11. Implementing File Systems

[ECE30021/ITP30002] Operating Systems

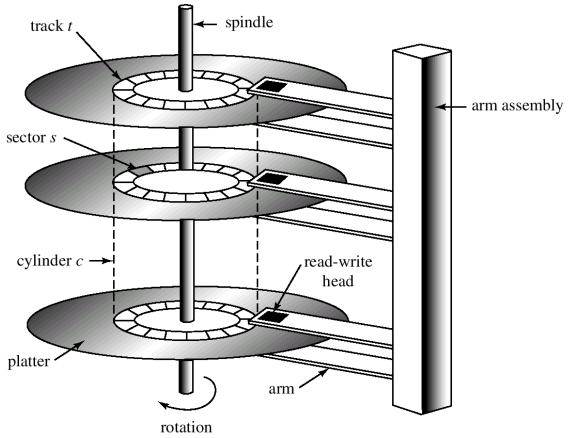
Agenda

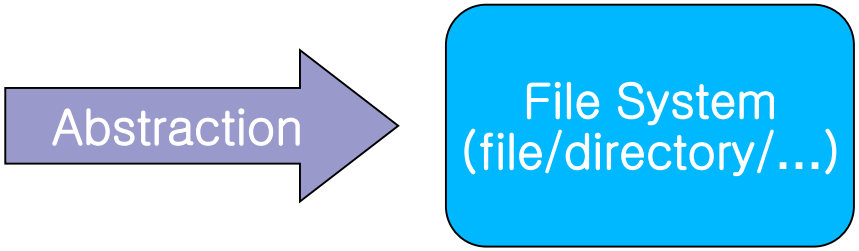
- File Concepts
- File System Mounting
- File System Structure
- File System Implementation
- Directory Implementation
- Allocation Methods
- Free Space Management

File System

- File system: a method for storing and organizing computer files and the data they contain to make it easy to find and access them.
  - File / directory structures

  Ex) ISO-9660(CDROM), UFS (UNIX), Ext2/Ext3/Ext4(Linux), FAT/FAT32/exFAT(Windows), NTFS (Windows)





File

- File: a named collection of related information that is recorded on secondary storage
  - Smallest allotment of logical secondary storage

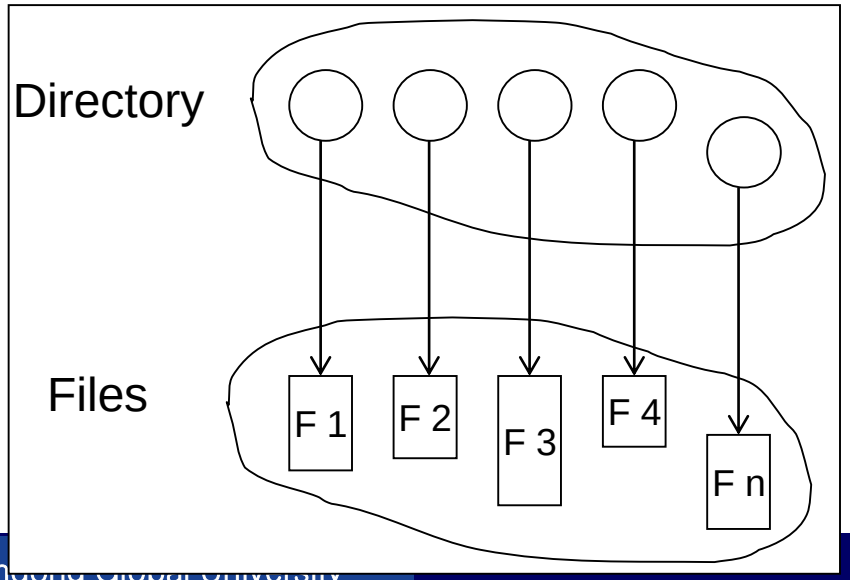
  cf) Physical HDD's are composed of platter, track, sector,

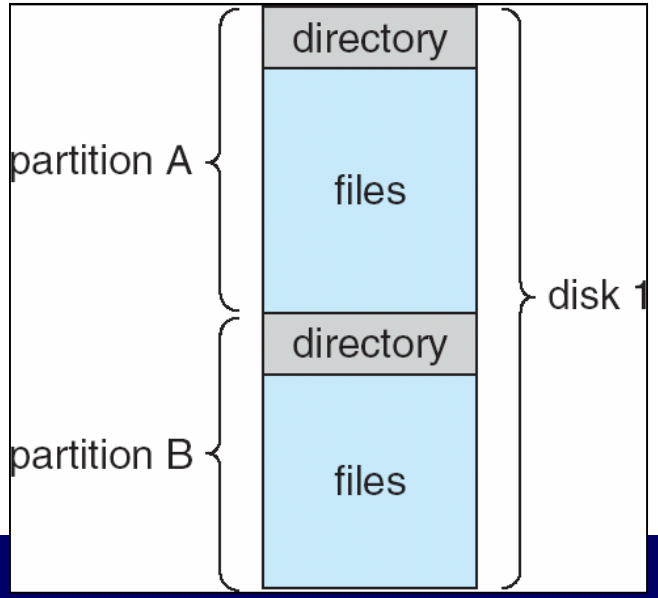
- Attributes of a file
  - Name, identifier, type, location, size, protection, time, date, userid, …

- Operations on a file
  - Create/write/read/repositioning/deletion/truncation/…

Directory

- Directory: an entity in a file system which contains a group of files and/or other directories.
  - Directory organizes and provides information about all files in system
  - Directory records information for files
    - Name, location, size, type, …
  - Some OS's treats a directory the same as a file (UNIX)



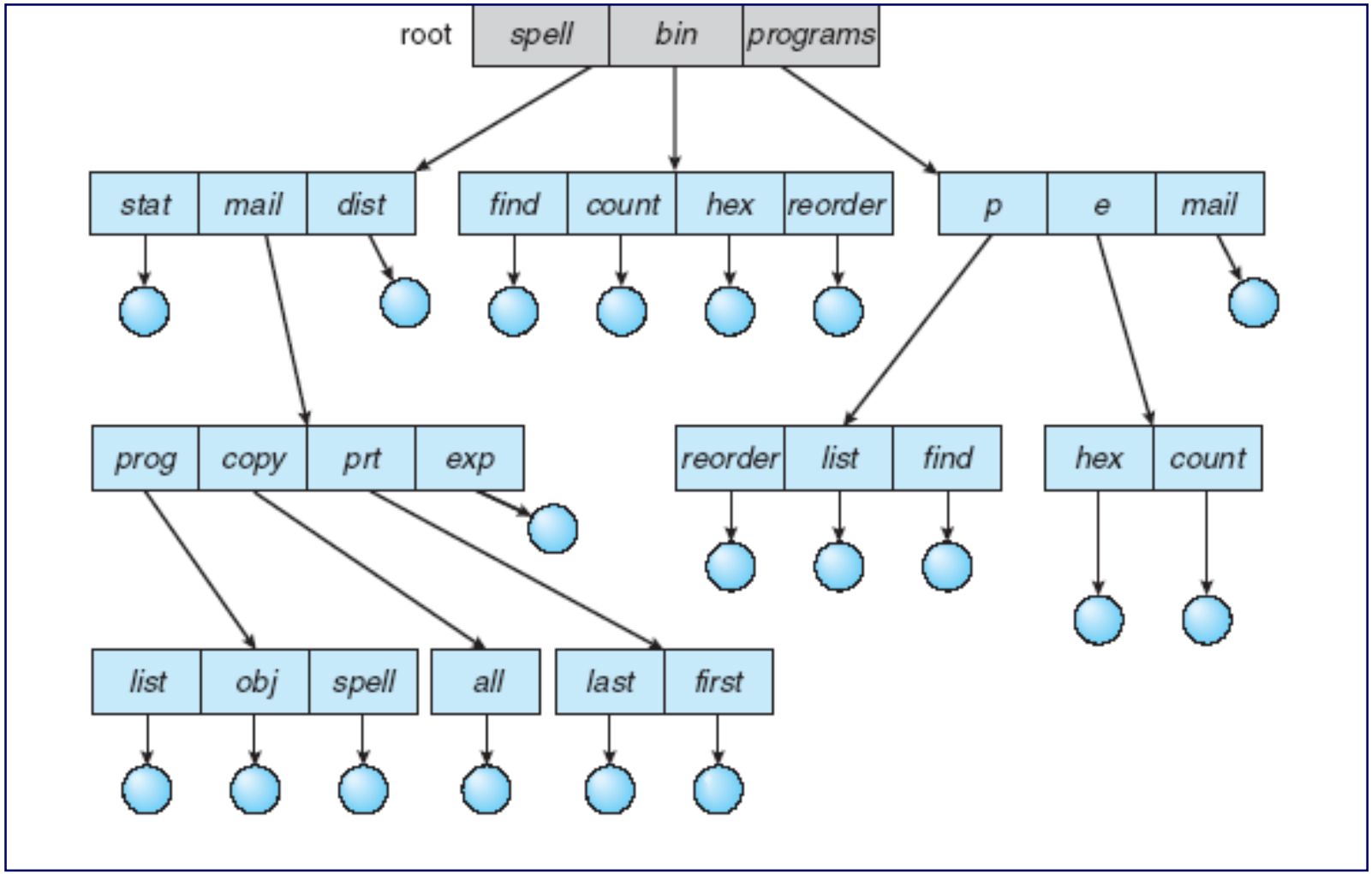


File Open/Close

- Motivation
  - File operation requires searching for entry for the file
  - To avoid search, many system requires open() system call be made before a file is first used.
    - open() searches the entry for the file and creates a corresponding entry in open-file table.
    - Thereafter, file operations are performed through the pointer to the entry in open-file table.

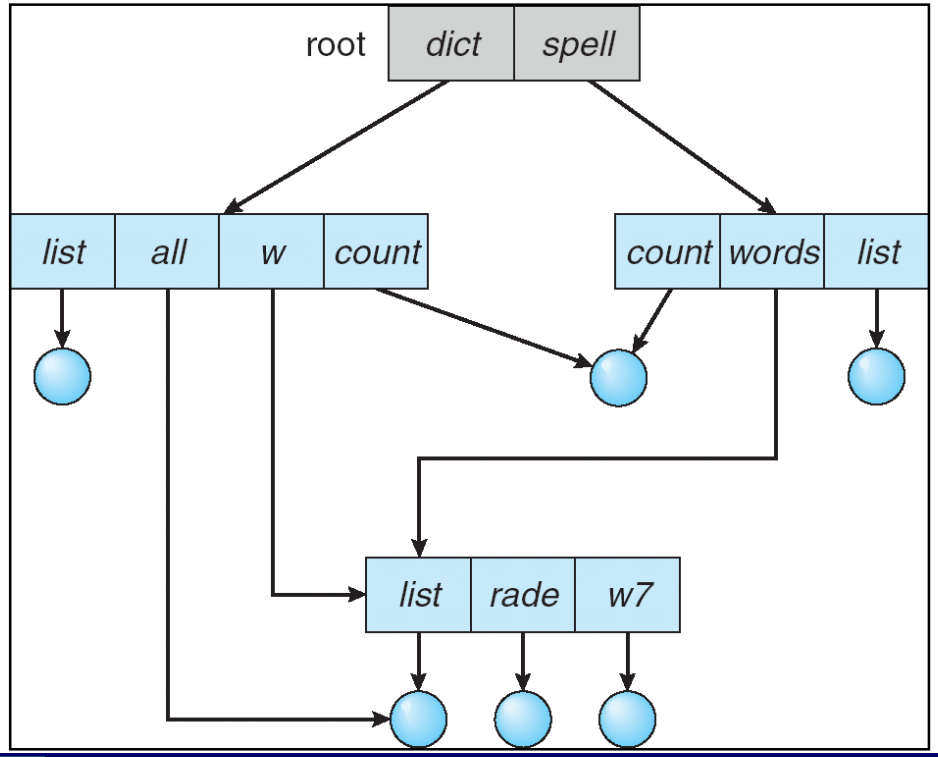
- Open-file table: a small table managed by OS containing information about open files
  - open() puts target file into open-file table
  - close() removes the entry from open-file table

Tree-Structured Directories



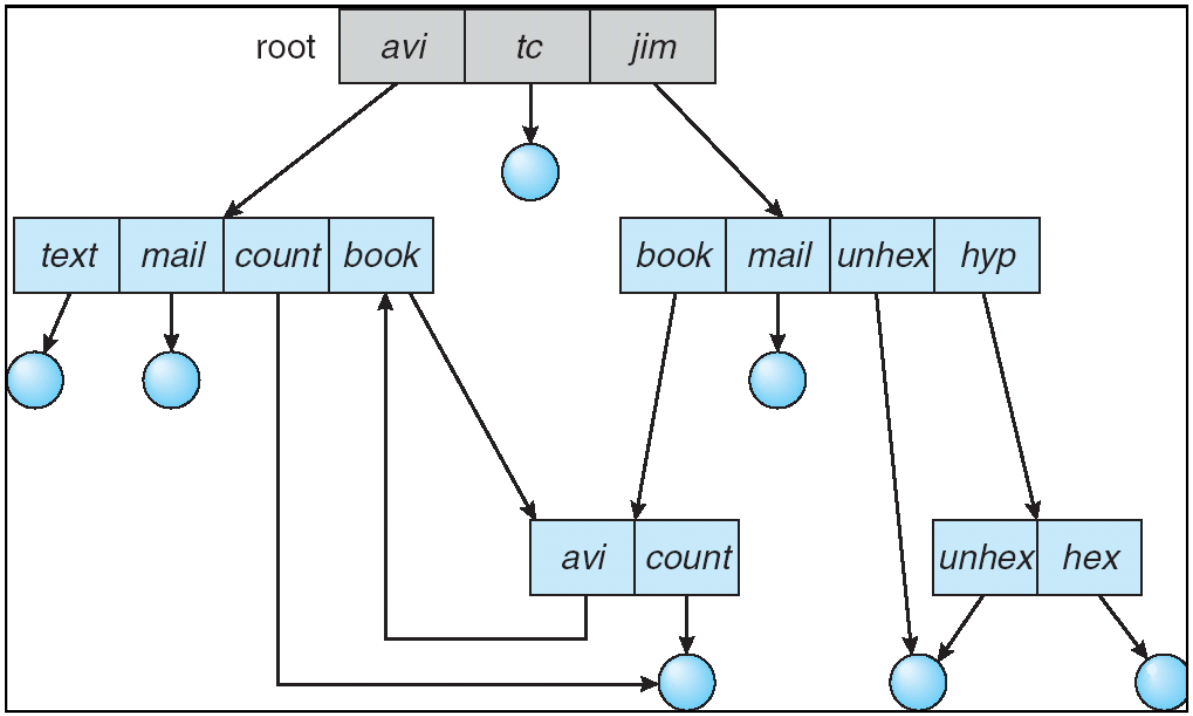
Acyclic-Graph Directories

- Subdirectories and files can be shared
  - Usually implemented by link (a pointer to another file or directory)
  - Cycle is not allowed



General Graph Directories

- Problems caused by cycles
  - A poor traversal algorithm can cause infinite loop
  - Deletion of self-referencing files/directory



Agenda

- File Concepts
- File System Mounting
- File System Structure
- File System Implementation
- Directory Implementation
- Allocation Methods
- Free Space Management

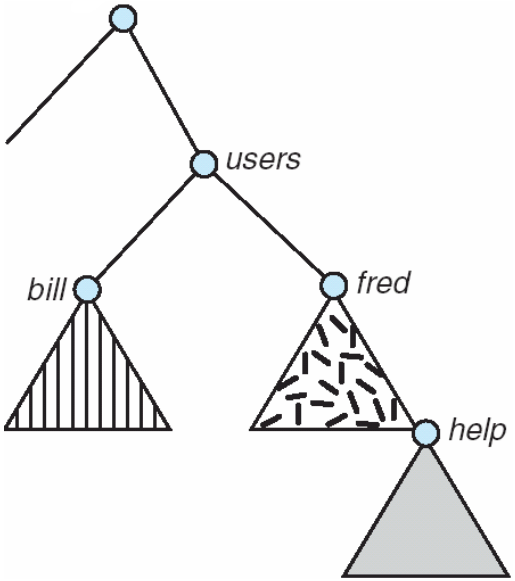
File-System Mounting

- Mount: making a file system ready for use by the OS, typically by reading certain index data structures from storage into memory ahead of time
  - A file system must be mounted before it can be available to the processes.
  - In UNIX, mount attaches the file system found on some device to the big file tree
  - In Windows, mounted volume is associated with the drive letter
- Mount point: location within the file structure where the file system is to be attached
  - Typically, mount point is an empty directory

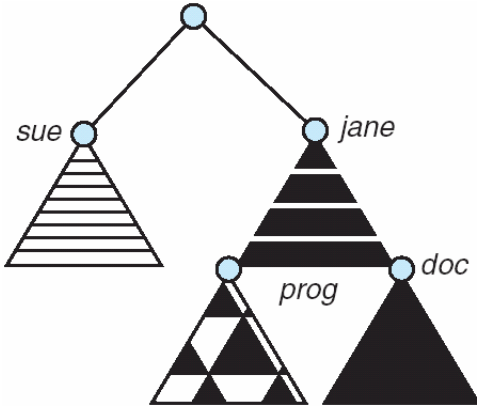
    Ex) home directories are usually mounted as /home

File-System Mounting

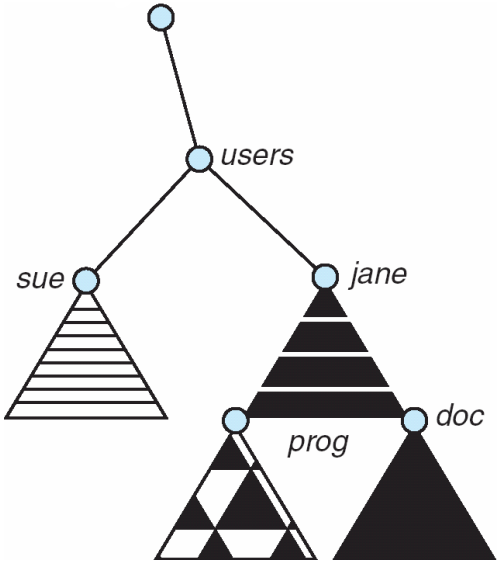
- An example of mounting



A file system



Unmounted volume



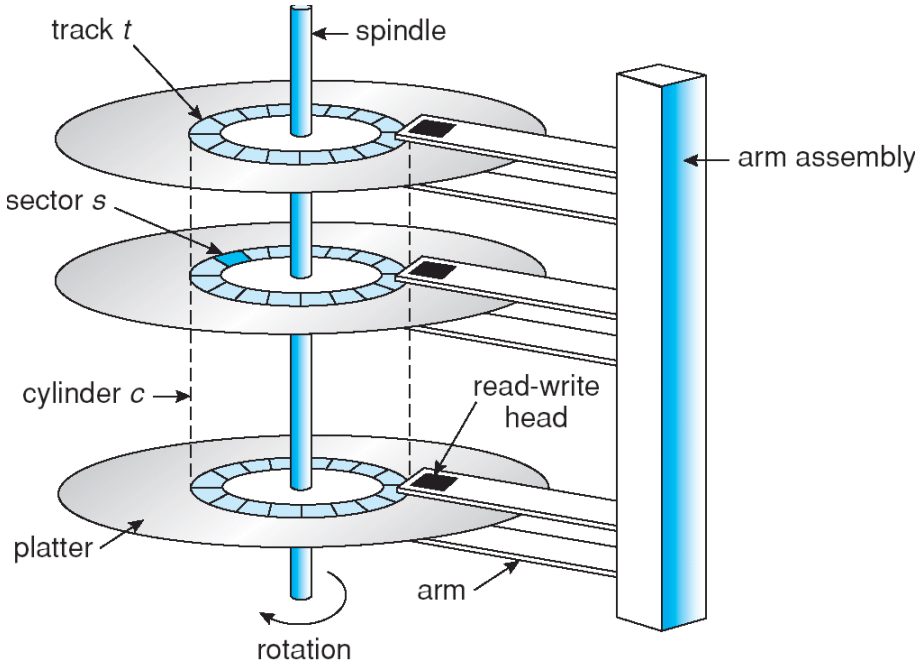
Mount point

Agenda

- File Concepts
- File System Mounting
- File System Structure
- File System Implementation
- Directory Implementation
- Allocation Methods
- Free Space Management

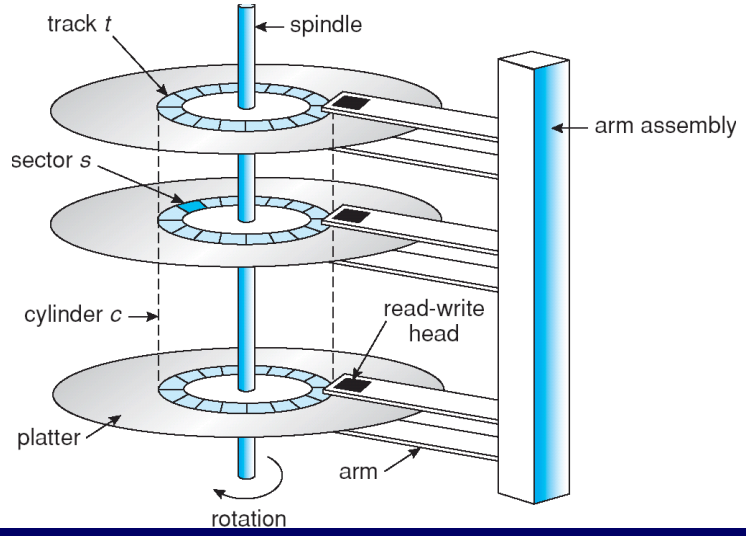
Disk Structure

- Disk structure
  - Platter – head – track – sector
  - Cylinder: set of tracks that are at one arm position



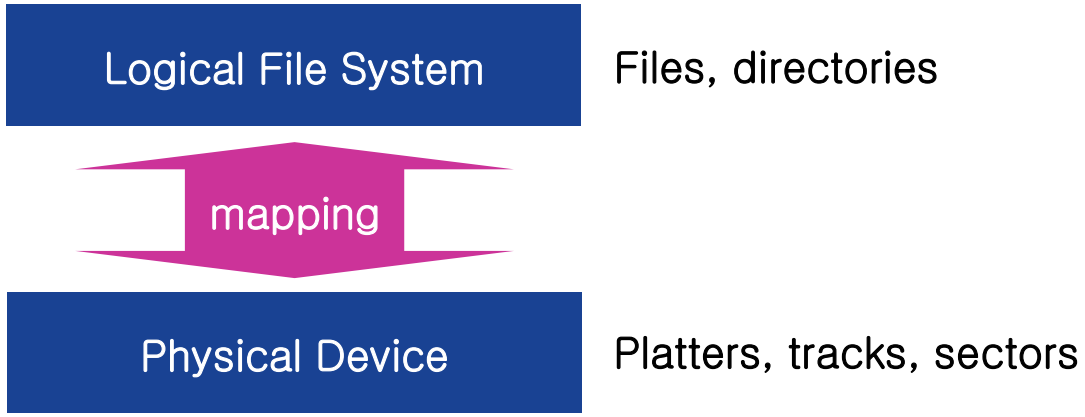
Properties of Disk

- Disks provide the bulk of secondary storage with the following properties
  - A disk can be rewritten in place
  - I/O transfers are performed in units of blocks
    - A block is composed of one or more sectors
      - 32~4096 bytes, usually, 512 bytes
  - A disk can access directly any given block of information



Issues of File System Design

- Defining how the file system look to the user
  - File and its attributes
  - Operations allowed on a file
  - Directory structure
- Creating algorithms and data structures to map logical file system onto the physical secondary-storage devices

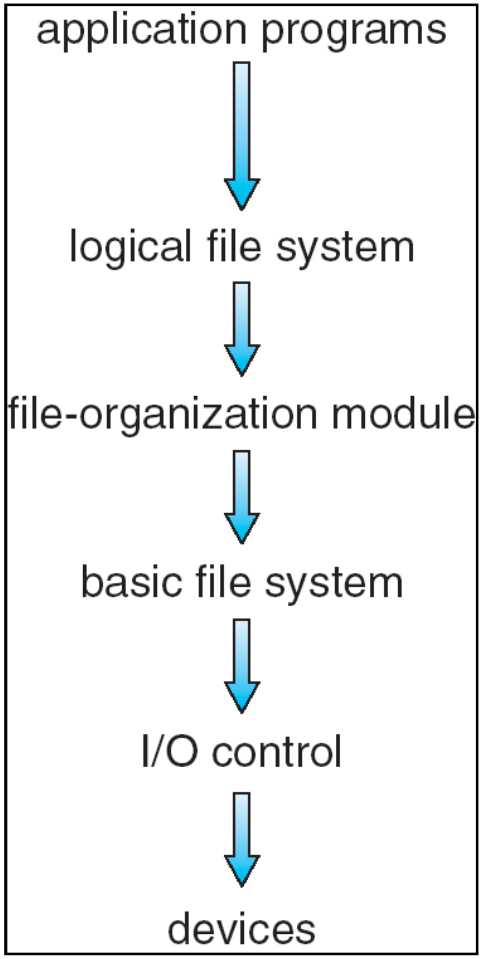


Layered File System

- File system is generally composed of many different levels
- I/O control: device drivers + interrupt handlers
  - Device driver: translator

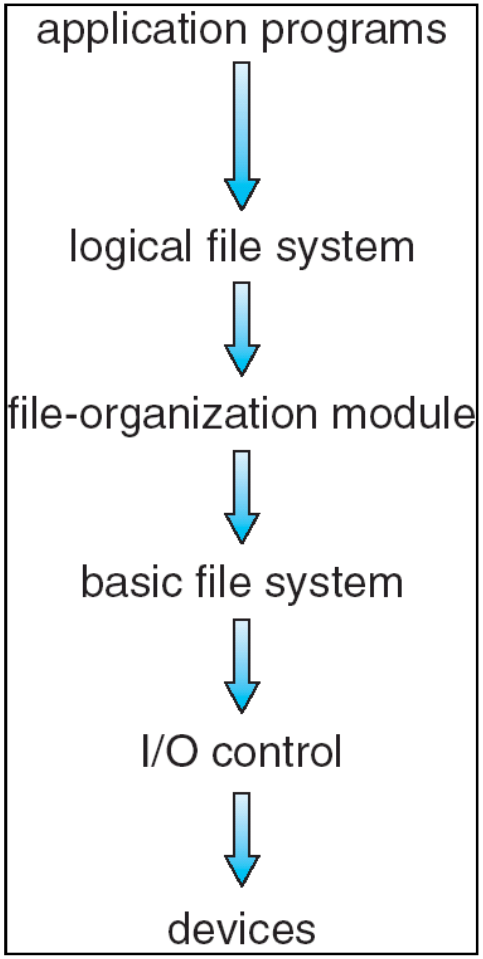
  Ex) High-level command “retrieve block 123”

  ➔ H/W specific instruction to access I/O controller’s memory
- Basic file system: issues generic command to device driver to read/write physical blocks
  - Interfaces to the device drivers for block level transfer routines
  - Manages the memory buffers and caches that hold various file-system, directory, and data blocks.

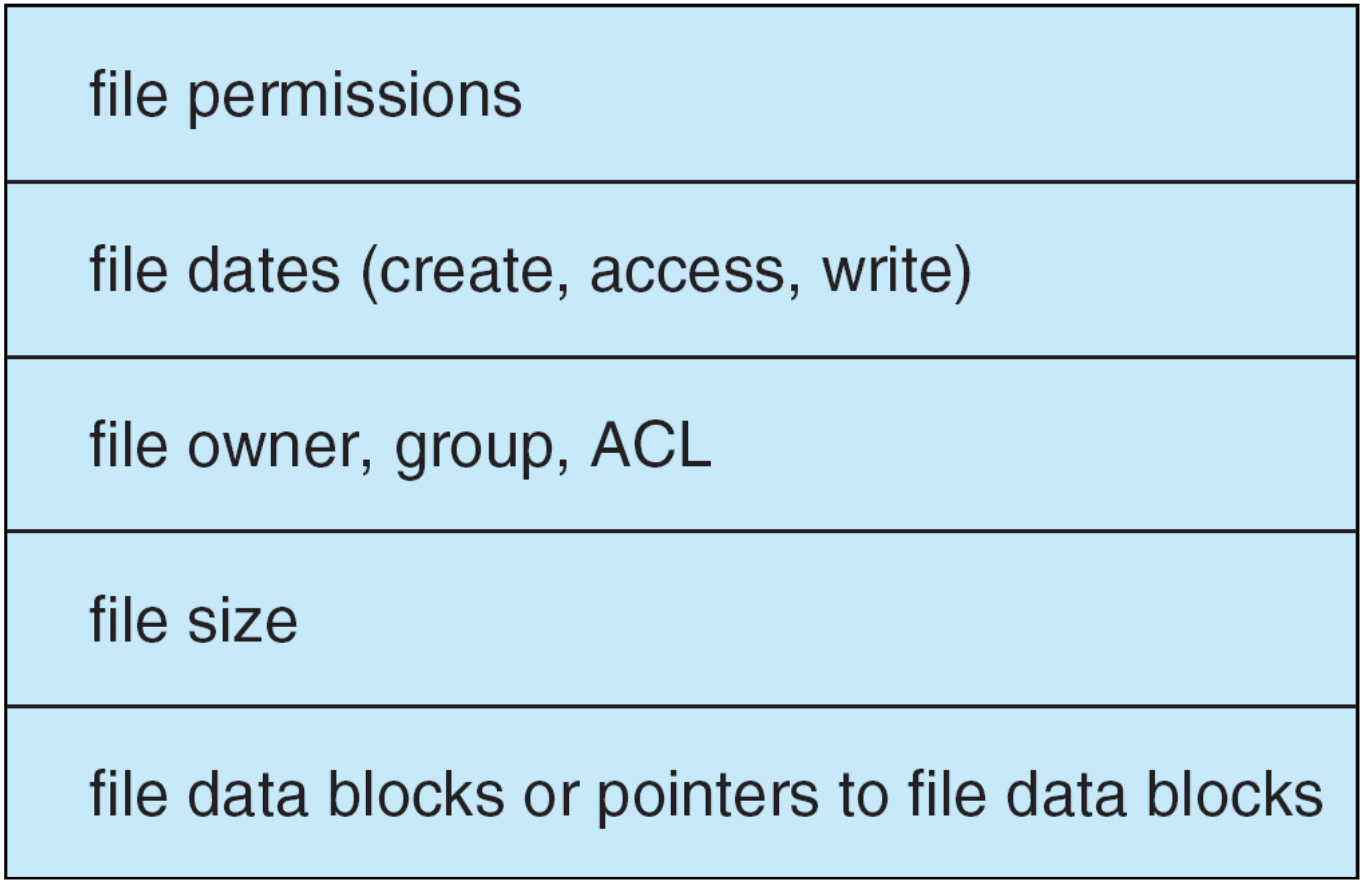


Layered File System

- File-organization module
  - Mapping from logical block address of a file to physical block address
  - Free-space manager
- Logical file system
  - Manages directory structure
    - File-control block (FCB) keeps information about file
      - Ownership, permission, location of file contents
  - Manages metadata information
    - Metadata: all file-system structure except the actual data



A Typical FCB



Agenda

- File Concepts
- File System Mounting
- File System Structure
- File System Implementation
- Directory Implementation
- Allocation Methods
- Free Space Management

Information Stored on Disk

- Boot control block (per volume): information need to boot an OS

  Ex) Boot block (UFS), partition boot sector (NTFS)
- Volume control block (per volume)
  - # and size of blocks, free block count/pointer, free FCB count/pointer

  Ex) Superblock (UFS), stored in MFT (NTFS)
- Directory structure (per file system)

  Ex) filename – inode(UFS), MFT(NTFS)
- FCB (per file): detail information about each file
  - Permission, ownership, size, location of data blocks, …

  Ex) inode (UFS), rows of MFT

Master File Table

- Master File Table (MFT): The place where information about every file and directory on an NTFS volume is stored
  - MFT is a relational database containing various attributes about different files.
  - MFT acts as the starting point and central management feature of an NTFS volume
    - Table of contents for the volume

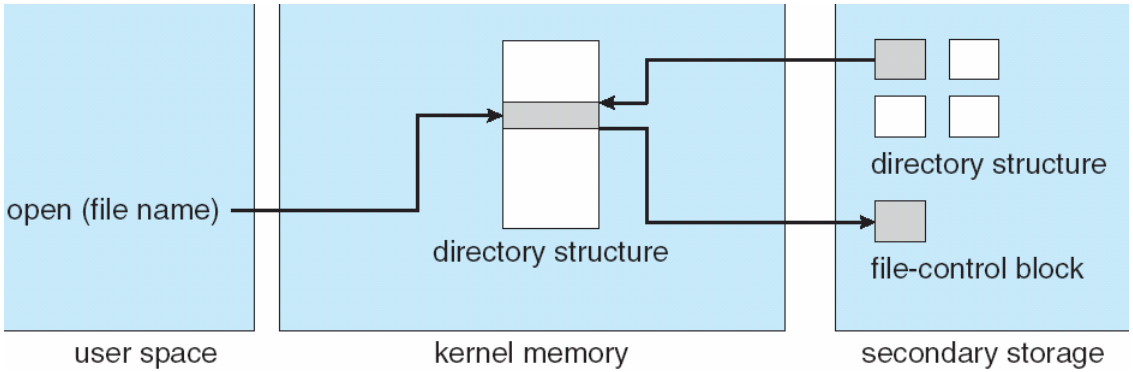
Information Stored In Memory

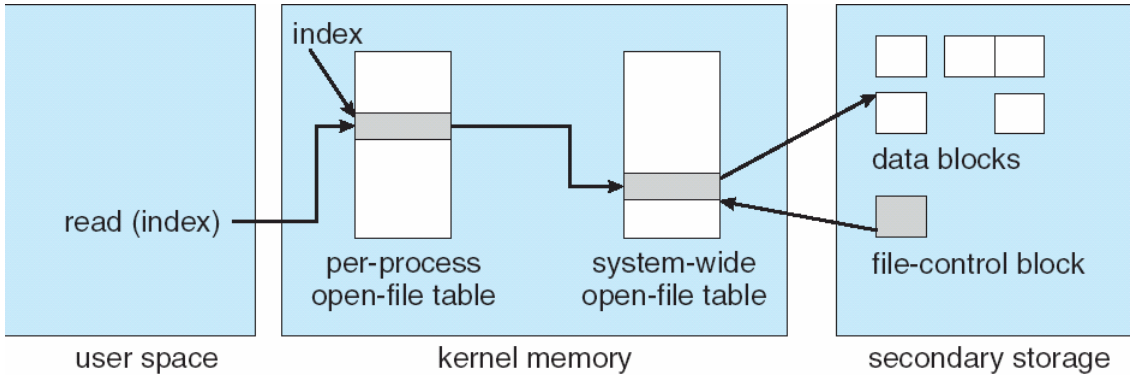
- In-memory mount table
- Cache of directory structure: directory information for recently accessed directories
- System-wide open-file table
  - Copy of <FCB of each open file> + <additional info.>
- Per-process open-file table
  - <Pointer to appropriate entry in system-wide open-file table> + <additional info.>

File Open

- open() system call
  1. Search directory structure.
     - Usually, directory structure is cached in memory
  2. FCB is copied in system-wide open-file table
     - FCB + <# of processes that have the file open>
  3. An entry is made in per-process open-file table
     - Pointer to the entry in system-wide open-file table + other fields
  4. Returns pointer to per-process open-file table
     - All file operations are performed via this pointer
     - File descriptor (UNIX), file handle (Windows)
  - If system-wide open-file table already the entry of the target file, 1 and are skipped.
- close() system call
  - Per-process table entry is removed
  - System-wide entry's open count is decreased
    - If it becomes zero, the entry is removed

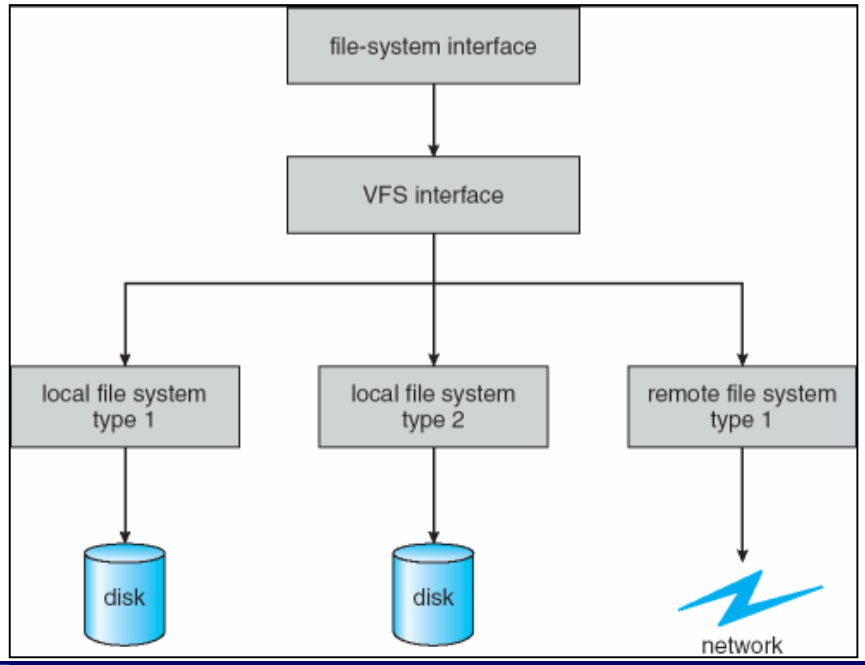
File-Open Table





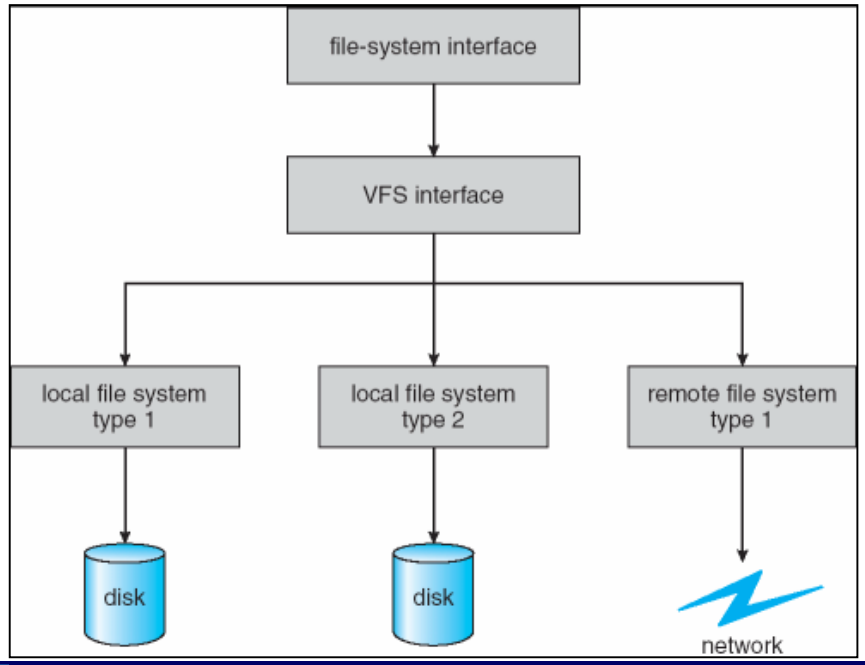
Virtual File Systems

- Modern OS's concurrently support multiple types of file systems
- Object-oriented technique
  - Data structures and procedures are used to isolate the basic system call functionality from implementation details



Virtual File Systems

- File system implementation consists of three major layers
  - File-system interface: open(), read(), write(), close(), …
  - Virtual file system (VFS)
    - Separates file-system-generic operations from their implementation
    - Files are represented by vnode
  - Local file system



Virtual File Systems

- Main object types in Linux VFS
  - inode object: for individual file
  - File object: for open file
  - Superblock object: for entire file system
  - Dentry object: for directory entry
- Every object has pointer to a function table
  - int open(…)
  - int close(…)
  - ssize_t read(…)
  - ssize_t write(…)
  - int mmap(…) – memory-map a file

Directory Implementation

- Linear list of file names with pointer to the data blocks.
  - Simple to program
  - Time-consuming to execute
    - Many OS's implement a software cache
- Hash Table – linear list with hash data structure.
  - Decreases directory search time
  - Problems
    - Collisions – situations where two file names hash to the same location
    - Fixed size
  - Remedies: linear probing / chained overflow hashing

Agenda

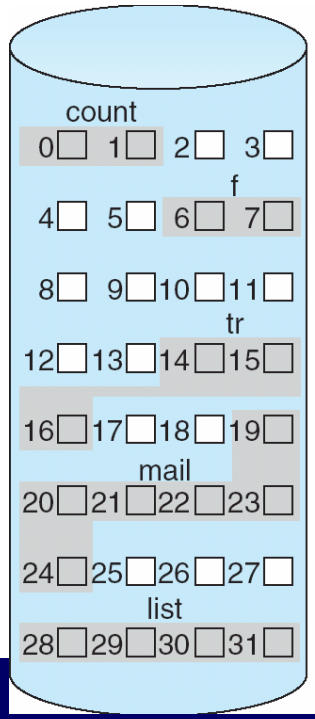
- File Concepts
- File System Mounting
- File System Structure
- File System Implementation
- Directory Implementation
- **Allocation Methods**
- Free Space Management

Allocation Methods

- How to allocate space to files?
  - Contiguous allocation
  - Linked allocation
  - Indexed allocation

Contiguous Allocation

- Each file occupy a set of contiguous blocks on 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

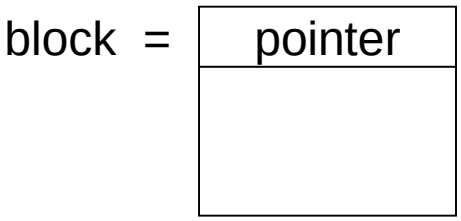


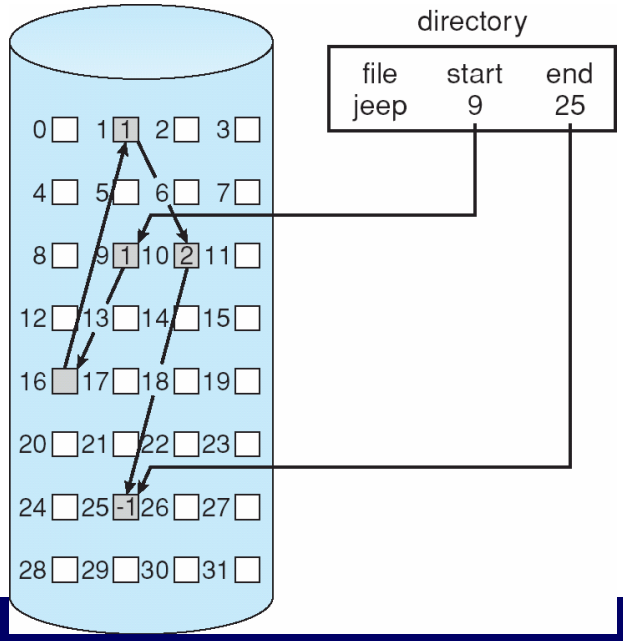
directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Linked Allocation

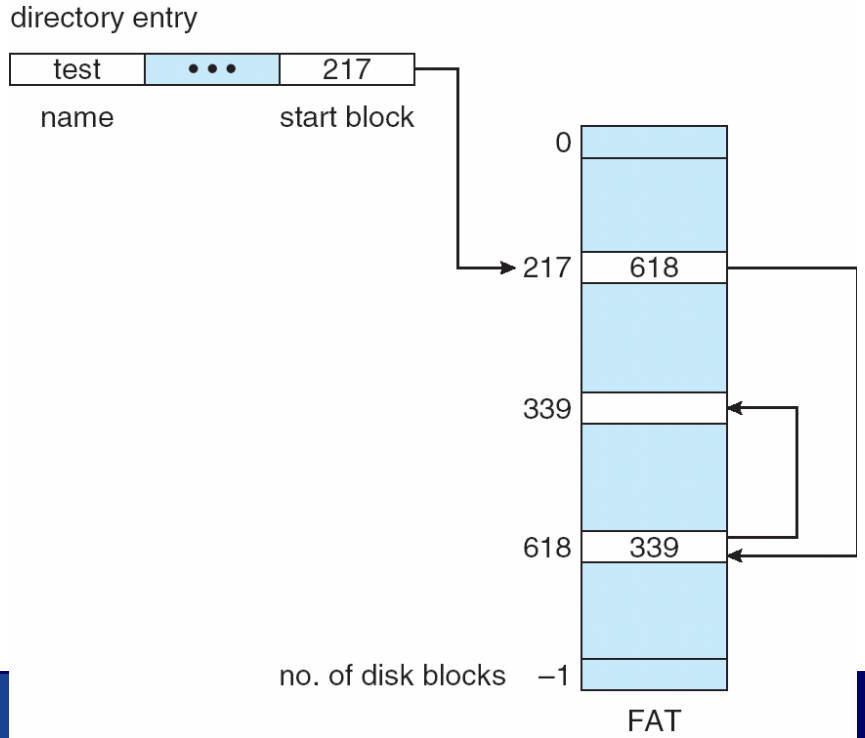
- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.
  - Simple – need only starting address
  - Free-space management system – no waste of space
  - Random access is inefficient





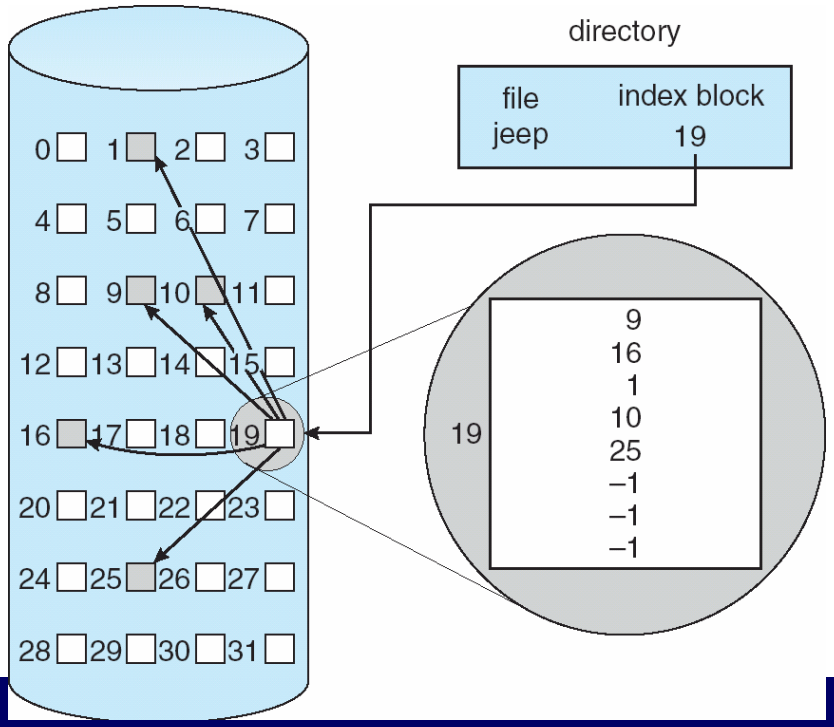
Linked Allocation

- Linked allocation using FAT (file-allocation table)
  - A section of disk is set aside to contain FAT
  - One entry for each block, containing next pointer (index)
  - Directory entry contains index of starting block of the file
  - Unused blocks are indicated by 0 value



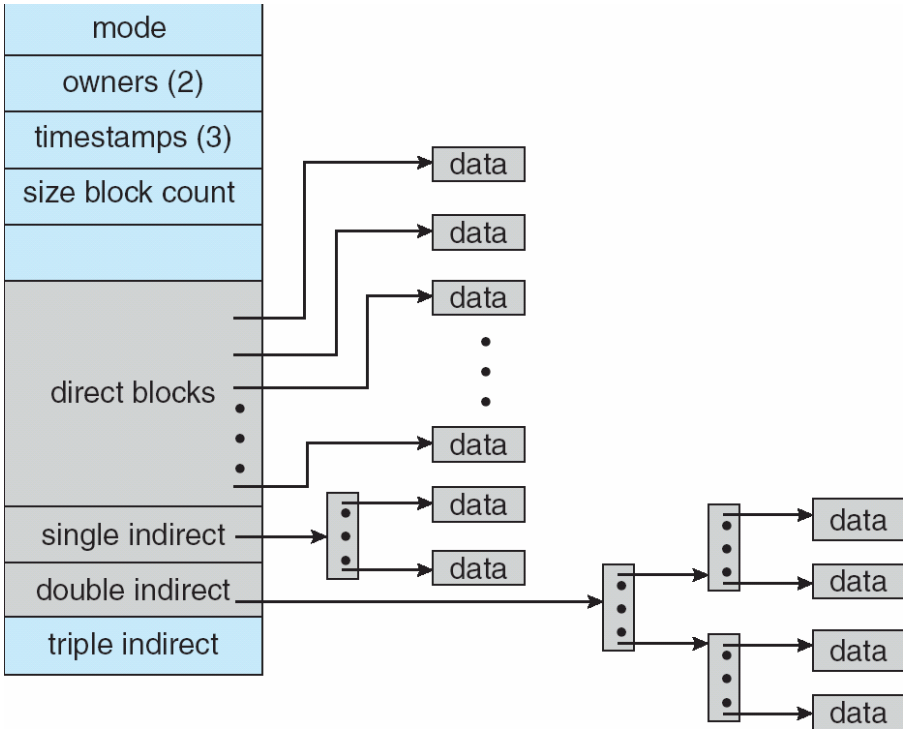
Indexed Allocation

- Brings all pointers together into the index block.
  - Each file has its own index block
  - Random access
  - Dynamic access without external fragmentation, but have overhead of index block.



Indexed Allocation

- Allocation of index block
  - Size of index block is variable
  - Even if the file use only a few block, the entire index block should be allocated
- Allocation methods
  - Linked scheme
  - Multilevel index
  - Combined scheme



UNIX inode (combined scheme)

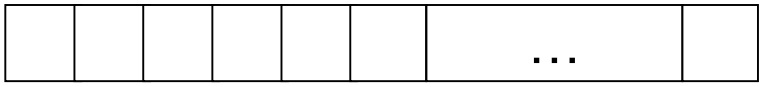
Agenda

- File Concepts
- File System Mounting
- File System Structure
- File System Implementation
- Directory Implementation
- Allocation Methods
- Free Space Management

Free-Space Management

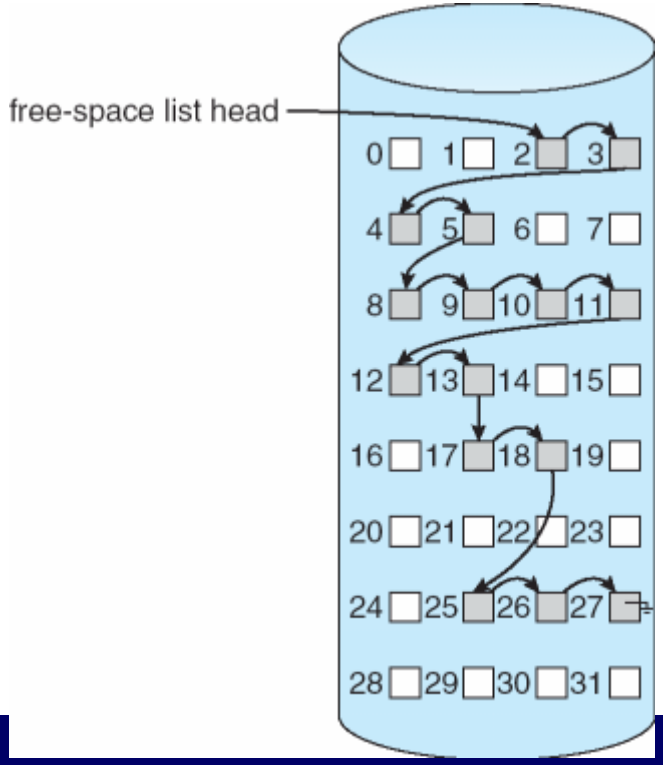
- OS maintains free-space list to record all free disk blocks

- Bit vector



$$bit[i] = \begin{cases} 0 \Rightarrow block[i] \text{ free} \\ 1 \Rightarrow block[i] \text{ occupied} \end{cases}$$

- Linked list



Free-Space Management

- Grouping
  - The addresses of n free blocks are stored in the first free block
  - Last block contains the address of another n free blocks
- Counting
  - Keep address of first free block and number of free contiguous blocks
  - Effective when the blocks were allocated by contiguous-allocation algorithm