Operating Systems 2023 Spring Term
Week 14-1

Dr. Emrah İnan (emrahinan@iyte.edu.tr)

File-System Interface

June 8, 2023

Week 13: Sample Glossary

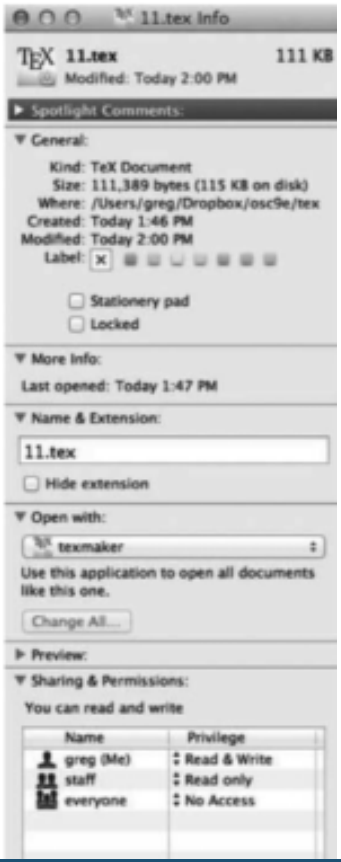
- **mirroring:** In storage, a type of RAID protection in which two physical devices contain the same content. If one device fails, the content can be read from the other. (on Page 1258)
- **mounting:** Making a file system available for use by logically attaching it to the root file system. (on Page 1258)
- **partition:** Logical segregation of storage space into multiple area; e.g., on HDDs, creating several groups of contiguous cylinders from the devices' full set of cylinders. (on Page 1261)

File-System Interface

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

File Concept

- Contiguous logical address space
- Types:
  Data -> numeric, character, binary
  Program
- Contents defined by 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 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
- Many variations, including extended file attributes such as file checksum
- Information kept in the directory structure

File Operations

- File is an abstract data type
- Create
- 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
- 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:

- Open-file table: tracks 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
  Similar to reader-writer locks
  Shared lock similar to reader lock – several processes can acquire 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
  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 args[]) 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();

      // 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
      sharedLock.release();
    } catch (java.io.IOException ioe) {
      System.err.println(ioe);
    }
    finally {
      if (exclusiveLock != null)
            exclusiveLock.release();
      if (sharedLock != null)
            sharedLock.release();
```

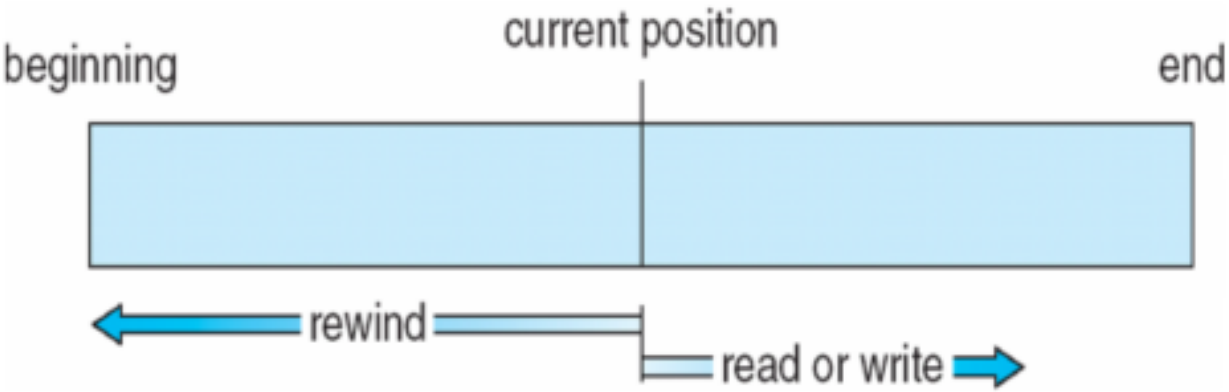
File Types – Name, Extension

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, perl, asm	source code in various languages
batch	bat, sh	commands to the command interpreter
markup	xml, html, tex	textual data, documents
word processor	xml, rtf, docx	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	rar, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, mp3, mp4, 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 first method by inserting appropriate control characters
- Who decides:
  Operating system
  Program

Sequential-access File



Access Methods

Temporarily interrupting a process with the intention of resuming the process at a later time

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 allocation problem in Ch 12

Simulation of Sequential Access on Direct-access File

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

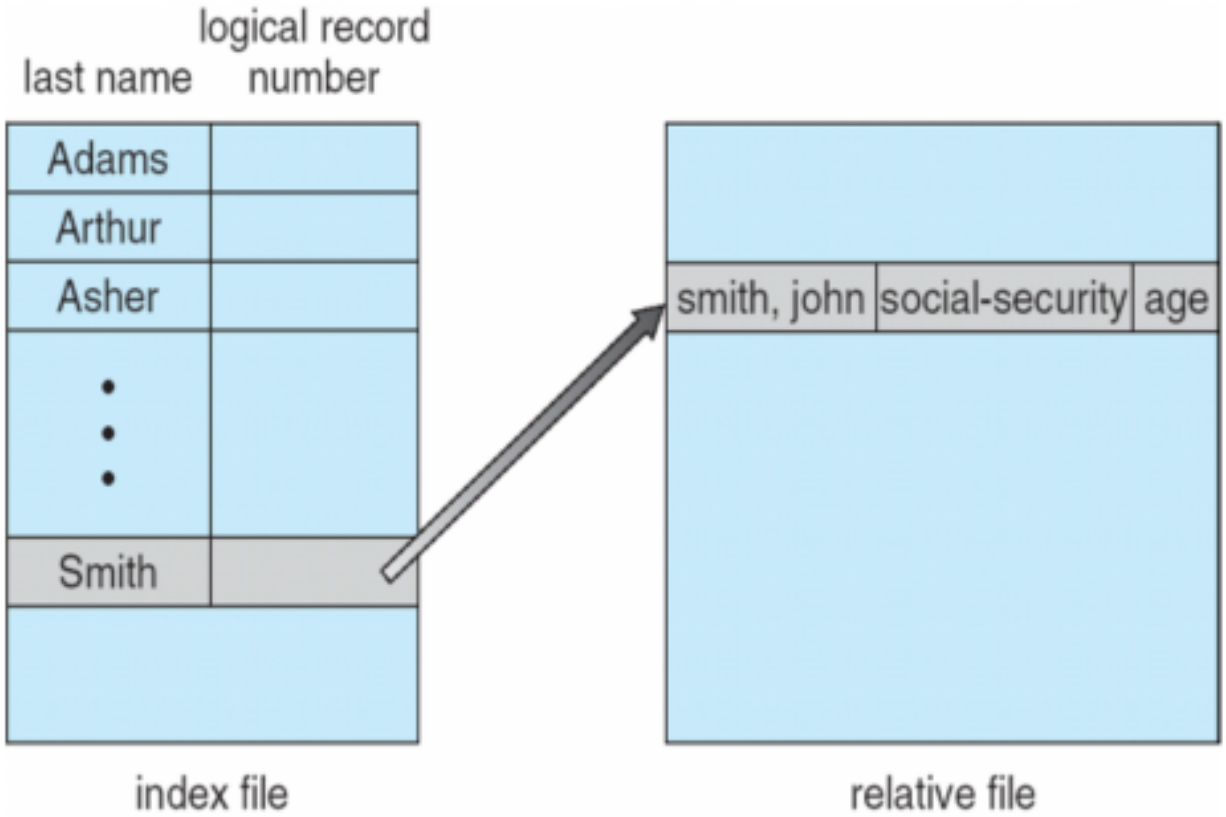
Relative block numbers allow OS to decide where file should be placed

See allocation problem in Ch 12

Other Access Methods

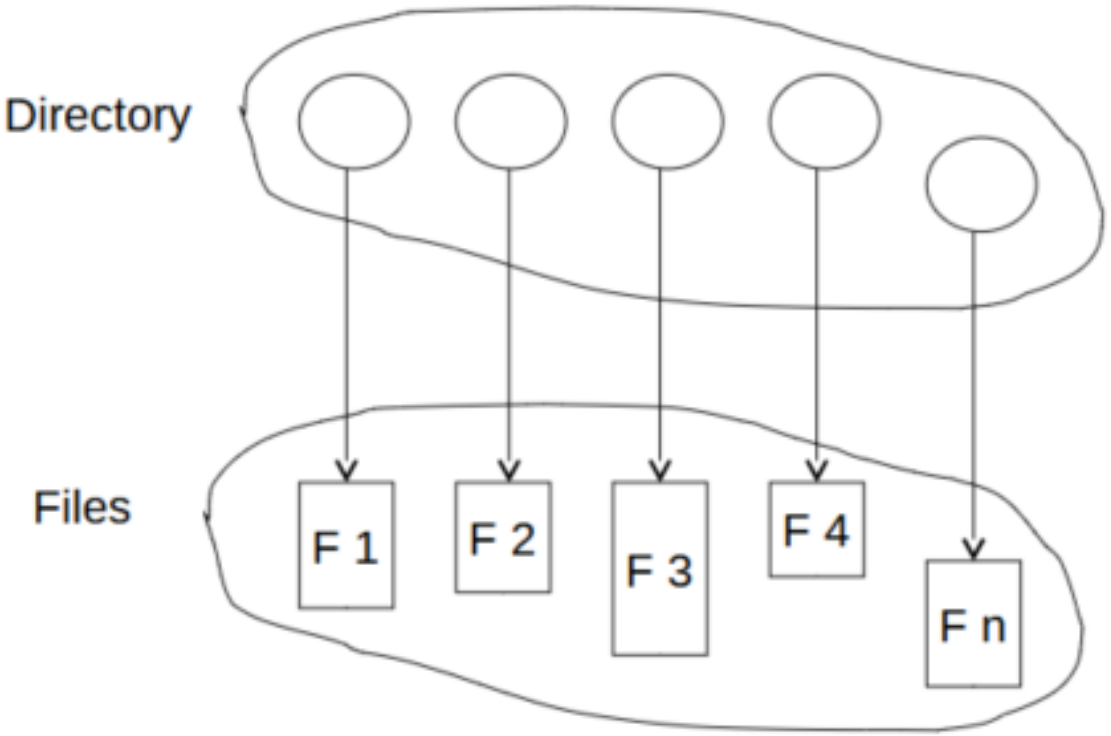
- Can be built on top of base methods
- General involve creation of an index for the file
- Keep index in memory for fast determination of location of data to be operated on (consider UPC code plus record of data about that item)
- If too large, index (in memory) of the index (on disk)
- IBM indexed 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



Example of Index and Relative Files

A collection of nodes containing information about all files

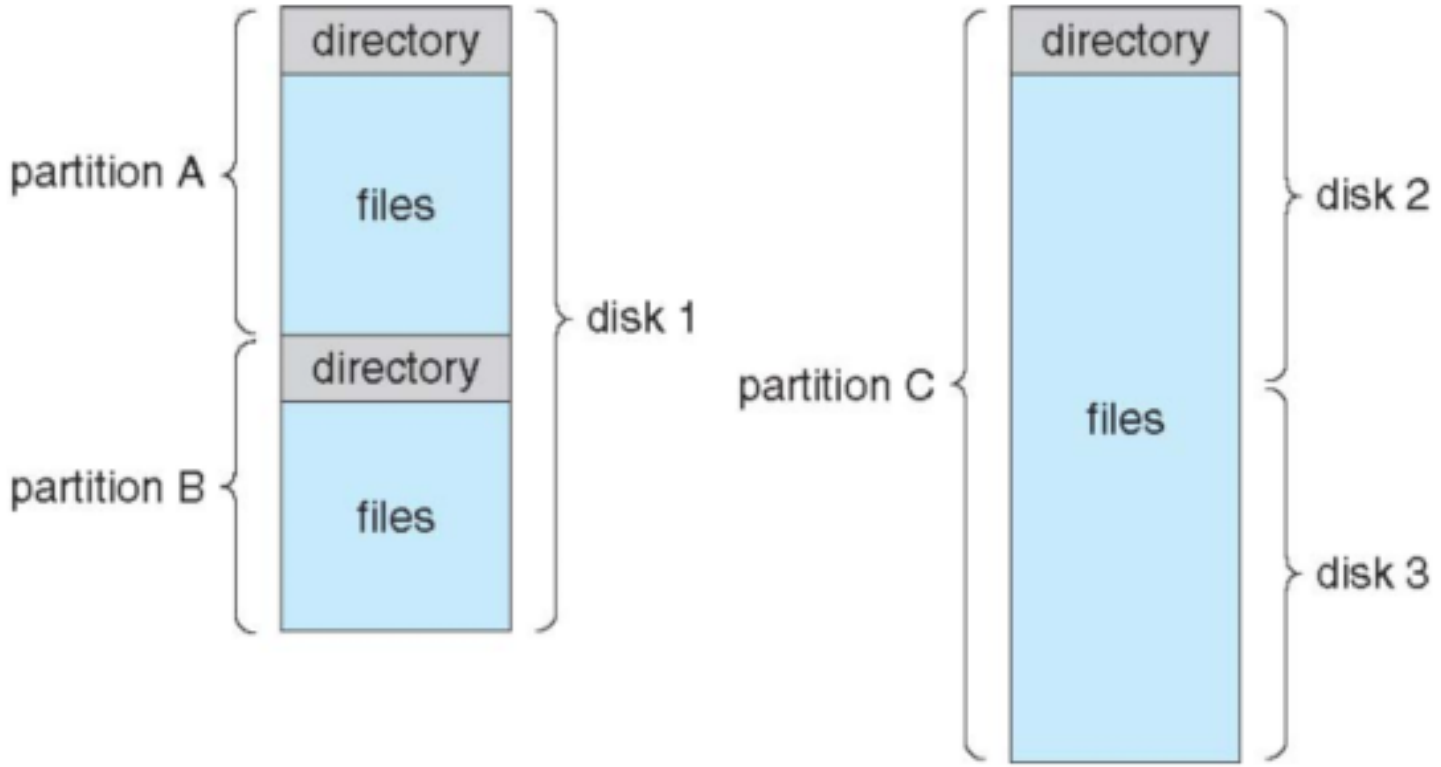


Both the directory structure and the files reside on disk

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
- Entity containing file system known as a volume
- Each volume containing file system also tracks that file system's info in device directory or volume table of contents
- As well as general-purpose file systems there are many special-purpose file systems, frequently all within the same operating system or computer

A Typical File-system Organization



Types of File Systems

- We mostly talk of general-purpose file systems
- But systems frequently have may 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 (tldp procfs)
  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

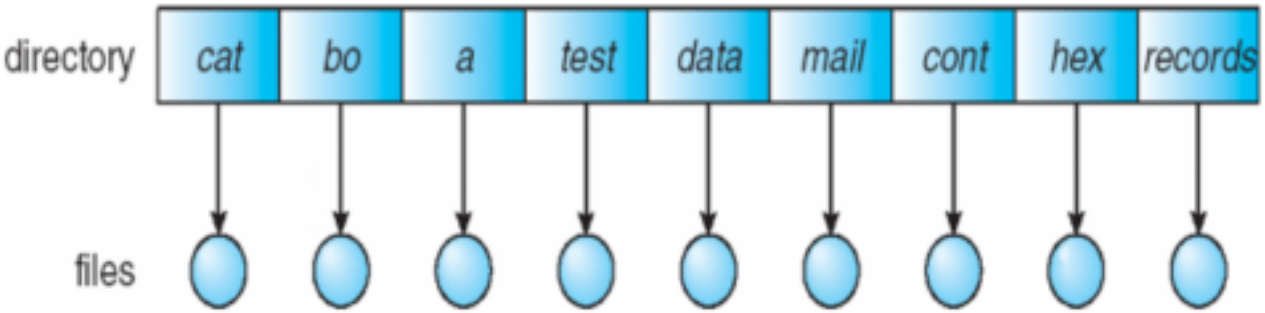
Directory Organisation

The directory is organised 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

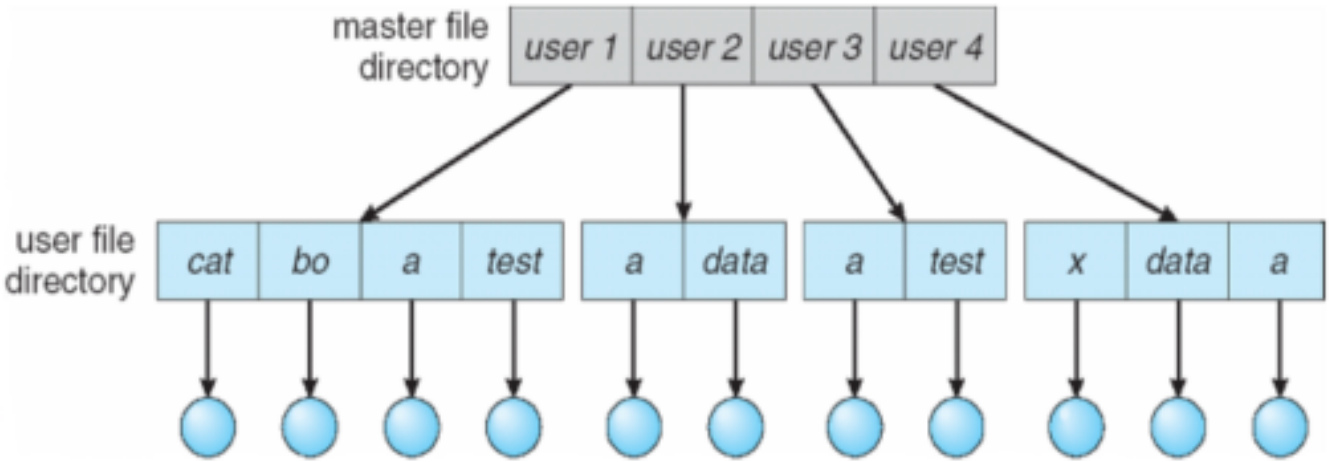


Naming problem

Grouping problem

Two-Level Directory

Separate directory for each user



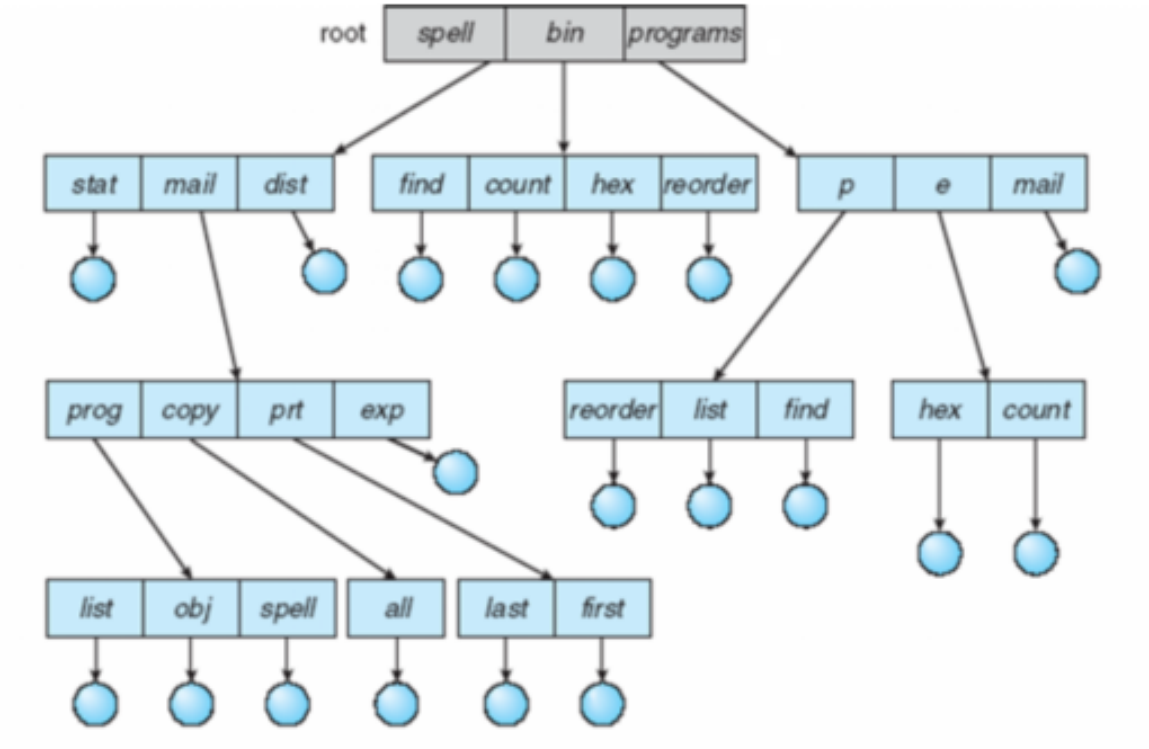
Path name

Can have the same file name for different user

Efficient searching

No grouping capability

Tree-Structured Directories I



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

Tree-Structured Directories II

Absolute or **relative** path name

Creating a new file is done in current directory

Delete a file

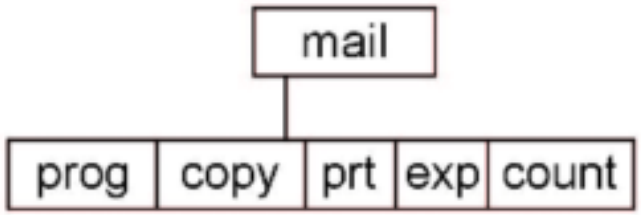
```
rm <file-name>
```

Creating a new subdirectory is done in current directory

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

Example: if in current directory `/mail`

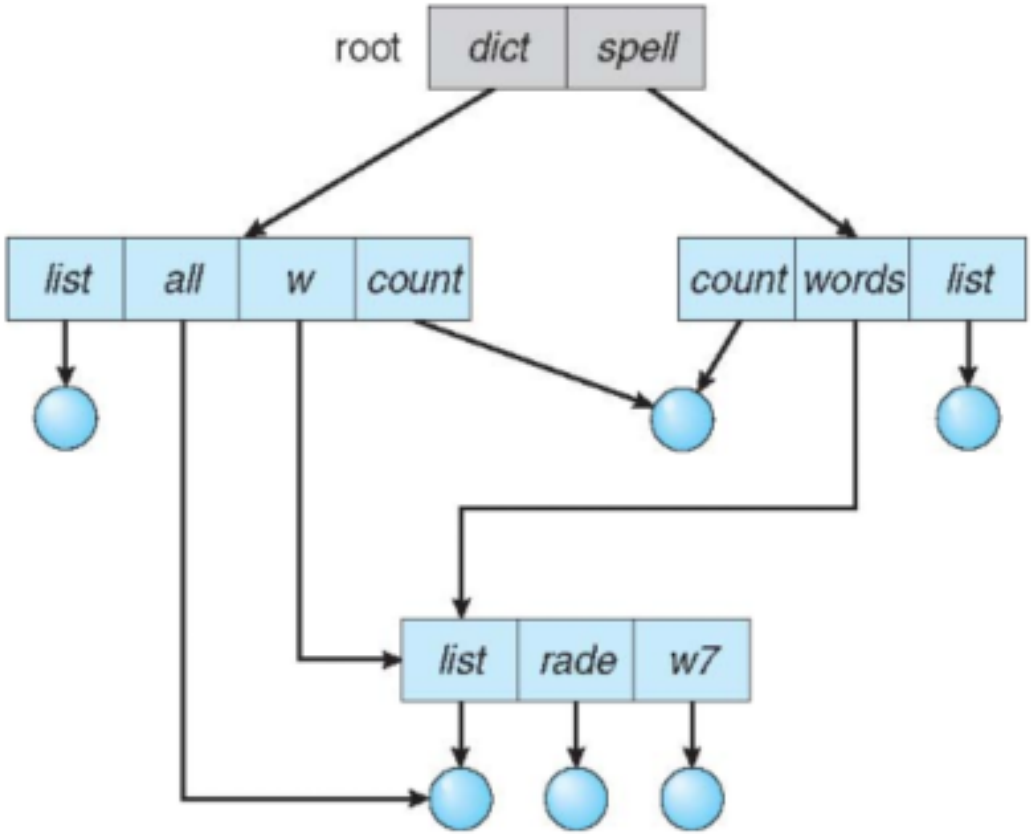
```
mkdir count
```



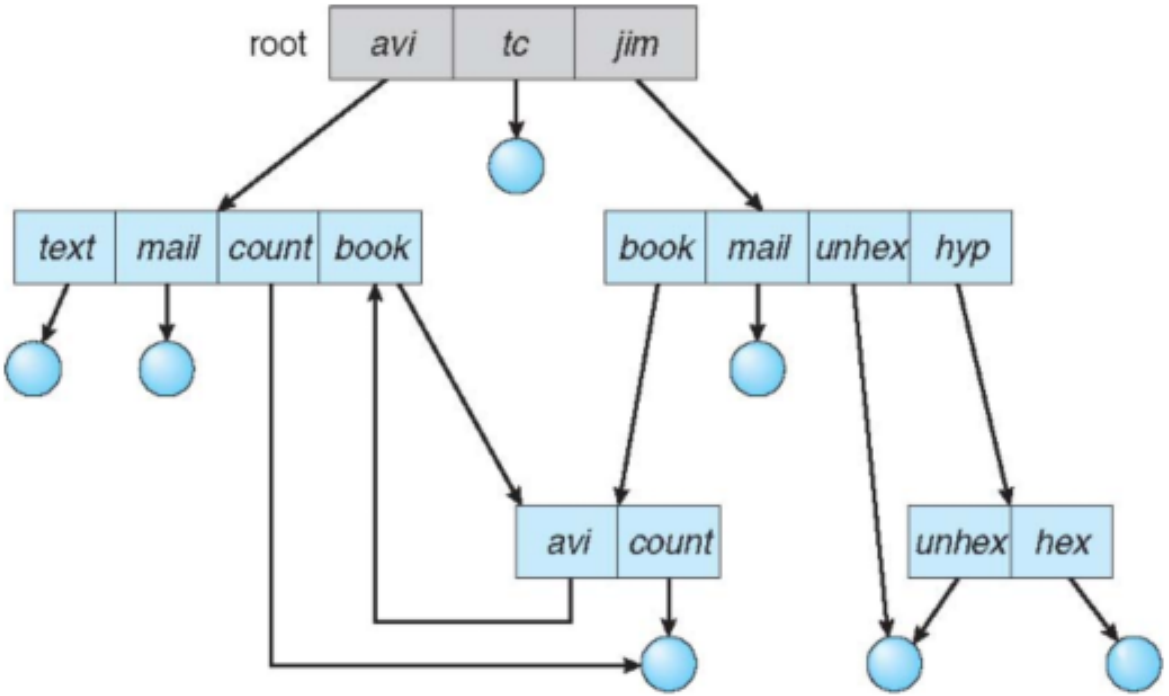
Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”

Acyclic-Graph Directories

Have shared subdirectories and files



General Graph Directory



How do we guarantee no cycles?

Allow only links to file not subdirectories

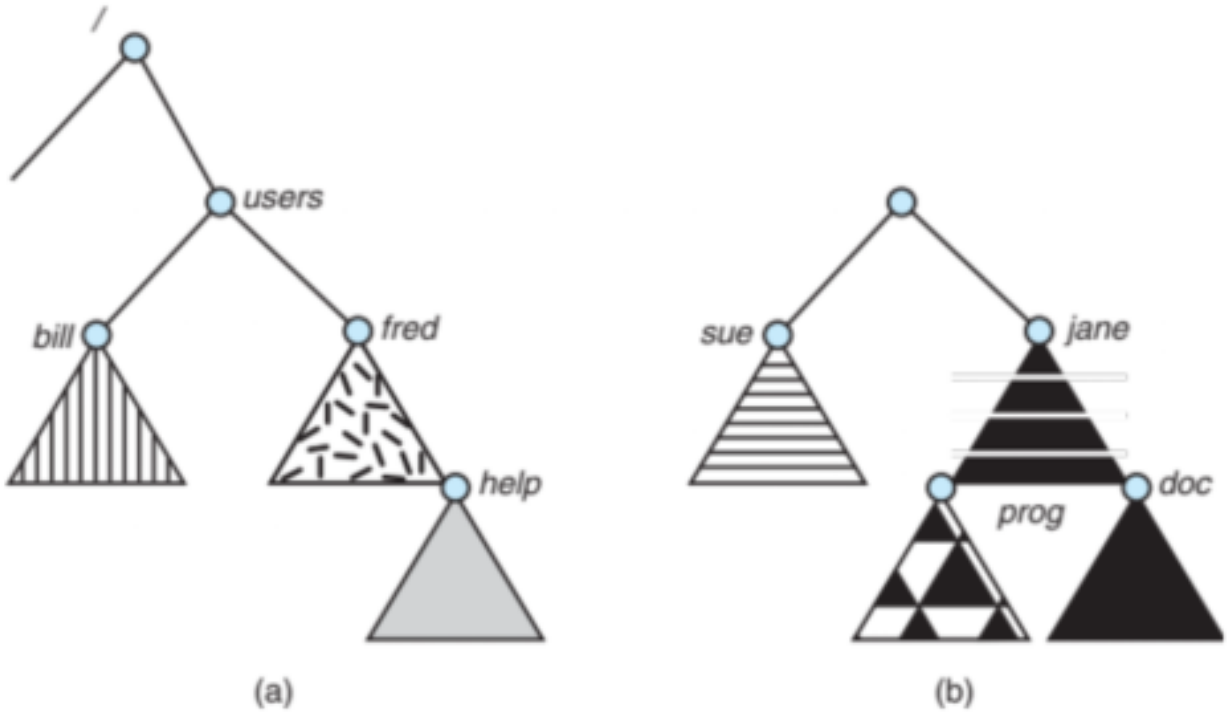
Garbage collection

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**



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

Protection-Access Lists and Groups

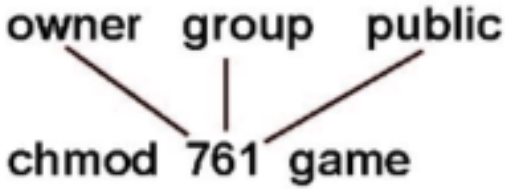
Mode of access: read, write, execute

Three classes of users on Unix / Linux

			RWX
a) **owner access**	7	⇒	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
```

ls -l, chmod, etc.

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 jwg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2017	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2017	program
drwx--x--x	4 tag	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/

The first field describes the protection of the file or directory. A d as the first character indicates a subdirectory. Also shown are the number of links to the file, the owner's name, the group's name, the size of the file in bytes, the date of last modification, and finally the file's name (with optional extension).