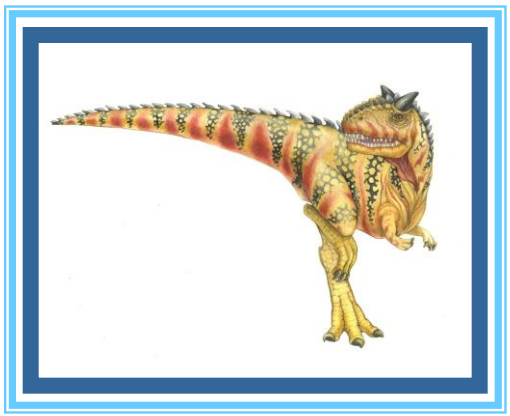
Chapter 13: File-System Interface

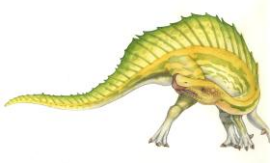


Silberschatz, Galvin and Gagne ©2018



Outline

- File Concept
- Access Methods
- Disk and Directory Structure
- Protection
- Memory-Mapped Files





Objectives

- To explain the function 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

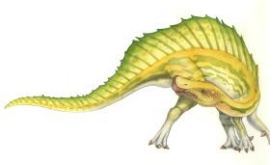


Troy University Resources



File Concept

- Contiguous logical address space
- Types:
  - Data
    - Numeric
    - Character
    - Binary
  - Program
- Contents defined by file's creator
  - Many types
    - **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

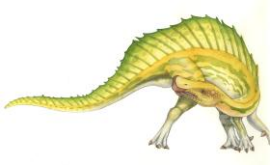


Troy University Resources



File info Window on Mac OS X

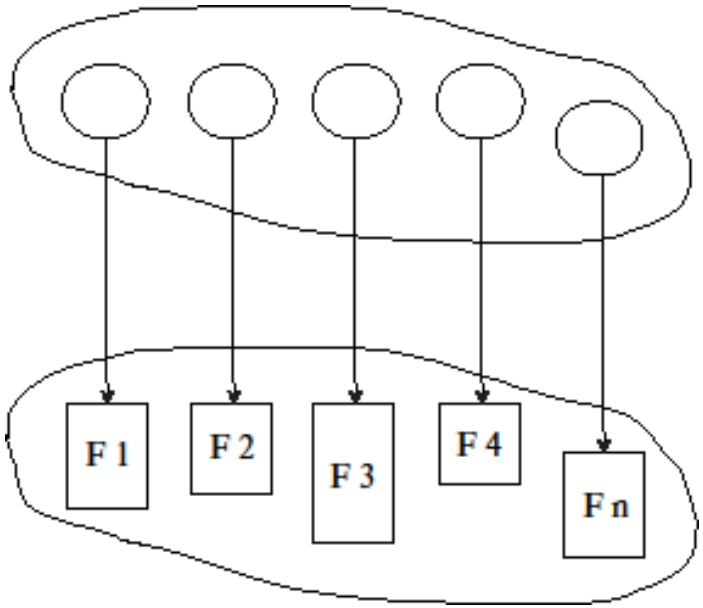






Directory Structure

- A collection of nodes containing information about all files



- Both the directory structure and the files reside on disk



Troy University Resources



File Operations

- **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)** – release the open-file reference and related resources. This does not guarantee durable data





Open Files

- Several pieces of data are needed to manage open files:
  - **Open-file table**: tracks open files
  - File pointer: current offset in an open-file description, which may be shared
  - **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

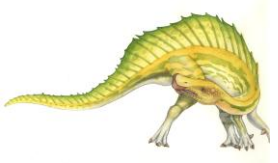


Troy University Resources



File Locking

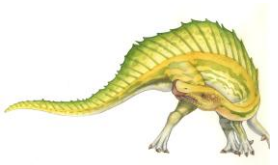
- Provided by some operating systems and file systems
  - Similar to reader-writer locks
  - **Shared lock** similar to reader lock – several processes can acquire concurrently
  - **Exclusive lock** similar to writer lock
- Mediates access to a file
- Mandatory or advisory:
  - **Mandatory** – access 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")) {
                long size = raf.length(), half = size / 2;
                FileChannel ch = raf.getChannel();
                // this locks the first half of the file - exclusive
                exclusiveLock = ch.lock(0, half, EXCLUSIVE);
                /** Now modify the data . . . */
                // release the lock
                exclusiveLock.release();
```

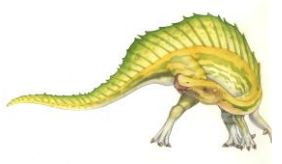


Troy University Resources



File Locking Example – Java API (Cont.)

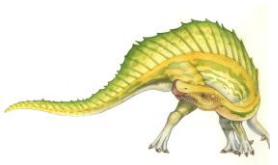
```
                    // this locks the second half of the file - shared
                    sharedLock = ch.lock(half, size - half, SHARED);
                    /** Now read the data . . . */
                    // release the lock
                    sharedLock.release();
            } catch (java.io.IOException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

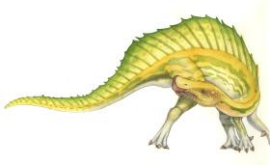


Troy University Resources



File Structure

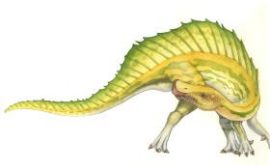
- None - sequence of words, bytes
- Simple record structure
  - Lines
  - Fixed length
  - Variable length
- Complex Structures
  - Formatted document
  - Relocatable load file
- Can simulate last two with first method by inserting appropriate control characters
- Who decides:
  - Operating system
  - Program





Access Methods

- Files may be byte streams or sequences **of records**
- **Sequential Access**
- **Direct Access**
- **Other Access Methods**

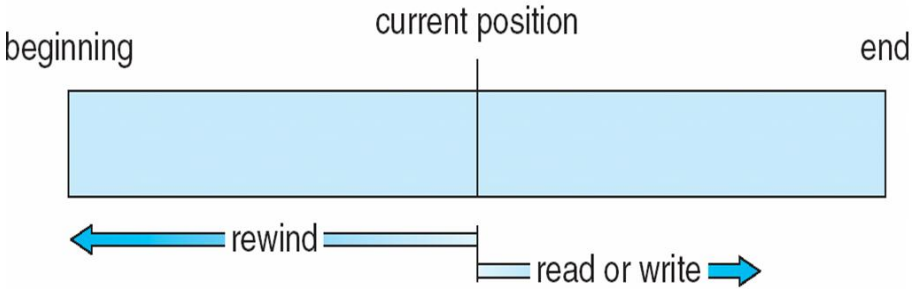


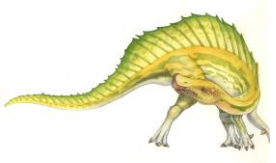
Troy University Resources



Sequential Access

- Operations
  - `read next`
  - `write next`
  - `Reset`
  - On sequential media, rewriting may truncate the remaining sequence
- Figure







Direct Access

- Operations
  - `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





Simulation of Sequential Access on Direct-access File

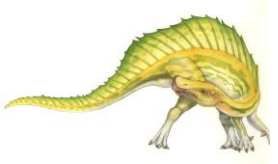
sequential access	implementation for direct access
reset	$cp = 0;$
read next	*read cp*; $cp = cp + 1;$
write next	*write cp*; $cp = cp + 1;$





Other Access Methods

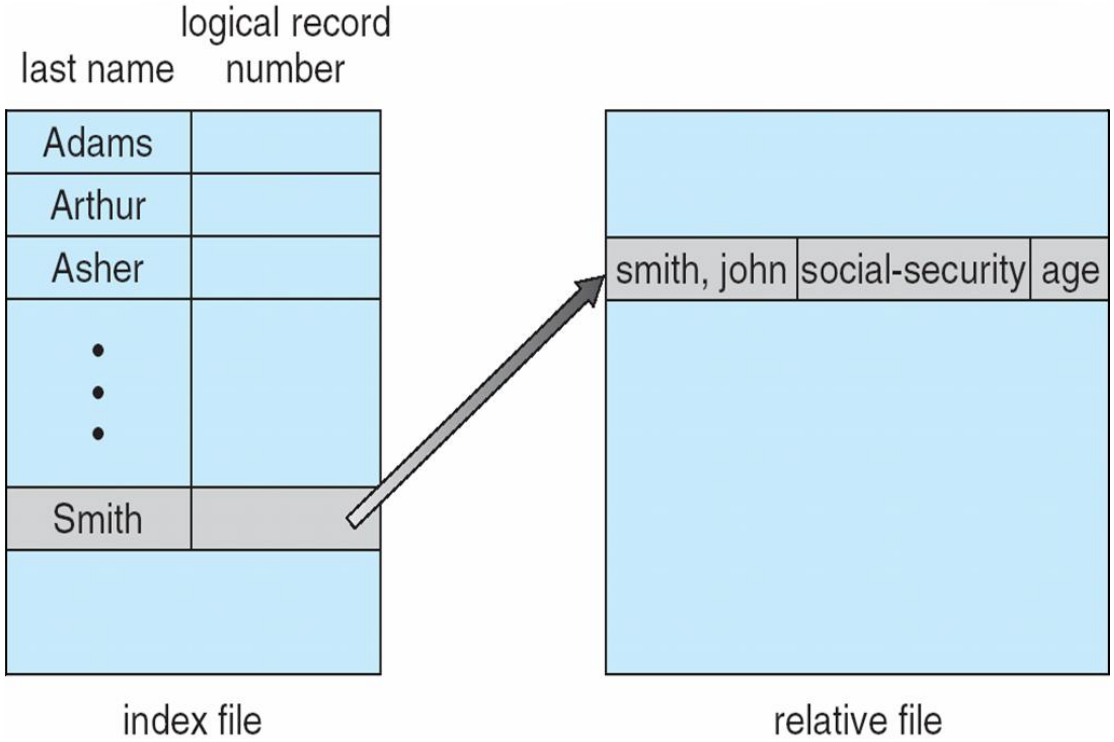
- Can be other access methods built on top of base methods
- General involve creation of an **index** for the file
- Keep index in memory for fast determination of location of data to be operated on (consider Universal Product Code (UPC code) plus record of data about that item)
- If the index is too large, create an in-memory index, which an index of a disk index
- 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)

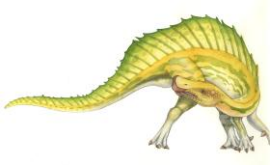


Troy University Resources



Example of Index and Relative Files

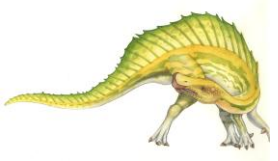






Disk Structure

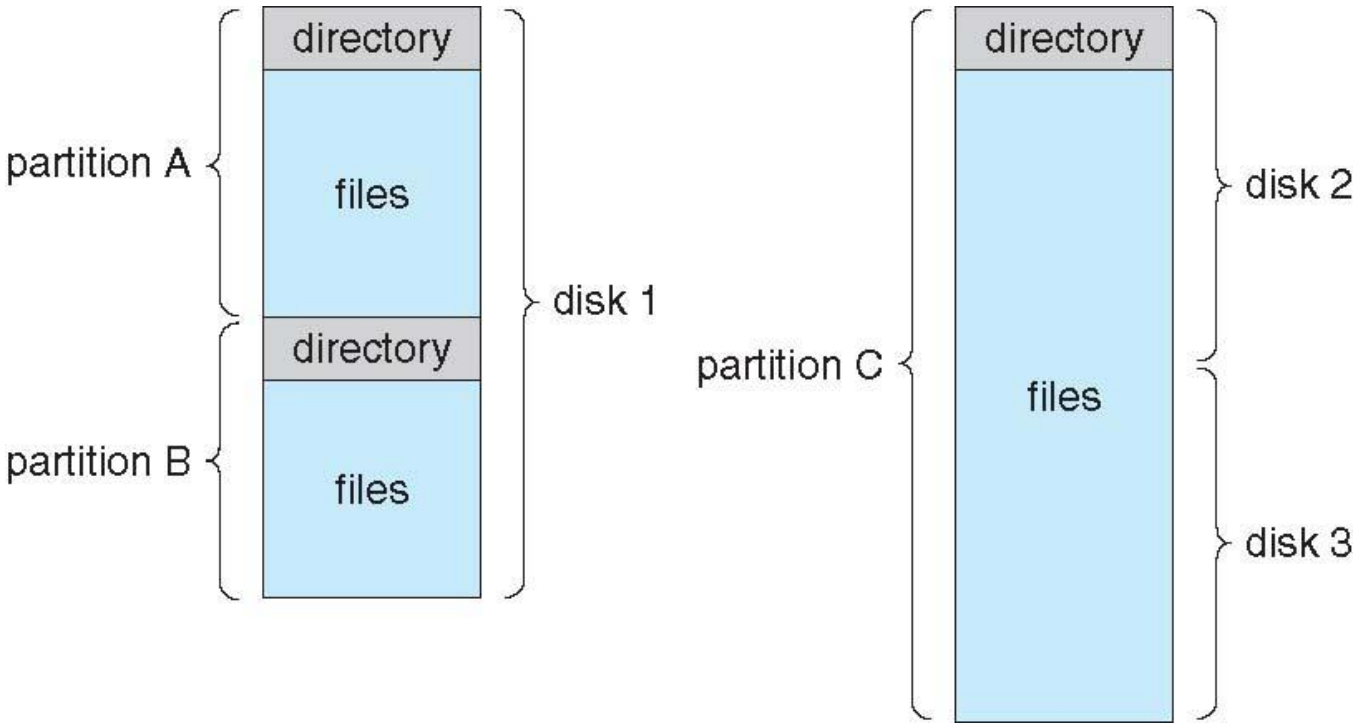
- Disk can be subdivided into **partitions**
- Disks or partitions can be **RAID** protected against failure
- Disk or partition can be used **raw** – without a file system, or **formatted** with a file system
- Partitions also known as minidisks, slices
- Entity containing file system is known as a **volume**
- Each volume containing a file system also tracks that file system's info in **device directory** or **volume table of contents**
- In addition to **general-purpose file systems** there are many **special-purpose file systems**, frequently all within the same operating system or computer

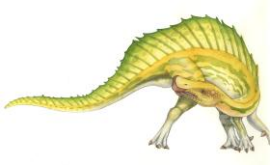


Troy University Resources



A Typical File-system Organization







Types of File Systems

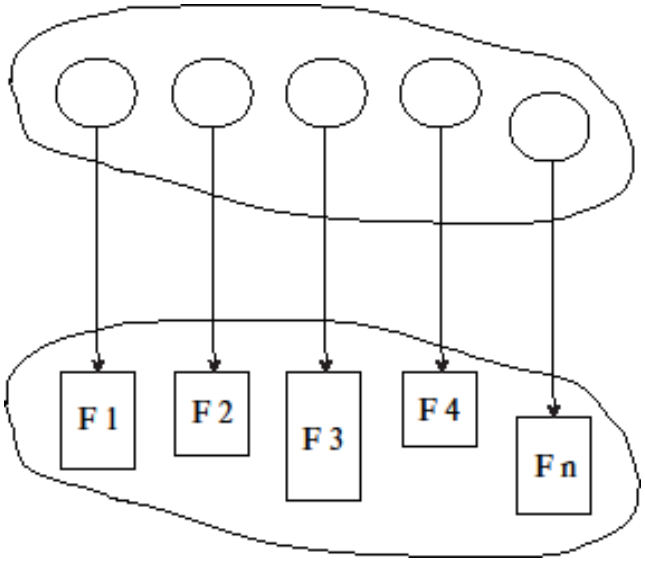
- We mostly talk of general-purpose file systems
- But systems frequently have may file systems, some general- and some special- purpose
- Consider Solaris has
  - tmpfs – memory-based volatile FS for fast, temporary I/O
  - objfs – interface into kernel memory to get kernel symbols for debugging
  - 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
  - ufs, zfs – general purpose file systems



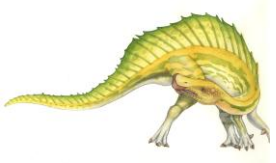


Directory Structure

- A collection of nodes containing information about all files



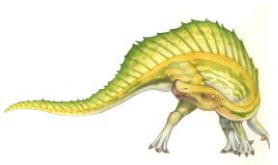
- Both the directory structure and the files reside on disk





Operations Performed on Directory

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



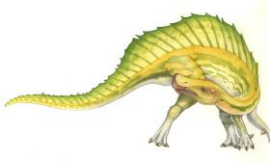
Troy University Resources



Directory Organization

The directory is organized logically to obtain

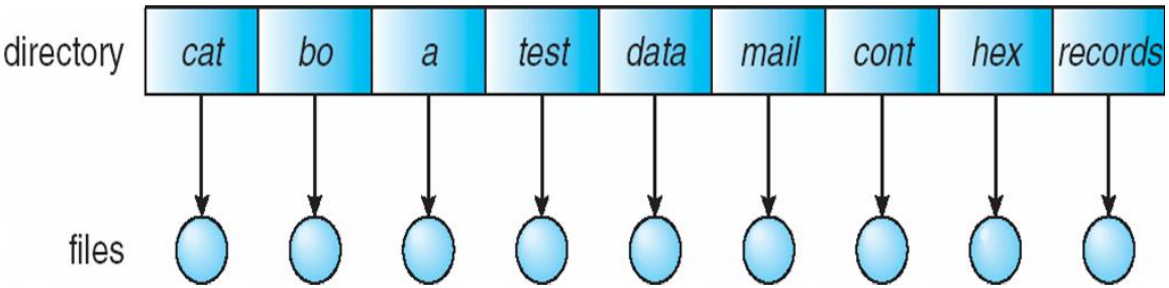
- Efficiency – locating a file quickly
- Naming – convenient to users
  - Two users can have same name for different files
  - The same file can have several different names
- Grouping – logical grouping of files by properties, (e.g., all Java programs, all games, …)





Single-Level Directory

- A single directory for all users



- Naming problem
- Grouping problem

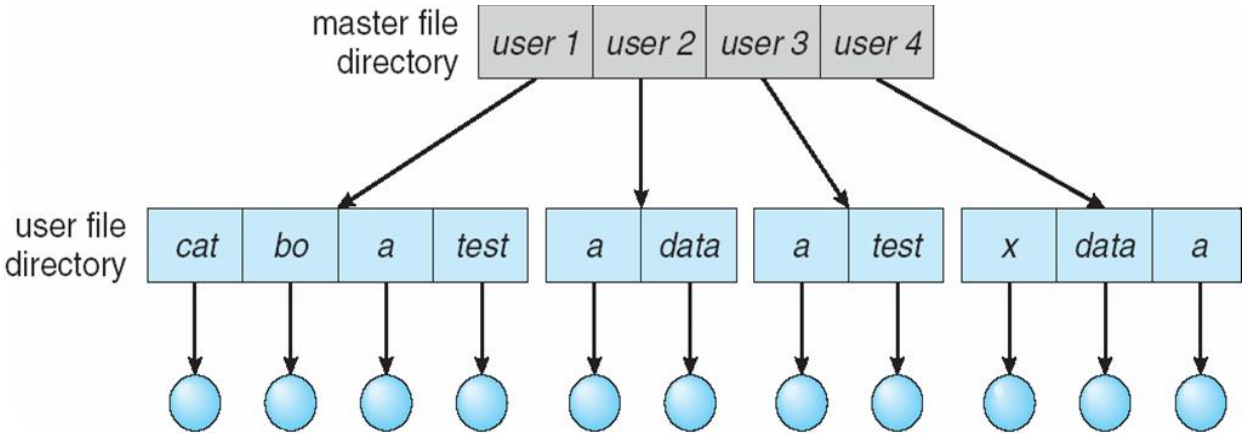


Troy University Resources

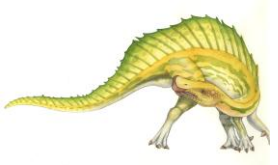


Two-Level Directory

- Separate directory for each user

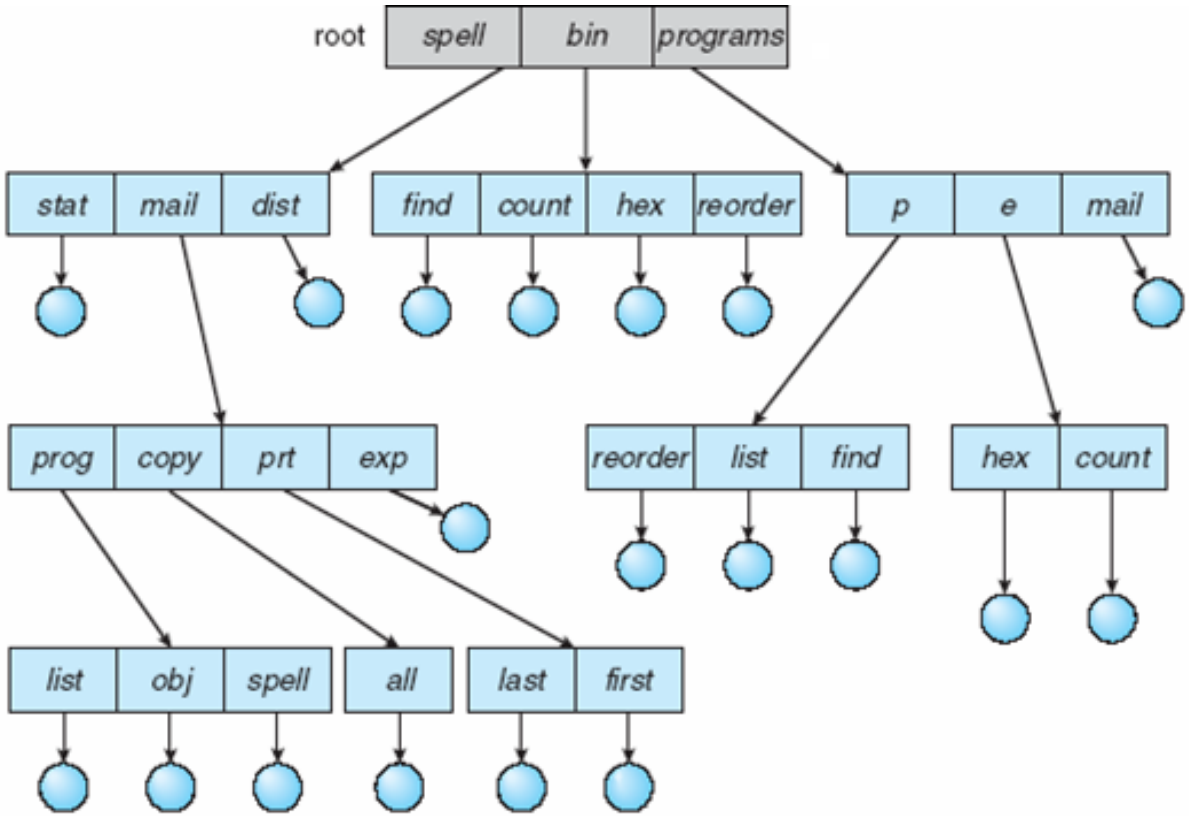


- Path name
- Can have the same file name for different user
- Efficient searching
- No grouping capability





Tree-Structured Directories

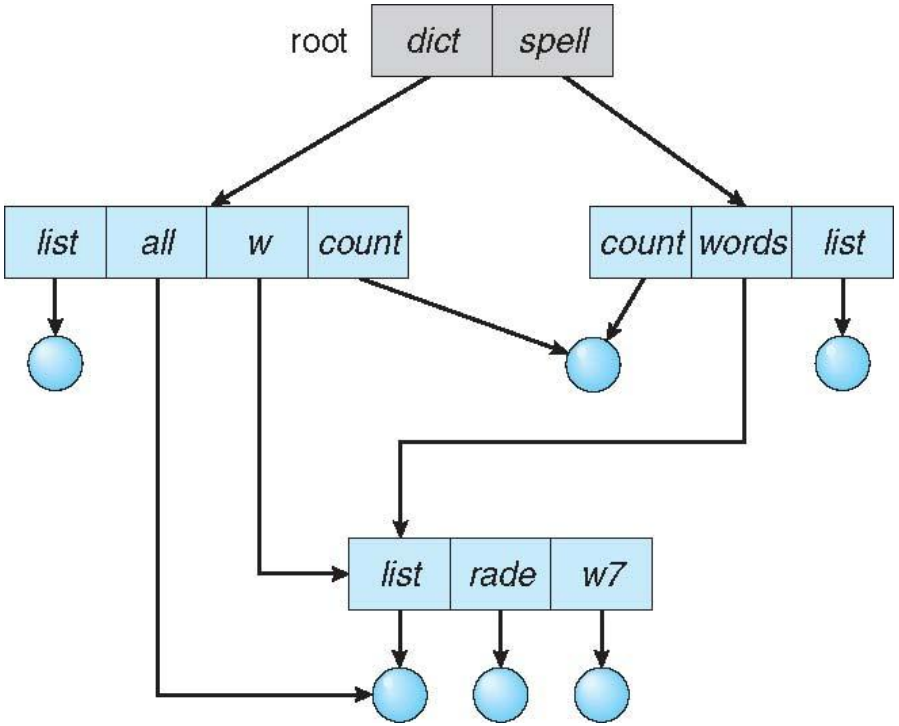


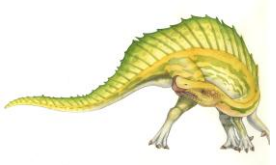




Acyclic-Graph Directories

- Have shared subdirectories and files
- Example







Acyclic-Graph Directories (Cont.)

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

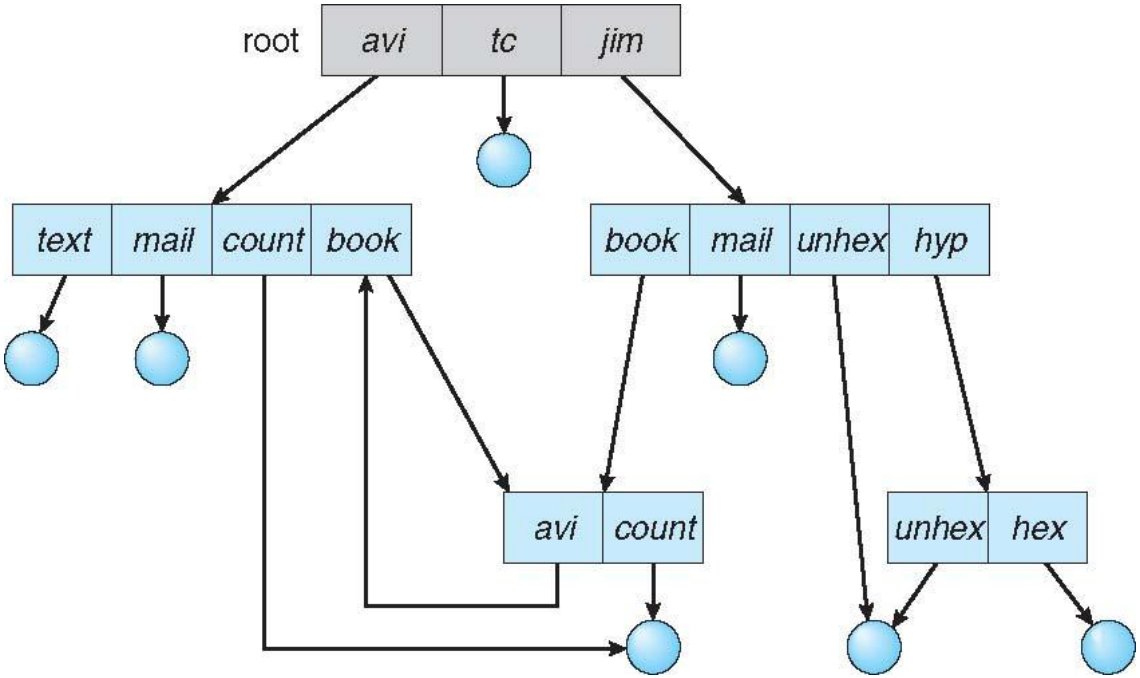
  Solutions:
  - Backpointers, so we can delete all pointers.
    - Variable size records a problem
  - Backpointers using a daisy chain organization
  - Entry-hold-count solution
- New directory entry type
  - **Link** – another name (pointer) to an existing file
  - **Resolve the link** – follow pointer to locate the file

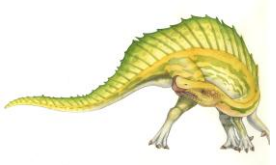


Troy University Resources



General Graph Directory

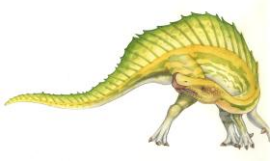






General Graph Directory (Cont.)

- How do we handle links and possible cycles?
  - Allow only links to files not subdirectories
  - **Garbage collection reclaims unreachable cycles**
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK



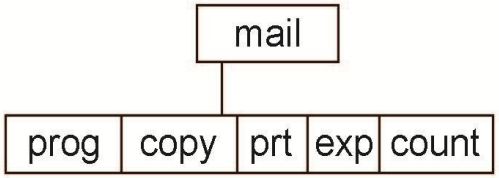
Troy University Resources



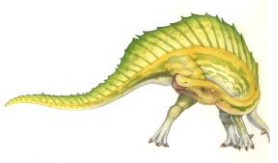
Current Directory

- Absolute paths start at /. Relative paths start at the current (working) directory
  - `cd /spell/mail/prog`
  - `cat list`
- Relative file names are interpreted from the current directory
- Example of creating a subdirectory
  - If in current directory is `/mail`
  - The command

    mkdir <dir-name>
  - Results in:



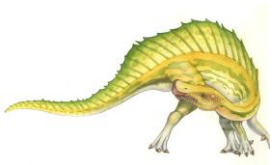
  - Recursive deletion of “mail” removes its entire subtree





Protection

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



Troy University Resources

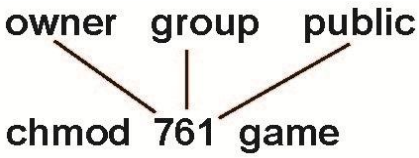


Access Lists and Groups in Unix

- 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 file (say *game*) or subdirectory, define an appropriate access.



- Attach a group to a file

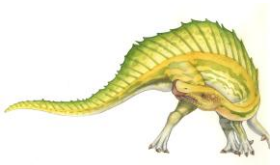
```
chgrp       G       game
```





A Sample UNIX Directory Listing

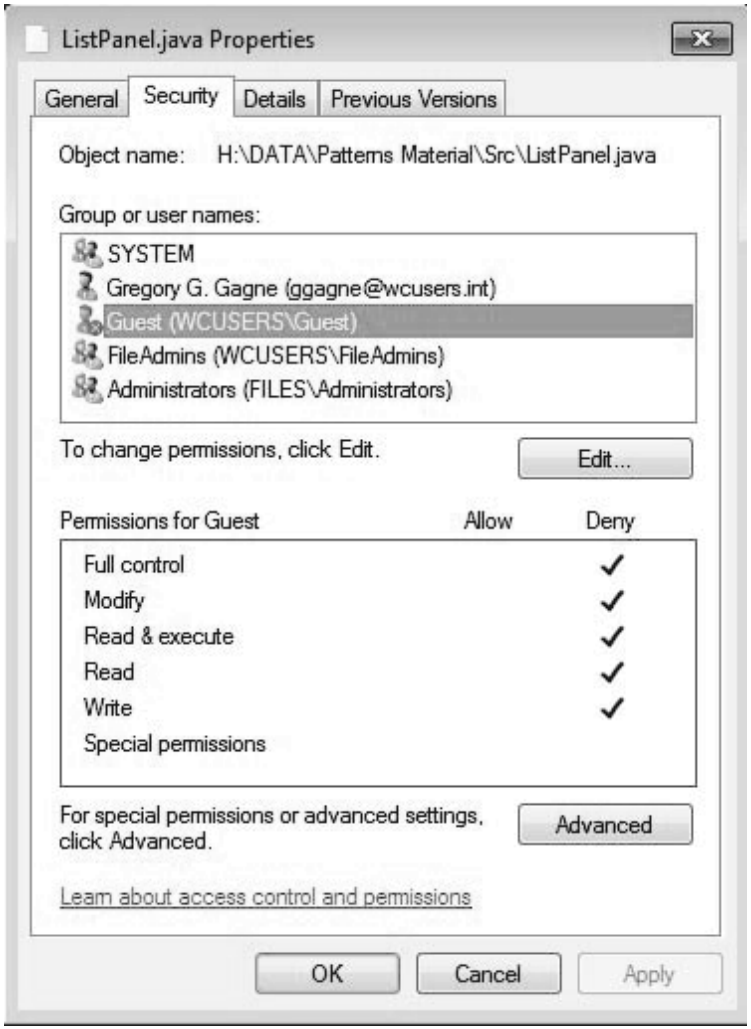
-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2003	program
drwx--x--x	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/

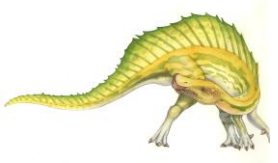


Troy University Resources



Windows 7 Access-Control List Management

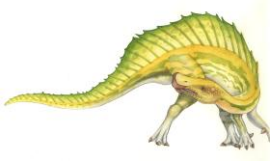






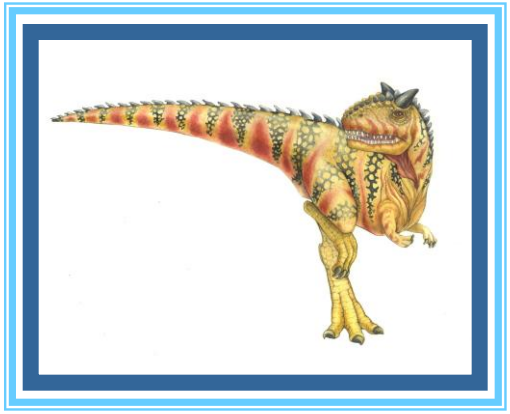
Memory-Mapped Files

- mmap() maps a file region into a process's virtual address space.
- File pages are loaded on demand. Reads and writes use ordinary memory accesses.
- MAP_SHARED changes can update the file and other shared mappings. MAP_PRIVATE uses copy-on-write.
- msync(MS_SYNC) requests synchronous write-back. munmap() removes the mapping.



Troy University Resources

End of Chapter 13



Silberschatz, Galvin and Gagne ©2018