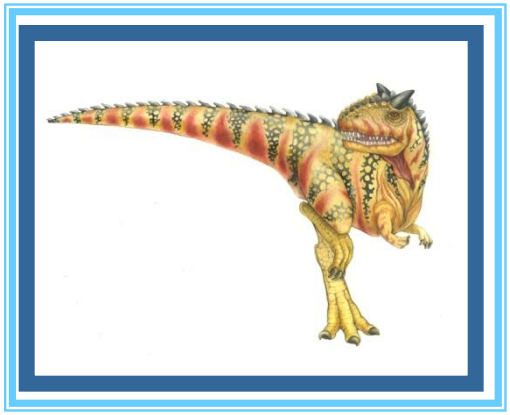
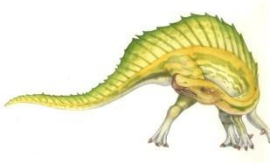
Chapter 10: File System



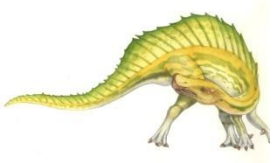
Chapter 10: File System

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



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



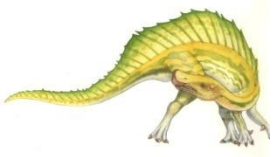
File System

- A file system (often also written as filesystem) is a method of storing and organizing computer files and their data.
- It organizes these files into a database for the storage, organization, manipulation, and retrieval by the computer's operating system.
- File systems are used on data storage devices such as hard disks or CD-ROMs to maintain the physical location of the files

File Concept

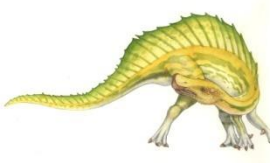
- Computers can store information on various storage media, such as magnetic disks, magnetic tapes, and optical disks.
- The operating system provides *a uniform logical view of information storage*.
- A file is *a named collection of related information that is recorded on secondary storage,* that is, data cannot be written to secondary storage unless they are within a file.
- In general , a file is *a sequence of bits, bytes, lines, or records, the meaning of which is defined by the file's creator and user*.
- Operating System mapped Files onto physical storage devices.

File is the logical storage unit



File Concept (cont.)

- ***Data files:*** may be: Numeric, alphabetic, alphanumeric or binary
- ***Text file:*** is a sequence of characters organized into lines.
- ***Source file:*** is a sequence of subroutines and executable statements.
- ***Object file:*** is a sequence of bytes organized into blocks understandable by the system's linker.
- ***Executable file:*** is a series of code sections that the loader can bring into memory and execute.

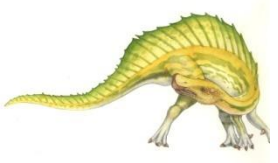


File Attributes

A file's attributes vary from one operating system to another but typically consist of these:-

- **Name** – Many OS support two part file names (name.extension)
  - File.bak (backup file); file.pdf (pdf file); file jpg (still picture in JPEG format)

  - Some OS do not enforce extensions (e.g., UNIX).
- **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



File Operations

File is an **abstract data type**

The six basic file operations are:

- **Create :**
  1. Find space in the file system for the file.
  2. Entry for the new file into the directory.
- **Write:** the system call specifying both the file name and the information to be written to the file.
  1. Search the directory to find the file's location
  2. Keep a write pointer to the location in the file where the next write take place
- **Read:** the system calls specifies the file name and where in memory the next block of the file should be put.
  1. Search the directory to find the file's location
  2. Keep a read pointer to the location in the file where the next write take place

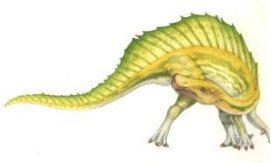
➢ Both write and read operation can use a per-process current-file-position pointer to saving space and reducing system complexity.

File Operations

- **Reposition within file:**
  1. Search The directory for the appropriate entry.
  2. Reposition the current-file-position pointer to a given value.
- **Delete:**
  1. search the directory for the named file.
  2. release all file space and erase the directory entry.
- **Truncate:** The user may want to erase the contents of a file but keeping its attributes. Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged except for file length, but lets the file be reset to length zero and its file space released**.**

File Operations

- To avoid searching the directory for the entry associated with the named file , the following system call will use:
  - Open(Fi) – search the directory structure on disk for entry Fi, and move the content of entry to memory
  - Close (Fi) – move the content of entry Fi in memory to directory structure on disk
  - OS keeps a small table in memory, called open-file table , containing information about all open files.
    - When a file operation is requested, it specified via an index into this table.
    - When the file is no longer used, it is closed by the process, and remove its entry from this table.

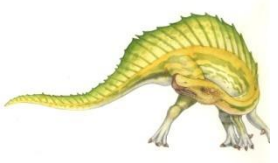


Open and close Files implementation

- The OS uses two levels of internal open file tables :
  - Pre-process table: process-dependent information about all files that a process has open , eg: current file pointer, access rights…
  - System-wide table: process-independent information about all files that has open, eg: file location, size, dates…
  - Each entry in the per-process table in turn points to a system-wide table.
- Several information 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

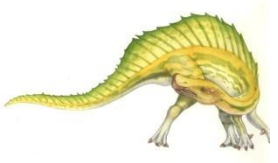
Open File Locking

- 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. Eg: Windows
  - **Advisory** – processes can find status of locks and decide what to do depending of the software developer. Eg: UNIX



File Types – Name, Extension

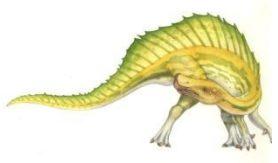
file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual 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



Internal File Structure

.

- None - sequence of words, bytes
- Simple record structure
  - Lines , Fixed length , Variable length
- Complex Structures
  - Formatted document
  - Relocatable load file
- To overcome the logical records variable length, using Packing a number of logical records into physical blocks
- The Packing can be done by the user's program or by the Operating system
- The file considered as a sequence of blocks.
- All file system suffer from internal fragmentation, Because disk space allocated in blocks, some portion of the last block of each file is wasted.



Access Methods

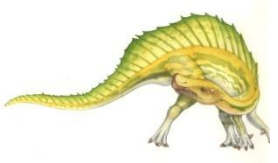
- **Sequential Access**

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

- **Direct Access**

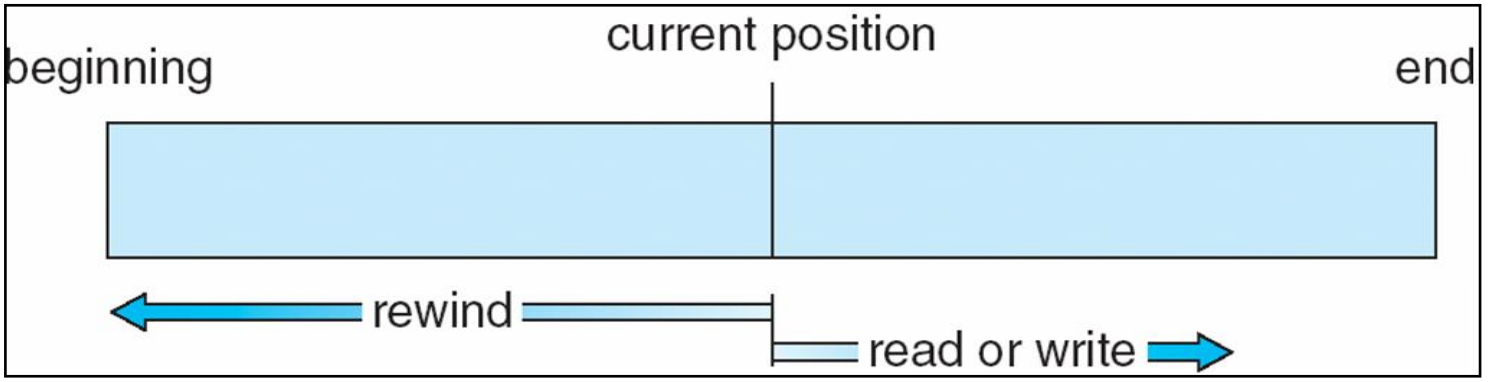
  read *n*
  write *n*
  position to *n*
  read next
  write next
  rewrite *n*

  n = relative block number



Sequential-access File

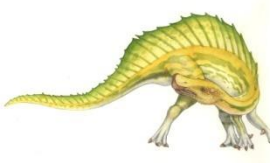
- Information in the file is processed in order, one record after the other



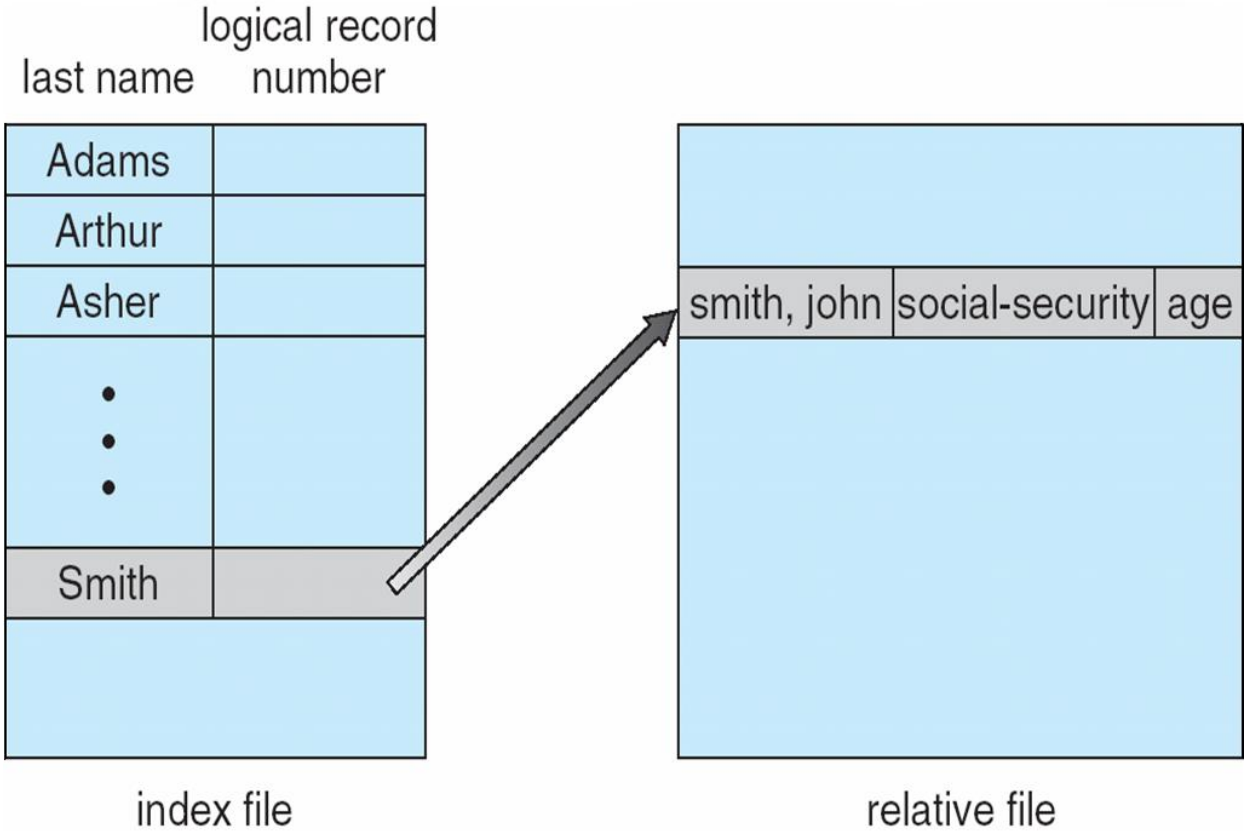
Direct-access File

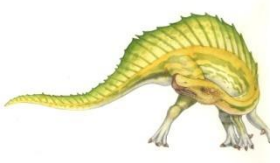
- A file is made up of fixed length that allow programs to read and write records rapidly in no particular order.
- This method is based on a disk model of a file, since disks allow random access to any file block.
- It is a great of use for immediate access to large amount of information " Databases".
- The file operations must be modified to include the block number as a parameter.

  we have ***read n***, where n is the block number rather than read next , and ***write n*** rather than write next.



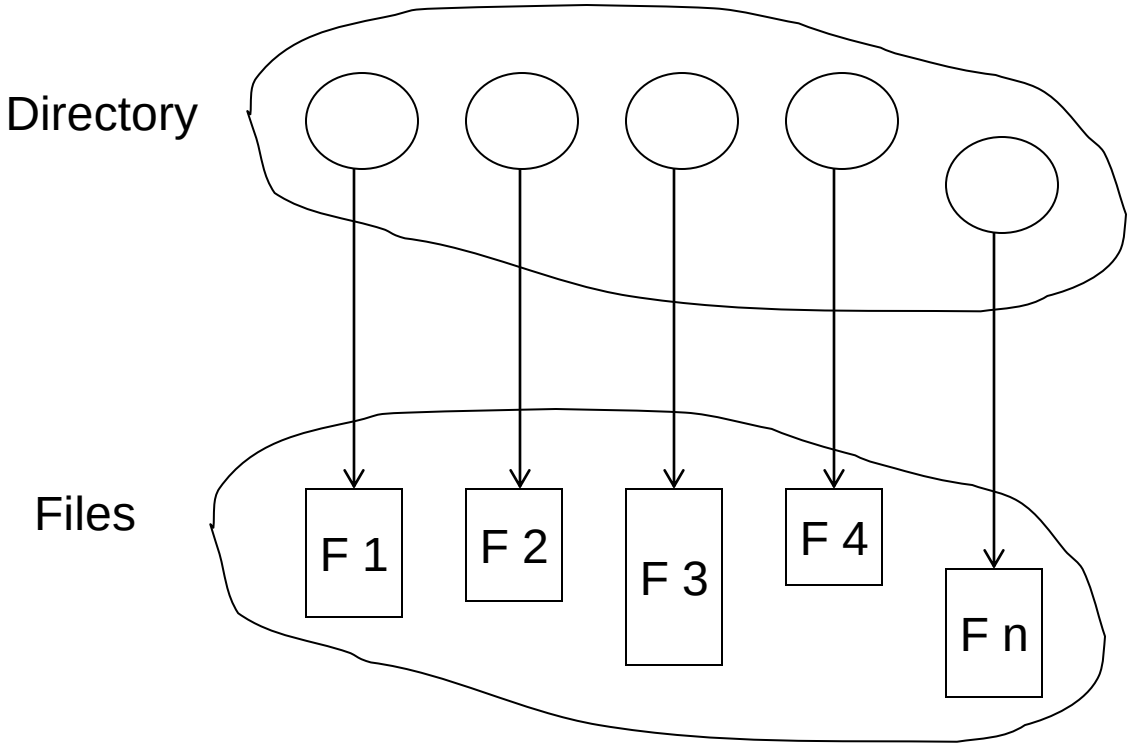
Example of Index and Relative Files





Directory Structure

- A collection of nodes containing information about all files

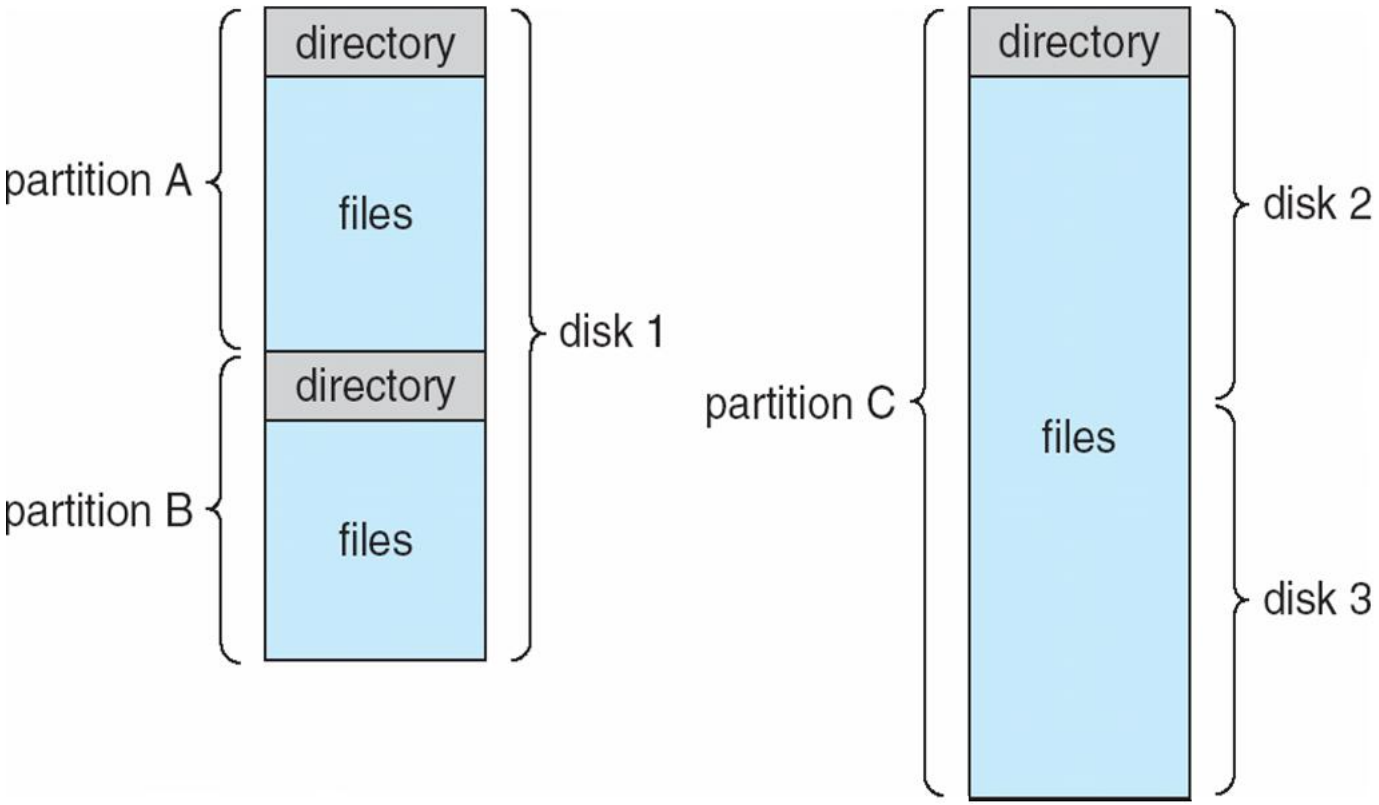


Both the directory structure and the files reside on disk
Backups of these two structures are kept on tapes

Disk Structure

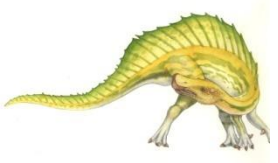
- Disk can be subdivided into partitions
- Disk or partition can be used raw – without a file system, or formatted with a file system
- Partitions also known as minidisks, slices
- Entity containing file system known as a volume
- Each volume containing file system also tracks that file system's info 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
  - tmpfs- a temporary file system created in volatile main memory and erased its contents if the system reboots.
  - objfs – a virtual file system gives debuggers access to kernel symbols.
  - lofs – a loop back file system allows one file system to be accessed in place of another one
  - See more in textbook.

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



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, …)

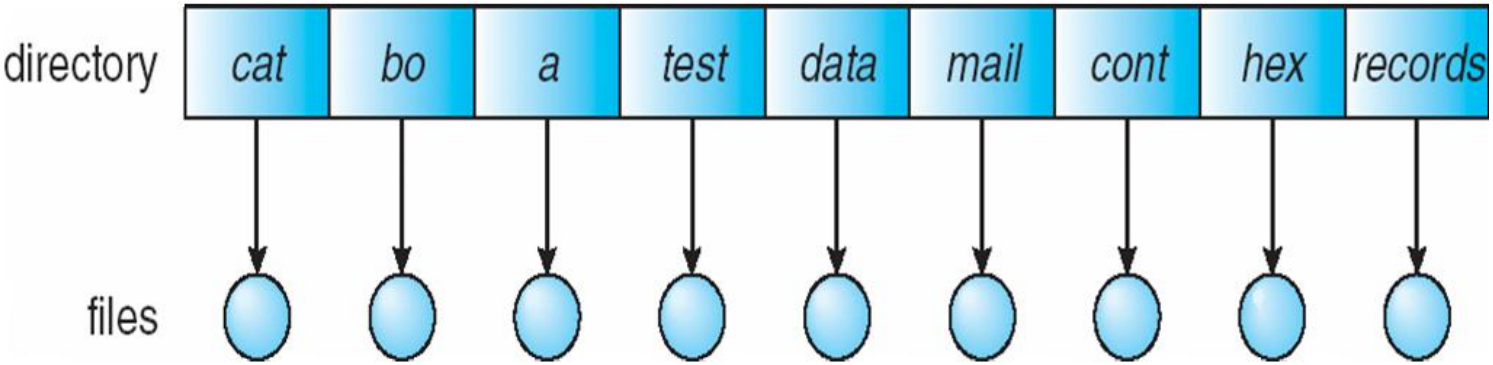
Logical directory structure schemes

The most common schemes for defining the logical structure of a directory:

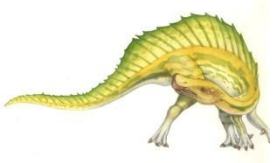
- Single-Level Directory
- Two-Level Directory
- Tree-Structured Directories
- Acyclic-Graph Directories
- General Graph Directory

Single-Level Directory

- A single directory for all users

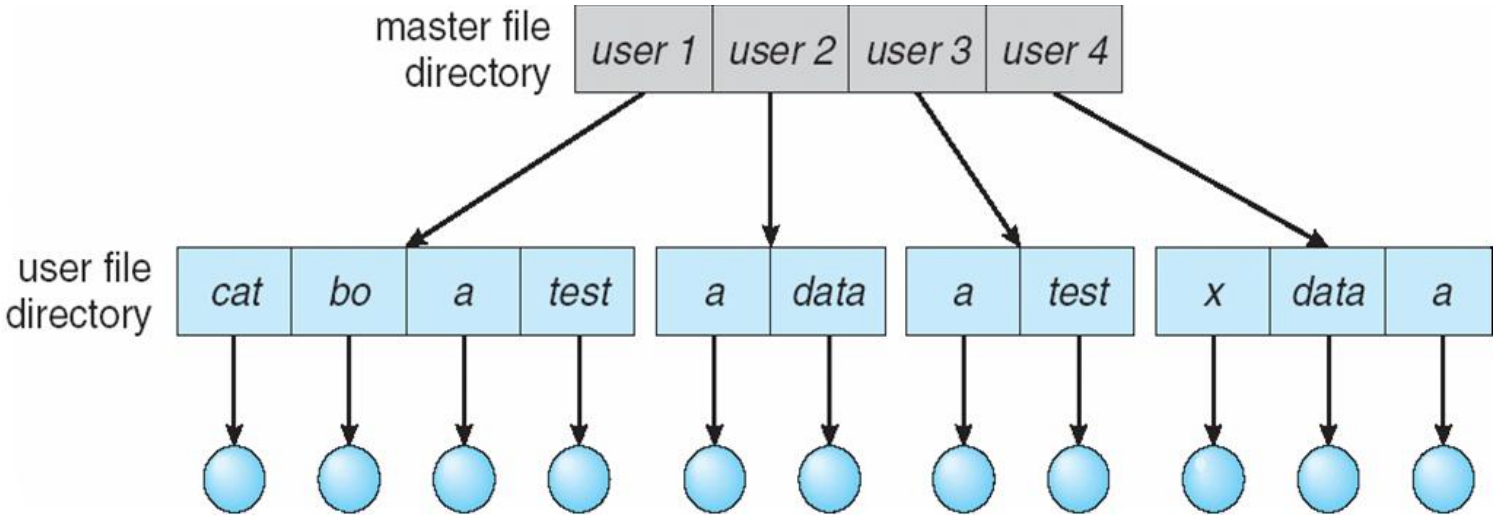


  - Naming problem
  - Grouping problem

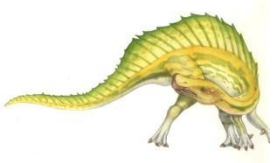


Two-Level Directory

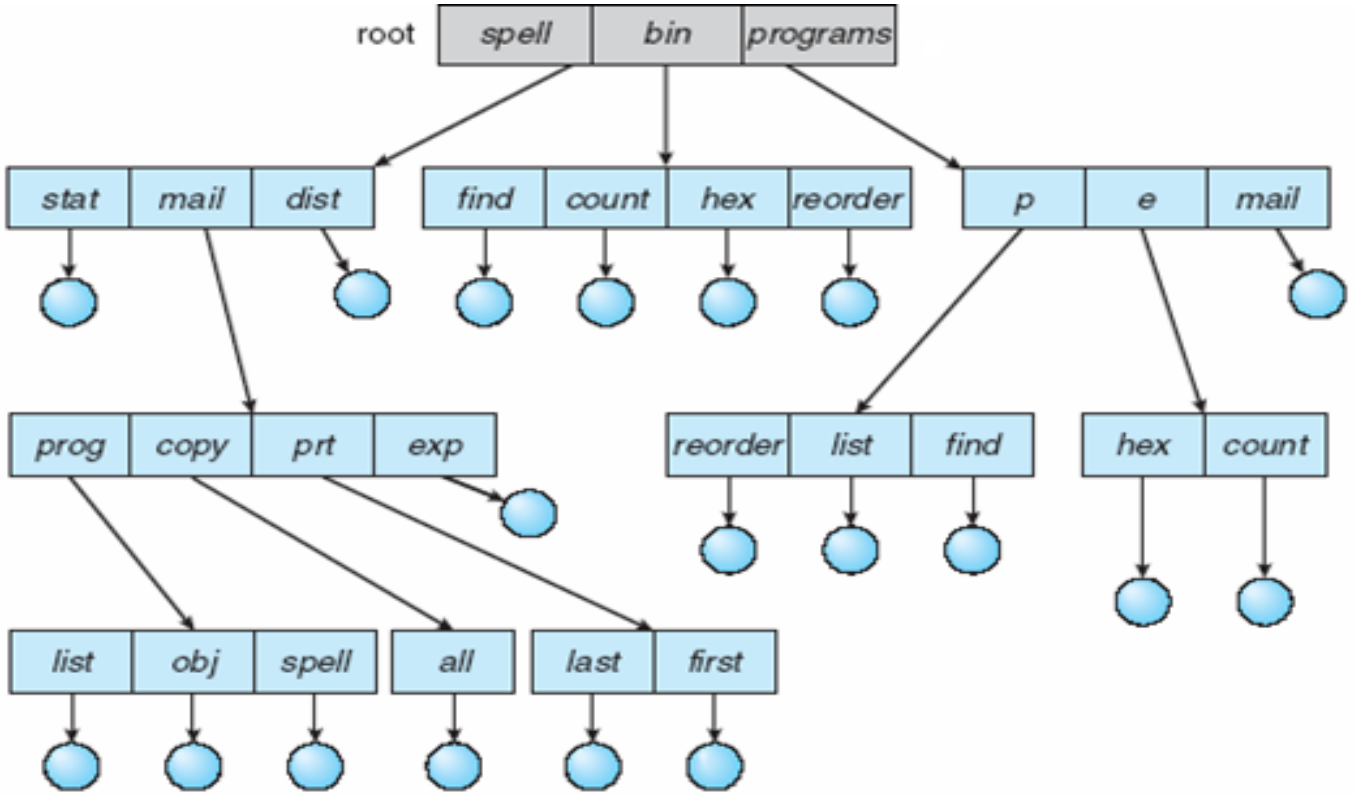
- Separate directory for each user



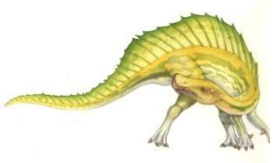
- One master file directory (MFD).
- Each user has their own user file directory (UFD).
- Each entry in the master file directory points to a user file directory.
- Each file in the system has a path name.
- Can have the same file name for different user
- Efficient searching
- No grouping capability



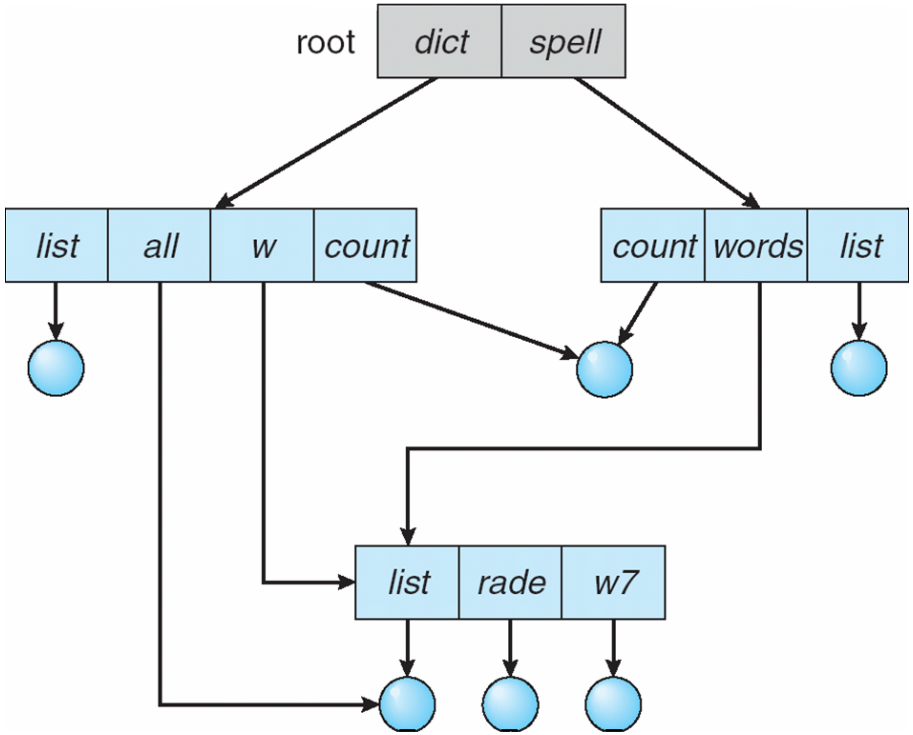
Tree-Structured Directories



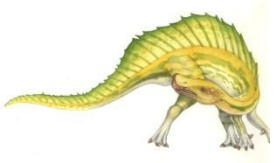
- Efficient searching
- Grouping Capability



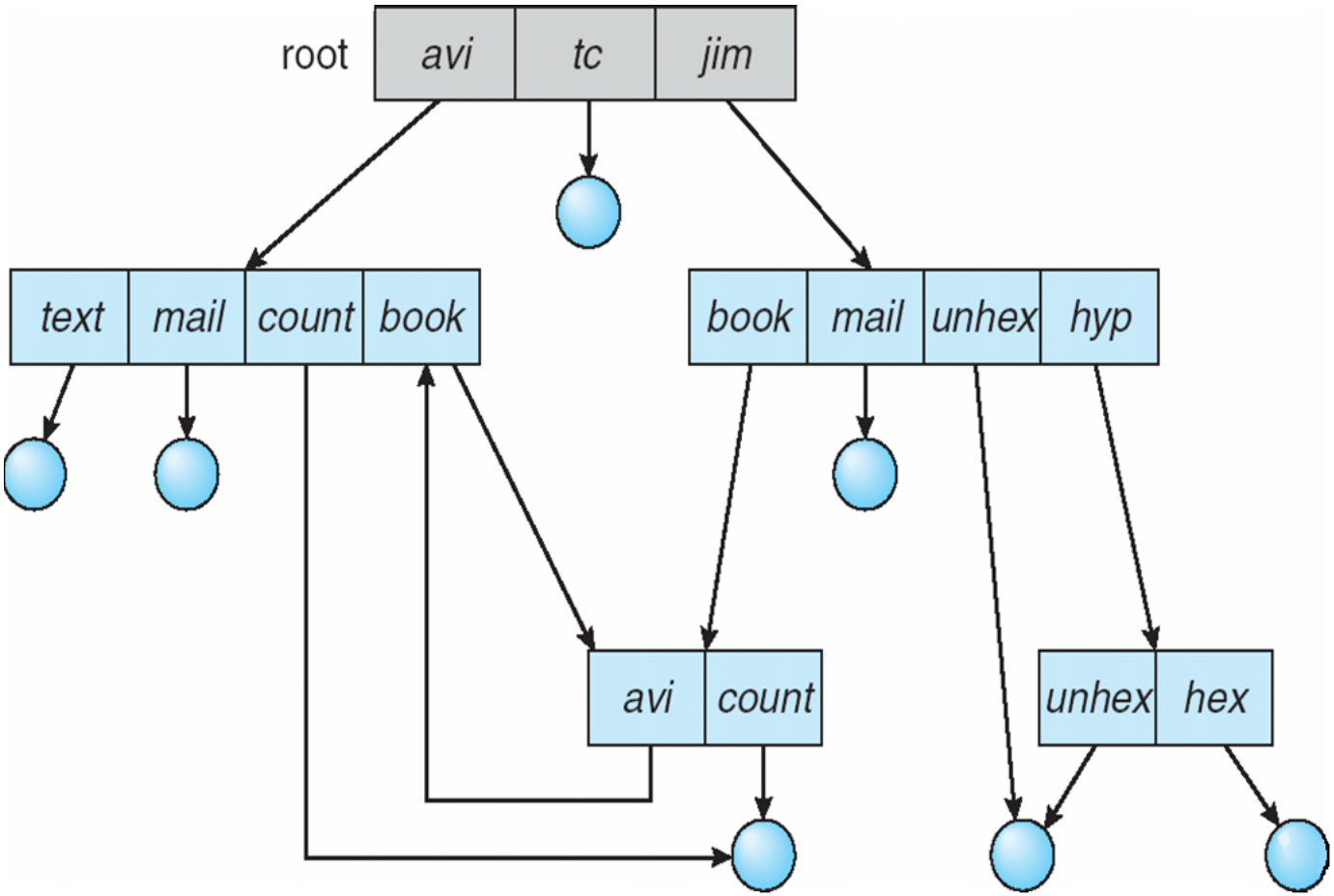
Acyclic-Graph Directories

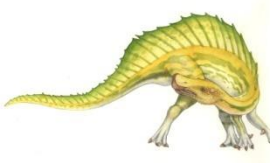


- Allow sharing of directories and files several times on the tree structure.
- Only once actual copy of the file or directory is stored.
- Can be accessed through one or more paths.



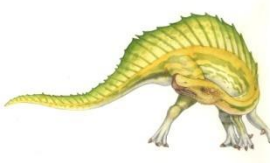
General Graph Directory (cont.)





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 it is OK



End of Chapter 10