File system interface

- File concept
- File access methods and operation
- Fie system organization
- Protection
- Consistency semantics

Last modification date: 17.10.2016

File concept

- File concepts
  - Logical unit of data storage
  - Abstract image of data, stored and retrieved by the system:
  - Named collection of data items, stored in secondary memory
- Operating system obligations:
  - Providing the mapping of abstract view of the file and the physical representation
  - File identification (hierarchical file system)
  - Providing file system API
  - Performing adequately file system related operations (efficiency, data protection)
- **POSIX:**
  - **File** - an object that can be written to, or read from, or both. A file has certain attributes, including access permissions and type. File types include regular file, character special file, block special file, FIFO special file, symbolic link, socket, and directory. Other types of files may be supported by the implementation.

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

File operations

- **create** – file creation
- **write** – storing data into a file
- **read** – retrieving data from a file
- **file seek** – changing current position
- **delete** – file removal (exact meaning might vary)
- **truncate** – file truncation
- **fd=open(F_i)** – finds entry of the file F_i in a file system catalogue and copies it to the main memory – provided access control allows for that.
- **close (fd)** – copies the local (memory) entry of the file F_i to the file system catalogue.

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. File type can be recognized by operating system, user or application.
- **Location** – pointer to file location on device
- **Size** – current file size. **NOTE**:

  1000B = 1kB; 1024B=1KB=1KiB (kibibyte);

  1000^2B=1MB; 1024^2B=1MiB (mebibyte)
- **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 same devices as files themselves. File format (→ type) can be represented in file name or kept at the beginning of the file (***magic numbers***), e.g.:

ELF exec: 0x7F, 'ELF' , Unix script: '#!' ; MS-DOS exec: 'MZ' (0x4D, 0x5A

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 session -** a sequence of file operations, starting with opening file access, ending with closing the session.

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

Access Methods

- **Sequential Access**

```
read next
write next
reset
no read after last write
        (rewrite)
```

- **Direct Access –** file is fixed length logical records

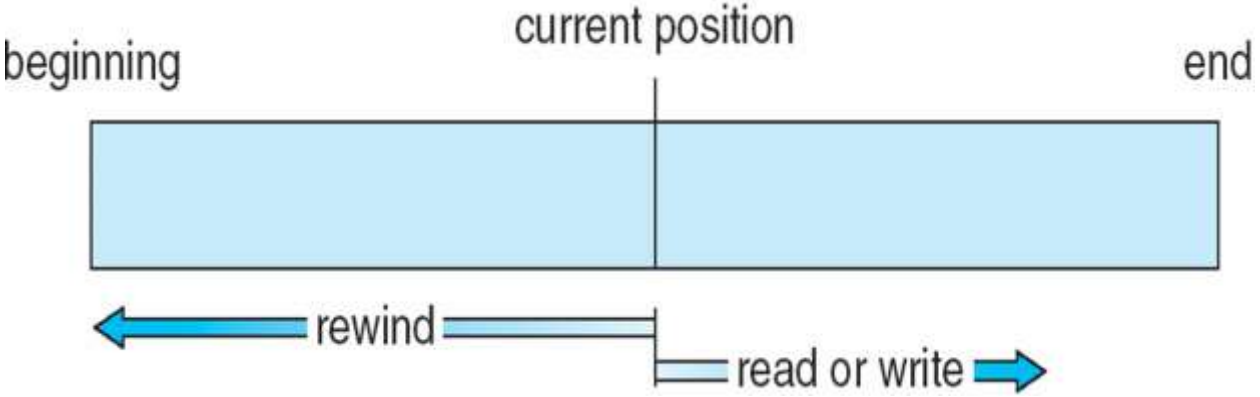
```
read n
write n
position to n
        read next
        write next
rewrite n
```

n = block numer of the file (relative)

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

Sequential File Access

- Sequential file access



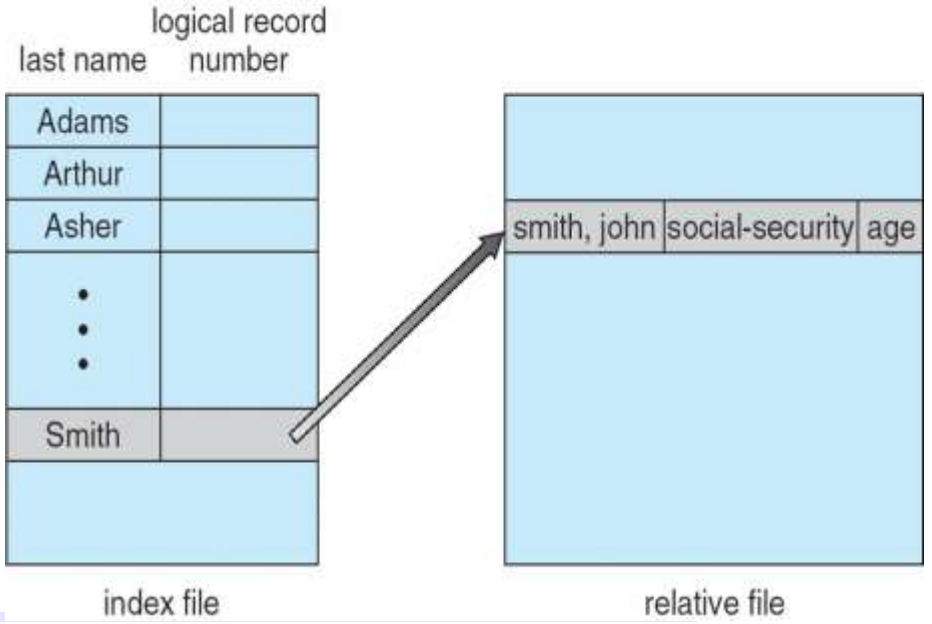
- Emulation of sequential file access with direct access

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

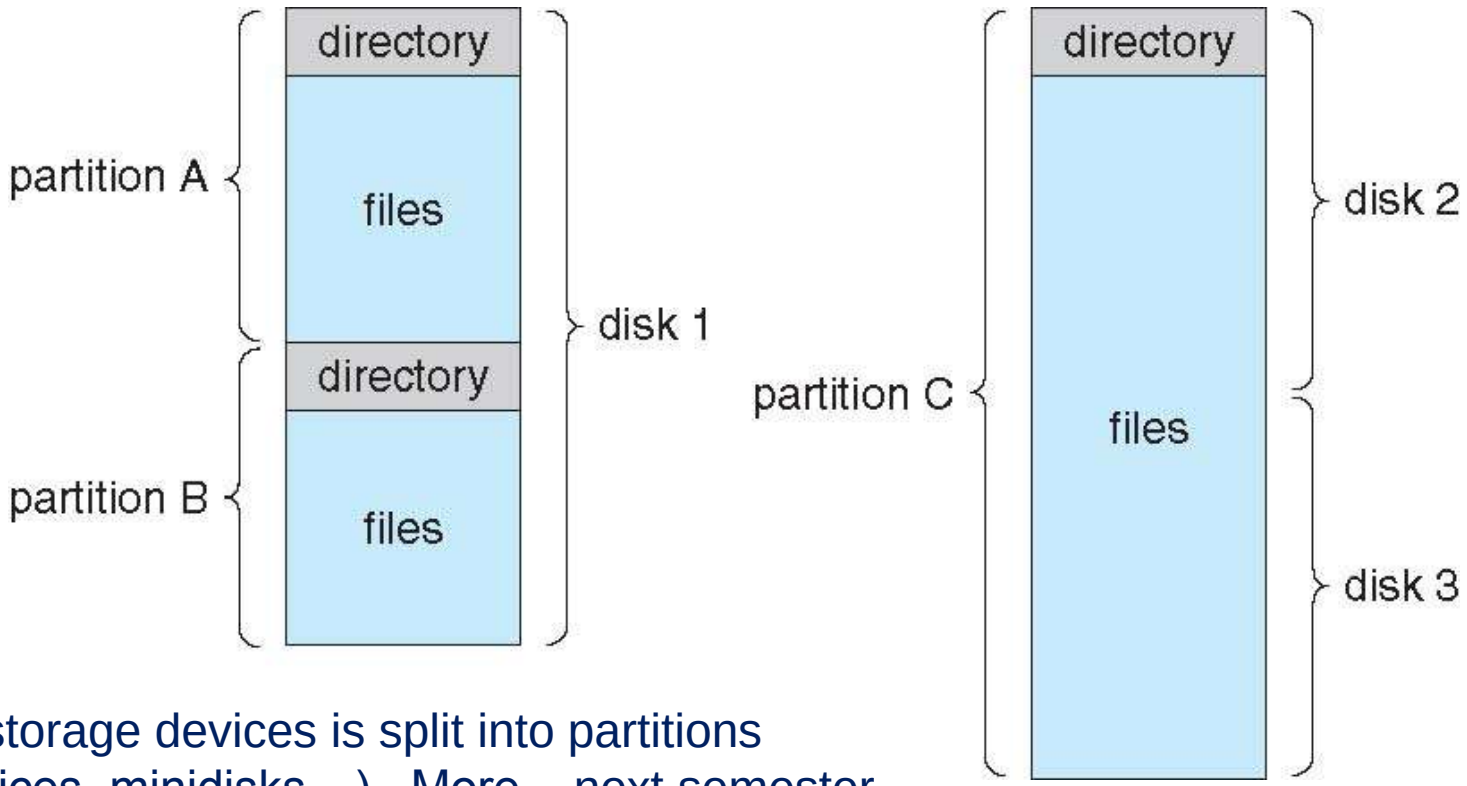
Other Access Methods

- Can be built on top of base methods
- 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 too large, index (in memory) of the index (on disk)
- 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 operating system:

  index + relative files →



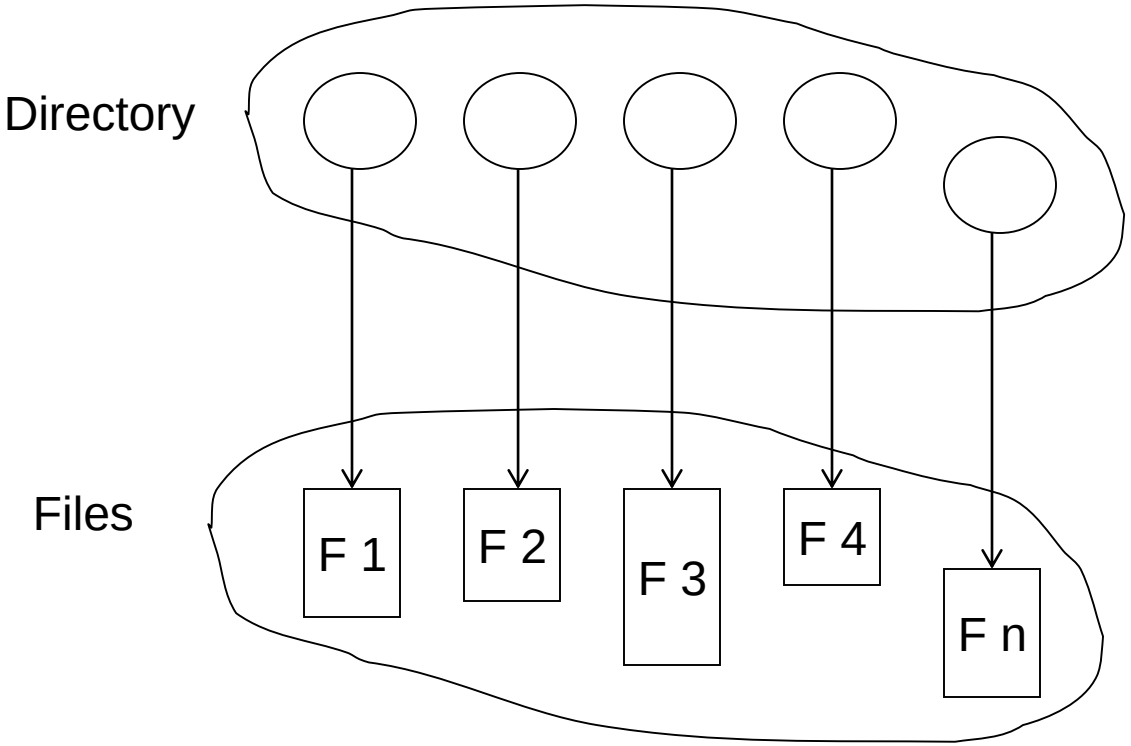
A typical file-system organization



- Memory of storage devices is split into partitions (volumes, slices, minidisks…). More – next semester.
- 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
- Formatted partition contains a header, the main directory and a storage space for sub-directories and files.

Directory structure

- A collection of nodes containing information about all files



Both the directory structure and the files reside on disk.
Copies of the directory and files should be backed onto tapes, CD-ROMs etc. to enable disaster recovery.

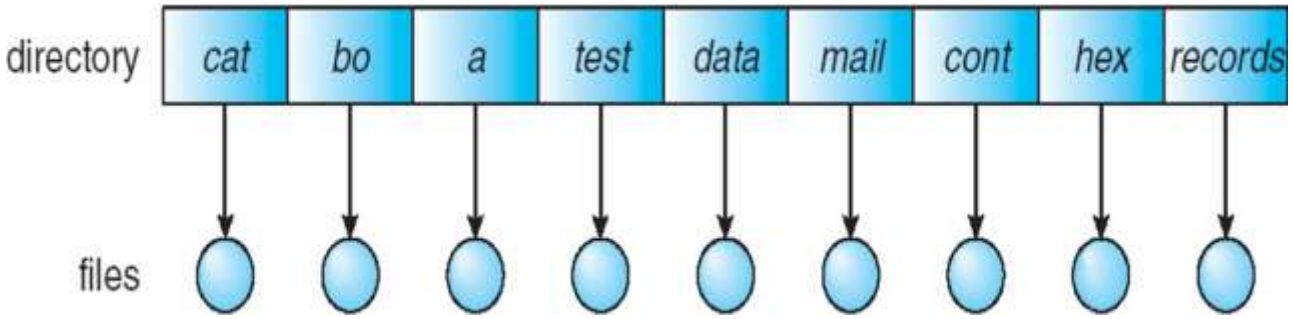
Directory organization

The directory is organized logically to obtain

- Efficiency of directory operations:
  - Finding a file (file path parsing) – the most common
  - File creation
  - File removal
  - File renaming
  - Traversing the file system
- 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

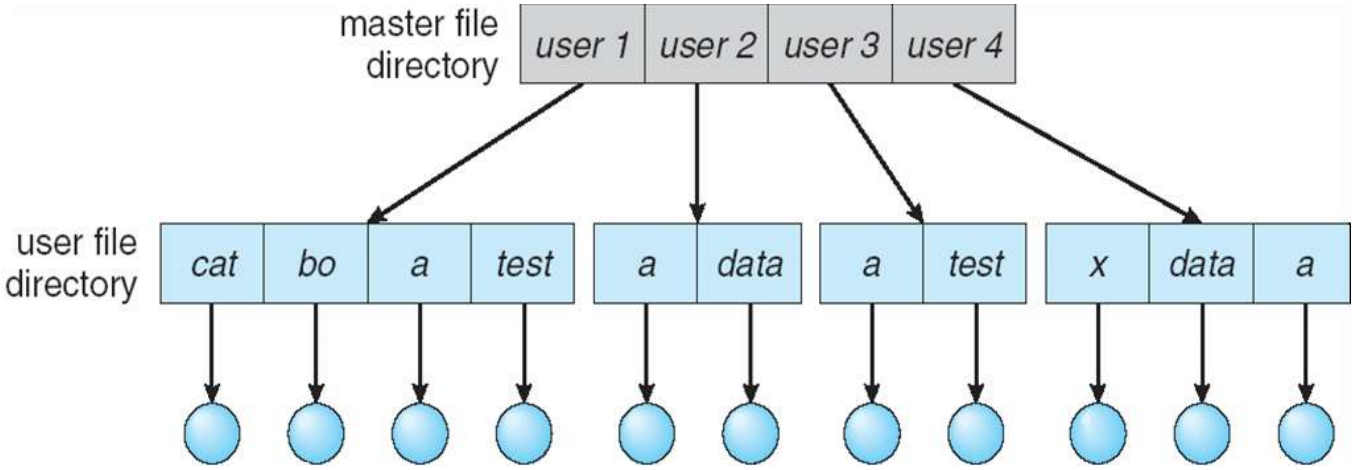
- A single directory for all users



- Naming problem
- Grouping problem

Two-level directory

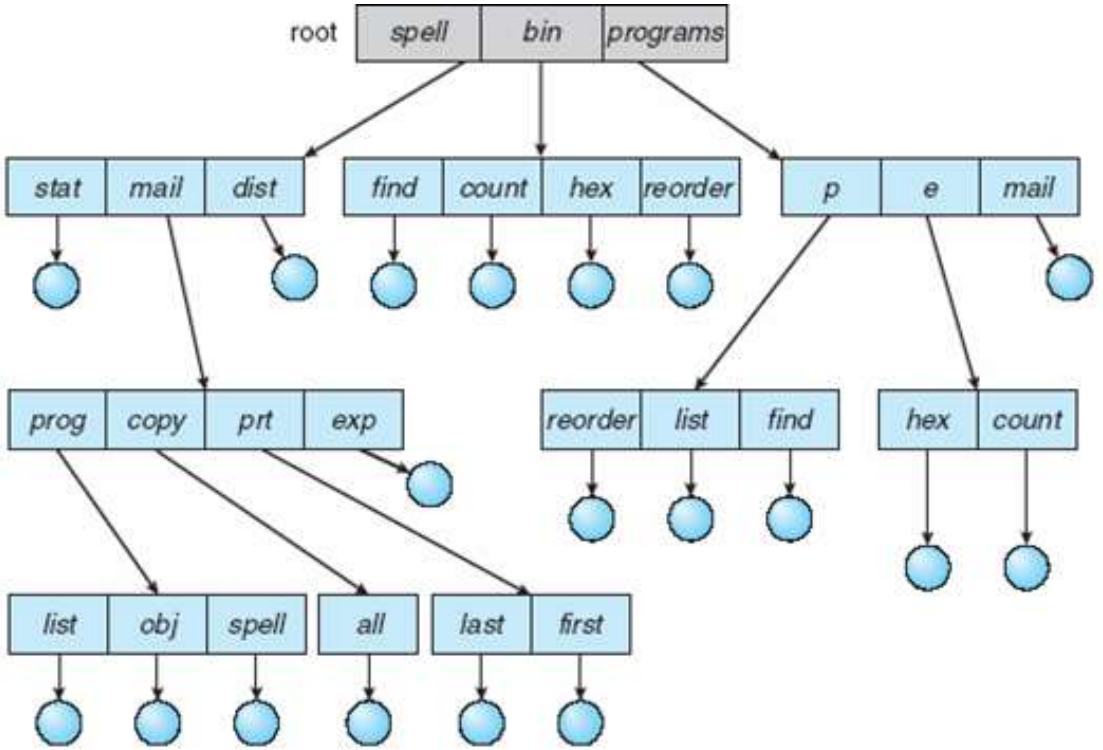
- Separate directory for each user



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

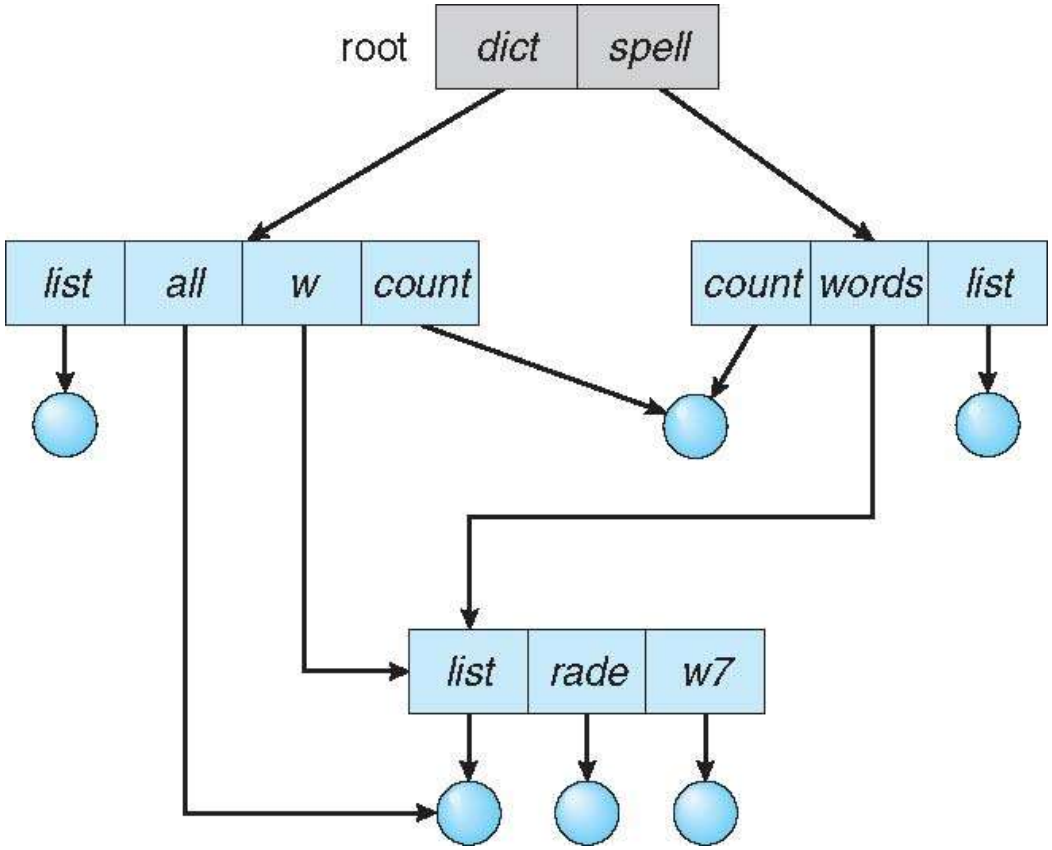
Tree-structured directories

- Efficient searching
- Grouping capability
- Current directory (working directory), e.g.:
  - `cd /spell/mail/prog`
  - `type list`
- **Absolute** or **relative** path name
- Removal of a directory involves trimming a tree (removal of files and subdirectories below). E.g. deleting “mail” ⇒ deleting the entire subtree rooted by “mail” (4 subdirectories and 6 files) →



Acyclic-graph directories

- Have shared subdirectories and files

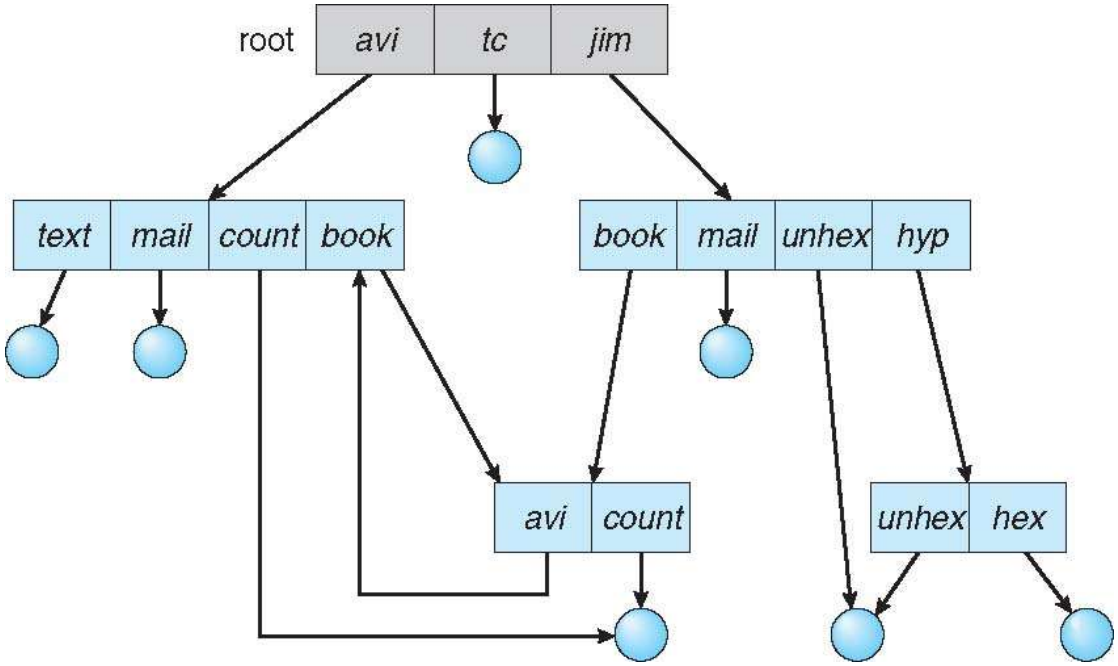


Acyclic-graph directories (cont.)

- 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

General graph directory

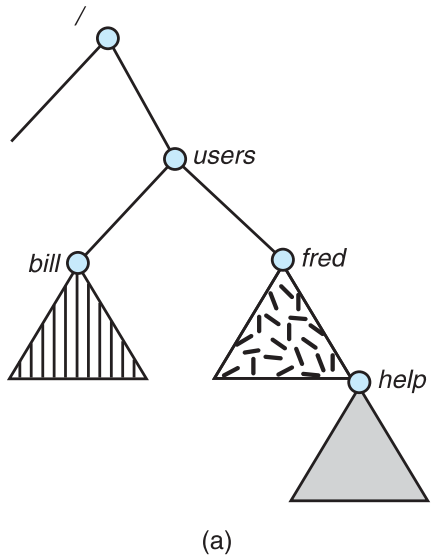


How do we guarantee no cycles?

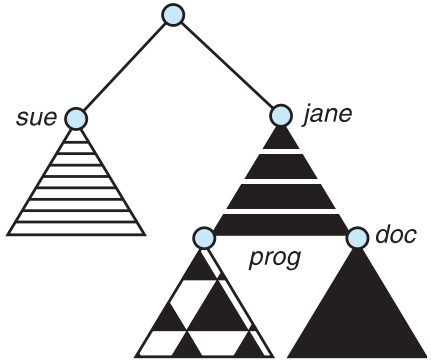
- Allow only links to files not subdirectories
- **Garbage collection**
- Every time a new link is added use a cycle detection algorithm to determine whether it is OK

File system mounting

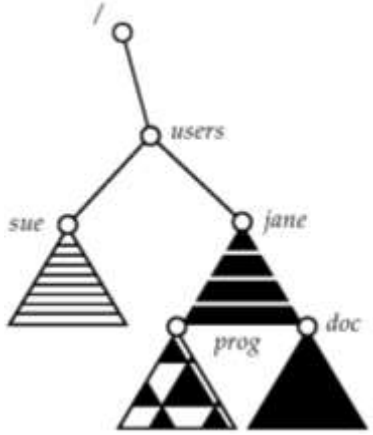
- A file system must be **mounted** before it can be accessed
- A unmounted file system (i.e., Fig. (b)) is mounted at a **mount point (**here**: /users**).The previous content and owner and mode of the mount point become invisible (Fig. (c)). Usually the mount point directory is empty prior to the mount.



(a)



(b)



(c)

File Sharing

- Sharing of files on multi-user systems is desirable
- On distributed systems, files may be shared across a network. Network File System (NFS) is a common distributed file-sharing method. To be discussed in **the second part of the course**.
- Sharing may be done through a **protection** scheme.
- **Consistency semantics** of the file sharing specifies how multiple users are to access a shared file simultaneously
  - Unix file system (UFS) implements:
    - Writes to an open file visible immediately to other users
    - Sharing file pointer (i-node) to allow multiple users to read and write concurrently
  - Session semantics (e.g. Andrew FS, VMS FS). Writes only visible to sessions starting after the file is closed
  - Semantic of shared read-only files

Protection

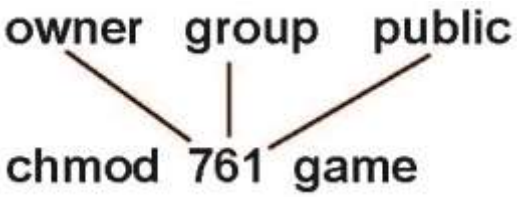
- File owner/creator (or a security officer) should be able to control what can be done by whom
- Types of access
  - **Read/Write/Execute**
  - **Append, Delete, List**
- If multi-user system
  - **User IDs** identify users, allowing permissions and protections to be per-user
    Group IDs allow users to be in groups, permitting group access rights
  - **Role-based access control** – change of user capability on-the fly, to perform dedicated set of operations (after authentication).
  - **Linux cgroups** – a kernel feature that limits, accounts for, and isolates the resource use for a collection of processes.
- **Access list (ACL)** – association of distinct user access rights with each file and directory. To decrease ACL overhead access groups can be implemented instead.

Access Groups

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



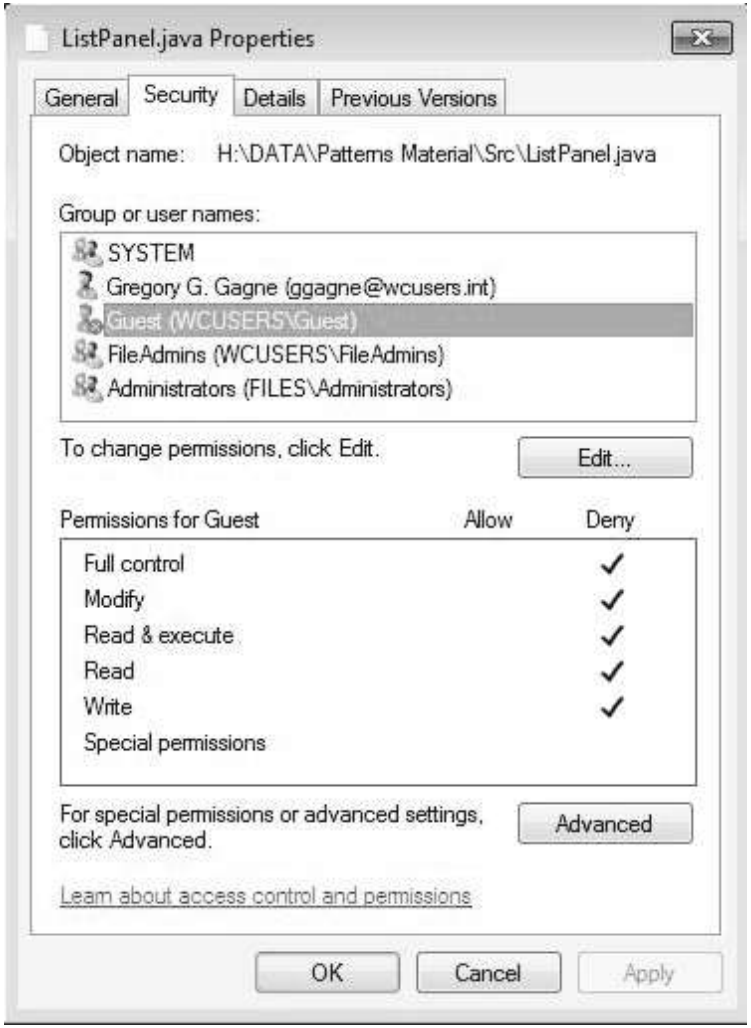
Attach a group to a file

```
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



To be discussed in the next semester