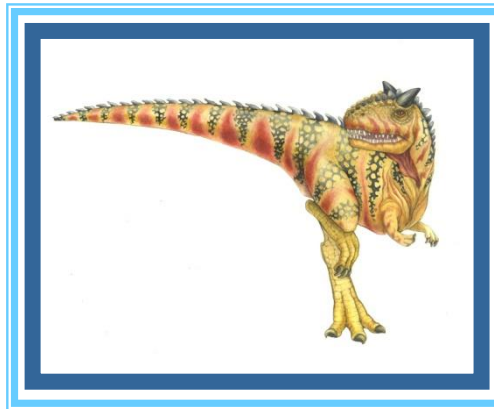# Chapter 7:
File-System Interface

Chapter 7: File-System Interface

- File Concept
- Access Methods
- Disk and Directory Structure

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

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

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
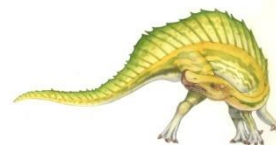
File Structure

- 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 first method by inserting appropriate control characters
- Who decides:
    - Operating system
    - Program
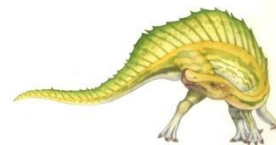
Sequential-access File
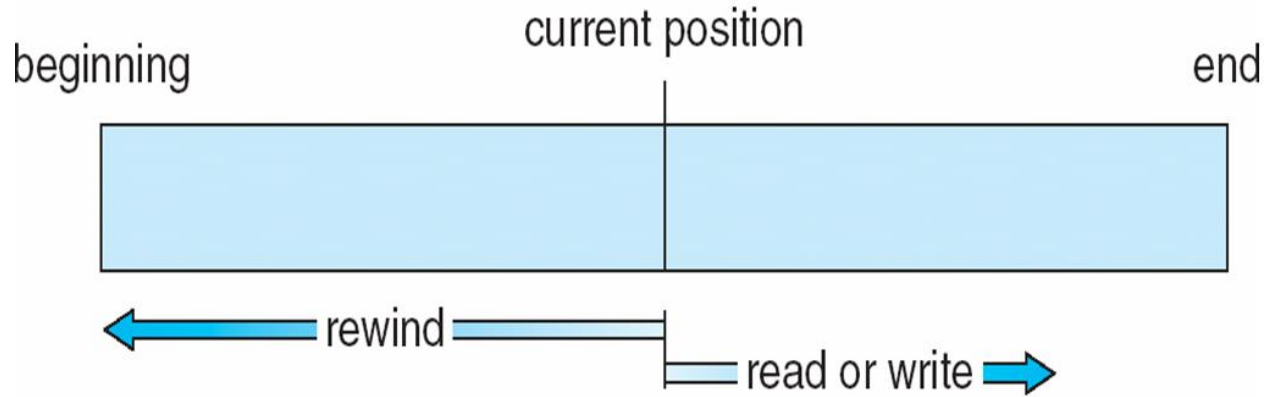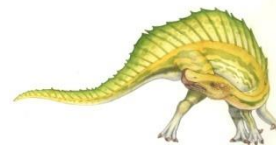
Access Methods

- **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 allocation problem in Ch 12

Directory Structure

- A collection of nodes containing information about all files

Directory

Files

F 1   F 2   F 3   F 4   F n
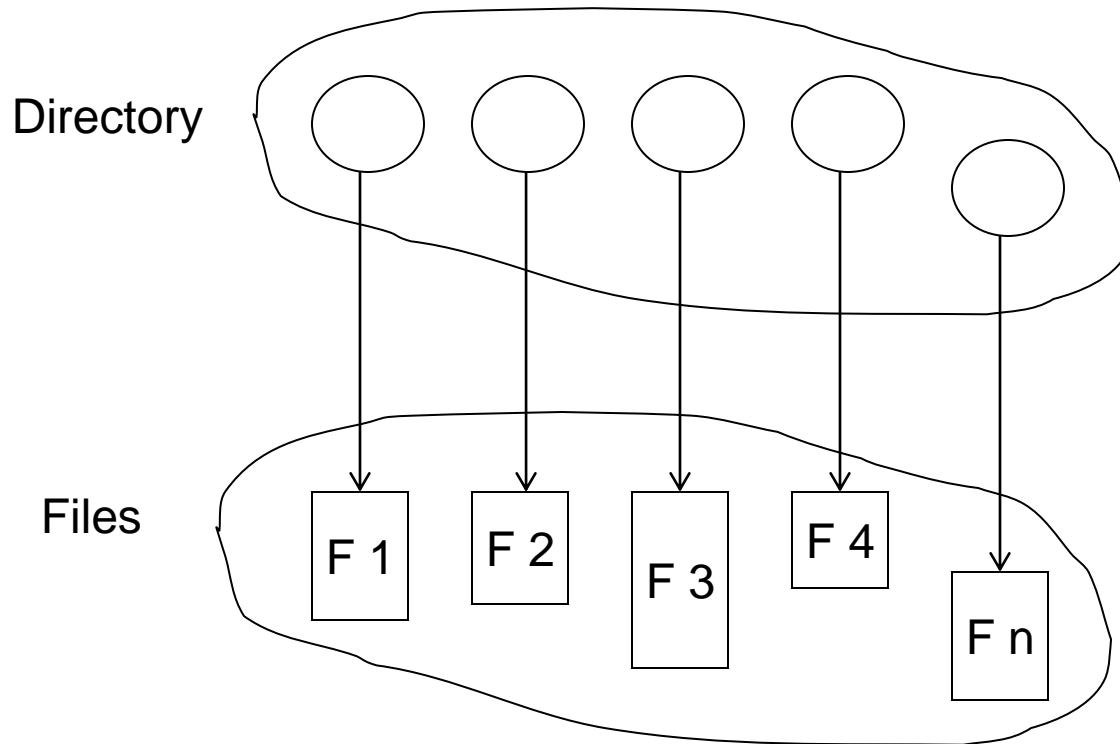
Both the directory structure and the files reside on disk

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
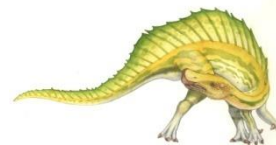
A Typical File-system Organization
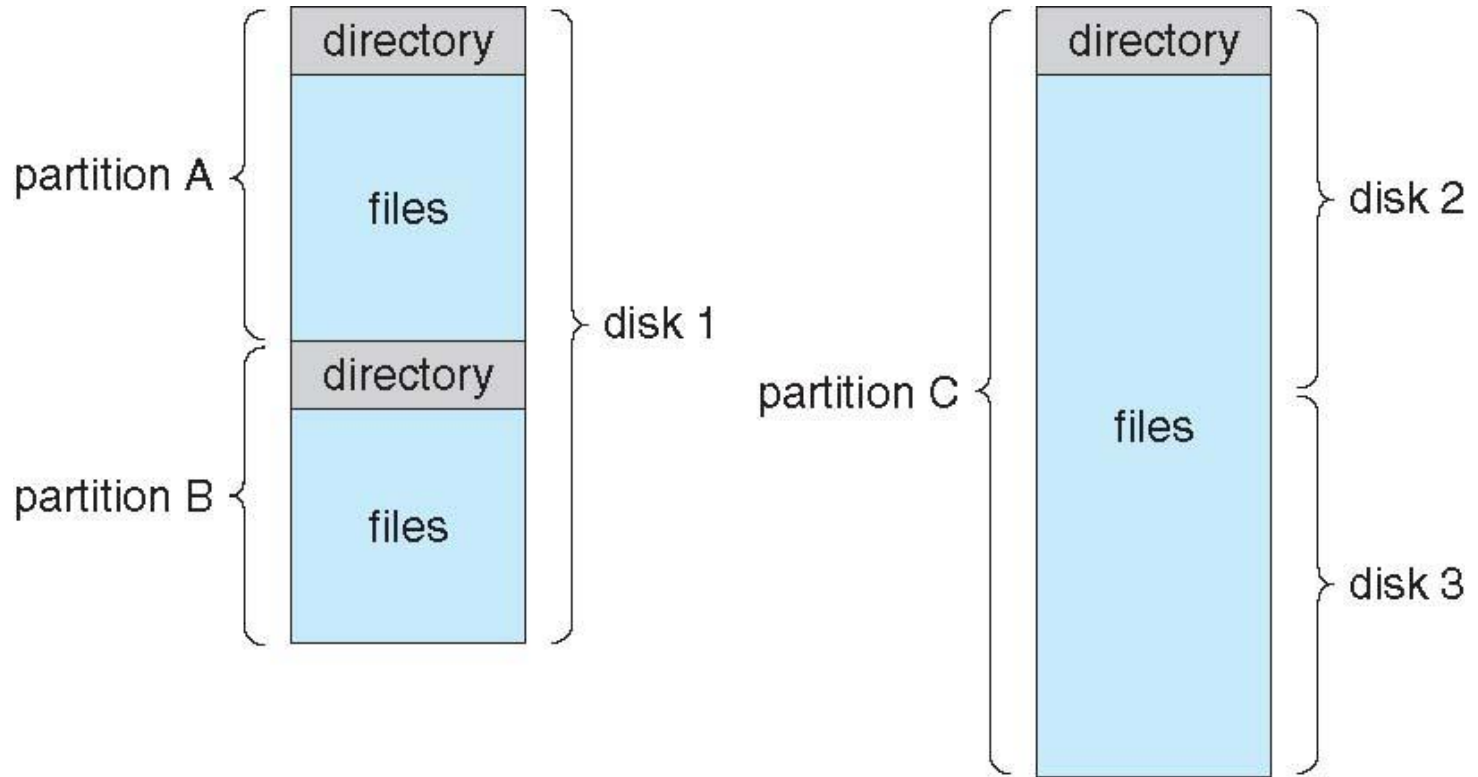
Operations Performed on Directory

- Search for a file

- Create a file

- Delete a file

- List a directory

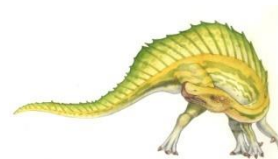- Rename a file

- Traverse the file system

Directory Organization

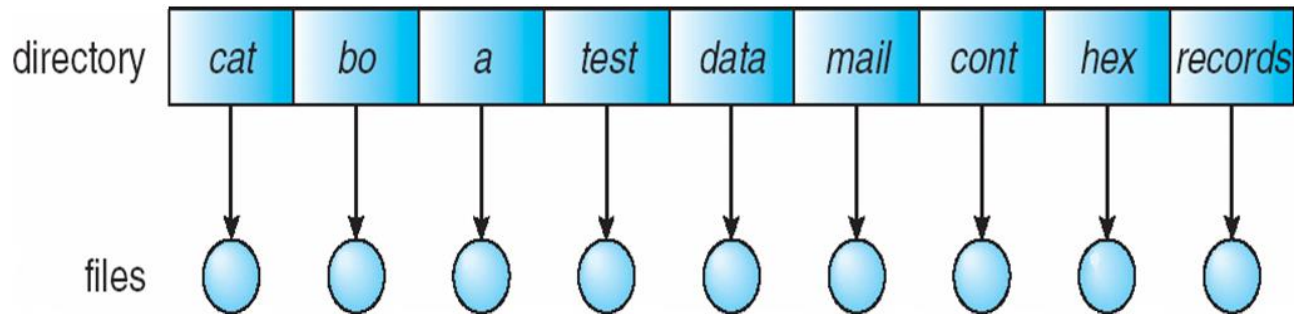The directory is organized logically to obtain

- 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, …)
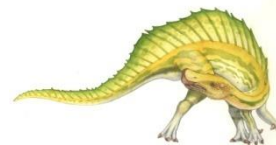
Single-Level Directory

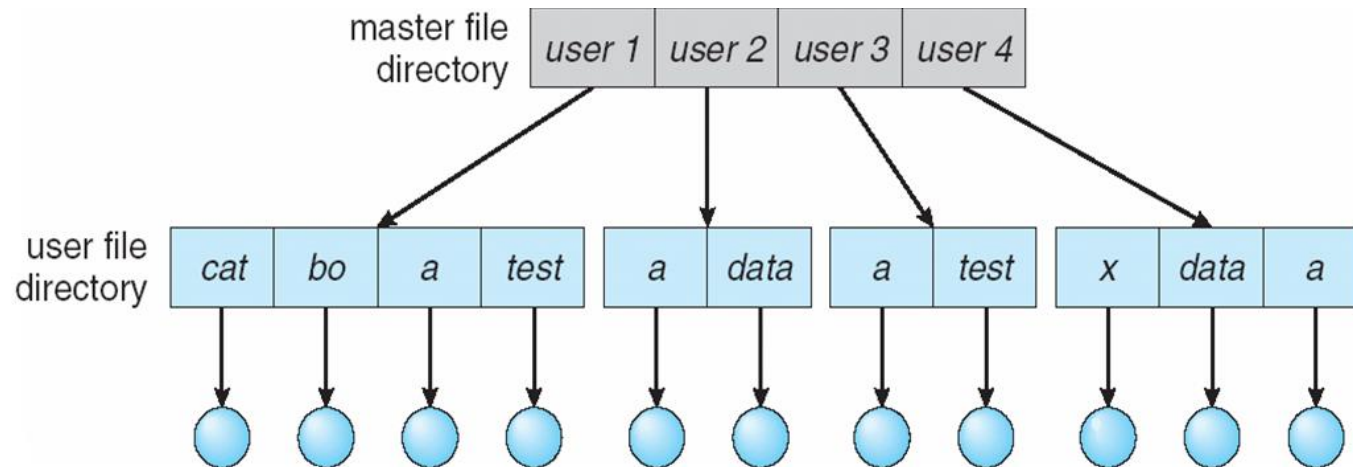☐ A single directory for all users

☐ Naming problem

☐ Grouping problem

Two-Level Directory
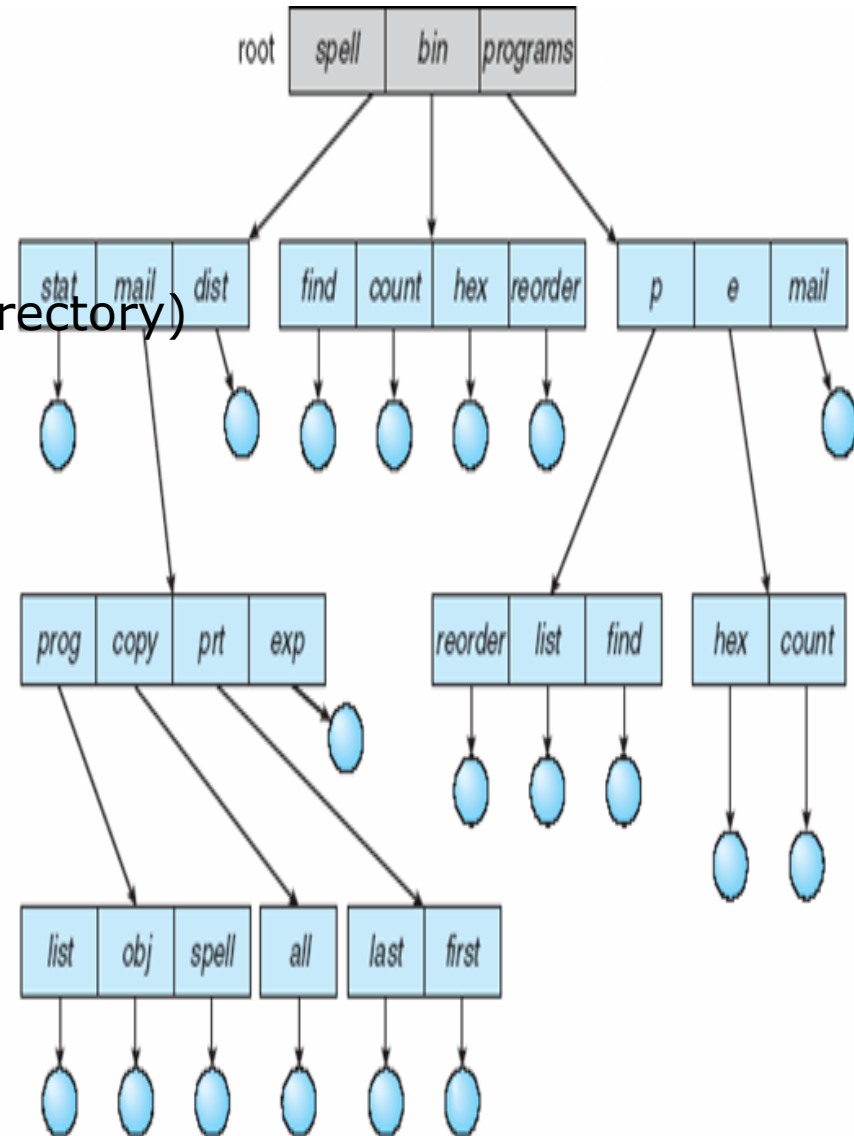
- Separate directory for each user

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

Tree-Structured Directories

- 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 current directory
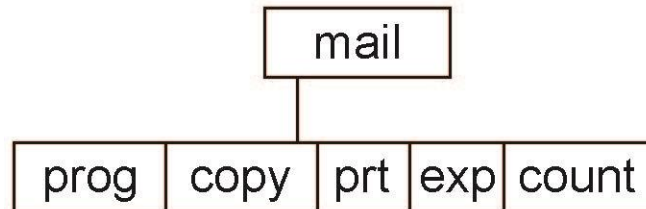- Delete a file

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

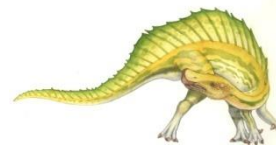- Creating a new subdirectory is done in current directory

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

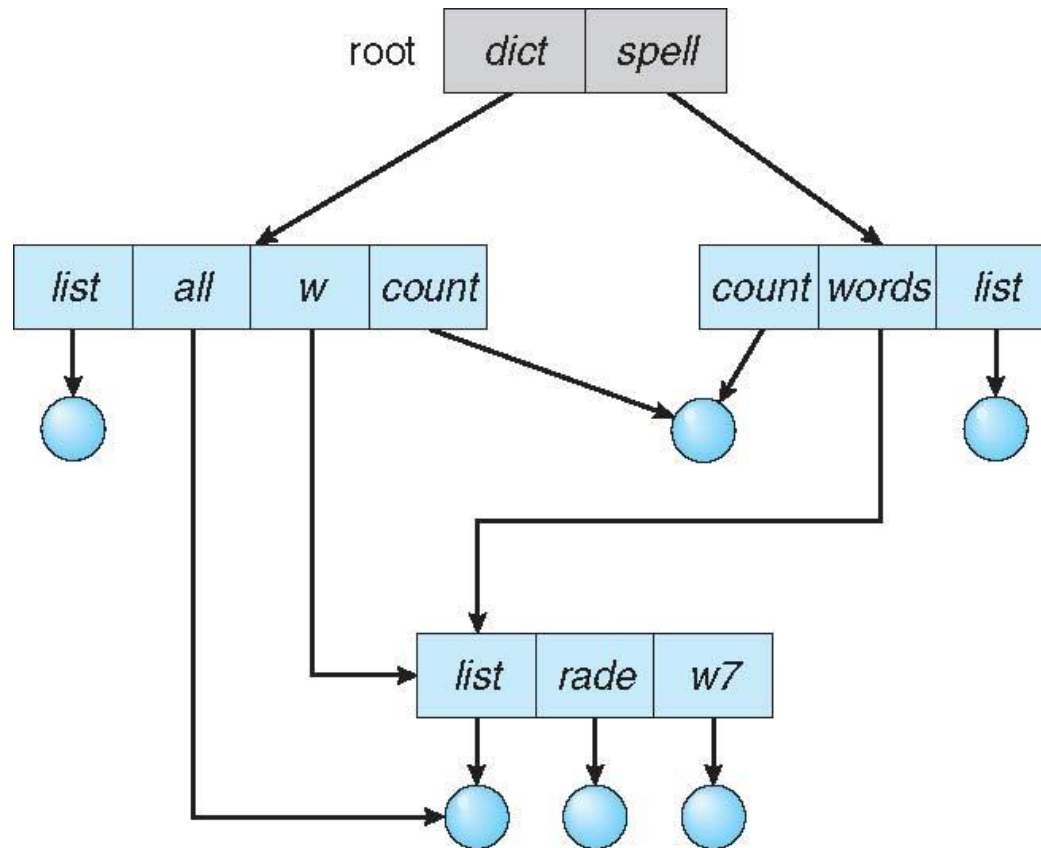Example: if in current directory `/mail`

```
mkdir count
```


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

Acyclic-Graph Directories

- Have shared subdirectories and files

Acyclic-Graph Directories (Cont.)

- Two different names (aliasing)
- If **dict** deletes **list** $\Rightarrow$ dangling pointer
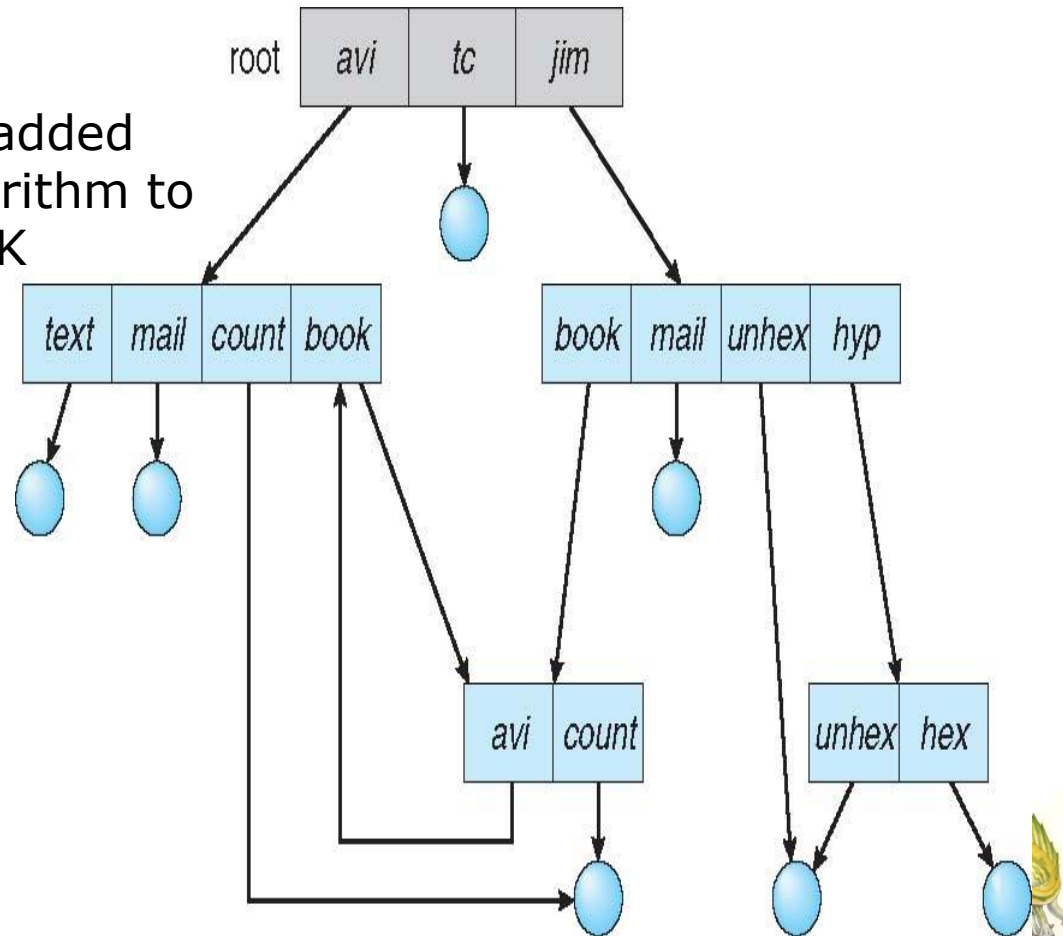
  Solutions:

    - 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
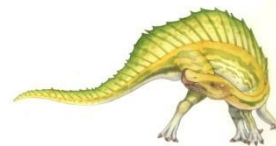
General Graph Directory

- 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

End of Chapter 7