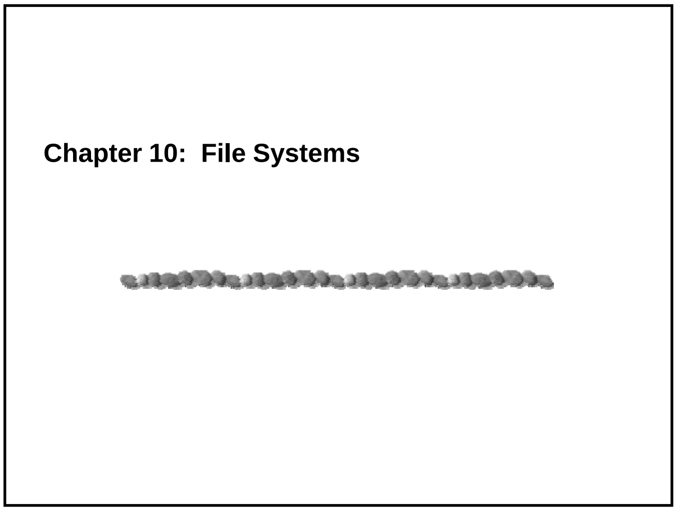
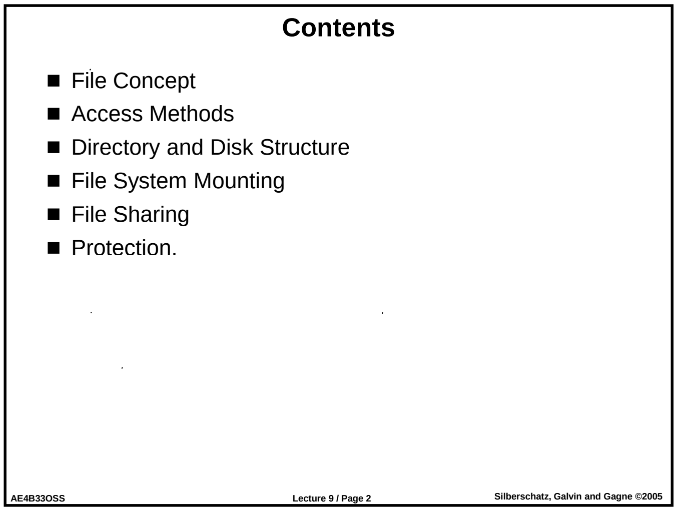
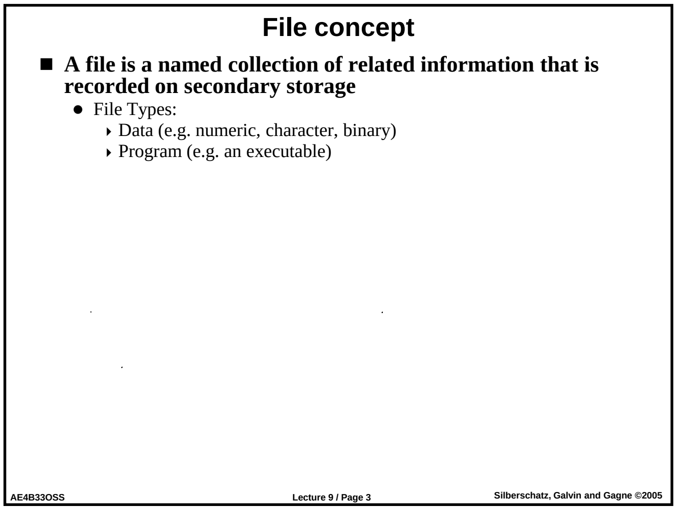


Chapter 10: File Systems



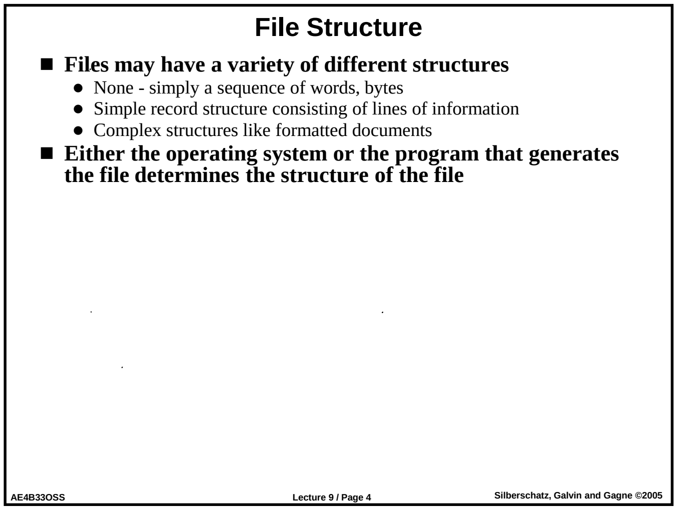
Contents

- File Concept
- Access Methods
- Directory and Disk Structure
- File System Mounting
- File Sharing
- Protection.



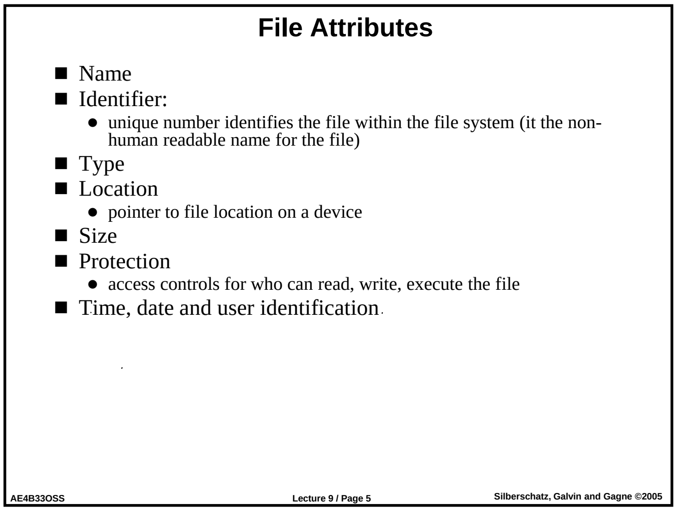
File concept

- **A file is a named collection of related information that is recorded on secondary storage**
  - File Types:
    - Data (e.g. numeric, character, binary)
    - Program (e.g. an executable)



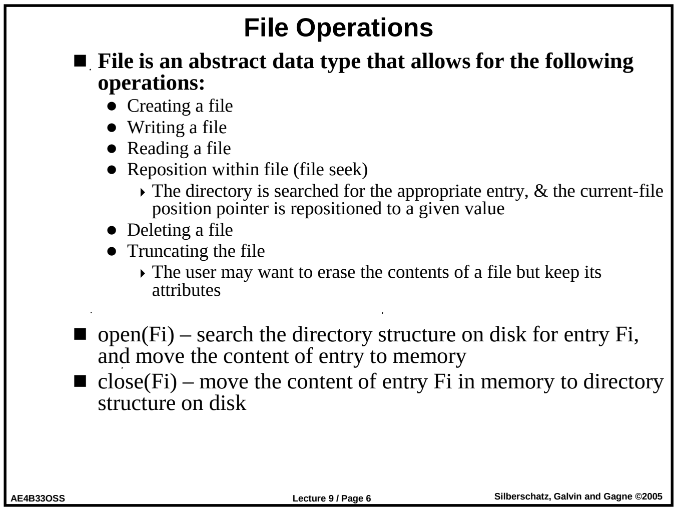
File Structure

- **Files may have a variety of different structures**
  - None - simply a sequence of words, bytes
  - Simple record structure consisting of lines of information
  - Complex structures like formatted documents
- **Either the operating system or the program that generates the file determines the structure of the file**



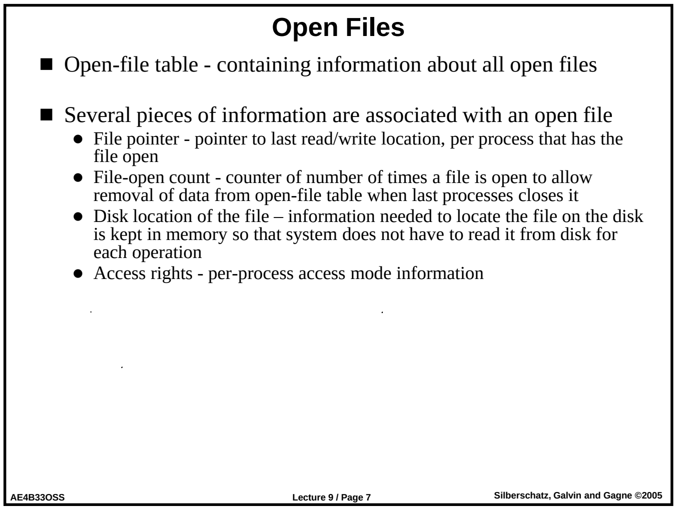
File Attributes

- Name
- Identifier:
  - unique number identifies the file within the file system (it the non-human readable name for the file)
- Type
- Location
  - pointer to file location on a device
- Size
- Protection
  - access controls for who can read, write, execute the file
- Time, date and user identification



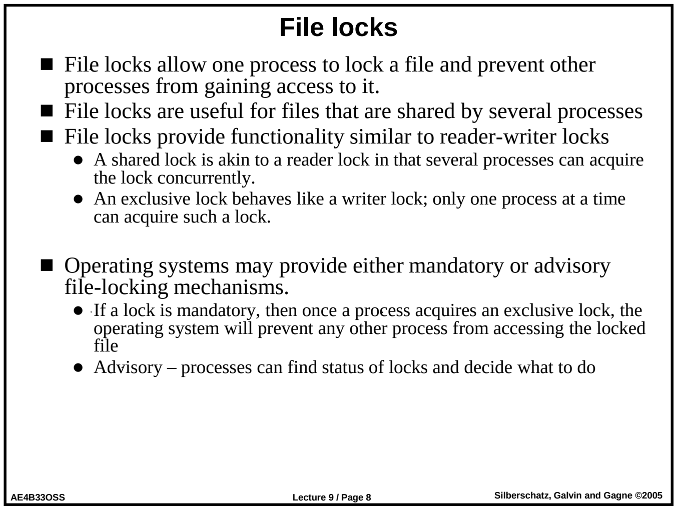
File Operations

- **File is an abstract data type that allows for the following operations:**
  - Creating a file
  - Writing a file
  - Reading a file
  - Reposition within file (file seek)
    - The directory is searched for the appropriate entry, & the current-file position pointer is repositioned to a given value
  - Deleting a file
  - Truncating the file
    - The user may want to erase the contents of a file but keep its attributes
- open(Fi) – search the directory structure on disk for entry Fi, and move the content of entry to memory
- close(Fi) – move the content of entry Fi in memory to directory structure on disk



Open Files

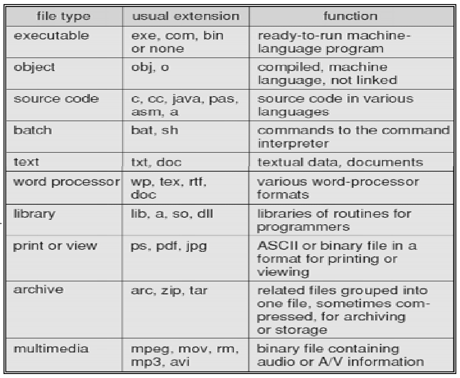
- Open-file table - containing information about all open files
- Several pieces of information are associated with an open file
  - File pointer - pointer to last read/write location, per process that has the file open
  - File-open count - counter of number of times a file is open to allow removal of data from open-file table when last processes closes it
  - Disk location of the file – information needed to locate the file on the disk is kept in memory so that system does not have to read it from disk for each operation
  - Access rights - per-process access mode information



File locks

- File locks allow one process to lock a file and prevent other processes from gaining access to it.
- File locks are useful for files that are shared by several processes
- File locks provide functionality similar to reader-writer locks
  - A shared lock is akin to a reader lock in that several processes can acquire the lock concurrently.
  - An exclusive lock behaves like a writer lock; only one process at a time can acquire such a lock.
- Operating systems may provide either mandatory or advisory file-locking mechanisms.
  - If a lock is mandatory, then once a process acquires an exclusive lock, the operating system will prevent any other process from accessing the locked file
  - Advisory – processes can find status of locks and decide what to do

File Types – Name, Extension



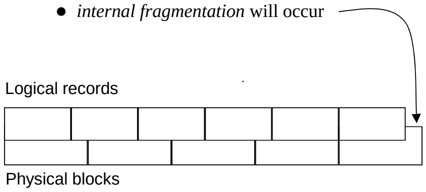
file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual 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 Structures

- Most modern OS support a minimal number of file structures directly
  - e.g. UNIX sees every file as a sequence of 8-bit bytes
- Benefits:
  - applications have more flexibility
  - simplifies the OS

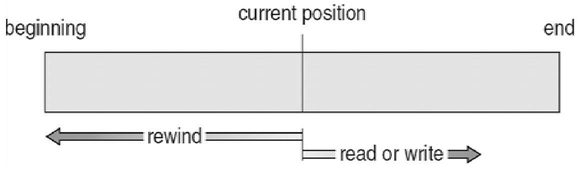
Internal File Structures

- It is unlikely that the physical record size will match the length of the desired logical record
- Logical record s may even vary in length
- *Packing* a number of logical records into physical blocks is a common solution to this problem
  - *internal fragmentation* will occur



Access Methods

- **Information stored in files can be accessed in different ways:**
  - Sequential Access - information in the file is processed in order, one record after another (size of record is dependent on OS, typically a byte)



  - Direct Access - records in a file can be accessed in any order (very useful for databases)

Access Methods

- **Sequential Access**
  - **read next: reads the next portion of the file**
  - **write next: appends to end of file**
  - **reset:**
- **Direct Access –** we have *read n, where n is* the relative block number, rather than *read next, and ·write n rather than write next*
  - An alternative approach is to retain *read next and write next, as with sequential* access, and to add an operation *position file to n, where n is the block number*. Then, to effect a *read n, we would position to n and then read next.*

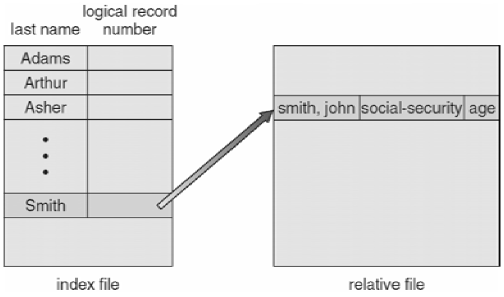
Simulation of Sequential Access on Direct-access File

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

cp = current position

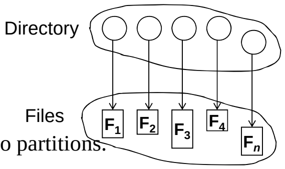
Example of Index and Relative Files

- Make an index file for the file, which contains pointers to various records
  - improves search time

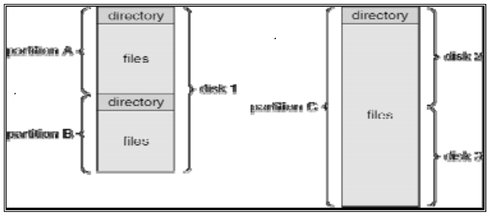


Directory Structure

- Directory is a collection of nodes containing information about files
  - Both the directory structure and the files reside on disk



- Disk can be subdivided into partitions.
  - Partitions also known as minidisks, slices.



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 types of file systems in the Solaris:
  - 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

Logical Organization the Directories

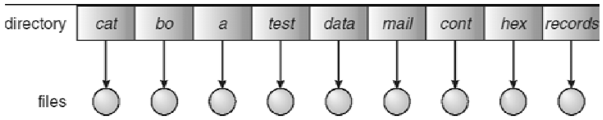
- Operations Performed on Directory
  - Search for a file
  - Create a file
  - Delete a file
  - List a directory
  - Rename a file
  - Traverse the file system
- Organize directories to get
  - **Efficiency** – locating a file quickly
    - The same file can have several different names
  - **Naming** – 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, …)

Five Possible Approaches

- Single-level Directory
- Two-level Directory
- Tree-structured Directories
- Acyclic Graph Directories
- General Graph Directory

Single-Level Directory

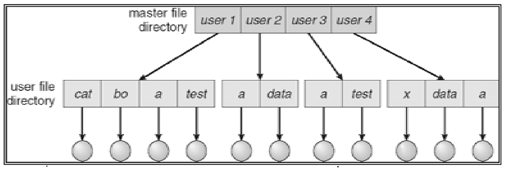
- A single directory for all users



- Advantages
  - Efficiency
- Disadvantages
  - Naming - collisions
  - Grouping – not possible
  - Sharing –not possible

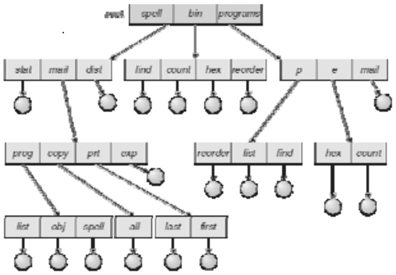
Two-Level Directory (a tree of height 2)

- Separate directory for each user
  - Each user has his own user file directory (UFD)
  - When a user logs on, system's master directory (MFD) will be searched. The MFD is indexed by user name or account number and each user points to UFD for that user



- Advantages
  - Efficient searching
  - Naming - same file name in a different directory; use of path name
- Disadvantages
  - Naming – collisions
  - Grouping capability – not possible except by user

Tree-Structured Directories (a tree of arbitrary height)



If we start with a two-level directory and allow users to create subdirectories, a tree-structured directory results.

Easy to see that simply adding new files and subdirectories to an existing tree-structured directory preserves the tree-structured nature.

- Path names can be of two types: absolute *and* relative.
- *An* absolute path begins at the root and follows a down to the specified file, giving the directory names on the path.
- A relative path name defines a path from the current directory.
- For example, in the tree-structured file system of fig. above if the current directory is *root/spell/mail, then the relative path name prt/first* refers to the same file as does the absolute path name *root/spell/mail/prt/first.*

Tree-Structured Directories

- Advantages
  - Efficient searching – current working directory
  - Naming - same file name in a different directory
  - Grouping capability
- Disadvantages
  - Structural complexity

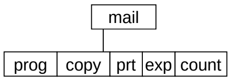
Tree-Structured Directories

- Current directory (working directory):
  - **cd** /spell/mail/prog
  - **type** list
- 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**

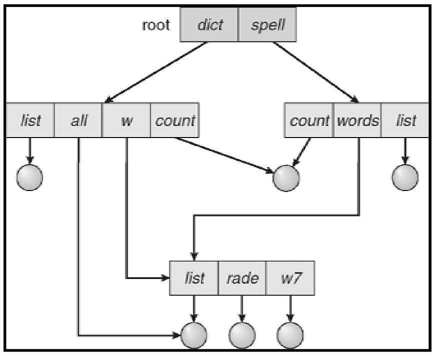
  mkdir count



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

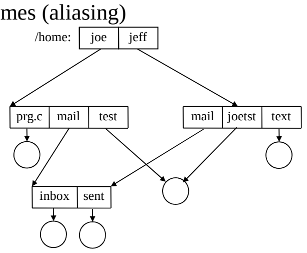
Acyclic-Graph Directories

- Have shared subdirectories and files



Acyclic-Graph Directories

- Entry may have two different names (aliasing)
  - an object can have different names
- Problem: dangling pointers
  - When 'joe' deletes file 'test', the directory item 'joetst' points wrong
  - Solution:
    - Each object has a counter containing a count of references.
    - The counter increments when a new reference is created and decrements when a reference is deleted.
      The object is erased when the counter drops to zero

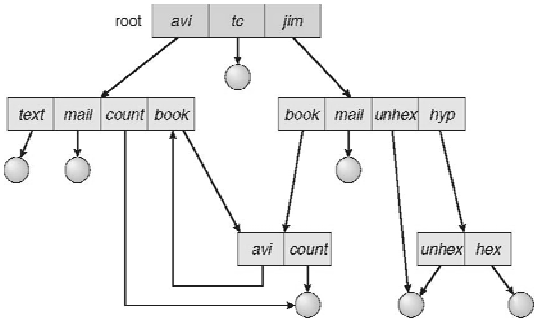


Acyclic-Graph Directories (Cont.)

- Same advantages as tree-structured directory
  - In addition, the same file or directory may have a reference that appears in two or more directories
- Disadvantage is that its structure is more complex
  - The same file or directory may be referred to by many names
  - Need to be cautious of dangling pointers when files are deleted
- Solutions to deletion
  - Just delete the link
  - Preserve the file until all links (i.e., references) are deleted

General Graph Directory

A serious problem with using an acyclic-graph structure is ensuring that there are no cycles
when we add links, the tree structure is destroyed, resulting in a simple graph structure

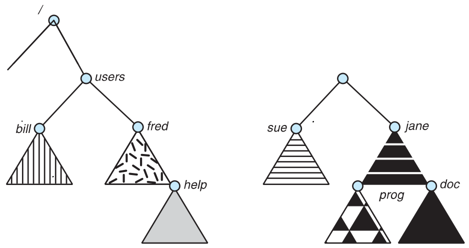


General Graph Directory

- How do we guarantee no cycles?
  - Allow only links to file not subdirectories
  - **Use Garbage collection (**to determine when the last reference has been deleted and the disk space can be reallocated)
    - Garbage collection involves traversing the entire file system, marking everything that can be accessed. Then, a second pass collects everything that is not marked onto a list of free space
    - very expensive and time consuming
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK

File System Mounting

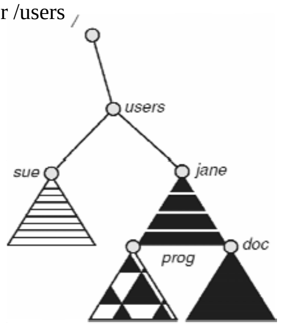
- A file system must be **mounted** before it can be accessed
  - Provide the operating system with the name of the device and a mount point
    - Mounting point is typically an empty directory on local machine
- Fig (a) shows an exiting file system
- Fig (b) shows an unmounted volume residing on */device/dsk.* At this point, only the files on the exiting file system can be accessed .



(a) (b)

Mount Point

Fig below shows the effects of mounting the volume residing on */device/dsk* over /users



File Sharing

- Sharing of files on multi-user systems is desirable
  - User IDs - identify users, allowing permissions and protections to be per-user
  - Group IDs - allow users to be in groups, permitting group access rights
    - POSIX **rwx**|**rwx**|**rwx** scheme
      U G O
    - ACL – Access Control Lists (Windows, some UNIXes)
- 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 – Remote File Systems

- **Uses networking to allow file system access between systems**
  - Manually via programs like FTP
  - Automatically, seamlessly using distributed file systems (DFS)
  - 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 (Network File System) is standard UNIX client-server file sharing protocol
  - CIFS (Common Internet File System) is standard Windows protocol
  - Standard operating system file calls are translated into remote calls
- Distributed Information Systems **(also known as 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 failure of the disk containing the file, corruption of directory structures or other disk management information(collectively called **metadata)**
- Remote file systems add new failure modes, due to network failure, server failure
- Recovery from failure can involve some kind of **state information** may be maintained on both the client and the server
  - If both server and client maintain knowledge of their current activities and open files, then they can seamlessly recover from a failure.
- **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) uses following consistency semantics
    - Writes to an open file by a user are visible immediately to other users who have this file open
    - Sharing file pointer to allow multiple users to read and write concurrently
  - AFS has following session semantics
    - Writes to an open file by a user are not visible immediately to other users that have same file open
    - Once the file is closed, the changes made to it are visible only in session starting later

Consistency Semantics

- Immutable-Shared-Files Semantics
- Once the a file is declared as *immutable shared* by its creator; it cannot be modified (read only)
  - An immutable file has two key properties: its name may not be reused, and its contents may not be altered

Protection

- How to keep information stored in a computer safe from physical damage (issue of reliability) and improper access (issue of protection)?
  - Reliability is generally provided by duplicate copies of files
  - Protection mechanisms provide controlled access by limiting the types of file access. Access is permitted or denied depending on several factors
- Several different Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**
- **File owner/creator should be able to control access to file**
  - What can be done to the file?
  - Who can do what to the file?

Access Control

- Access Lists can be used to control mode of access to a file (i.e. read, write, execute permissions)
- Three classes of users (use 3 bits per class)
  - **Owner:** user who created the file
  - **Group:** set of users who are sharing the file & need similar access is group
  - **Public (Universe):** All other users in the system constitute the universe

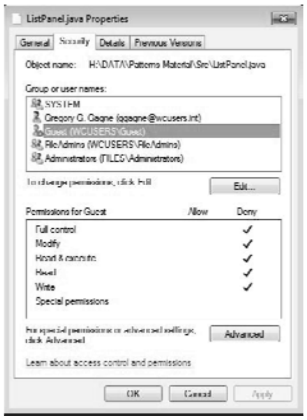
	RWE
a) owner access 7	1 1 1
b) group access 6	1 1 0
c) public access 1	0 0 1

- Users can be added to a group that allows them access to certain files
- On Unix/Linux systems, mode of access can be changed by owner using chmod command

Other protection Approaches

- Associate A password with each file
  - too hard to remember
- A password per subdirectory
  - too course-grained

Windows 7 Access-Control List Management



Object name: H:\DATA\Patterns Material\Src\ListPanel.java

Group or user names:

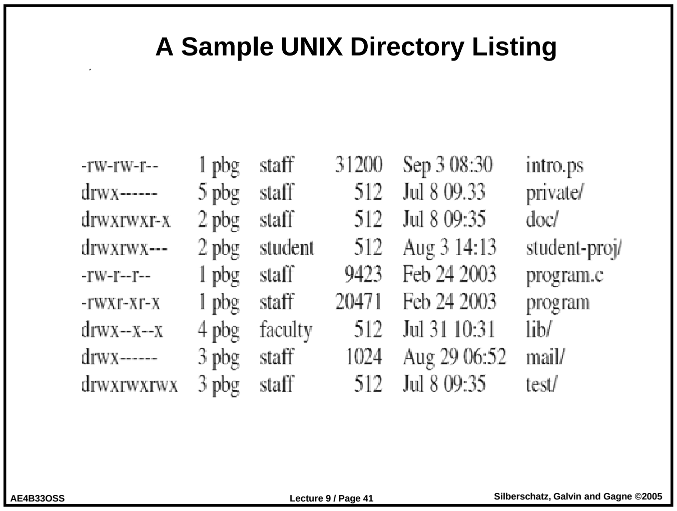
SYSTEM
Gregory G. Gagne (ggagne@wcusers.int)
Guest (WCUSERS\Guest)
FileAdmins (WCUSERS\FileAdmins)
Administrators (FILES\Administrators)

To change permissions, click Edit.

Permissions for Guest	Allow	Deny
Full control		✓
Modify		✓
Read & execute		✓
Read		✓
Write		✓
Special permissions		

For special permissions or advanced settings, click Advanced.

Learn about access control and permissions



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/
```