CISC 7310X

C12: Files and Directories: User's Perspective

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Design File Systems

- User's perspective
- System's perspective

File Systems: Requirements

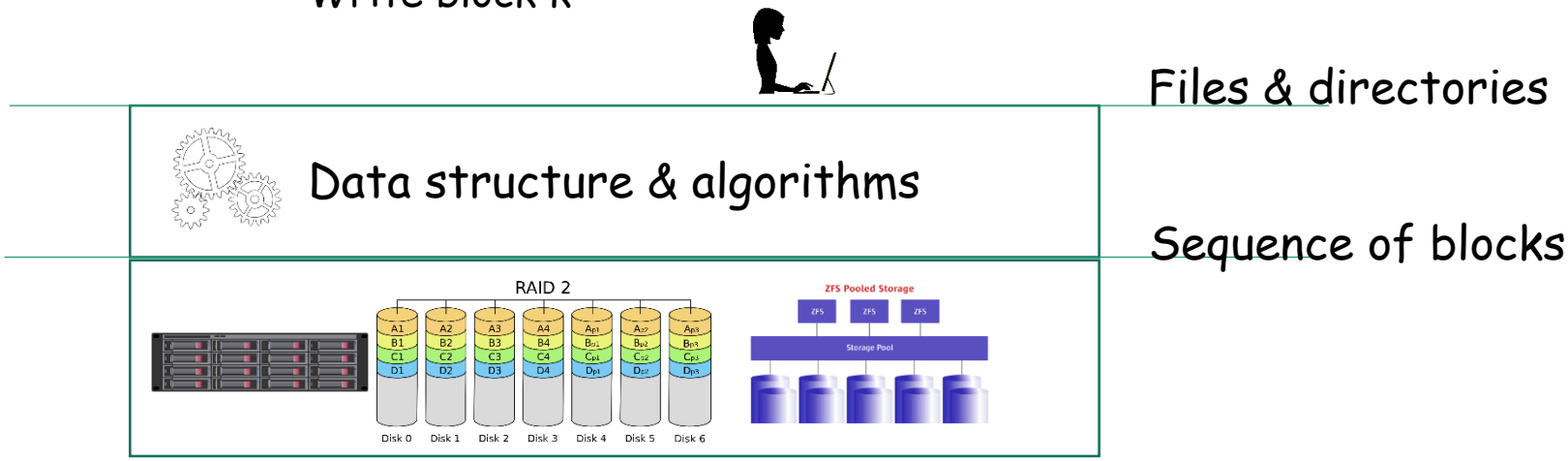
- Long term data storage
  1. It must be possible to store a very large amount of data.
  2. Data must survive termination of process using it.
  3. Multiple processes must be able to access data concurrently.

Common Queries

1. How do you find information?
2. How do you keep one user from reading another user's data?
3. How do you know which blocks are free?

Storage Devices: Disks

- Long-term storage devices are abstracted as "disks"
- A disk is abstracted as a linear sequence of fix-sized blocks
  - Two operations
    - Read block k
    - Write block k

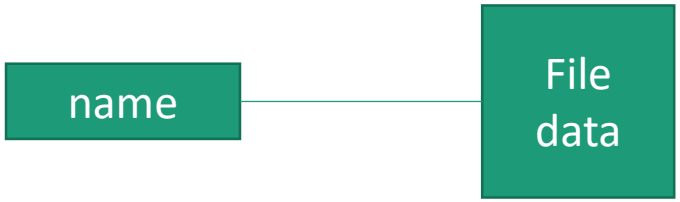


User's Perspective

- Files
  - Concept
  - File naming, structure, types, access, attributes, operations
  - Programming interfaces
- Directories
  - Hierarchical directory systems
  - Path names
  - Directory operations

File: Concept

- Abstraction mechanism
  - Logical units of data created by processes
    - Smallest allotment of logical secondary storage
  - Data stored in those units can be read it back later
    - Where and how the data are stored are abstracted away
    - A unit is identified with a "name"



Characterizing Files

- Information in a file is defined by its creator
  - A few dimensions to characterize files
    - Regular files & directories
    - Regular file
      - File names, and file name extensions (file names)
      - Structures of file content
      - Formatted/unformatted (binary or text) file

Regular Files and Directories

- Many OSes support a few types of files
  - Regular files: user data
  - Directories: contains data about the structure of the file system

File Naming

- System dependent
- Characters allowed?
- Length of filenames?
- Case sensitive or insensitive?
  - e.g., Windows is case insensitive; while Unix isn't
- Multi-part filenames?
  - Most support two-part filenames
    - Second part is file extension
    - Some OSes or applications enforces it (Windows, C & Java compilers), some do not (Unix)

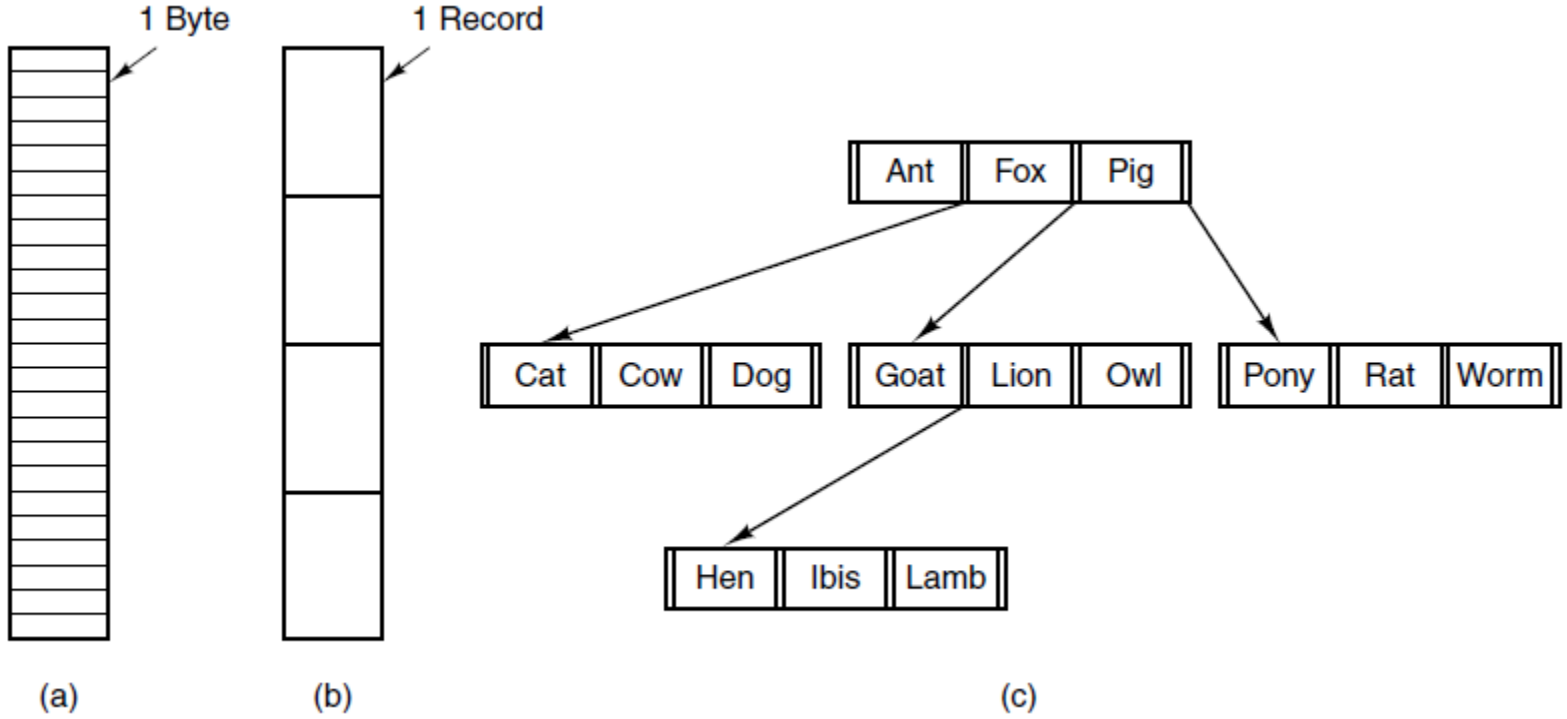
File Extension: Examples

Extension	Meaning
.bak	Backup file
.c	C source program
.gif	Compuserve Graphical Interchange Format image
.hlp	Help file
.html	World Wide Web HyperText Markup Language document
.jpg	Still picture encoded with the JPEG standard
.mp3	Music encoded in MPEG layer 3 audio format
.mpg	Movie encoded with the MPEG standard
.o	Object file (compiler output, not yet linked)
.pdf	Portable Document Format file
.ps	PostScript file
.tex	Input for the TEX formatting program
.txt	General text file
.zip	Compressed archive

- [Figure 4-1 in Tanenbaum & Bos, 2014]

File Structure

- Byte sequence, record sequence, or tree



- [Figure 4-2 in Tanenbaum & Bos, 2014]

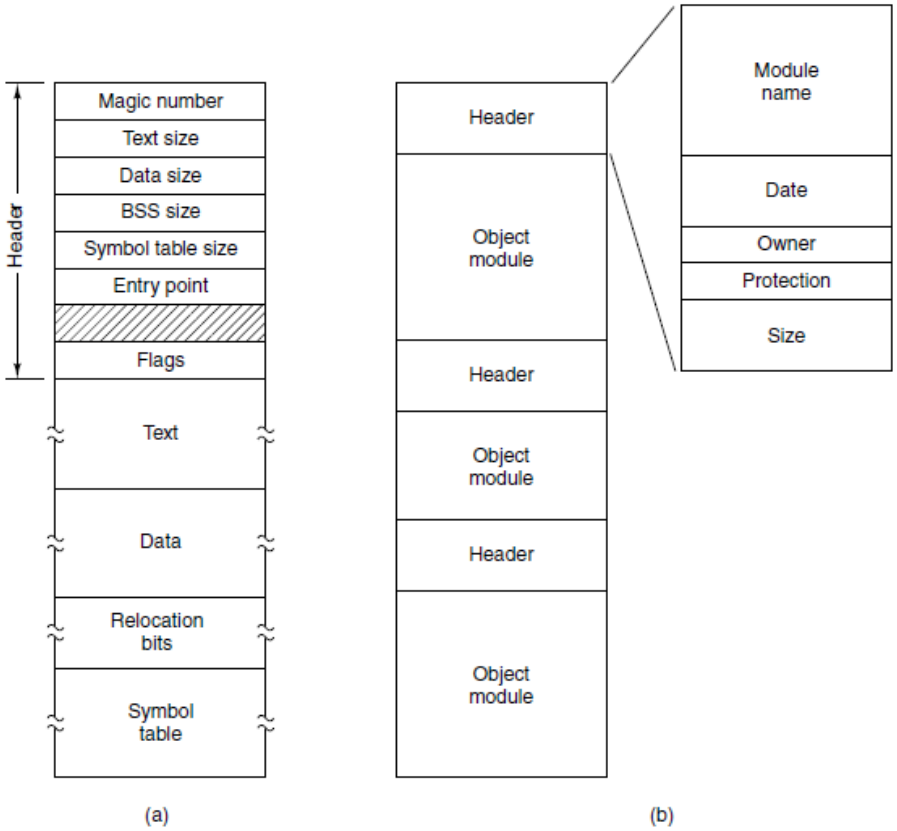
Byte, Record, and Tree

- Byte sequence
  - Read and write a byte; maximal flexibility
- Record sequence
  - Read and write a record
- Tree
  - A tree of records, each has a key field, allowing efficient searching

Regular Files

- Character/text files
- Binary files

Binary File: Example



- (a) executable (b) an archive [Figure 4-2 in Tanenbaum & Bos, 2014]

File Access

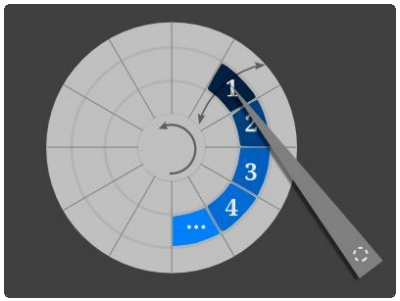
- Sequential access
- Random-access
  - Essential for many applications, such as, a database system

Sequential Access

- Generally, provides
  - Read, write, and rewind

Random Access File

- Generally, provides
  - Length
    - the size of the file
  - File pointer/Cursor
    - an index into the implied array, pointing to the byte next read reads from or next write writes to.
    - Each read or write results an advancement of the pointer
    - The file pointer can be obtained
  - Seek:
    - Set the file pointer
  - Generally, unformatted files (binary files)



File Attributes

- An OS associates additional information about a file
  - Example: the data and time the file was last modified, and the size of the file
  - Attributes or metadata

File Attributes: Examples

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file was last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

- [Figure 4-4 in Tanenbaum & Bos, 2014]

File Operations

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

Lock

- Some OSes provides facilities for locking an open file (or sections of a file).
- File locks
  - A process locks a file and prevents other processes from gaining access to it
- Some OSes provides one, some, or all:
  - Shared lock (reader lock): several processes can acquire the lock concurrently (for reading)
  - Exclusive lock (writer lock): only one process at a time can acquire such a lock
  - Mandatory (e.g., Windows) or advisory (e.g., Unix)

Questions?

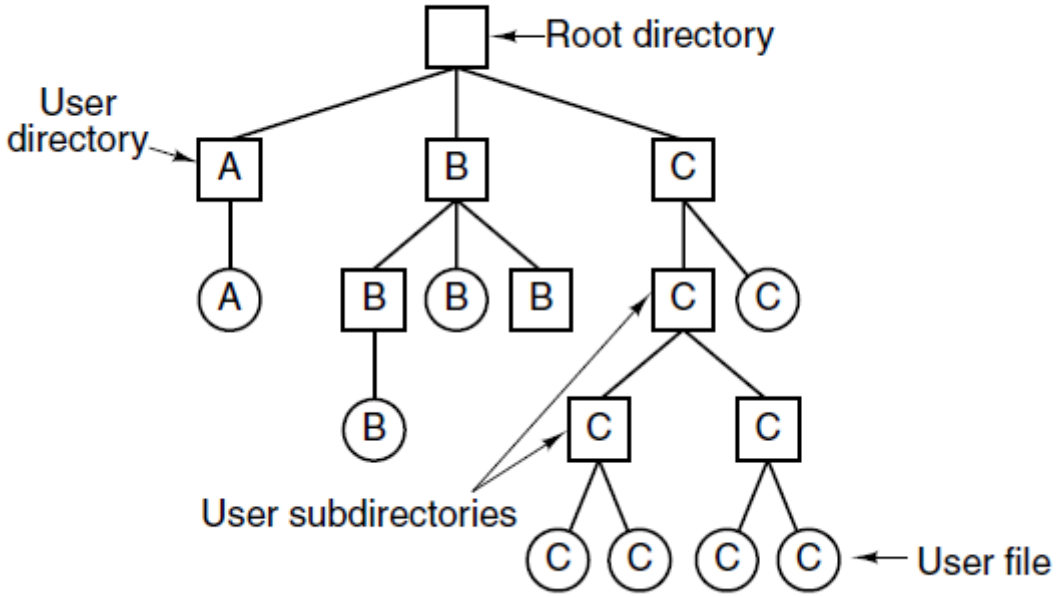
- Concept
- File naming, structure, types, access, attributes, operations
- Programming interfaces

Directories

- Directories or folders are files whose data are about regular files and organizations
- Most OSes use hierarchical directory systems

Hierarchical Directory System

- Root directory
- Node
  - User directory
  - File



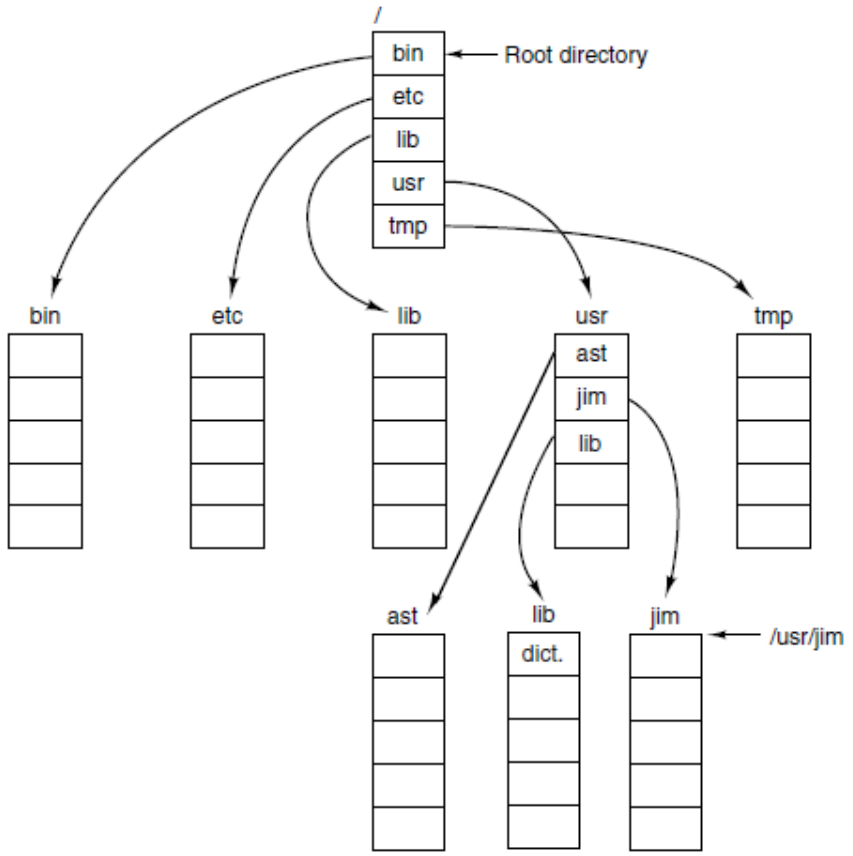
- [Figure 4-7 in Tanenbaum & Bos, 2014]

Path Names

- A path name of a file (or directory) is a traversal of the file system tree or the directory tree to the file (or directory)
  - Any traversal is a valid path name
- Absolute path
- Relative path

Directory Tree

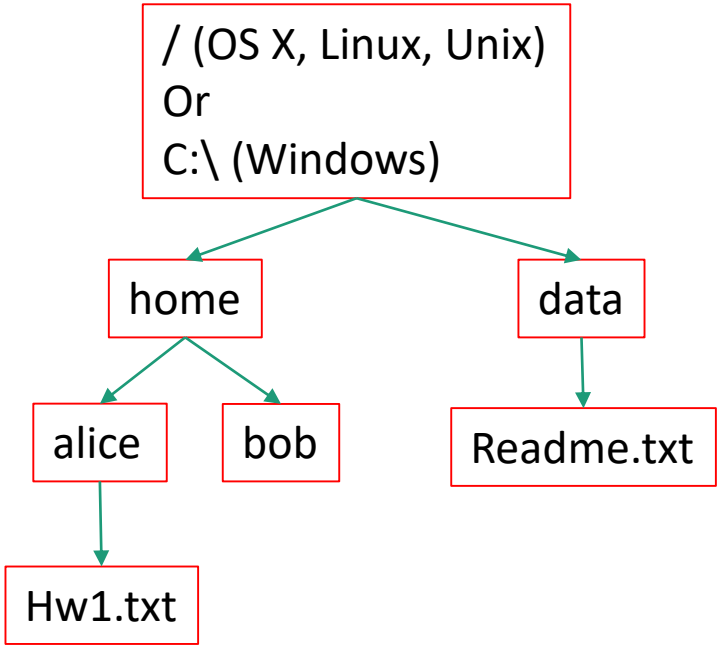
- File system tree
  - Starting at the root
- A subtree
  - Directory tree



- [Figure 4-7 in Tanenbaum & Bos, 2014]

Path Name: Examples

- File system tree traversal
  - Example: identify Hw1.txt
  - OS X
    - /home/alice/Hw1.txt
  - Windows
    - C:\home\alice\Hw1.txt
  - Delimiter
    - Windows: "\"
    - Unix-like: "/"

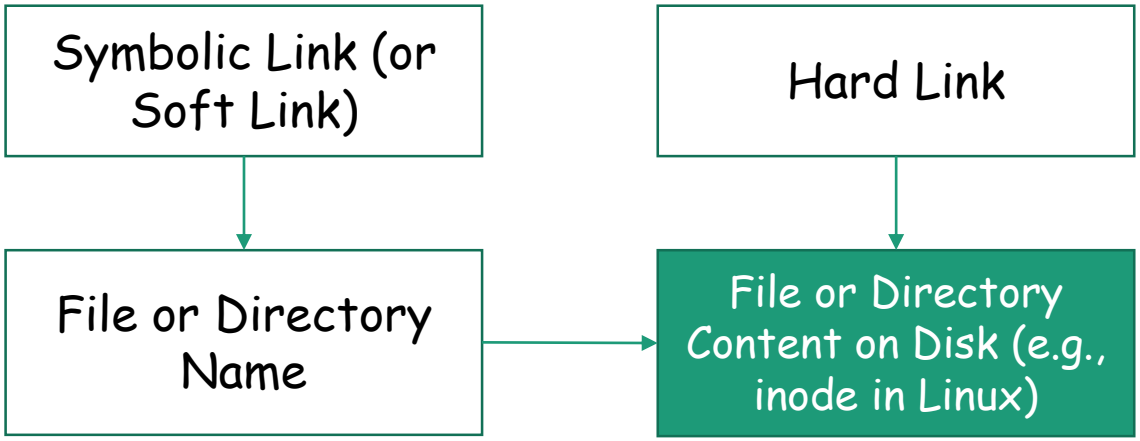


Relative and Absolute Path

- Absolute path
  - Contains the root element and the complete directory list required to locate the file
    - Example: /home/alice/Hw1.txt or C:\home\alice\Hw1.txt
- Relative path
  - Needs to be combined with another path in order to access a file.
  - Example
    - alice/Hw1.txt or alice\Hw1.txt, without knowing where alice is, a program cannot locate the file
  - "." is the path representing the current working directory
  - ".." is the path representing the parent of the current working directory

Symbolic Link and Hard Link

- A file-system object (source) that points to another file system object (target).
  - Symbolic link (soft link): an "alias" to a file or directory name
  - Hard link: another name of a file or directory



Transparency to Users

- Links are transparent to users
  - The links appear as normal files or directories, and can be acted upon by the user or application in exactly the same manner.
- Create symbolic links from the Command Line
  - Unix-like: ln
  - Windows: mklink

Unix-like OS: Example

- Unix-like (e.g., Linux, OS X): "#" leads a comment. do the following on the terminal,
  - echo "hello, world!" > hello.txt # create a file, the content is "hello, world!"
  - ln -s hello.txt hello_symlink.txt # create a soft link to hello.txt
  - ls -l hello_symlink.txt # list the file, what do we observe?
  - cat hello_symlink.txt # show the content using the symbolic link, what do we observe?
  - ln hello.txt hello_hardlink.txt # create a hard link
  - ln -l hello_hardlink.txt # observation?
  - cat hello_hardlink.txt # observation?
  - mv hello.txt hello2.txt # rename hello.txt
  - ls -l hello_symlink.txt # observation?
  - ln -l hello_hardlink.txt # observation?
  - cat hello_symlink.txt # observation?
  - cat hello_hardlink.txt # observation

Window: Example

- On Windows, it requires elevated privilege to create file symbolic link. Do not type the explanation in "()".
  - echo "hello, world!" > hello.txt (create a file, the content is "hello, world!")
  - mklink hello_symlink.txt hello.txt (create a soft link to hello.txt)
  - dir hello_symlink.txt (list the file, what do we observe?)
  - more hello_symlink.txt (show the content using the symbolic link, what do we observe?)
  - mklink /h hello_hardlink.txt hello.txt (create a hard link to hello.txt)
  - dir hello_hardlink.txt (observation?)
  - more hello_hardlink.txt (observation?)
  - move hello.txt hello2.txt (rename hello.txt)
  - dir hello_symlink.txt (observation?)
  - dir hello_hardlink.txt (observation?)
  - more hello_symlink.txt (observation?)
  - more hello_hardlink.txt (observation?)

Directory Operations

- Create
- Delete
- Opendir
- Closedir
- Readdir
- Rename
- Link
- Unlink

Questions

- Hierarchical directory systems
- Path names
- Directory operations

Mounting

- A file system must be "mounted" before it can be available to processes
  - A directory structure may be built from multiple volumes (multiple file system instances)
  - Mounting procedure
    - Device (where the file system resides)
    - Mount point (the location within the file structure where the file system is to be attached)
    - (optionally) file system type
  - Mounting may be explicit (upon request from a user) or implicit (automatically and via discovery)

Protection

- Reliability
  - Keep files safe from physical damage
    - Via redundancy
- Access control

Types of Access

- OSes provide controlled access by limiting the types of file access that can be made
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

Access Control

- Common approaches are identify-based approaches
  - Users is assigned an identify, a data structure in the OS
- Many security models have been examined

Common Concepts

- Classification of users
  - Owner
  - Group
  - Universe
- Access control list
- Examples: Windows and Unix
  - Windows: access-control list
  - Unix: permissions

Questions

- A few concepts of file system mounting
- A few concepts about protection and access control

File I/O: API

- System calls
  - Unix APIs
- Windows APIs
- Java APIs

Java I/O Streams

- Sequential access of files

Java I/O Streams

- Sequential access of regular files
- A stream is a sequence of data associated with an input source or an output destination.
  - Input source or output destination
    - Files, network end point, standard I/O, memory array, programs
  - A program uses an *input stream* to read data from a source, one item at a time
  - A program uses an *output stream* to write data to a destination, one item at time
  - In java.io package, since JDK 1.0

Java Block-based I/O

- Java Channel interface provides block-based I/O
  - Read, write fixed blocks
  - Channel in java.nio package, since JDK 1.4

Asynchronous I/O in Java

- AsynchronousChannel interface
  - AsynchronousChannel in java.nio package, since JDK 1.7
  - Often referred to as a part of Java NIO.2

Examples

- A few example programs

Questions

- File system APIs
- Examples in Java and JVM

Assignment

- Project 3