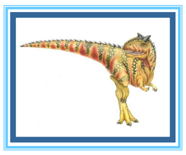
Chapter 10: File System



Chapter 10: File System

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

Objectives

- To explain the functions 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

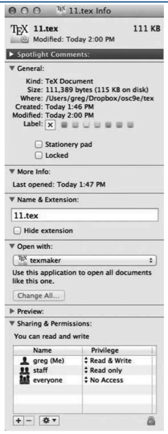
File Concept

- Contiguous logical address space
- Types:
  - Data
    - numeric
    - character
    - binary
  - Program
- Contents defined by the file's creator
  - Many types, consider text file, source file, executable file

File Attributes

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies files within a file system
- **Type** – needed by systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing, and etc.
- **Time, date, and user identification** – data for protection, security, and usage monitoring
- Information about files are kept in a directory structure, which is maintained on the disk. Part of it can be cached in main memory
- Many variations, including extended file attributes such as file checksum
- Information kept in the directory structure

File info Window on Mac OS X



11.tex Info

11.tex 111 KB
Modified: Today 2:00 PM

Spotlight Comments:

General:

Kind: TeX Document
Size: 111,389 bytes (115 KB on disk)
Where: /Users/greg/Dropbox/osc9e/tex
Created: Today 1:46 PM
Modified: Today 2:00 PM
Label: ✕

☐ Stationery pad
☐ Locked

More Info:

Last opened: Today 1:47 PM

Name & Extension:

11.tex

☐ Hide extension

Open with:

texmaker

Use this application to open all documents like this one.

Change All...

Preview:

Sharing & Permissions:

You can read and write

Name	Privilege
greg (Me)	Read & Write
staff	Read only
everyone	No Access

File Operations

- File is an **ADT** or **abstract data type**
- **Create** – create a file
- **Write –** at write pointer location
- **Read –** at read pointer location
- **Reposition within file** - seek
- **Delete**
- **Truncate**
- **Open(F_i)** – search the directory structure on disk for entry F_i, and move the content of entry to memory, preparing file for subsequent access
- **Close (F_i)** – move the content of entry F_i in memory to directory structure on disk

- Such operations involve the changes of various OS data structures

Open Files

- Several data structures are needed to manage open files:
  - **Open-file table**: tracks open files, system-wide open-file table, and per-process open-file table
  - File pointer: pointer to last read/write location, per process that has the file open
  - **File-open count**: counter of number of times (processes) that the file is open – to allow removal of data from the 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 Locking

- Provided by some operating systems and file systems
  - Similar to reader-writer locks
  - **Shared lock** similar to reader lock – several processes can acquire it concurrently
  - **Exclusive lock** similar to writer lock

- Mediates access to a file

- Mandatory or advisory:
  - **Mandatory** – access is denied depending on locks held and requested. Window OS uses mandatory lock
  - **Advisory** – processes can find status of locks and decide what to do – programmers decide. Unix systems use advisory lock

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

- 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 the first method by inserting appropriate control characters
- Who decides:
  - Operating system
  - Program

Access Methods

- **Sequential Access**

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

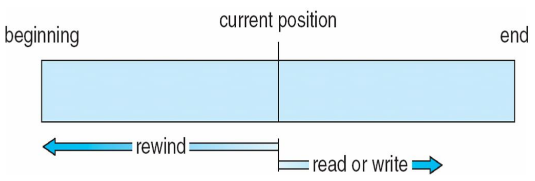
- **Direct Access –** file is fixed length logical records

```
read n
write n
position to n
        read next
        write next
rewrite n
```

n = relative block number

- Relative block numbers allow OS to decide where file should be placed
  - See disk block allocation problem in Chapter 11

Sequential-access File



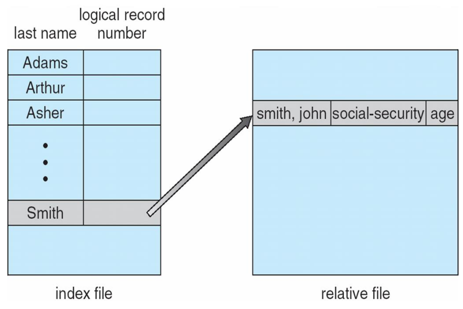
Simulation of Sequential Access on Direct-access File

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

Other Access Methods

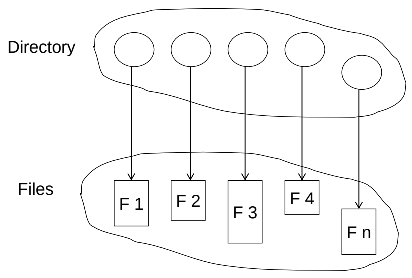
- Other access methods can be built on top of a direct-access method
- General involve creation of an **index** for the file
- Keep index in memory for fast determination of location of data to be operated on (consider UPC code plus record of data about that item)
- If too large, index (in memory) of the index (on disk)
- IBM indexed sequential-access method (ISAM)
  - Small master index, points to disk blocks of secondary index
  - File kept sorted on a defined key
  - All done by the OS
- VMS operating system provides index and relative files as another example (see next slide)

Example of Index and Relative Files



Directory Structure

- A collection of nodes containing information about all files

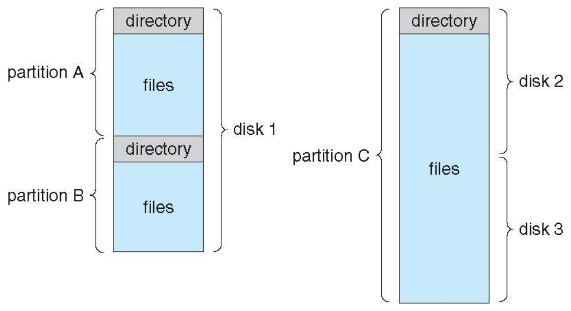


Both the directory structure and the files reside on disk

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
- An entity containing a file system known as a volume
- Each volume containing the file system also tracks that file system info in device directory or volume table of contents
- Other than general-purpose file systems, there are many special-purpose file systems, frequently all within the same operating system or computing systems

A Typical File-system Organization



Types of File Systems

- We mostly deal with general-purpose file systems
- But systems frequently have many file systems, some general- and some special- purpose
- Consider Solaris has
  - 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

Operations Performed on Directory

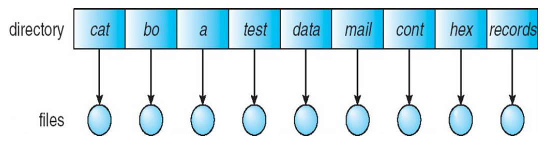
- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

Organize the Directory (Logically) to Obtain

- Efficiency – locating a file quickly
- 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, …)

Single-Level Directory

- A single directory for all users

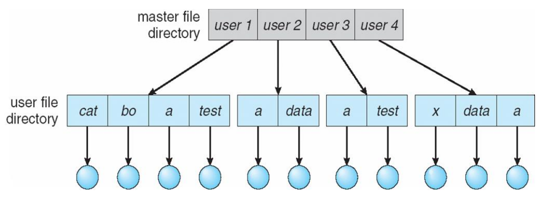


Naming problem

Grouping problem

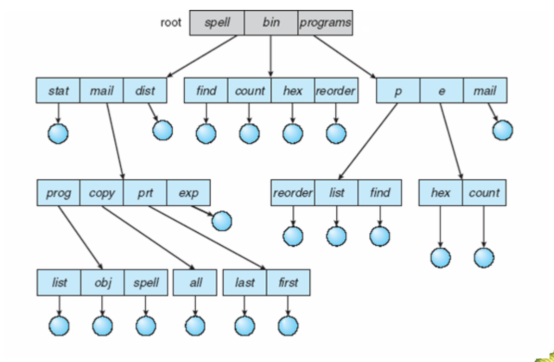
Two-Level Directory

- Separate directory for each user



- Path name - /user1/cat
- Can have the same file name under different users
- More efficient searching than single-level directory
- No grouping capability

Tree-Structured Directories



Tree-Structured Directories (Cont.)

- Efficient searching
- Grouping Capability
- Current directory (working directory)
  - `cd /spell/mail/prog`
  - `type list`

Tree-Structured Directories (Cont)

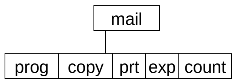
- **Absolute** or **relative** path name
- Creating a new file is done in the current directory
- Delete a file in the current directory

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

  ```
  mkdir <dir-name>
  ```

  Example: if in current directory `/mail`

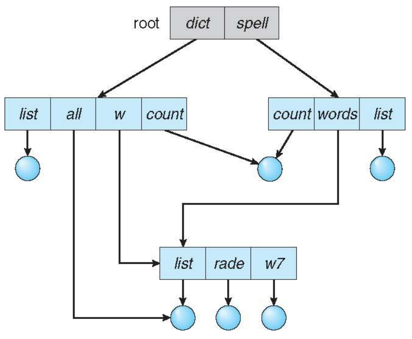
  ```
  mkdir count
  ```



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

Acyclic-Graph Directories

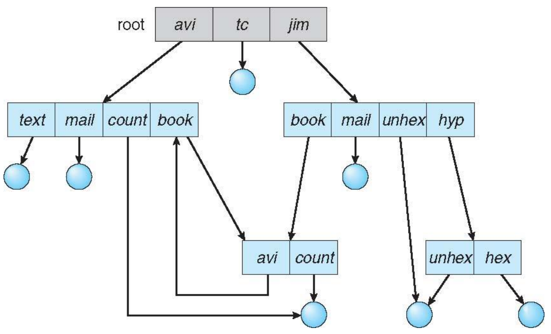
- Have shared subdirectories and files – more flexible and complex



Acyclic-Graph Directories (Cont.)

- New directory entry type
  - Link – another name (pointer) to an existing file
  - Resolve the link – follow pointer to locate the file
- Two different names (aliasing)
  - Ensure not traversing shared structures more than once
- Deletion might lead to that dangling pointers that point to empty files or wrong files
- Yet there is also difficulty ensuring there is no cycles in a graph – complexity associated with it

General Graph Directory

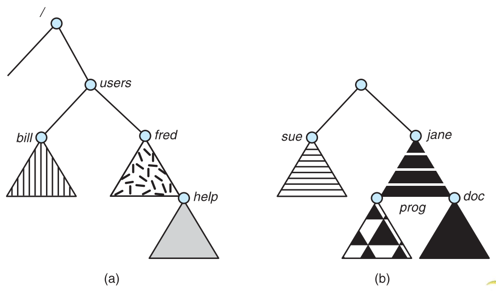


General Graph Directory (Cont.)

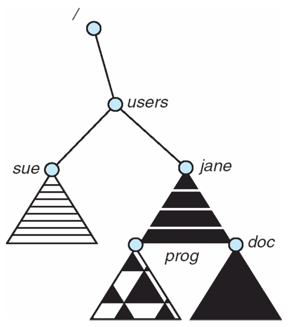
- How do we guarantee no cycles?
  - Allow only links to file not subdirectories
  - Every time a new link is added use a cycle detection algorithm to determine whether there is a cycle or not

File System Mounting

- A file system must be **mounted** before it can be accessed – just like a file must be opened before it is used
- A unmounted file system (i.e., Fig. 10-11(b)) is mounted at a **mount point**



Mount Point



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
- If multi-user system
  - **User IDs** identify users, allowing permissions and protections to be per-user
    Group IDs allow users to be in groups, permitting group access rights
  - Owner of a file / directory
  - Group of a file / directory

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

Protection

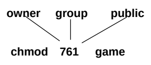
- File owner/creator of the file should be able to control:
  - what can be done
  - by whom
- Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users on Unix / Linux

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

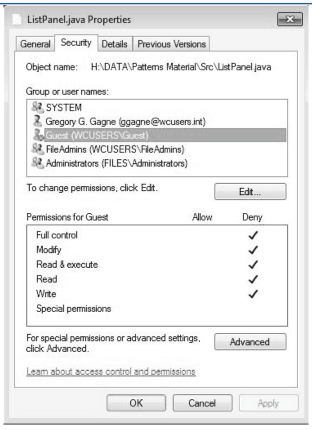
- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say *game*) or subdirectory, define an appropriate access.



Attach a group to a file

```
chgrp     G     game
```

Windows 7 Access-Control List Management



ListPanel.java Properties

General | Security | Details | Previous Versions

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/
```