Lecture Overview

- File systems
  - User concepts
    - File
    - Directory
      - Structure
    - Protection
  - File system structure
  - Implementing files
  - Implementing directories

File Systems

- File systems are very important
  - They provide online storage and access to both programs and data
  - For users, the file system is one of the most visible aspects of the operating system
- There are three essential requirements for a file system
  - It must be possible to store large amounts of information
  - Information must survive the termination of the process using it
  - Multiple processes must be able to access the information concurrently
- *First we examine the concepts of the user's view of a file system*

File Concept

- Contiguous logical address space
- Types
  - Data (e.g., numeric, character, binary)
  - Program
- Structure
  - None - sequence of words, bytes
  - Simple record structure
    - Lines, fixed length, variable length
  - Complex Structures
    - Formatted document
    - Relocatable load file

File Concept

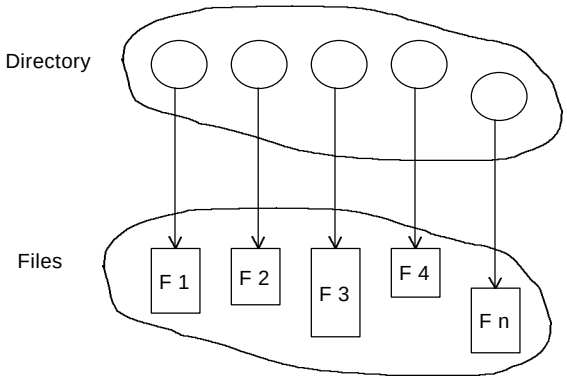
- File attributes
  - **Name** – only information kept in human-readable form
  - **Type** – needed for systems that support different types
  - **Location** – pointer to file location on device
  - **Size** – current file size
  - **Protection** – controls who can do reading, writing, executing
  - **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 Concept

- File operations
  - *create*
  - *write*
  - *read*
  - *reposition* - move current read/write position within file
  - *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

Directory Concept

- A list of files and metadata about files
  - Directories special files that are stored on the disk also

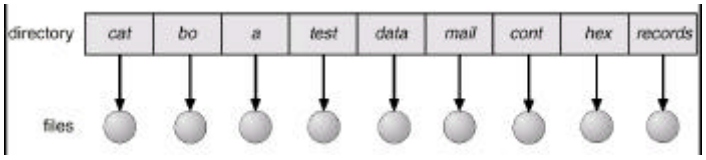


Directory Concept

- Directory operations
  - Search for a file
  - Create a file
  - Delete a file
  - List a directory
  - Rename a file
  - Traverse the file system
- Directories are used to provide logical organization

Directory Concept

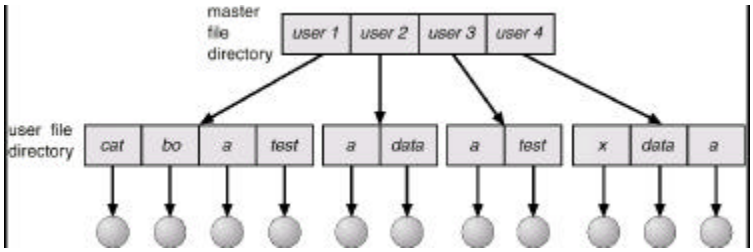
- Logical directory organization
  - Single-level directory



  - Naming problems
  - Grouping problems

Directory Concept

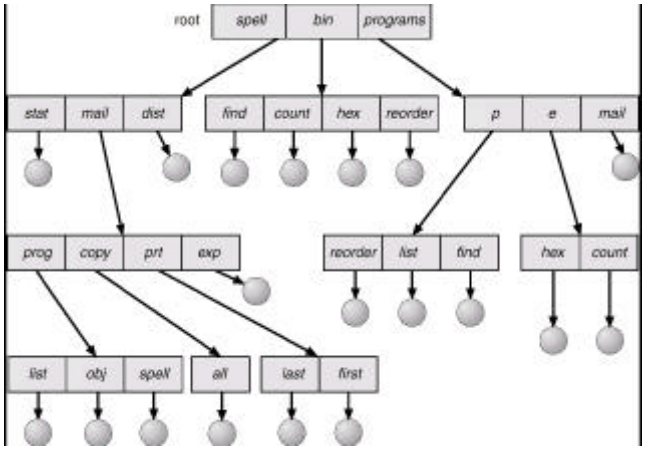
- Logical directory organization
  - Two-level directory (a directory for each user)



- Introduces path name
- Can have the same file name for different user
- No grouping capability

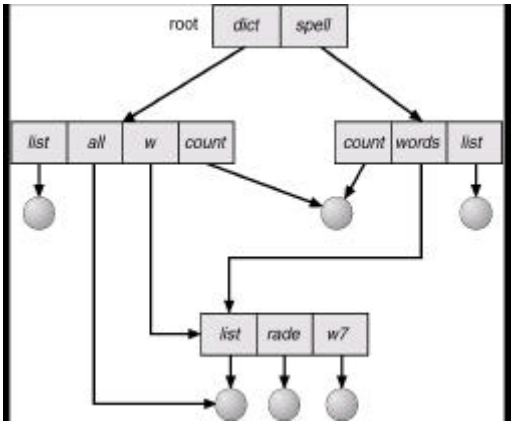
Directory Concept

- Logical directory organization
  - Tree-structured directory



Directory Concept

- Logical directory organization
  - Acyclic-graph directory (File and directory sharing)



Directory Concept

- Logical directory organization
  - Acyclic-graph directory (File and directory sharing)
    - Use linking
      - Hard links - update reference count in directory data
      - Soft links (symbolic links) -
    - Two different names for same file (aliasing)
    - Can create dangling pointers

File System Protection

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

File System Protection

- Mode of access: read, write, execute
- Three classes of users
  - Owner, group, and public (i.e., everyone else)
- Read/write bit approach

	RWX		
a) owner access	7	⇒	1 1 1
	RWX		
b) groups access	6	⇒	1 1 0
	RWX		
c) public access	1	⇒	0 0 1

- Access Control List (ACL) approach
  - Similar to above, but more flexible -- no practical limit on classes of users

File System Structure

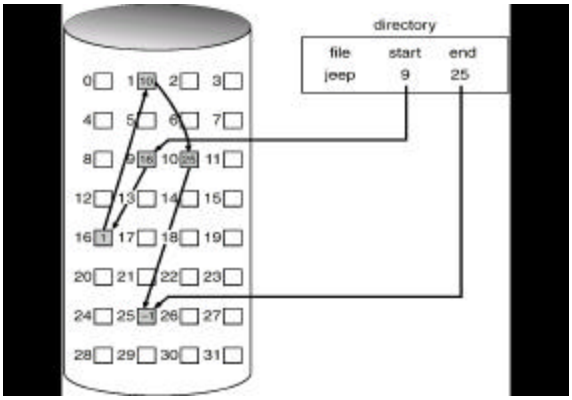
- Now that we have seen how the file system looks to the user, how is this logical view mapped to the physical storage device?
- General disk organization
  - Most disks are divided into partitions
    - One partition is marked as “active”
  - Sector zero of the disk is called the *master boot record (MBR)* which is used to boot the computer
    - The end of the MBR contains the partition table
    - When booting the BIOS reads and executes the MBR
      - The MBR program locates the active partition and reads its first block, called the *boot block*, and executes it (this program loads the operating system)

Implementing Files

- Contiguous allocation
  - Each file occupies a set of contiguous blocks on the disk
  - Simple – only starting location (block #) and length (number of blocks) are required
  - Random access
  - Wasteful of space (dynamic storage-allocation problem)
  - Files cannot grow
  - Mapping from logical to physical

Implementing Files

- Linked allocation
  - Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk

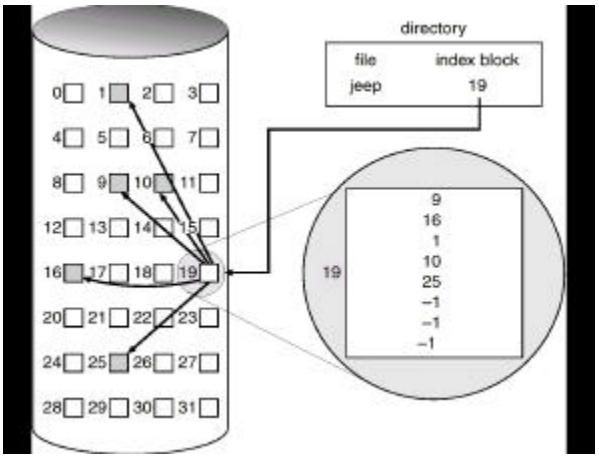


Implementing Files

- Linked allocation
  - Simple – need only starting address
  - Free-space management system – no waste of space
  - No random access
  - Mapping
  - *File-allocation table (FAT)*
    - Variation of linked allocation
    - Links are kept in a table at beginning of partition
    - Disk-space allocation used by MS-DOS and OS/2

Implementing Files

- Indexed allocation
  - Bring all pointers together into the *index node (i-node)*



Implementing Files

- Free space management
  - Maintain a free space list
  - Approaches
    - Bit vectors - bit flag for each block
    - Linked list - a list of free blocks
    - Grouped linked list - a list of groups of contiguous free blocks of the same size
    - Counting - a list of groups of contiguous free blocks where the size varies according to usage

Implementing Directories

- Linear list of file names with pointer to the data blocks
  - Simple to program
  - Time-consuming to execute
- Hash Table - linear list with hash data structure
  - Decreases directory search time
  - *Collisions* - situations where two file names hash to the same location
  - Fixed size