File-System Interface

Gordon College
Stephen Brinton

File-System Interface

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

File Concept

- Contiguous logical address space

  OS maps a file onto a physical device.

- Types:
  - Data
    - numeric
    - character
    - binary
  - Program

- Basic Definition "File": named collection of related information that is recorded on secondary storage

File Structure

- Different Types of File Structures:
  - None - sequence of words, bytes
  - Simple record structure
    - Lines (text file)
    - Fixed length
    - Variable length
  - Complex Structures
    - Formatted document (word document)
    - Relocatable load file
- Can simulate last two with first method by inserting appropriate control characters
- Who decides the structure: the creator
  - 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

```
drwxr-xr-x   7 brinton  brinton      238 Nov 27 19:31 brinton_HW6_DPL
drwxr-xr-x  13 brinton  brinton      442 Oct 18  2007 brinton_hw2_files
drwx------   9 brinton  brinton      306 Oct 10  2007 brinton_hw3_DPL
-rw-r--r--@  1 brinton  brinton    27415 Oct 23  2007 cheatsheet.pdf
-rw-r--r--   1 brinton  brinton  1049019 Oct 17  2007 denotational_semantics.pdf
-rw-r--r--   1 brinton  brinton   302100 Oct 17  2007
    foundations_functional_programming.pdf
-rw-r--r--   1 brinton  brinton   591836 Oct 17  2007 lambda_calculus.pdf
-rw-r--r--@  1 brinton  brinton   635904 Oct 18  2007 lecture04.ppt
-rw-r--r--@  1 brinton  brinton   935406 Oct 23  2007 offline.pdf
-rw-r--r--   1 brinton  brinton  1109552 Oct 17  2007 prolog.pdf
-rw-r--r--@  1 brinton  brinton   119786 Oct 23  2007 schemeQuickRef.pdf
-rw-r--r--   1 brinton  brinton  3647033 Oct 17  2007 semantics_applications.pdf
drwxr-xr-x  10 brinton  brinton      340 Nov 27 16:18 timpcore
drwxr-xr-x   6 brinton  brinton      204 Nov 24 15:30 tuscheme
-rw-r--r--   1 brinton  brinton   294375 Oct 17  2007 type_systems.pdf
```

File Operations

File is an abstract data type and should have at least 6 basic operations:

- **Create**
- **Write**
- **Read**
- **Reposition within file**
- **Delete**
- **Truncate** (reset length to zero and release file space)

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

- Data needed to manage open files:
  - File pointer: pointer to last read/write location, unique to each process that has the file open {used when offset not included with read/write}
  - 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: data access information
  - Access rights: per-process access mode information

Open 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 (Win OS)
  - **Advisory** – processes can find status of locks and decide what to do (up to developers) (Unix)

C# example

```
using(FileStream fileStream = new FileStream(
        "Test#@@#.dat", FileMode.OpenOrCreate,
        FileAccess.ReadWrite, FileShare.ReadWrite))


try
                {
                   fileStream.Lock(textLength - 1, byteCount);
                   Console.WriteLine("The specified part " +
                      "of file has been locked.");
                }
                catch(IOException e)
                {
                   Console.WriteLine(
                      "{0}: The specified part of file is" +
                      " already locked.", e.GetType().Name);
                }
```

File Locking Example – Java API

```
import java.io.*;
import java.nio.channels.*;
public class LockingExample {
    public static final boolean EXCLUSIVE = false;
    public static final boolean SHARED = true;
    public static void main(String arsg[]) throws IOException {
        FileLock sharedLock = null;
        FileLock exclusiveLock = null;
        try {
            RandomAccessFile raf = new RandomAccessFile("file.txt", "rw");
            // get the channel for the file
            FileChannel ch = raf.getChannel();
            // this locks the first half of the file - exclusive
            exclusiveLock = ch.lock(0, raf.length()/2, EXCLUSIVE);
            /** Now modify the data . . . */
            // release the lock
            exclusiveLock.release();
```

File Locking Example – Java API (cont)

```
            // this locks the second half of the file - shared
            sharedLock = ch.lock(raf.length()/2+1, raf.length(),
                                 SHARED) ;
            /** Now read the data . . . */
            // release the lock
            exclusiveLock.release();
        } catch (java.io.IOException ioe) {
            System.err.println(ioe);
        }finally {
            if (exclusiveLock != null)
            exclusiveLock.release();
            if (sharedLock != null)
            sharedLock.release();
        }
    }
}
```

File Types

- Should the OS recognize and support file types?
- How?
  - Include type as part of name (file extension)
  - Magic number – UNIX – stored at beginning of file
  - Name of program in file - Mac OS X
- Examples of use:
  - Open a file and it automatically is associated with a program
  - TOPS-20 OS: execute a program, if modified then OS recompiles and then executes

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 Structure

File types often used to indicate internal structure of file

Executable – OS expects a certain structure: dynamically load it into memory

Disadvantage

OS can become massive

Force a file to conform to one of the expected types?

All OS must support the executable file type

Internal File Structure

Disk-system has well-defined block size determined by size of sector.

- Logical records can vary in length
- Physical records are based on block (sector size) on disk

Internal Fragmentation – wasted space at the end of the last physical record used.

Packing – placing a number of logical records into physical blocks

Access Methods

How do you choose the correct method to use?

Sequential Access (most common)

read next	write next
reset (more to initial)	advance (advance forward)

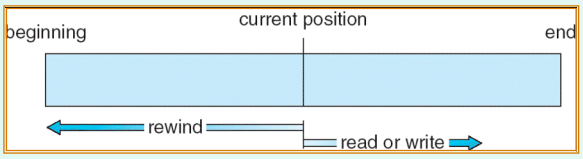
Direct Access

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

viewed as a numbered sequence of blocks or records
On top of direct access: index file or hash function

"read n" get L bytes starting at n*L (logical record length = L)
n = relative block number

Accessing Files

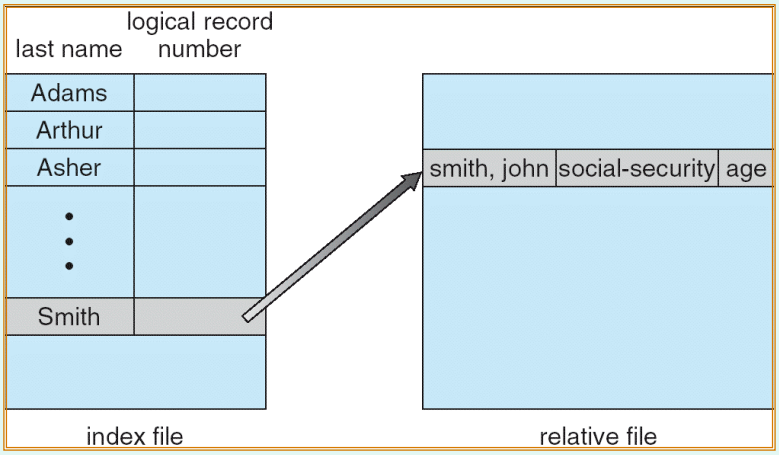


Sequential Access the file pointer is advanced after each operation

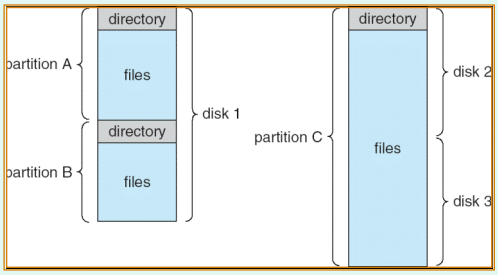
Doing sequential access on a direct-access file

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



Directory Structure

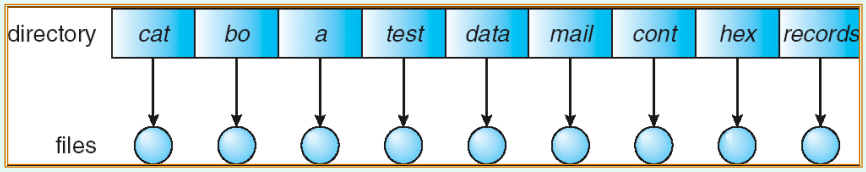


A Typical File-system Organization

Directory Operations:
Search for a file
Create a file
Delete a file
List a directory
Rename a file
Traverse the file system

Single-Level Directory

A single directory for all users: simple and "flat"

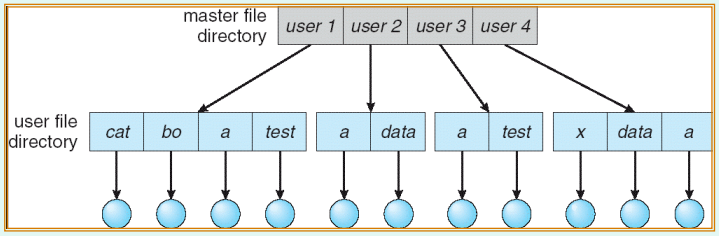


Limitations:

1. Naming problem
2. Grouping problem

Two-Level Directory

Separate directory for each user



- ⇨ Same file name - different users allowed (name collision problem solved)
- ⇨ Efficient searching
- ➨ No grouping capability (no user directories)

Path – user name & filename

Search Path – sequence of directories searched

Search Path

Windows

Sets the command path in the PATH environment variable, which is the set of directories used to search for executable files. Used without parameters, **path** displays the current command path.

Syntax

path [[**%path%**] [*Drive***:**]*Path* [**;**...]]

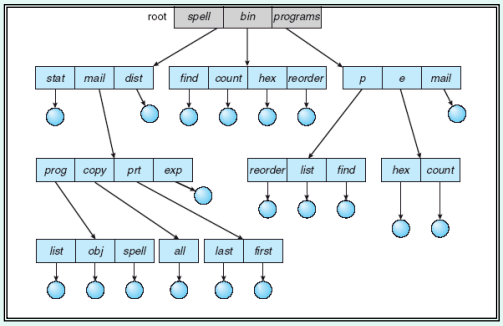
Linux

$ **echo $PATH**

The directories to search for executables in absence of an absolute or relative pathname containing a / character

Tree-Structured Directories

- Efficient searching
- Grouping Capability
- Current directory (working directory)
  - cd /spell/mail/prog
  - ./othello



Tree-Structured Directories (Cont)

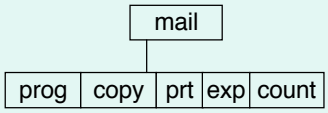
- **Absolute** or **relative** path name
- Creating a new file is done in current directory
- Delete a file

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

  mkdir <dir-name>

  Example: if in current directory /mail

  mkdir count

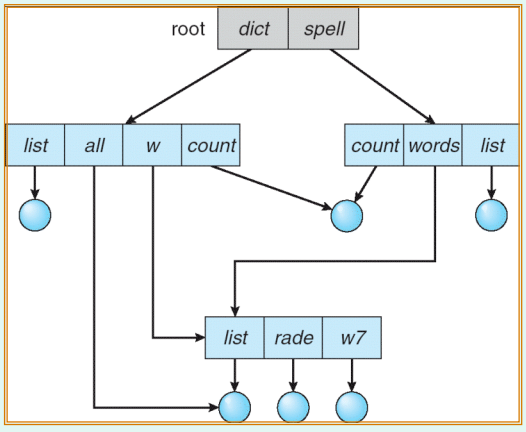


Deleting "mail" ⇒ deleting the entire subtree rooted by "mail"

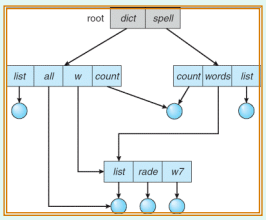
Acyclic-Graph Directories

Allows directories to share subdirectories and files

Must contain no cycles

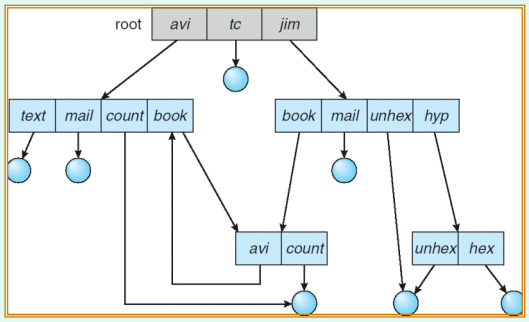


Acyclic-Graph Directories (Cont.)



- Two different names (aliasing)
- If */dict/all/* is deleted ⇒ dangling pointer "list"
  Solutions:
  - Backpointers, so we can delete all pointers
  - Entry-hold-count solution – keep count in file info
  - Let user keep up with the problem – delete dangling pointers
- New directory entry type (linux - hard and symbolic links)
  - **Link** – another name (pointer) to an existing file
  - **Resolve the link** – follow pointer to locate the file

General Graph Directory

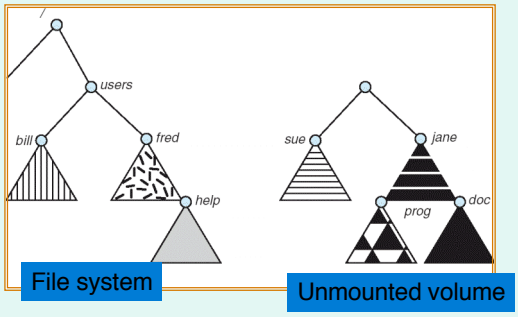


How do we guarantee no cycles?

- Allow only links to file not subdirectories
- Garbage collection – walk the graph and pickup garbage
- Every time a new link is added use a cycle detection algorithm to determine whether it is OK

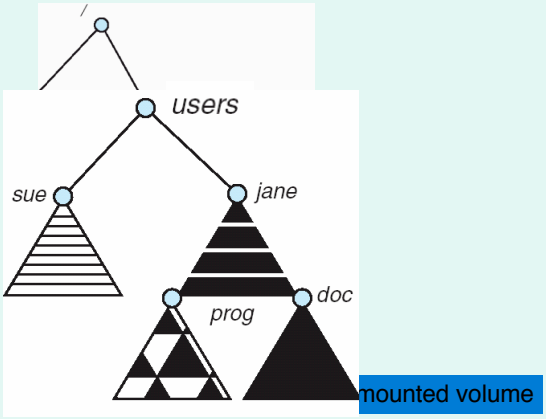
File System Mounting

- A file system must be **mounted** before it can be accessed
- A unmounted file system (bottom right) is mounted at a **mount point**



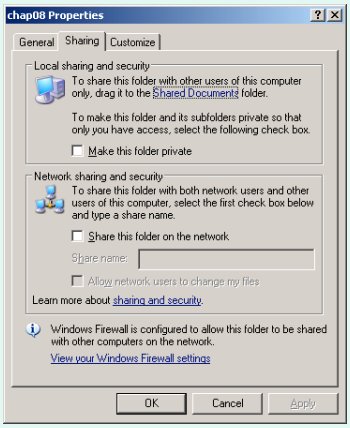
File System Mounting

- A file system must be **mounted** before it can be accessed
- A unmounted file system (bottom right) is mounted at a **mount point**



File Sharing

- Sharing of files on multi-user systems is desirable
- Sharing may be done through a **protection** scheme
  - owner, group, universe
- On distributed systems, files may be shared across a network
- Network File System (NFS) is a common distributed file-sharing method



Windows file sharing

File Sharing – Multiple Users

- **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 (spoof) 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

- 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 include all information in each request, allowing easy recovery but less security

File Sharing – Consistency Semantics

- **Consistency semantics** specify how multiple users are to access a shared file simultaneously
  - Similar to process synchronization algorithms
    - Tend to be less complex due to disk I/O and network latency (for remote file systems)
    - File session – open, file operations, close
  - Unix file system (UFS) implements:
    - Writes to an open file visible immediately to other users of the same open file
    - Sharing file pointer to allow multiple users to read and write concurrently (one type of mode)

Protection

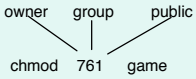
- File owner/creator should be able to control:
  - what can be done
  - by whom
- Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

Access Lists and Groups

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

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

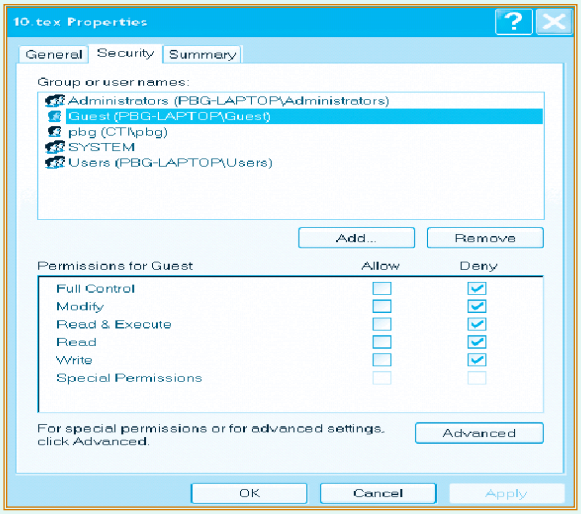
- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say *game*) or subdirectory, define an appropriate access.



Attach a group to a file:

chgrp G game chown :G game

Windows XP Access-control List Management



A Sample UNIX Directory Listing

```
drwx------+ 11 brinton  brinton    374 Apr 18 09:07 Desktop
drwx------+ 17 brinton  brinton    578 Apr 10 20:13 Documents
drwx------+ 19 brinton  brinton    646 Mar 18 17:21 Downloads
drwx------+ 38 brinton  brinton   1292 Feb 26 09:08 Library
drwx------+  5 brinton  brinton    170 Dec 25 12:22 Movies
drwx------+ 14 brinton  brinton    476 Dec  7 14:51 Music
drwx------+  6 brinton  brinton    204 Feb  1 15:14 Pictures
drwxr-xr-x+  5 brinton  brinton    170 Aug 10  2007 Public
drwxr-xr-x+  9 brinton  brinton    306 Mar  6 14:33 Sites
-rw-r--r--@  1 brinton  brinton  33280 Feb 22 15:04 Things Needed.doc
drwx------   9 brinton  brinton    306 Oct 10  2007 brinton_hw3_DPL
-rwx------@  1 brinton  brinton  21504 Jan 25 15:55 coversheet.doc
drwxr-xr-x  12 brinton  brinton    408 Apr 18 12:08 eclipse
-rwx------   1 brinton  brinton  21504 Jan 25 17:11 lab3-writeup.doc
-rw-r--r--@  1 brinton  brinton  72805 Apr 25  2005 project5.pdf
drwxr-xr-x  14 brinton  brinton    476 Mar  1 10:09 shared-windows
drwxr-xr-x  22 brinton  brinton    748 Nov 23 14:39 sml
```