File-System Interface

Unit-IV

Session 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

Session Outcomes

At the end of this session, participants will be able to

- Discuss File Concepts, Access Methods
- Disk and Directory Structure
- File-System Mounting
- File Sharing Protection

Agenda

Presentation Outline

File Concept

- Contiguous logical address space
- Types:
  - Data
    - numeric
    - character
    - binary
  - Program
- Contents defined by file's creator
  - Many types
    - Consider **text file, source file, executable 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
- Many variations, including extended file attributes such as file checksum
- Information kept in the directory structure

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

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

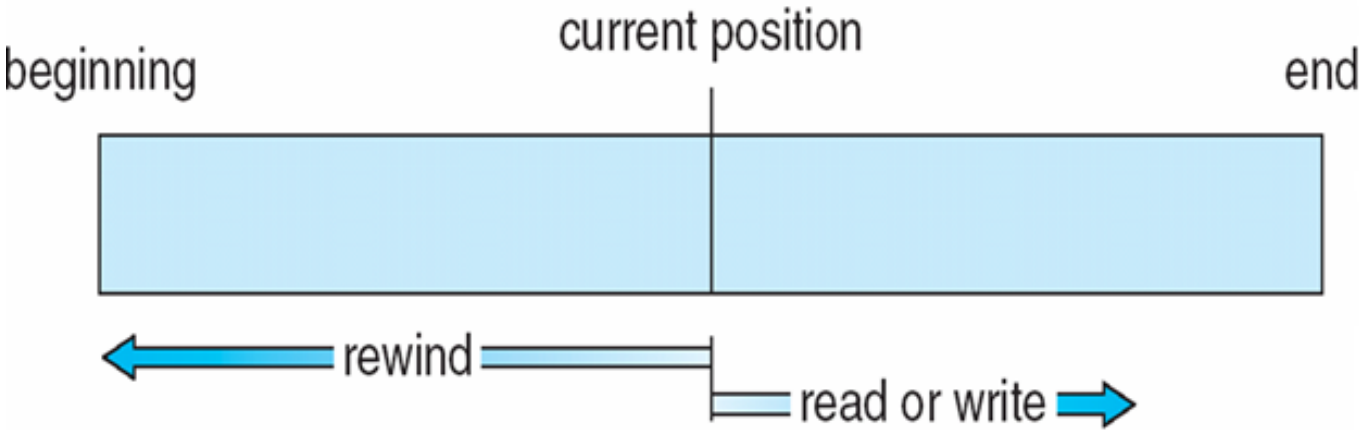
Open File Locking

- Provided by some operating systems and file systems
- **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
- Who decides:
- Operating system

Sequential-access File



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 = relative block number

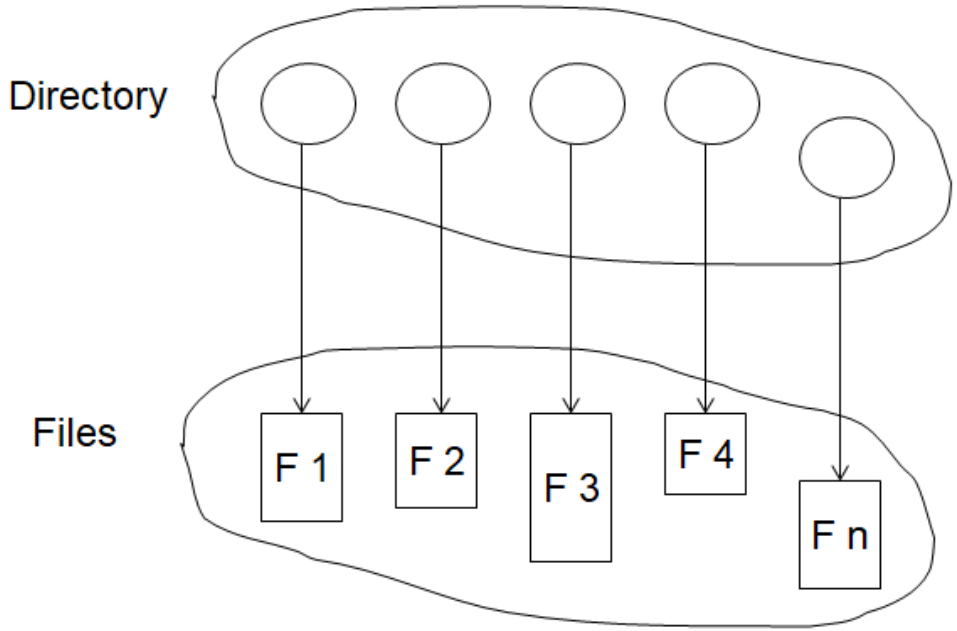
- Relative block numbers allow OS to decide where file should be placed

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*; $cp = cp + 1;$

Directory Structure

- A collection of nodes containing information about all files

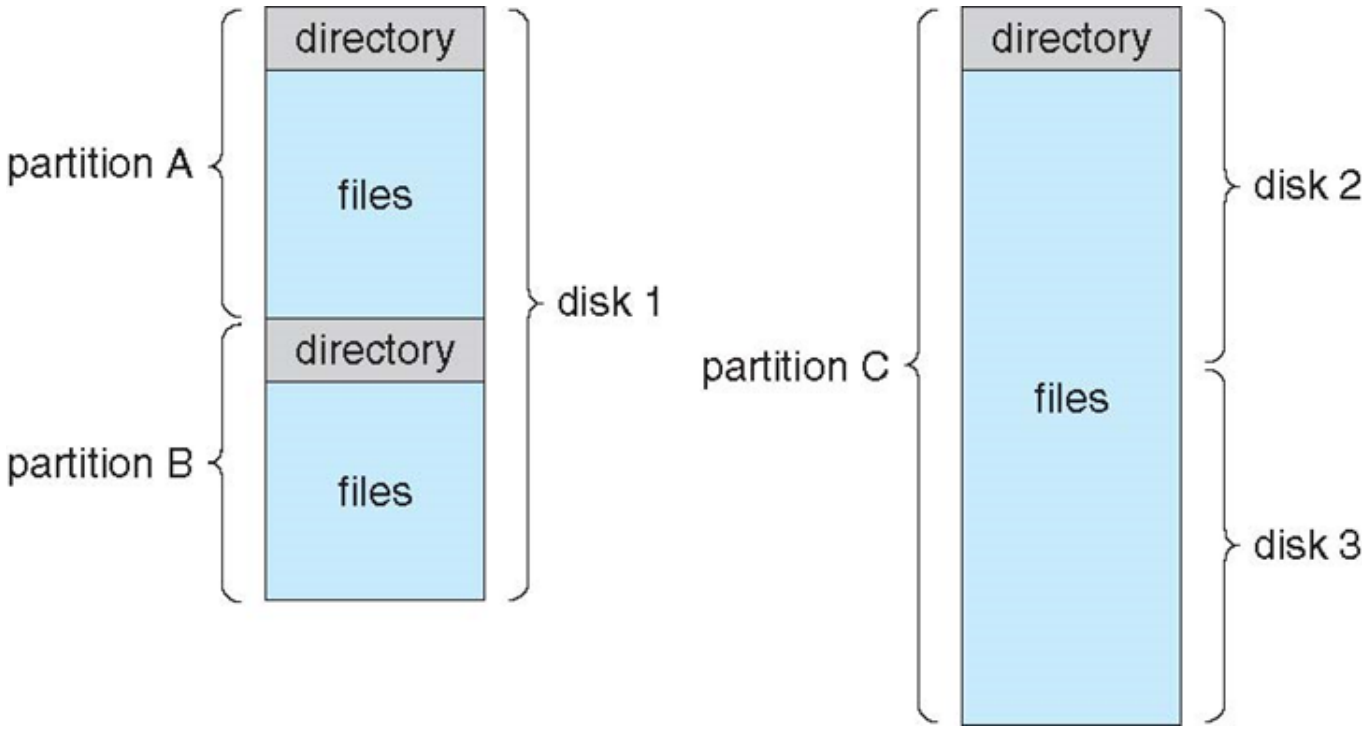


Both the directory structure and the files reside on disk

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**

A Typical File-system Organization



Types of 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

Operations Performed on Directory

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

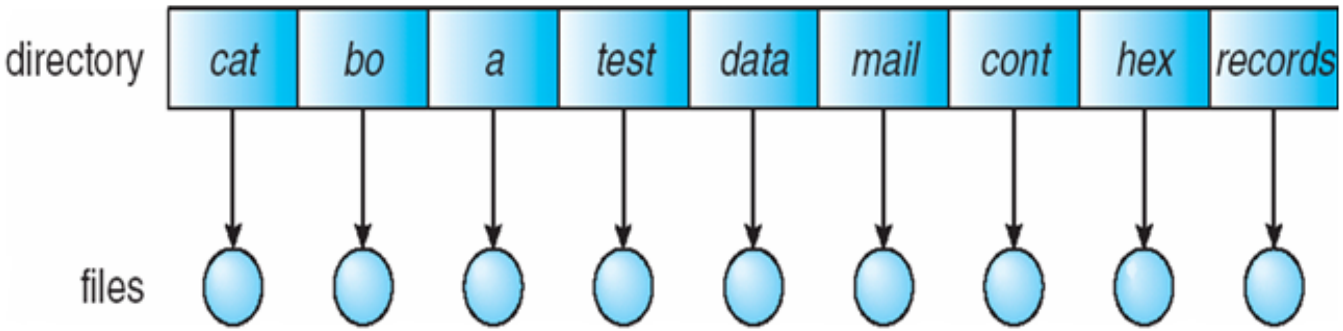
Directory Organization

The directory is organized 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, -

Single-Level Directory

- A single directory for all users

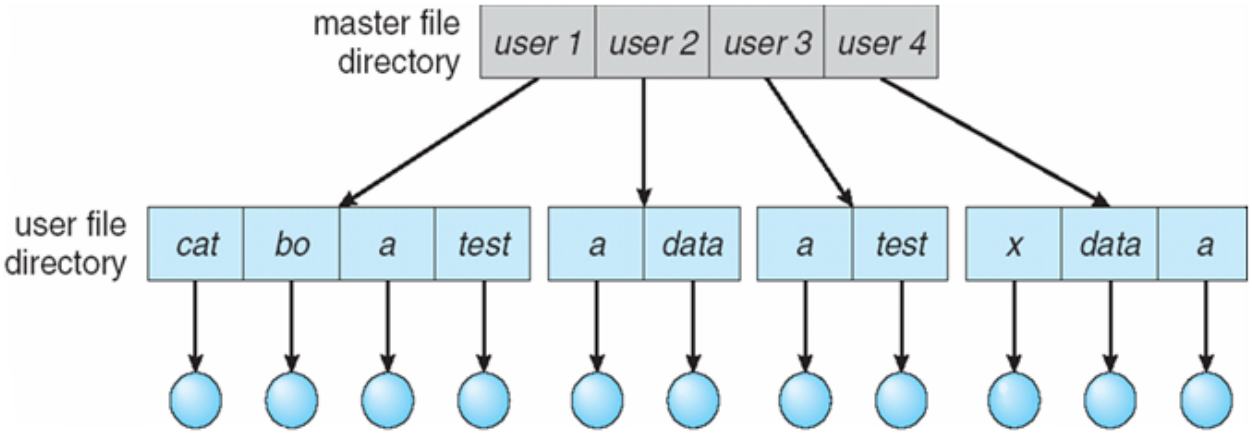


- Naming problem
- Grouping problem

Two-Level Directory

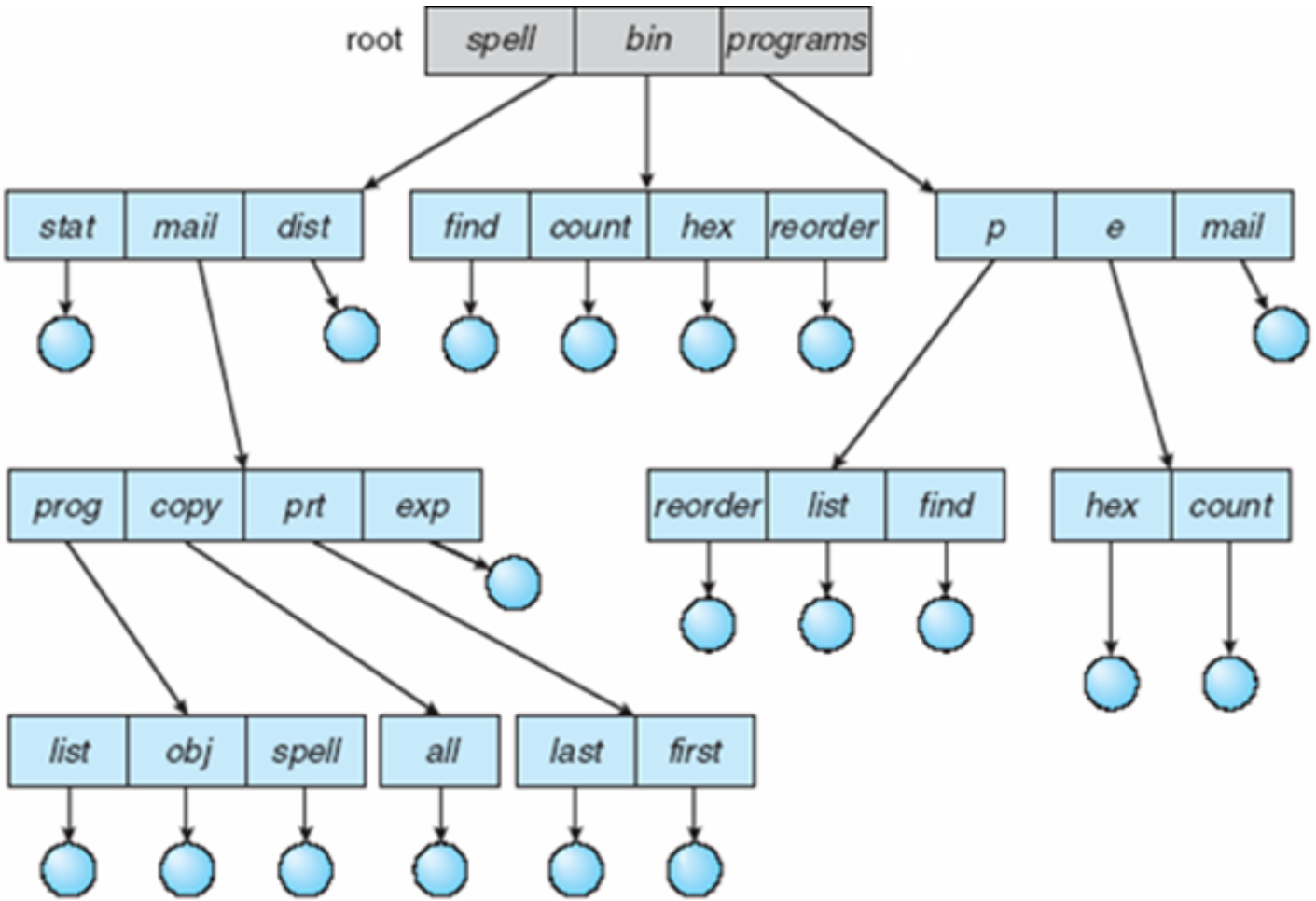
Two-Level Directory

- 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

Tree-Structured Directories

Tree-Structured Directories (Cont)

- **Absolute** or **relative** path name
- Creating a new file is done in current directory
- Delete a file

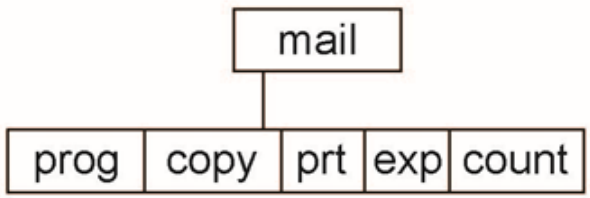
```
rm <file-name>
```

- Creating a new subdirectory is done in current directory

```
mkdir <dir-name>
```

Example: if in current directory `/mail`

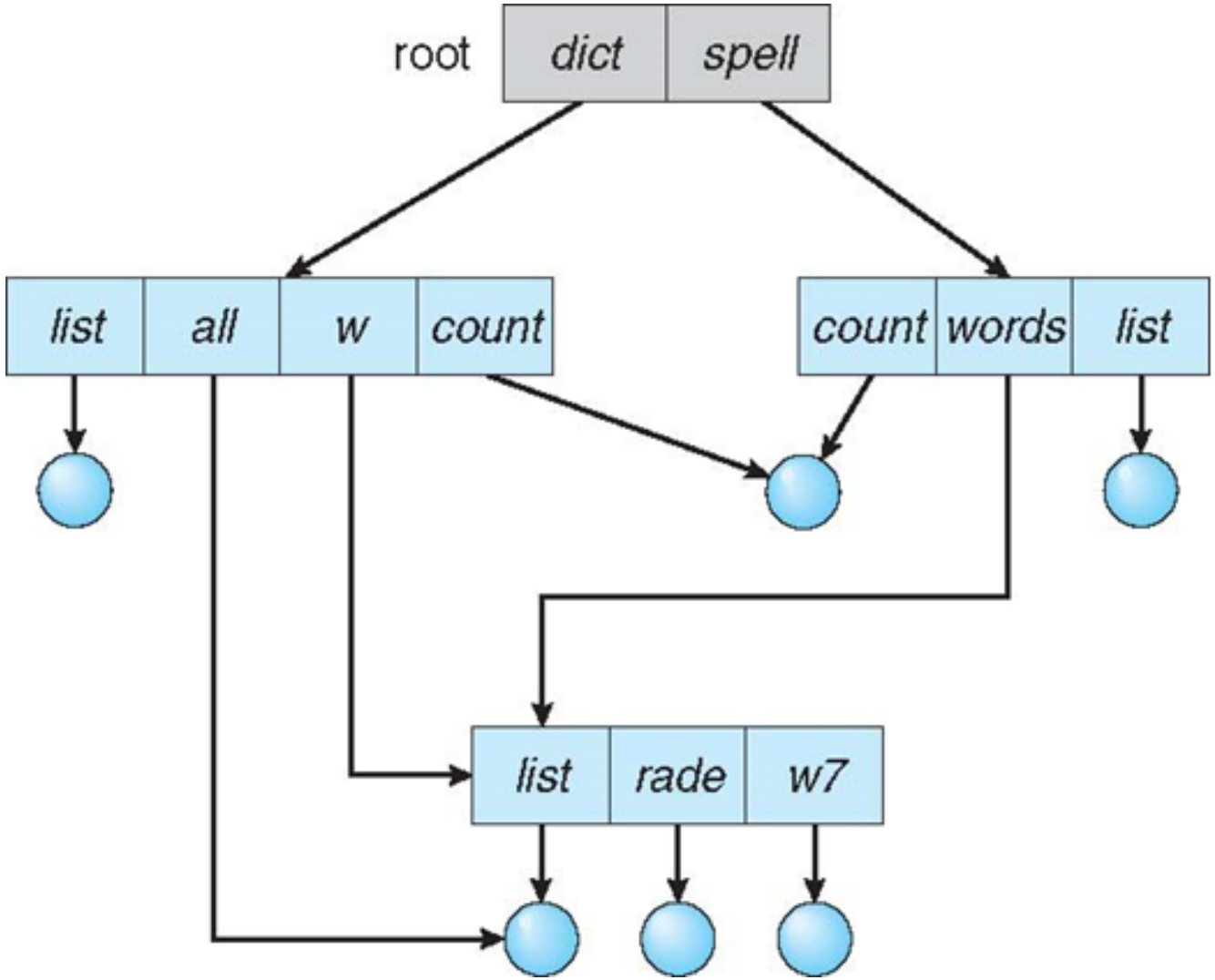
```
mkdir count
```



Deleting "mail" ⇒ deleting the entire subtree rooted by "mail"

Acyclic-Graph Directories

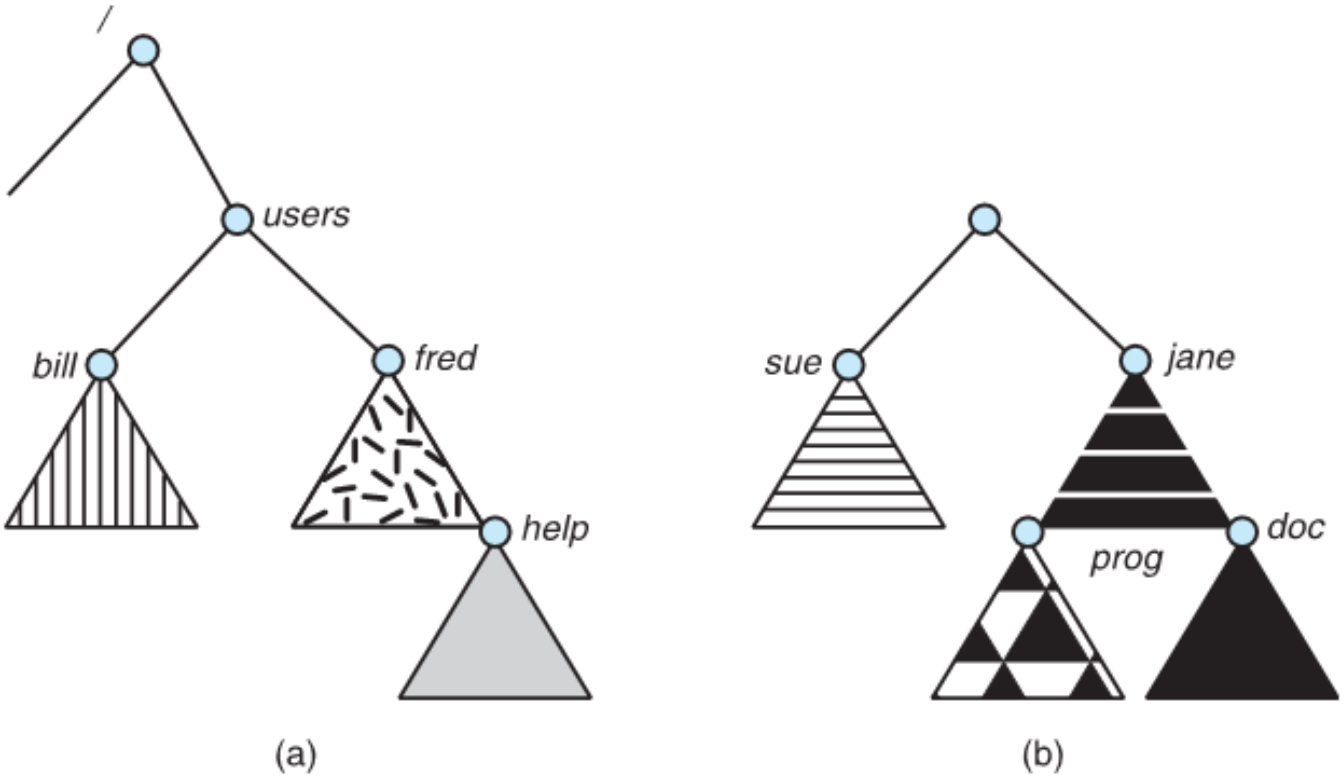
Have shared subdirectories and files



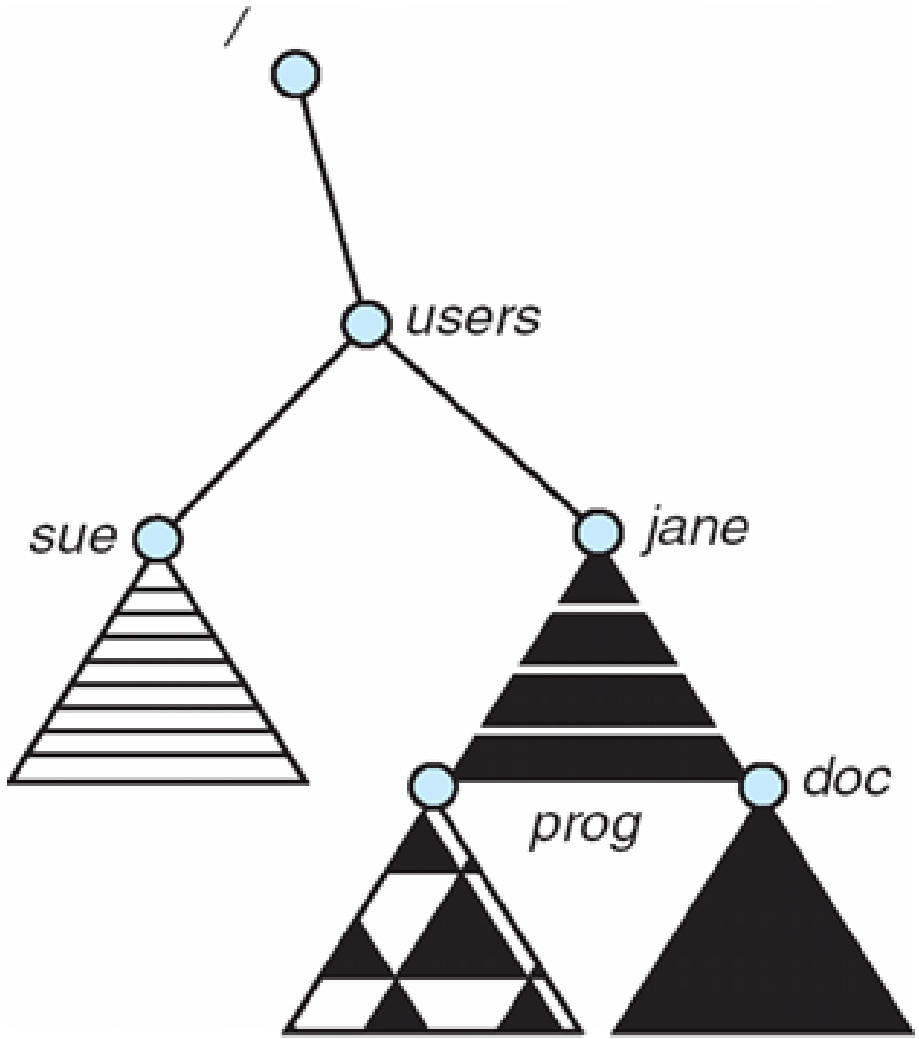
File System Mounting

A file system must be mounted before it can be accessed

A unmounted file system is mounted at a mount point



Mount Point



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
- In 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 or directory
  - Group of a file or directory

File Sharing – Remote File Systems

- 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** is standard Windows protocol
  - Standard operating system 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, 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

Protection

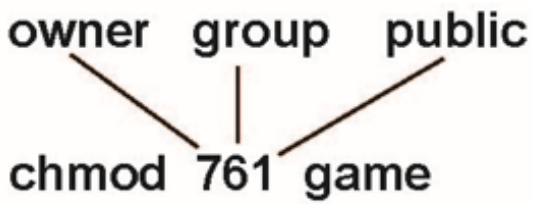
- File owner/creator should be able to control:
  - what can be done
  - by whom
- Types of access
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

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
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.



Summary

Discussed

- File operations
- File structures
- Disk Structures, types of file systems
- Directory organization

Test Your Understanding

- What is the mount point?
  a) an empty directory at which the mounted file system will be attached
  b) a location where every time file systems are mounted
  c) is the time when the mounting is done
  d) none of the mentioned
- In distributed file system ————directories are visible from the local machine.
  a) protected b) local c) private d) remote
- To organise file systems on disk ————
  a) they are split into one or more partitions
  b) information about files is added to each partition
  c) they are made on different storage spaces
  d) all of the mentioned