BILKENT UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING
CS342 OPERATING SYSTEMS

LECTURE 10
FILE SYSTEMS - INTERFACE
(CHAPTER 10)

Dr. İbrahim Körpeoğlu
http://www.cs.bilkent.edu.tr/~korpe

References

- *The slides here are adapted/modified from the textbook and its slides: Operating System Concepts, Silberschatz et al., 7th & 8th editions, Wiley.*

REFERENCES

- Operating System Concepts, 7th and 8th editions, Silberschatz et al. Wiley.
- Modern Operating Systems, Andrew S. Tanenbaum, 3rd edition, 2009.

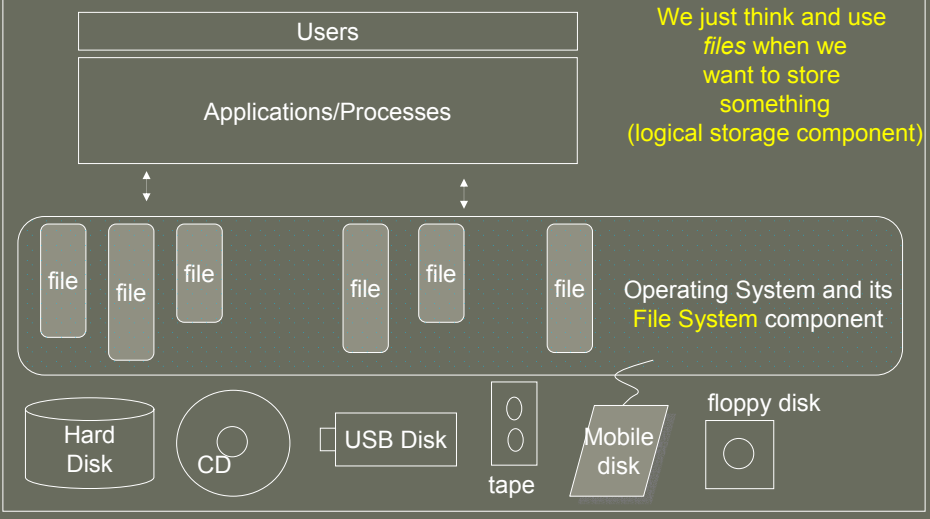
Outline

- File Concept
- Access Methods
- Directory Structure
- File-System Mounting
- File Sharing
- Protection

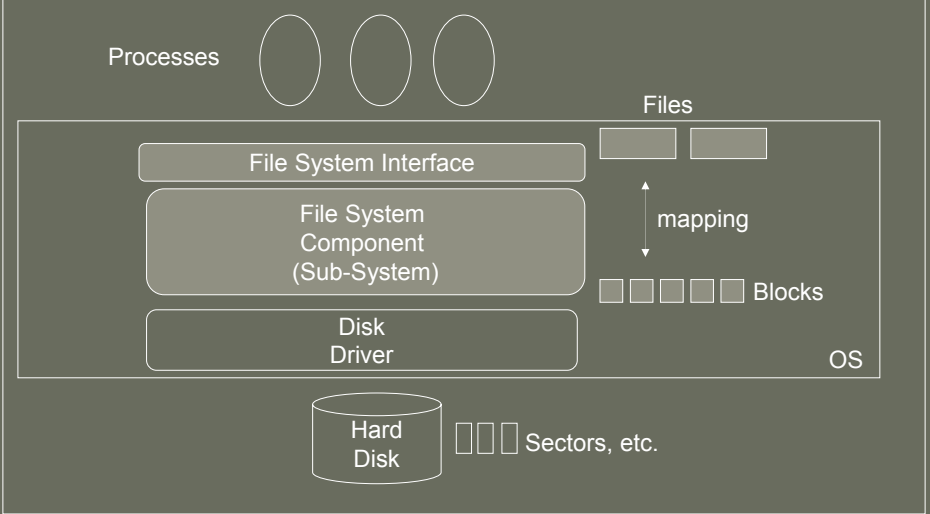
Objectives

- 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

File Concept

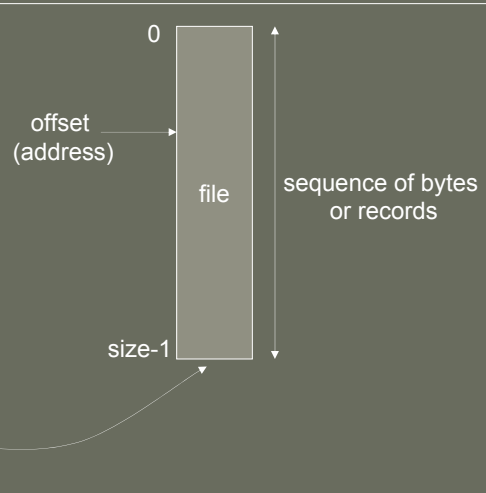


File Concept



File Concept

- Contiguous logical address space (a storage)
- Types:
  - Data
    - numeric
    - character
    - binary
  - Program



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

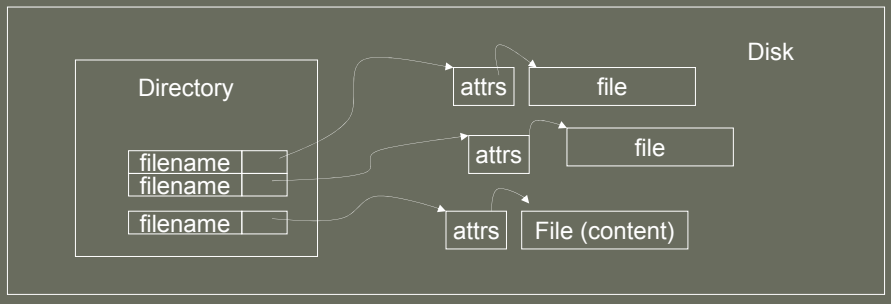
File Attributes

- **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

Files and Directories

- There are two basic things are stored on disk as part of the area controlled by the file system
  - files (store content)
  - directory information (can be a tree): keeps info about files, their attributes or locations

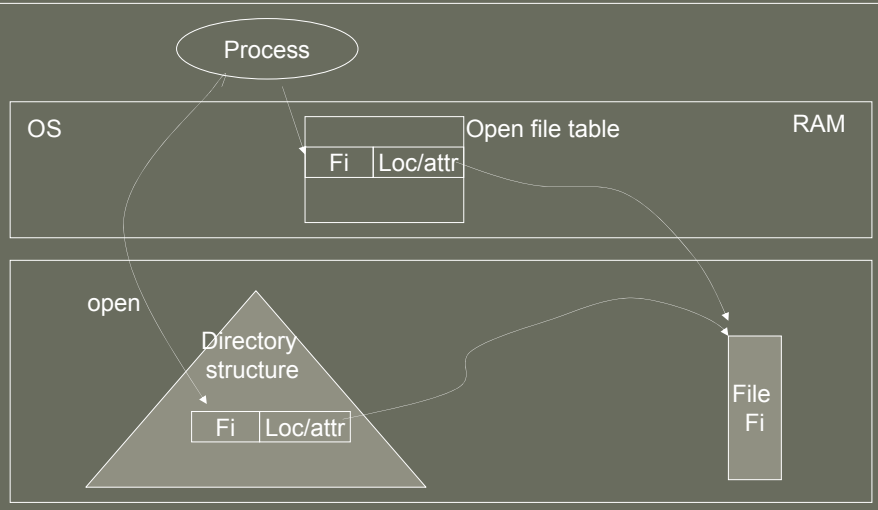


File Operations

- File is an **abstract data type**
- **Common Operations that are supported by the Operating System:**
  - Create
  - Write
  - Read
  - Reposition within file
  - 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

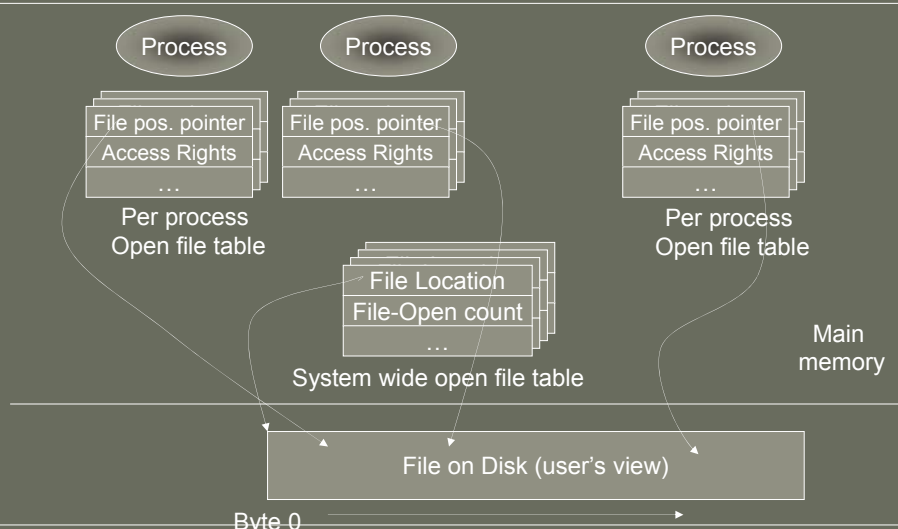
Open File



Open Files

- Several pieces of data are needed to manage open files:
  - File position 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 information



File Locking

- Provided by some operating systems and file systems
- 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 Types

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

Access Methods

- **Sequential Access**

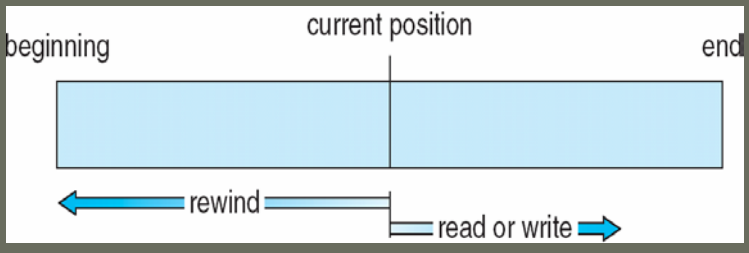
  read next
  write next
  reset
  no read after last write
  (rewrite)

- **Direct Access**

  read *n*
  write *n*
  position to *n*
  read next
  write next
  rewrite *n*

  n = relative block number

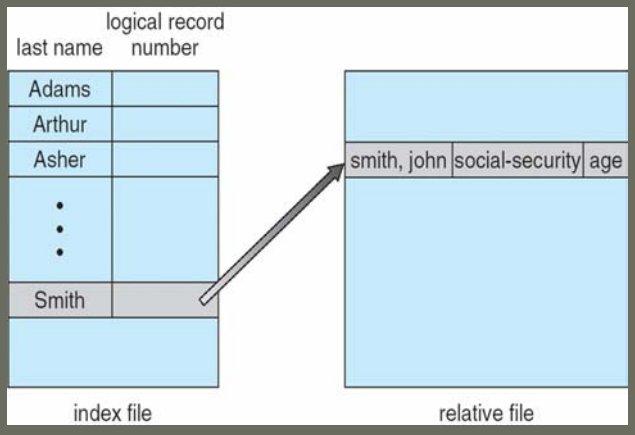
Sequential-access File



Simulation of Sequential Access on Direct-access File

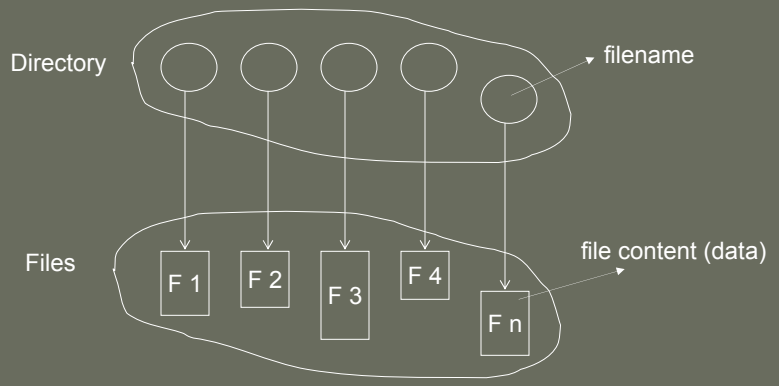
sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

Example of Index and Relative Files



Directory Structure

- A collection of nodes containing information about all files



Both the directory structure and the files reside on disk
Backups of these two structures are kept on tapes

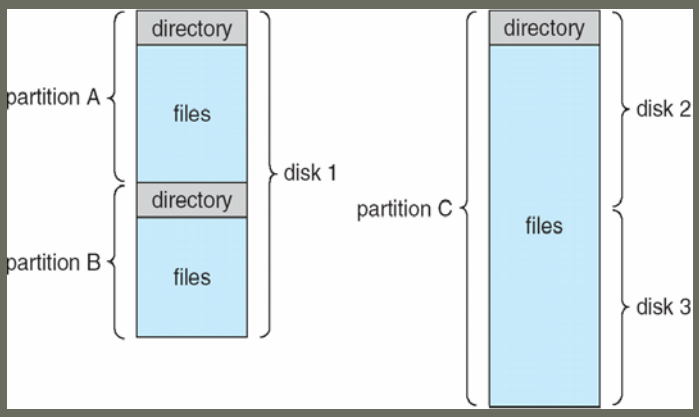
Disk Structure

- 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

- As well as general-purpose file systems there are many special-purpose file systems, frequently all within the same operating system or computer

A Typical File-system Organization



Operations Performed on Directory

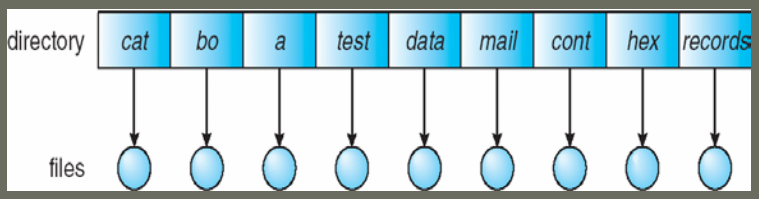
- Search for a file
  - Given filename, find out the corresponding directory entry
- Create a file
- Delete a file
- Rename a file
- List a directory
  - List the names of files in that directory. For each file; more information may be printed out.
- Traverse the file system
  - Starting from root directory, go though all directory entries, including the subdirectories and their entries, recursively.

Organize the Directory (Logically) to Obtain

- Efficiency – locating a file quickly
- Convenient Naming – convenient to users
  - Two users can have same name for different files
  - The same file can have several different names
- Enabling Grouping – logical grouping of files by properties, (e.g., all Java programs, all games, …)

Single-Level Directory

- A single directory for all users

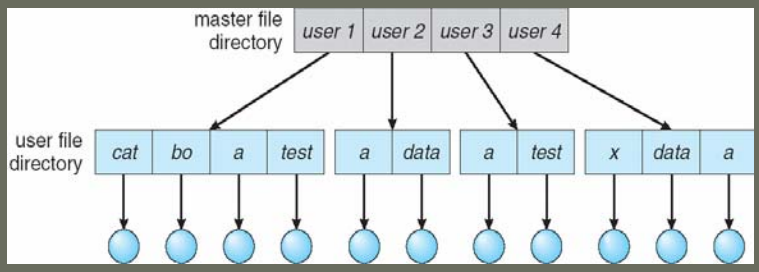


Naming problem

Grouping problem

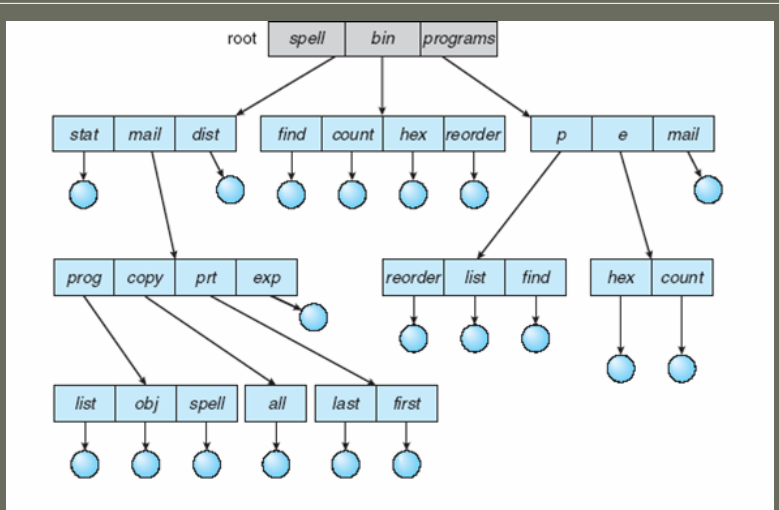
Two-Level Directory

- Separate directory for each user



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

Tree-Structured Directories



Tree-Structured Directories (Cont)

- Efficient searching
  - A pathname indicated where a file is. Parse the pathname and follow those subdirectories indicated in the pathname

    "/usr/home/ali/projects/cs342/file.txt"
- Grouping Capability
- Current directory (working directory)
  - cd /spell/mail/prog
  - type list

Tree-Structured Directories (Cont)

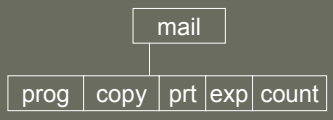
- **Absolute** or **relative** path name
- Creating a new file is done in current directory
- Delete a file

  rm <file-name>
- Creating a new subdirectory is done in current directory

  mkdir <dir-name>

  Example: if in current directory /mail

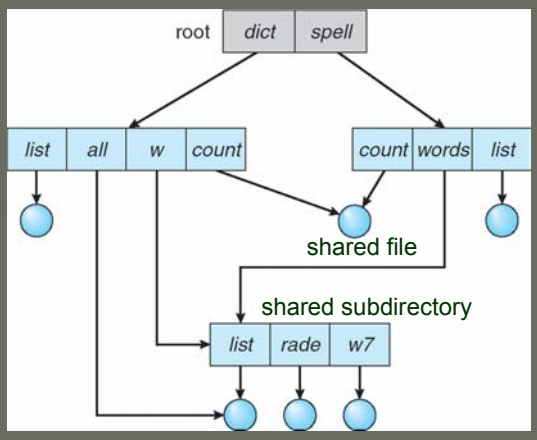
  mkdir count



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

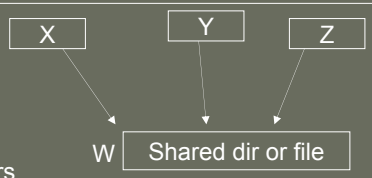
Acyclic-Graph Directories

- Have shared subdirectories and files



Acyclic-Graph Directories (Cont.)

- Two different names (aliasing)



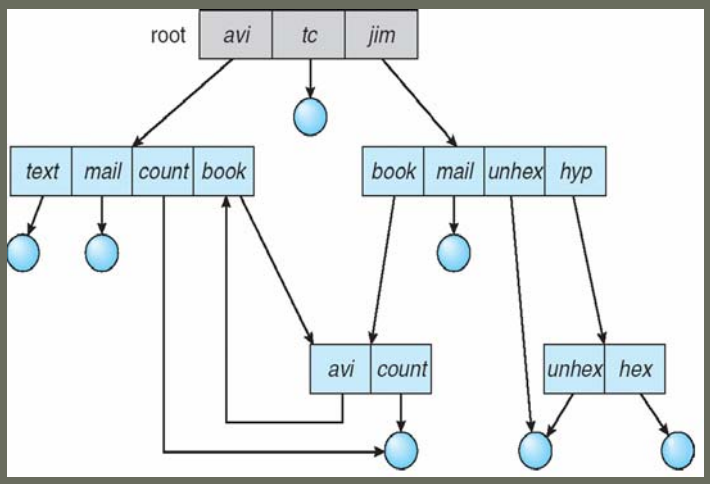
- If *dict* deletes *list* ⇒ dangling pointer
  Solutions:
  - Backpointers, so we can delete all pointers
    Variable size records a problem
  - Use reference count

- New directory entry type
  - **Link** – another name (pointer) to an existing file
  - **Resolve the link** – follow pointer to locate the file

a sequence of entries

a directory	dir	dir	file	file	file	link	file

General Graph Directory



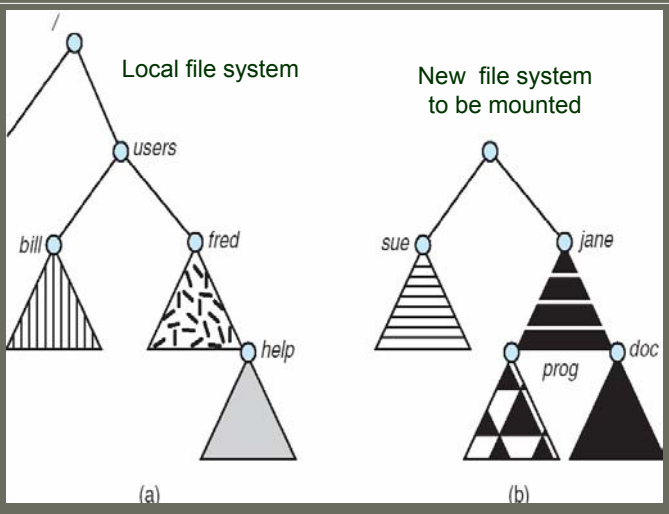
General Graph Directory (Cont.)

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

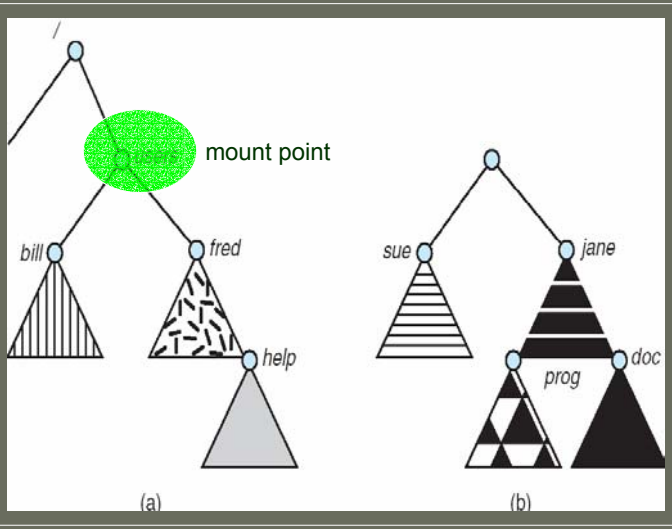
File System Mounting

- A file system must be **mounted** before it can be accessed
- Mounting a new file system means placing the new file system into a location in the local directory tree (local file system) that becomes accessible at system boot time.
- A unmounted file system is mounted at a **mount point**
- Mount point: the place in the local directory tree where the new file system is placed. The root of that file system will be that place in the local file system

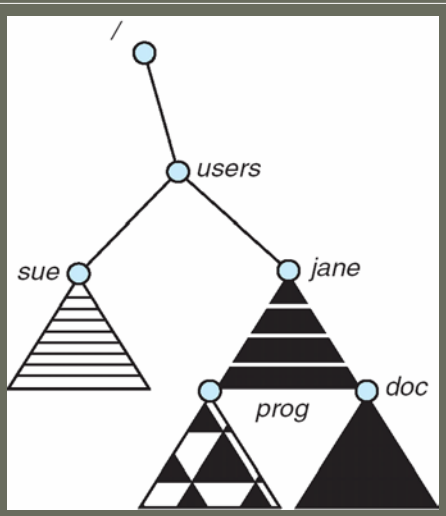
(a) Existing. (b) Unmounted Partition



Mount Point



Local file system appearance after mounting



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

File Sharing

Each user that has an account in the computer has a username and a unique user ID (UID)

The administrator can create groups. A group may have a set of usernames (users) associated with it. Each group has a unique group ID (GID)

Example: group os_team: ali, veli, selcuk, ….

File attributes for a file

…
UID (user ID)
GID (group ID)
User (owner) permissions
Group permissions
Other people permissions
……

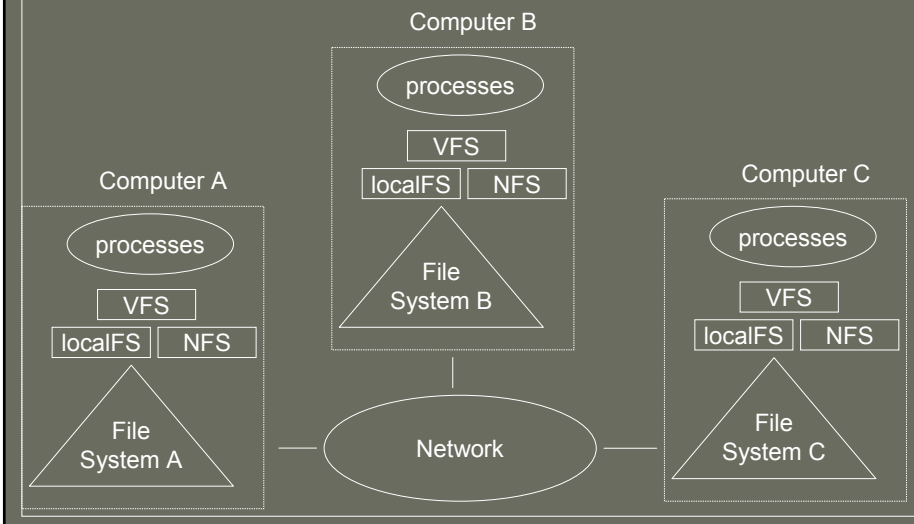
File Sharing – Multiple Users Protection

- Protection is based on the use of UserIDs and GroupIDs.
- Each file has associated protection bits (permissions) for userID and groupID.
  - User ID: read, write, execute?
  - Group ID: read, write, execute?
- **User IDs** identify users, allowing permissions and protections to be per-user
- **Group IDs** allow users to be in groups, permitting group access rights

File Sharing – Remote File Systems

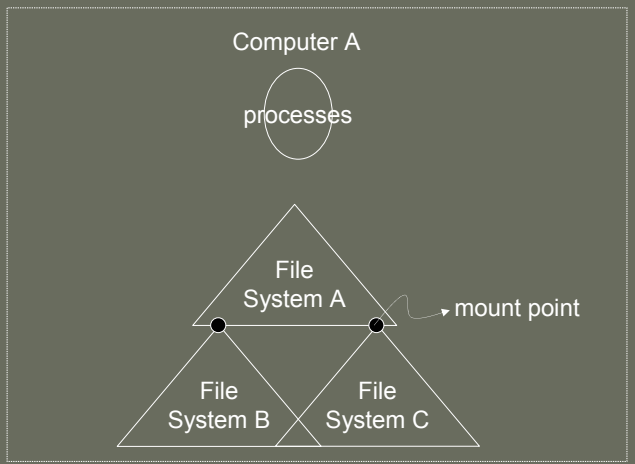
- Uses networking to allow file system access between systems
  - Manually via programs like FTP
  - Automatically, seamlessly using **distributed file systems**
  - Semi automatically via the **world wide web**
- **Client-server** model allows clients to mount remote file systems from servers
  - Server 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

Distributed File System



Distributed File System

File System view at computer A after remote file systems are mounted



File Sharing – Failure Modes

- Remote file systems add new failure modes, due to network failure, server failure
- How to recover from failures?
- Recovery from failure can involve state information about status of each remote request
  - State information: while files are opened; what is the file position pointer, etc.

File Sharing – Consistency Semantics

- **Consistency semantics** specify 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 of the same open file
  - AFS has session semantics
    - Writes only visible to sessions starting after the file is closed

Protection

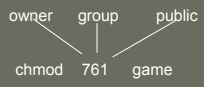
- File owner/creator should be able to control:
  - what can be done (read, write, execute….)
  - by whom (owner, others, group member…)
- Types of access (what can be done)
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users

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

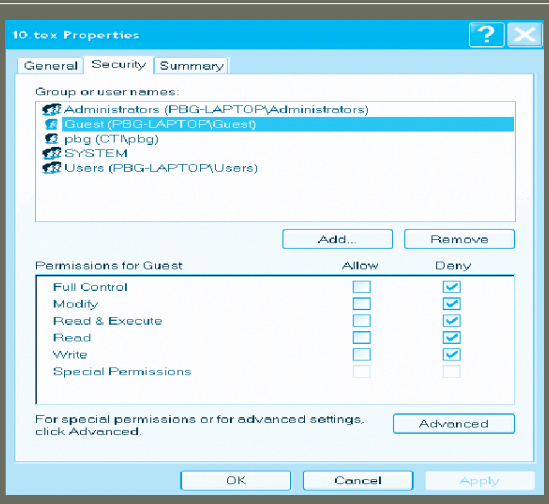
- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say *game*) or subdirectory, define an appropriate access.



Attach a group to a file

chgrp G game

Windows XP Access-control List Management



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/

End of Lecture