

CSMC 412

Operating Systems Prof. Ashok K Agrawala

© 2004 Ashok Agrawala
Set 11

File-System Interface

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

File Concept

- Contiguous logical address space
- Types:
 - Data
 - numeric
 - character
 - binary
 - Program

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

File Attributes

- **Name** – only information kept in human-readable form
- **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

File Operations

- Create
- Write
- Read
- file seek – reposition within file
- Delete
- Truncate
- $\text{Open}(F_i)$ – search the directory structure on disk for entry F_i and move the content of entry to memory
- $\text{Close}(F_i)$ – move the content of entry F_i in memory to directory structure on disk

Open Files

- Several pieces of data are needed to manage open files:
 - 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: cache of 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
 - **Advisory** – processes can find status of locks and decide what to do

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 – Name, Extension

file type	usual extension	function
executable	exe, com, bin or none	read 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, rrf, doc	various word-processor formats
library	lib, a, so, dll, mpeg, mov, rm	libraries of routines for programmers
print or view	arc, zip, tar	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	binary file containing audio or A/V information

Access Methods

■ Sequential Access

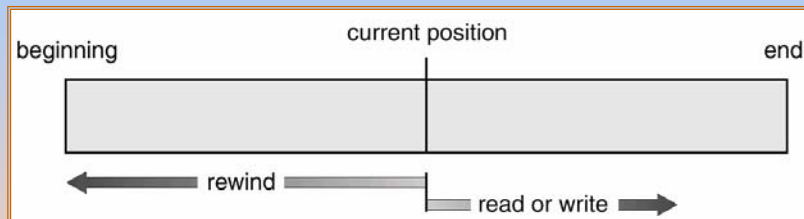
read next
write next
reset
no read after last write
(rewrite)

■ Direct Access

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

n = relative block number

Sequential-access File



Operating System Concepts

11.13

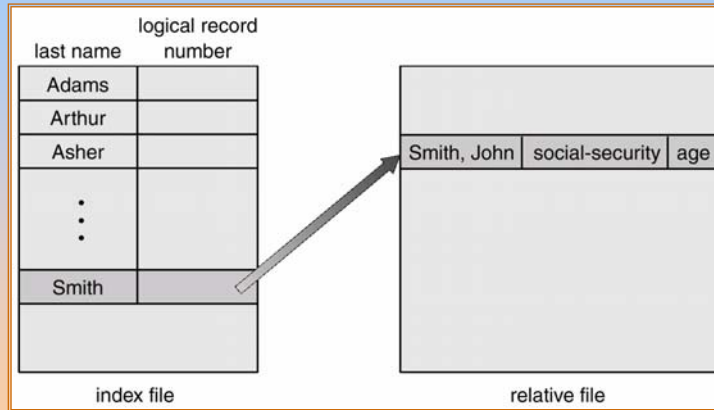
Simulation of Sequential Access on a Direct-access File

sequential access	implementation for direct access
<i>reset</i>	<i>cp = 0;</i>
<i>read next</i>	<i>read cp;</i> <i>cp = cp+1;</i>
<i>write next</i>	<i>write cp;</i> <i>cp = cp+1;</i>

Operating System Concepts

11.14

Example of Index and Relative Files

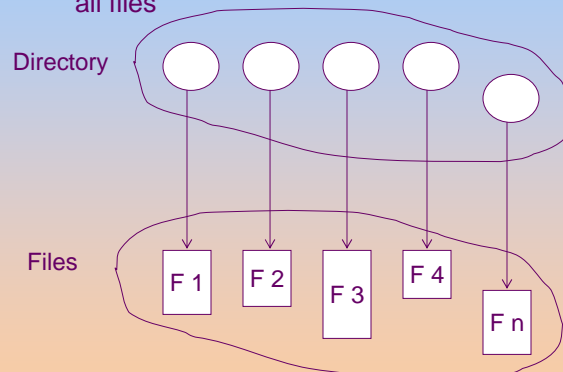


Operating System Concepts

11.15

Directory Structure

- A collection of nodes containing information about all files

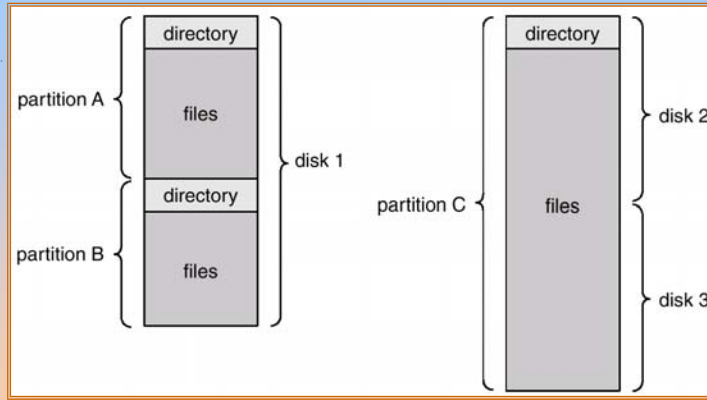


Both the directory structure and the files reside on disk
Backups of these two structures are kept on tapes

Operating System Concepts

11.16

A Typical File-system Organization



Information in a Device Directory

- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed (for archival)
- Date last updated (for dump)
- Owner ID
- Protection information (discuss later)

Operations Performed on Directory

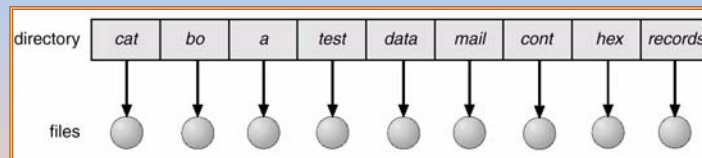
- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

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

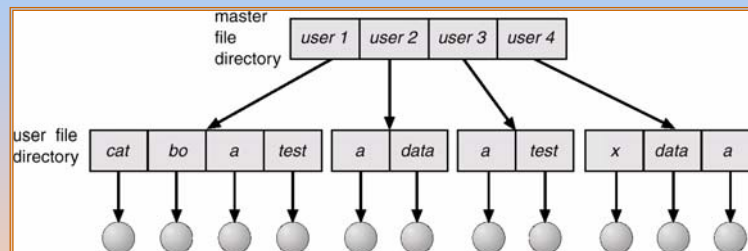


Naming problem

Grouping problem

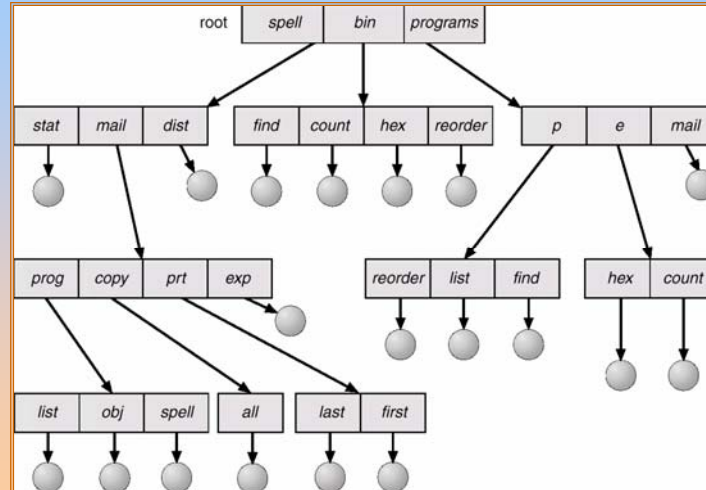
Two-Level Directory

- Separate directory for each user



- Path name
- Can have the same file name for different user
- Efficient searching
- No grouping capability

Tree-Structured Directories



Operating System Concepts

11.23

Tree-Structured Directories (Cont)

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

Operating System Concepts

11.24

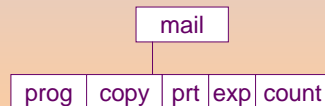
Tree-Structured Directories (Cont)

- **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

`rm <file-name>`

`mkdir <dir-name>`

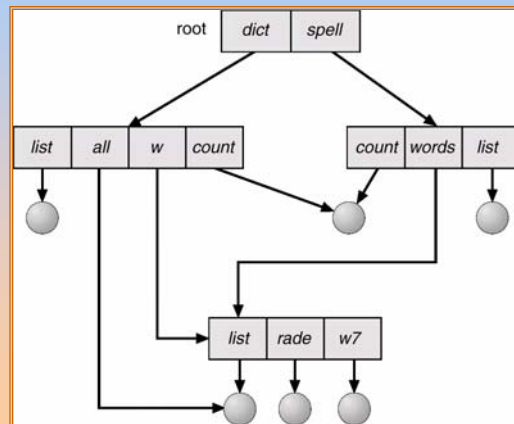
Example: if in current directory `/mail`
`mkdir count`



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

Acyclic-Graph Directories

- Have shared subdirectories and files



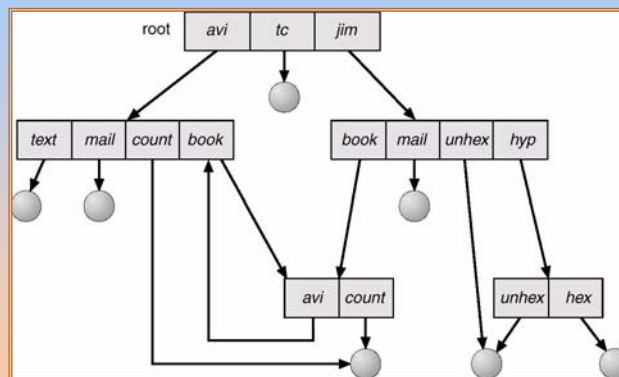
Acyclic-Graph Directories (Cont.)

- Two different names (aliasing)
- If *dict* deletes *list* ⇒ dangling pointer

Solutions:

- ↪ Backpointers, so we can delete all pointers
- ↪ Variable size records a problem
- ↪ Backpointers using a daisy chain organization
- ↪ Entry-hold-count solution

General Graph Directory



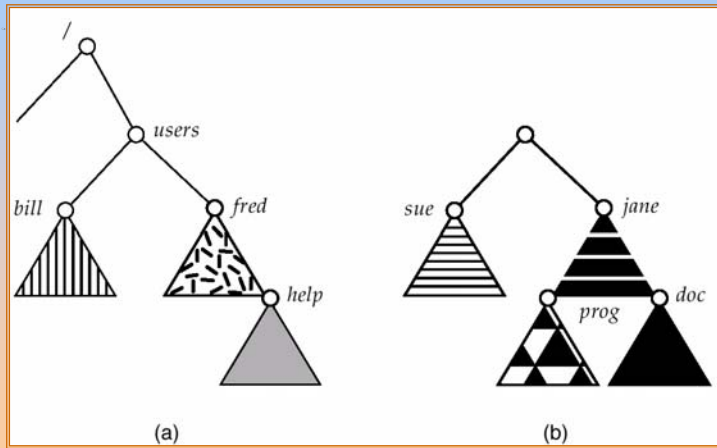
General Graph Directory (Cont.)

- How do we guarantee no cycles?
 - Allow only links to file not subdirectories
 - Garbage collection
 - 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 (i.e. Fig. 11-11(b)) is mounted at a **mount point**

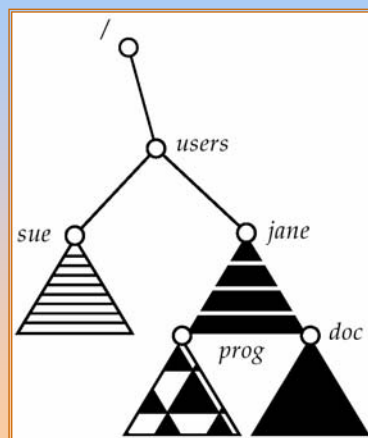
(a) Existing. (b) Unmounted Partition



Operating System Concepts

11.31

Mount Point



Operating System Concepts

11.32

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

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 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 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

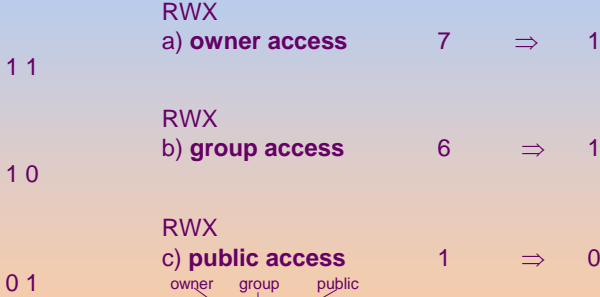
- **Consistency semantics** specify how multiple users are to access a shared file simultaneously
 - Similar to Ch 7 process synchronization algorithms
 - ▢ Tend to be less complex due to disk I/O and network latency (for remote file systems)
 - Andrew File System (AFS) implemented complex remote file sharing semantics
 - Unix file system (UFS) 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
 - AFS has session semantics
 - ▢ Writes only visible to sessions starting after the file is closed

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



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