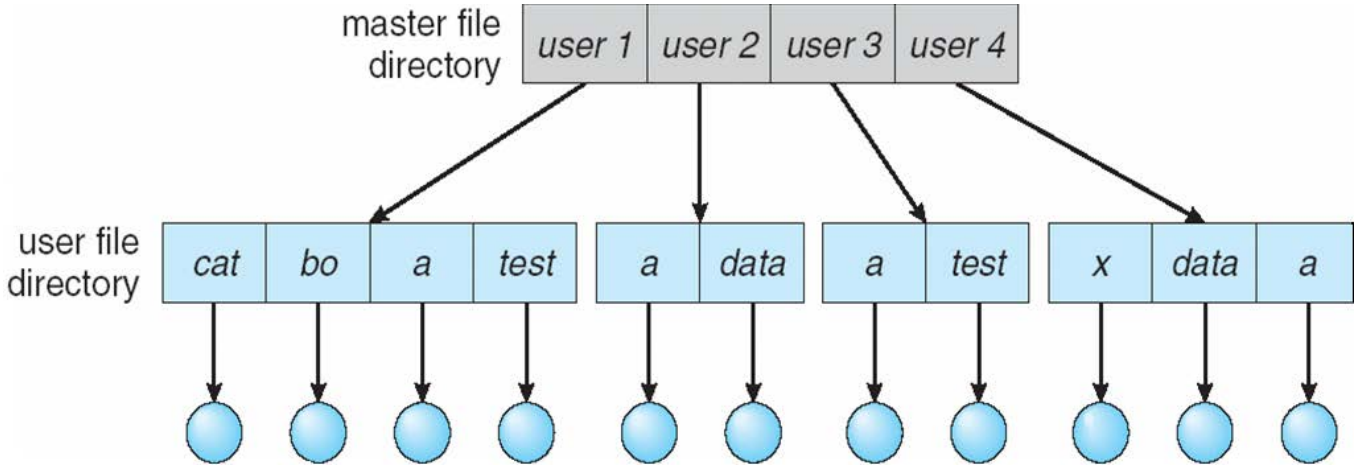
1) Single-level directory

- A single directory for all users
- Naming problem: they must have unique names
- Grouping problem



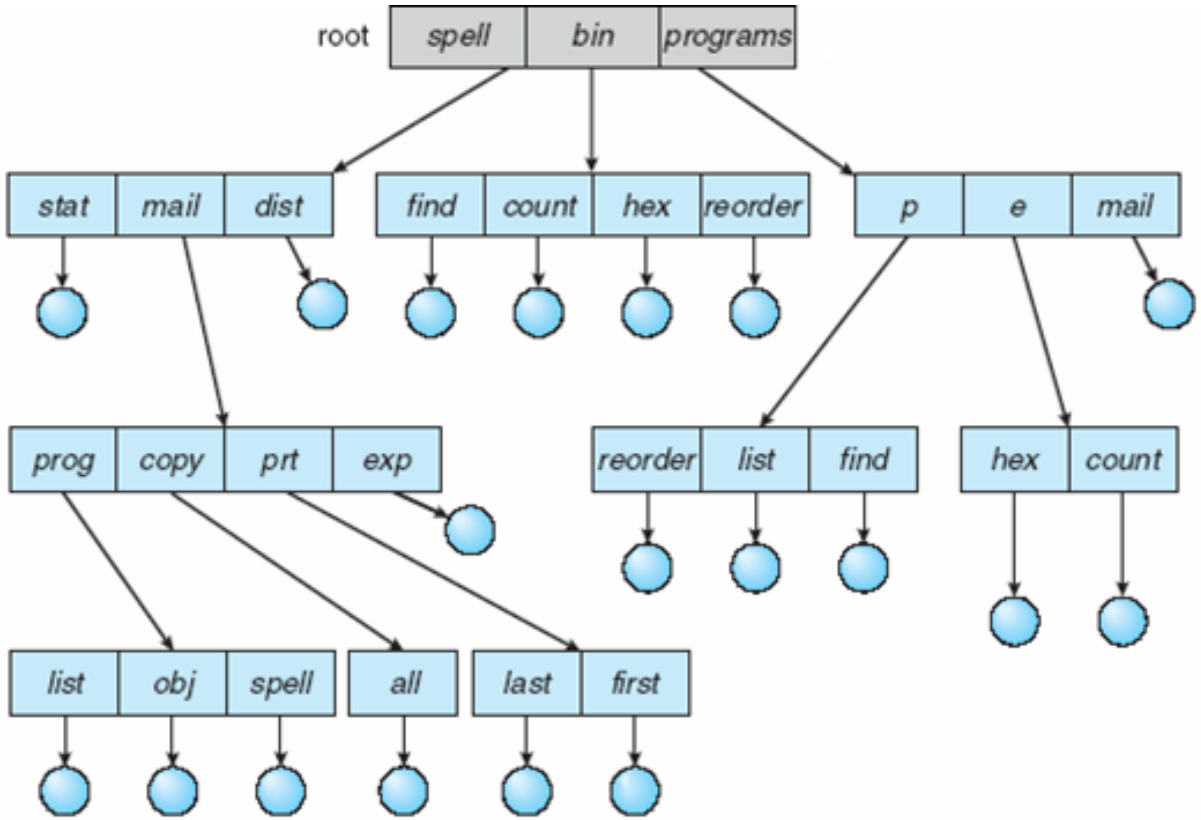
2) Two-level directory

- Separate directory for each user
- Can have the same file name for different users
- Efficient searching
- Path name: two level path, e.g., /userN/file.txt
- No grouping capability
- Sharing problem



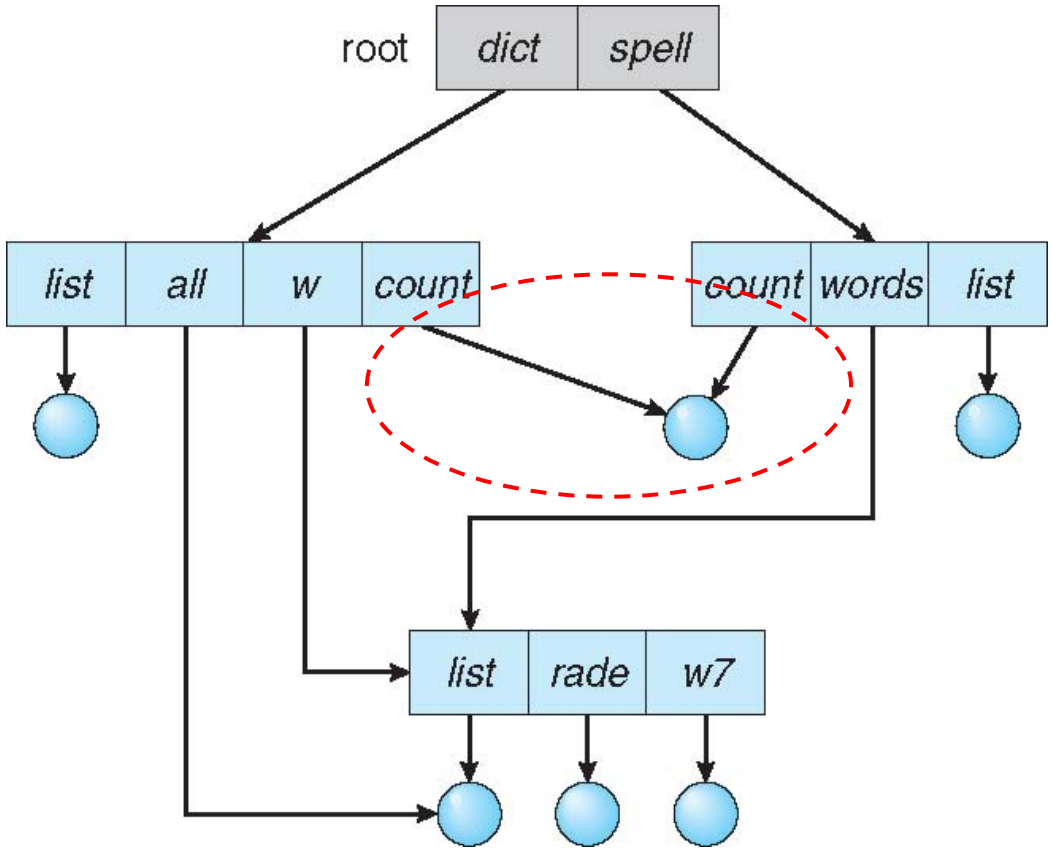
3) Tree-structured directories

- Efficient searching
- Grouping capability
- Two types of path names
  - Absolute path name
    - Begins at the root and follows a path down to the specified file
  - Relative path name
    - A path from the current directory
- Deleting a directory
  - 1. Not allowed if is not empty
  - 2. Have an option of delete internal nodes
- Sharing problem



4) Acyclic-graph directories

- Have shared subdirectories and files
  - Only one actual file exists, so any changes made by one person are immediately visible to the other
- Methods of shared files implementation
  - 1) Link
    - Another name (pointer) to an existing file
    - Resolve the link: follow pointer to locate the file
  - 2) Duplicate all information about the file
    - Both entries are identical and equal
    - Consistency problem (why?)
- Deletion & Traversing problems (?)



Deletion possibilities?

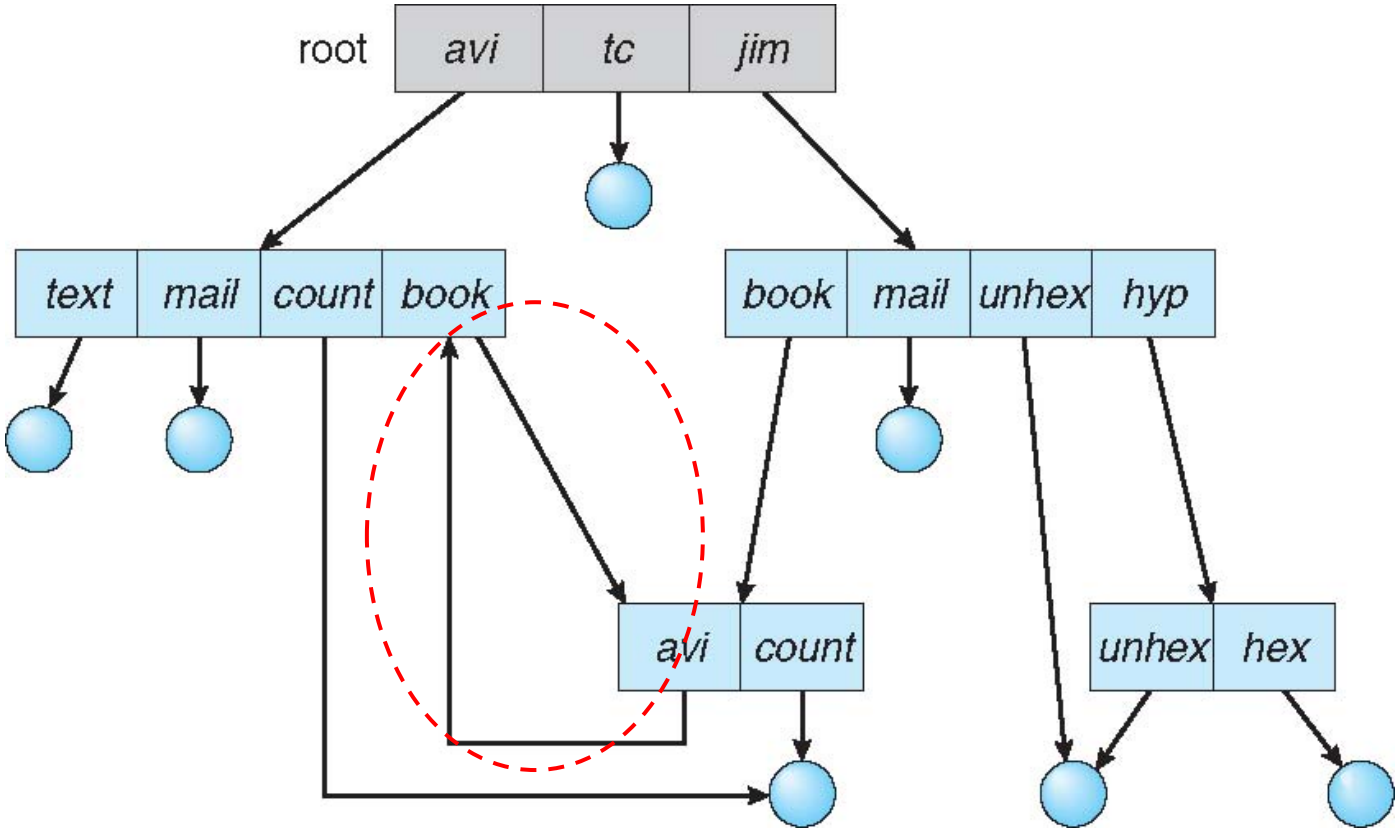
- 1) Remove the file content whenever anyone deletes it
  - Dangling pointers: pointing to the nonexistent file
  - What if the remaining file pointers contain actual disk addresses?
  - Easy with soft-links (symbolic links)

- 2) Preserve the file until all references to it are deleted
  - Hard links
  - Counting number of references

4) General graph directories

- Remove problem of no cycles

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

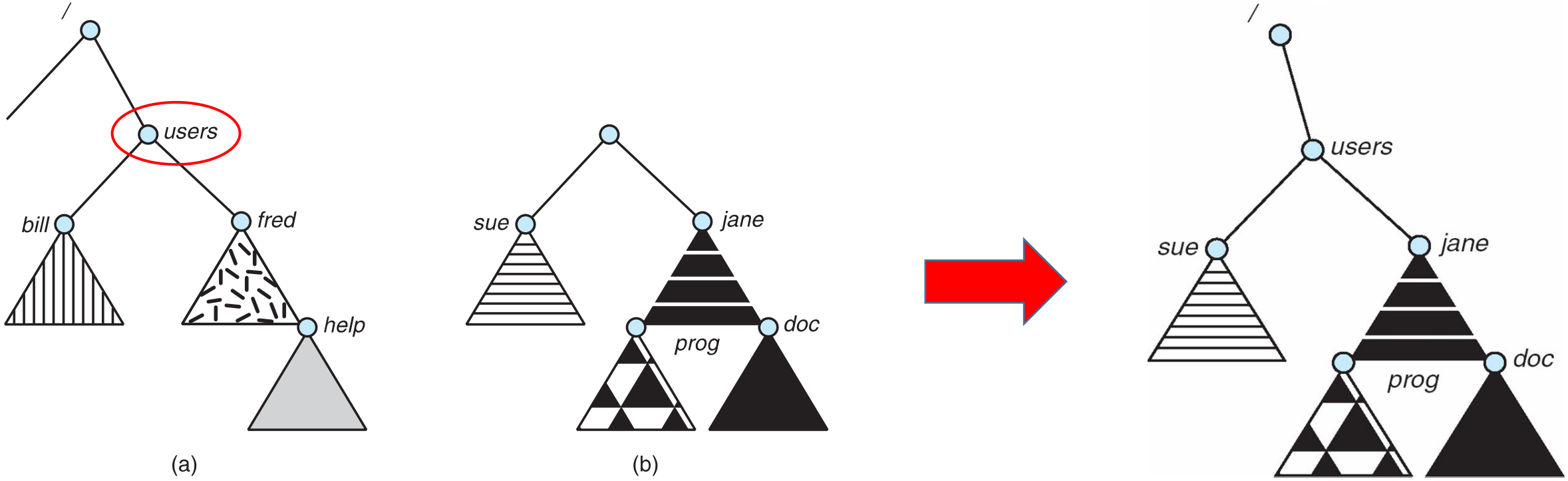


File System Mounting

File system mounting

- A file system must be mounted before it can be accessed
- A unmounted file system is mounted at a mount point
- Mount point
  - The location within the file structure where the file system is to be attached

File system mounting & mount point



(a) Existing FS (b) unmonted FS

The mounted FS

File Sharing & Protection

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 system

- Uses networking to allow file system access between systems
  - 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 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 FS) is standard Windows protocol
  - Standard OS file calls are translated into remote calls
- Distributed Information Systems (distributed naming services) 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 (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 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)
    - 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

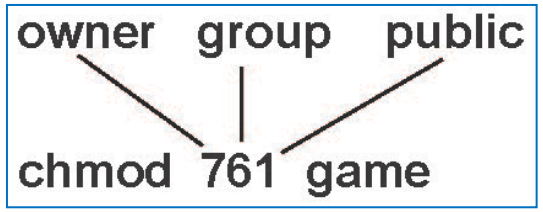
- File owner/creator should be able to control
  - What can be done
  - By whom
- Types of access
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

Access lists and groups

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

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



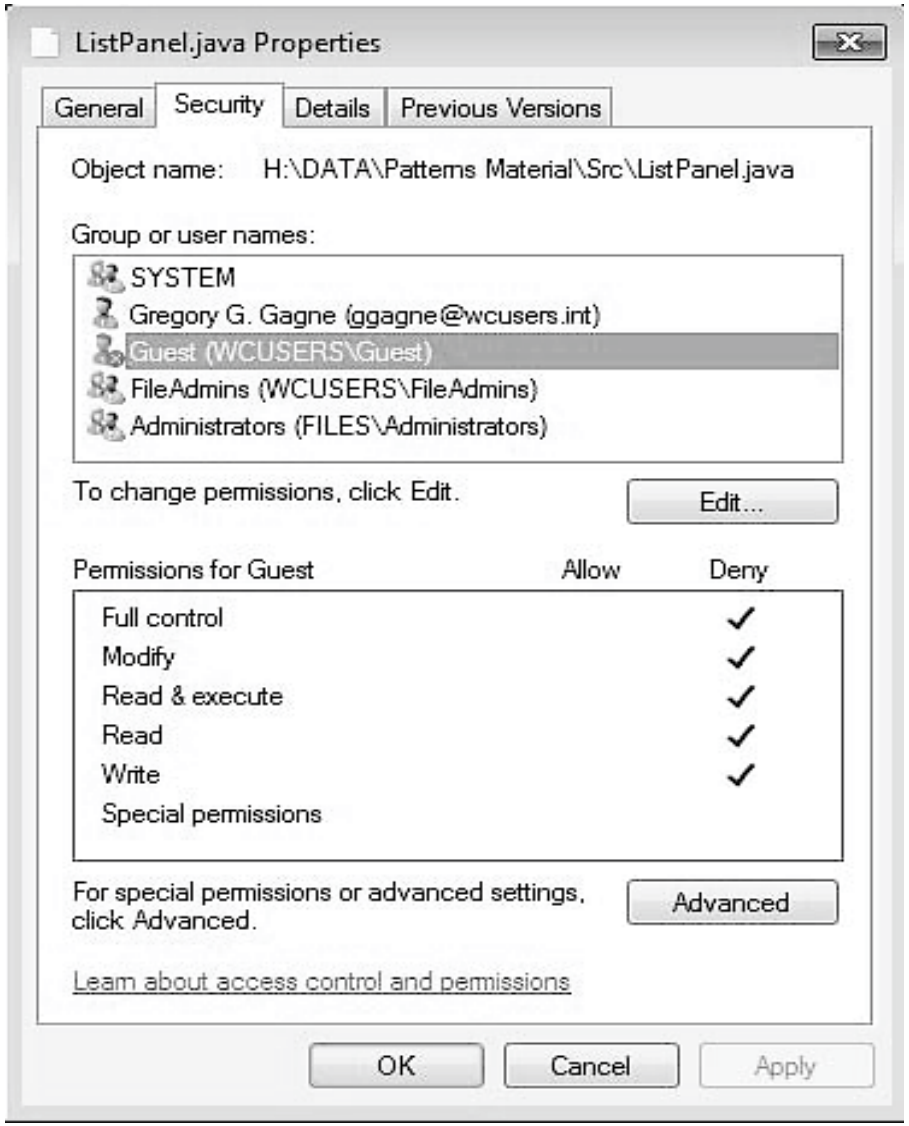
Attach a group to a file

```
chgrp    G    game
```

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 7 access control list management



Questions?