CIS 415: Operating Systems

File Systems

Spring 2012

Prof. Kevin Butler

UNIVERSITY OF OREGON

- Last class:
  - Virtual Memory
- Today:
  - Files

A System Problem?

- Got some data in your program
  - Want to keep it for a while
- Got a long running program
  - Want to prevent loss of data if it crashes
- Got a lot of programs, system resources, data, etc. stored
  - Want a mechanism to refer to them all

File System Interface

UNIVERSITY OF OREGON

- Most visible part of the OS
- Consists of
  - Files
  - Directories
- And sometimes
  - Partitions

What is a file?

- A repository for data
- Is long lasting (until explicitly deleted).
- Also, may refer to a system resource (device)

Why not just an address space?

UNIVERSITY OF OREGON

- You may want data to persist longer than a process
- You may want data that is larger than a virtual address space
- Easier to share the data across processes.

Two aspects to consider

UNIVERSITY OF OREGON

- User's view
  - Naming, type, structure, access, attributes, operations, …
- System implementation

Naming

- Typically x.y
- x is supposed to give some clue about contents
- y is supposed to be the nature of the file.

Structure

- Byte stream
- Sequence of Records
- Indexed Records

Types of File Objects

UNIVERSITY OF OREGON

- Regular files (containing data)
- Directories
- Character special files (access a character at a time)
- Block special files (access a block at a time)

Attributes

UNIVERSITY OF OREGON

- protection, creator, owner, creation time, access time, current size, max size, record length, lock flags, ...

File Operations

UNIVERSITY OF OREGON

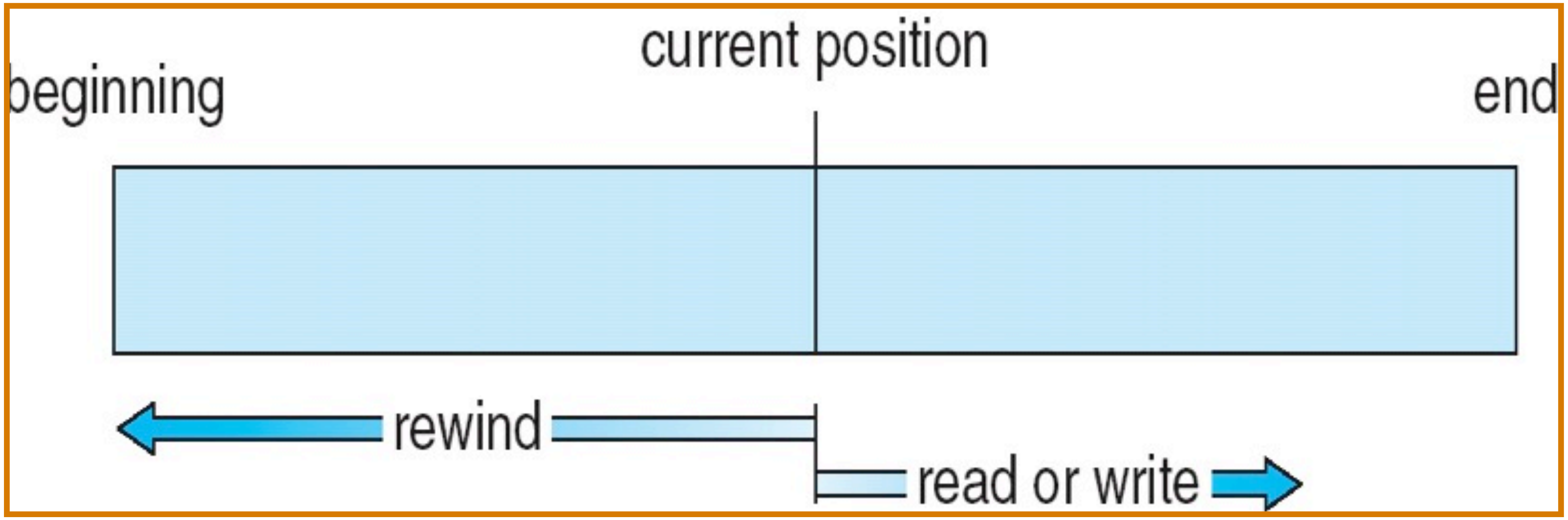
- Create, Delete, Open, Close, Read, Write, Append, Seek, Get attributes, Set attributes, Rename

- Exercise: Get acclimated to UNIX file system calls
  - What are the file system calls when you write a new document in a text editor? What if you go back to modify it later?

File Operations

- Sequential Access
  - reset
  - read next (advance file pointer automatically)
  - write next (advance file pointer automatically)
- Direct Access
  - read *n*
  - write *n*
  - position to *n*
  - read next
  - write next
- *n* = relative block number

Sequential File Access



Sequential File Access

UNIVERSITY OF OREGON

Simulation of 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;

Directory

- A way of organizing files.
- Each directory entry has:
  - File/directory name
  - A way (pointer) to get to the data blocks of that file/directory

Filesystem details

- Flat (only 1 directory) vs. hierarchical file system
- File names: relative vs. absolute
- Directory Operations:
  - Create
  - Delete directory
  - Open Dir
  - Close Dir
  - Read Dir
  - Rename
  - Link (allow a file to appear in more than 1 directory)
  - Unlink

Links in UNIX

- Makes a file appear in more than 1 directory.
- Is a convenience in several situations.
- 2 types of links:
  - Soft links
  - Hard links

Soft links

UNIVERSITY OF OREGON

- Create a file which contains the name of the other file.
- Use this path name to get to the actual file when it is accessed.
- Problem: Extra overhead of parsing and following components till the file is reached.

Hard links

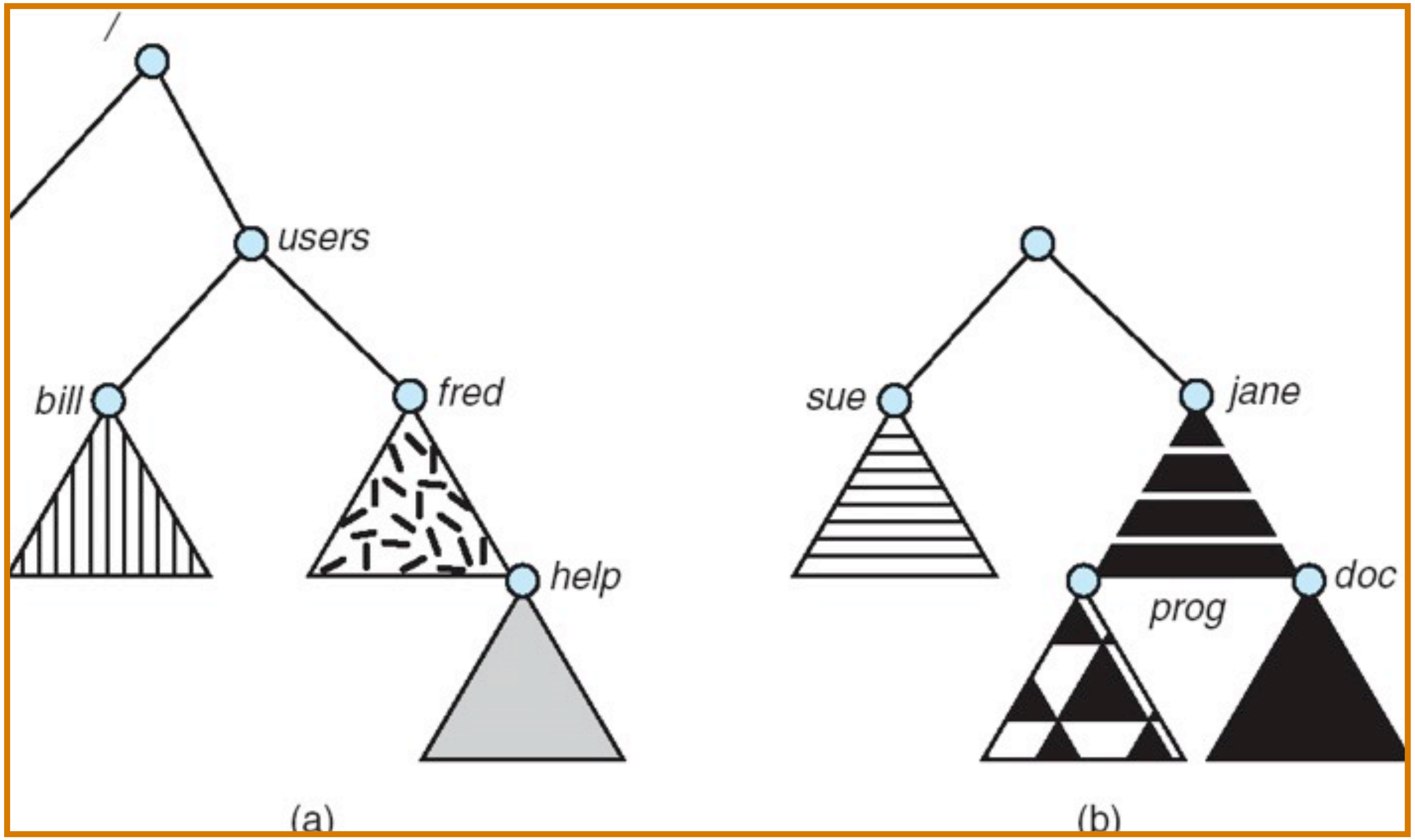
UNIVERSITY OF OREGON

- Create a directory entry and make a reference that file.
  - Others may reference the same file
- Problem: What if the creator wants to delete the file?
  - There are still other references to the file, potentially.
  - Cannot free up until all the other references are removed.
  - Done by keeping a counter that is incremented for each hard link. On removing a link, decrement counter. Only if counter is 0, remove the file.

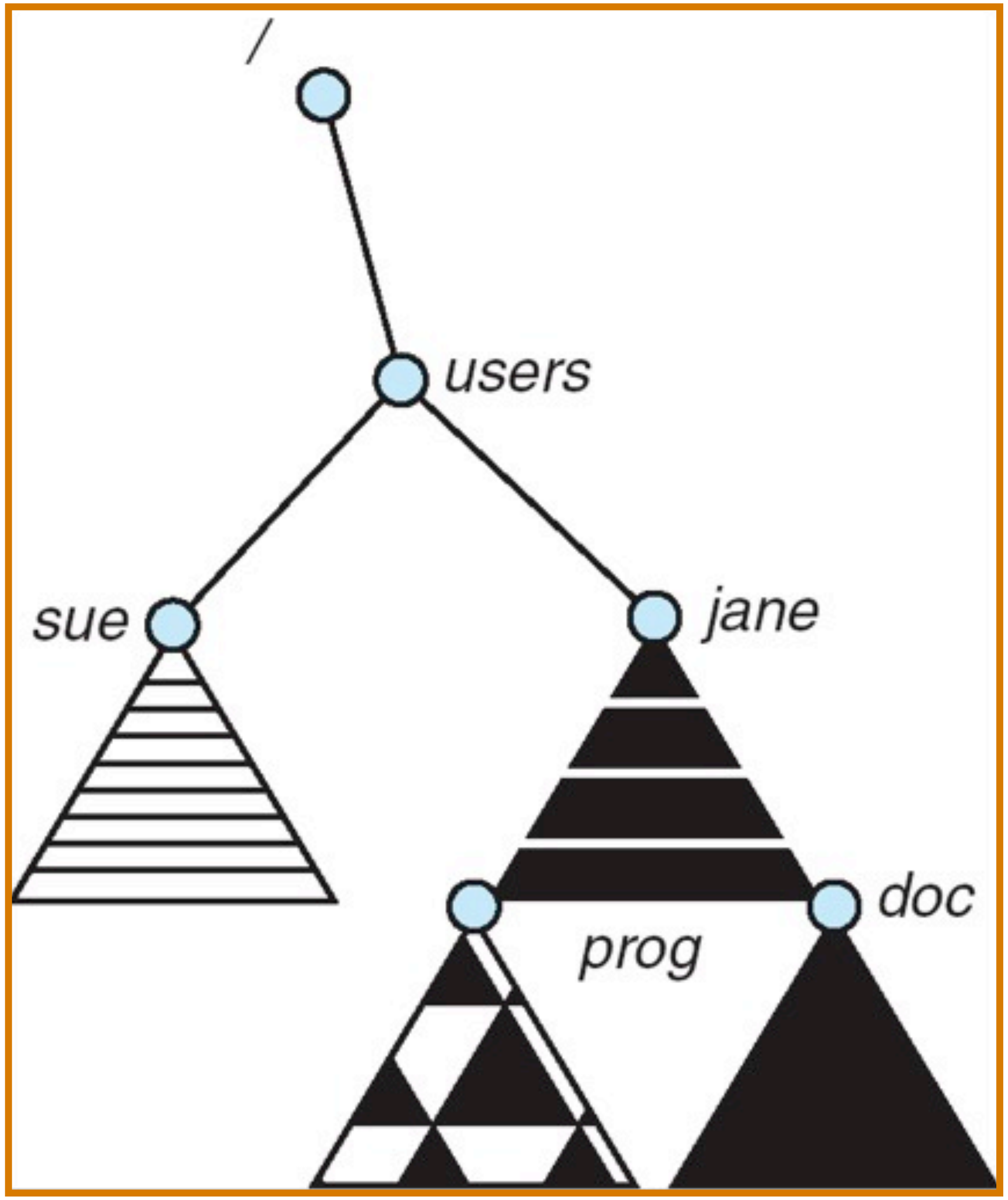
Partitions

- A way of organizing directories
- Each partition contains a:
  - File system of directories
- Examples:
  - Root file system '/'
  - Boot file system '/boot'
  - User's homes '/home'

File System Mounting



File System Mounting



File Sharing

- In a multi-user system,
  - There is interest in sharing files
- System files
  - Shared by all
  - Examples?
- Per user files
  - May want to work with others
  - Or with particular groups of users

File Sharing

UNIVERSITY OF OREGON

- Where are the files to share?
- Files and Links
  - Short cut through the file system
    - *Hard* and *soft*
- Directories
  - Must provide the other user or group access to your directory
- Remote file systems
  - Access files on another machine
  - Must provide the other user or group access to your machine and directory

User and Group Identity

UNIVERSITY OF OREGON

- User identity
  - UID in UNIX
  - Security ID in Windows NT
- Group identity
  - GID in UNIX
  - Group ID in Windows NT
- Give users and/or groups access to your files to share them

Protection

UNIVERSITY OF OREGON

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

Access Control: Mode Bits

UNIVERSITY OF OREGON

- Three classes of users: public, group, owner
- Three types of access permissions:
  - read, write, execute

- Example:

	Owner	Group	Public
rwx=	111	101	101
Octal	7	5	5

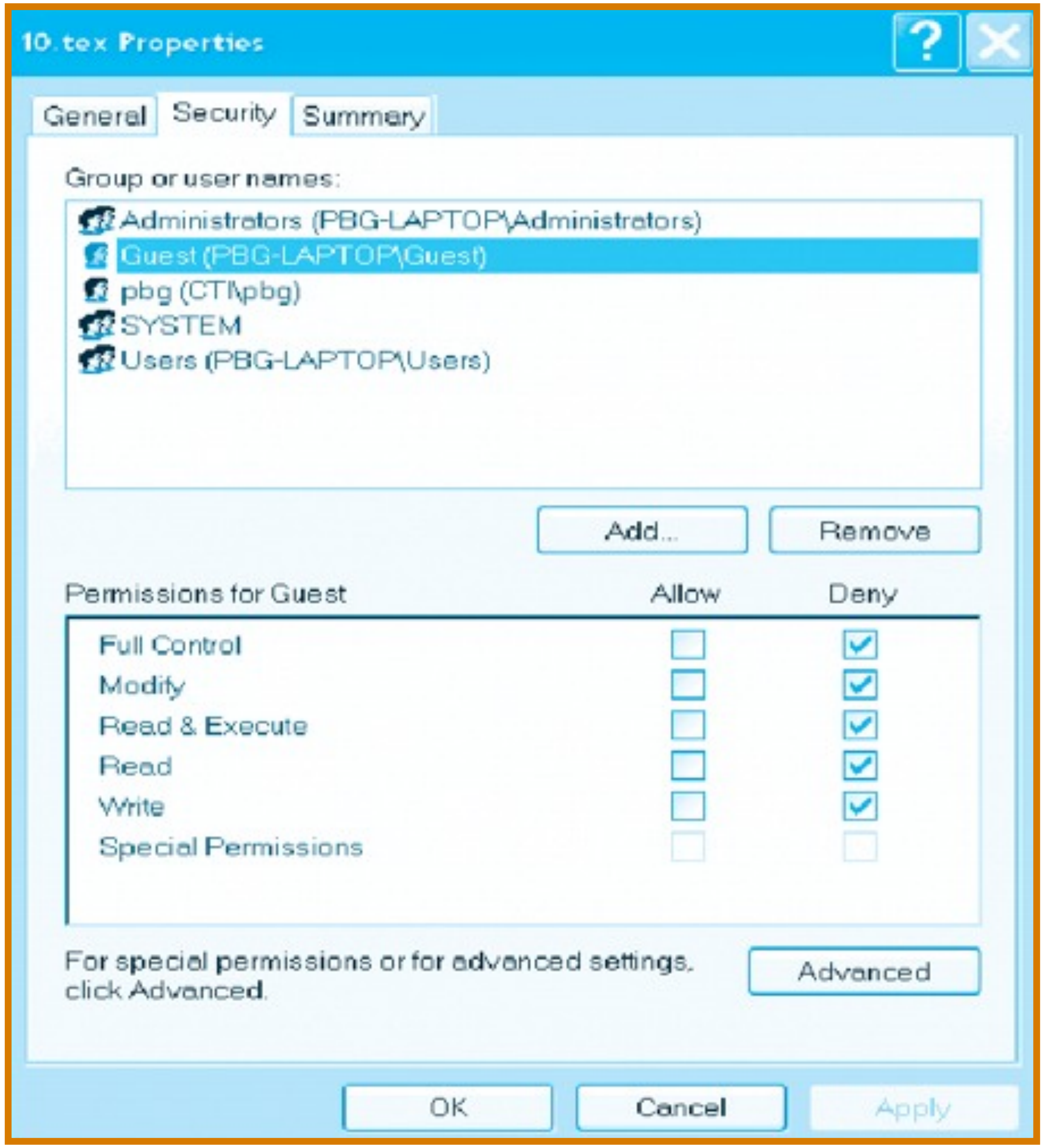
What if no exec access and only owner can read/write?

UNIX File Permissions

UNIVERSITY OF OREGON

-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/

WinXP Access Control Lists



Access Control/Authorization

UNIVERSITY OF OREGON

- An ***access control*** system determines what *rights* a particular *entity* has for a set of *objects*
- It answers the question
  - E.g., do you have the right to read /etc/passwd
  - Does Alice have the right to view the CIS website?
  - Do students have the right to share project data?
  - Does Prof. Butler have the right to change your grades?
- An *Access Control Policy* answers these questions

Simplified Access Control

UNIVERSITY OF OREGON

- Subjects are the active entities that do things
  - E.g., you, Alice, students, Prof. Butler
- Objects are passive things that things are done to
  - E.g., /etc/passwd, CS website, project data, grades
- Rights are actions that are taken
  - E.g., read, view, share, change

Policy Goals

UNIVERSITY OF OREGON

- Secrecy
  - Don’t allow reading by unauthorized subjects
  - Control where data can be written by authorized subjects
    - Why is this important?
- Integrity
  - Don’t permit dependence on lower integrity data/code
    - Why is this important?
  - What is “dependence”?
- Availability
  - The necessary function must run
  - Doesn’t this conflict with above?

The Access Matrix

UNIVERSITY OF OREGON

- An access matrix is one way to represent policy.
  - Frequently used mechanism for describing policy
- Columns are objects, subjects are rows.
- To determine if S_i has right to access object O_j, find the appropriate entry.
- Succinct descriptor for O(|S|*|O|) entries
- There is a matrix for each right.

	O_1	O_2	O_3
S_1	Y	Y	N
S_2	N	Y	N
S_3	N	Y	Y

Access Control

- Suppose the private key file for J is object O_1
  - Only J can read
- Suppose the public key file for J is object O_2
  - All can read, only J can modify
- Suppose all can read and write from object O_3
- What's the access matrix?

	O_1	O_2	O_3
J	?	?	?
S_2	?	?	?
S_3	?	?	?

Trusted Processes

UNIVERSITY OF OREGON

- Does it matter if we do not trust some of J's processes?

	O_1	O_2	O_3
J	R	RW	RW
S_2	N	R	RW
S_3	N	R	RW

Secrecy

UNIVERSITY OF OREGON

- Does the following protection state ensure the secrecy of J's private key in O_1?

	O_1	O_2	O_3
J	R	RW	RW
S_2	N	R	RW
S_3	N	R	RW

Integrity

- Does the following access matrix protect the integrity of J's public key file O_2?

	O_1	O_2	O_3
J	R	RW	RW
S_2	N	R	RW
S_3	N	R	RW

Protection vs Security

- Protection
  - Security goals met under *trusted* processes
  - Protects against an error by a non-malicious entity
- Security
  - Security goals met under *potentially malicious* processes
  - Protects against any malicious entity
  - Hence, For J:
    - Non-malicious process shouldn't leak the private key by writing it to O_3
    - A potentially malicious process may contain a Trojan horse that can write the private key to O_3

Least Privilege

UNIVERSITY OF OREGON

- Limit permissions to those required and no more
- Consider three processes for user J
  - Restrict privilege of the process J_1 to prevent leaks

	O_1	O_2	O_3
J_1	R	R	N
J_2	N	RW	N
J_3	N	R	RW

Distributed File System

UNIVERSITY OF OREGON

- Remote directories are visible from the local machine
  - *Server* has the files
  - *Client* makes file requests
- Share by partition
  - Mount to remote file system
  - Convert local file request to client-server request to access remote file
  - Like an RPC

Distributed File System

- Examples
  - Network file system (NFS)
    - What we use
  - Andrew file system (AFS)
    - Other universities use
  - Distributed Computing Environment (DCE)
    - Commercial system
  - Distributed file access
    - WWW, ftp, CIFS (Samba), …

Distributed File Systems

UNIVERSITY OF OREGON

- Consistency is a major issue
- Scenario
  - Suppose I open a file on an NFS partition
  - And suppose you open a file on an NFS partition
- And we both write the file
  - Who updates the file?
  - What if our writes overlap?
- Similar to concurrency control, but spanning machines
  - How much expense is necessary to enforce concurrency?
  - Is it worth it?

Summary

- File System Interface
  - Files
  - Directories
  - Partitions
- Operations on the interface
  - Mounting (partitions)
  - Sharing (files)
  - Protection (files)
- Distributed file systems

- Next time: File System Implementation