SCS 1214: Operating Systems

Dr. Dinuni Fernando, PhD

Based on Operating System Concepts by A.Silberschatz, P.Galvin, and G.Gagne

File Systems

Learning 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

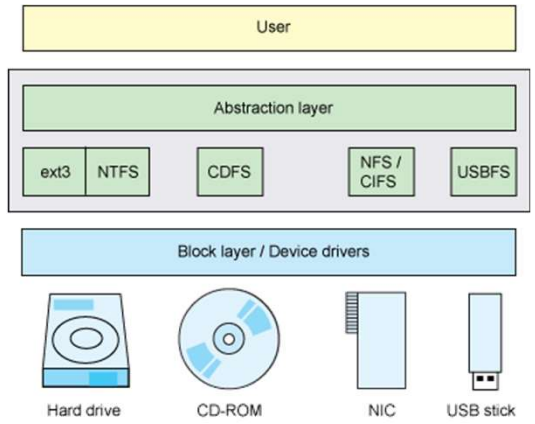
- Computers store information on various storage media
  - NVM devices, HDD, magnetic tapes, optical disks
- OS provides a uniform logical view of stored information.
  - OS abstracts from physical properties that defines logical storage unit File.
- File is a named collection of related information that is recorded on secondary storage.
- From user's perspective, a file is small allotment of logical storage.
  - Data cannot write to secondary storage unless they are within a file.

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**

Disks and the Operating system

- Disks are physical devices:
  - Errors, bad blocks, missed seeks, etc.
- The job of the OS is to hide this mess from higher level software
  - Low-level device control (initiate a disk read, etc.)
  - Higher-level abstractions (files, databases, etc.)
- The OS may provide different levels of disk access to different clients
  - Physical disk (surface, cylinder, sector)
  - Logical disk (disk block #)
  - Logical file (file block, record, or byte #)



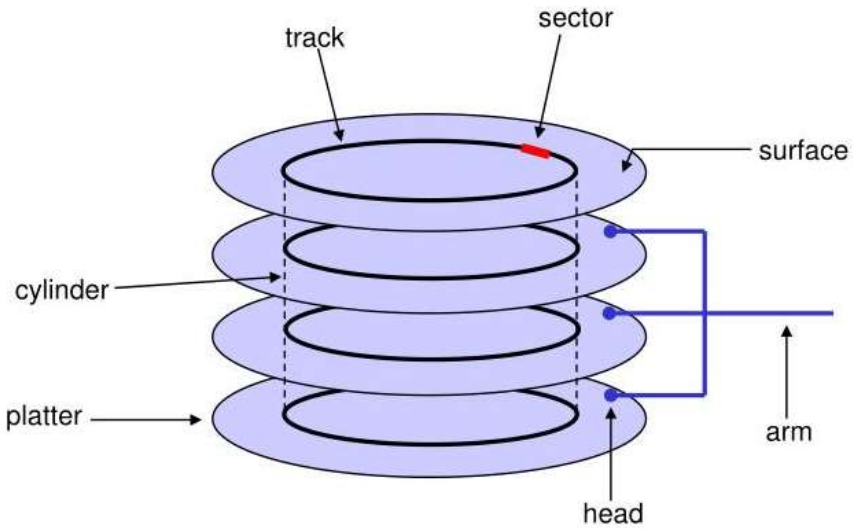
Disks basics

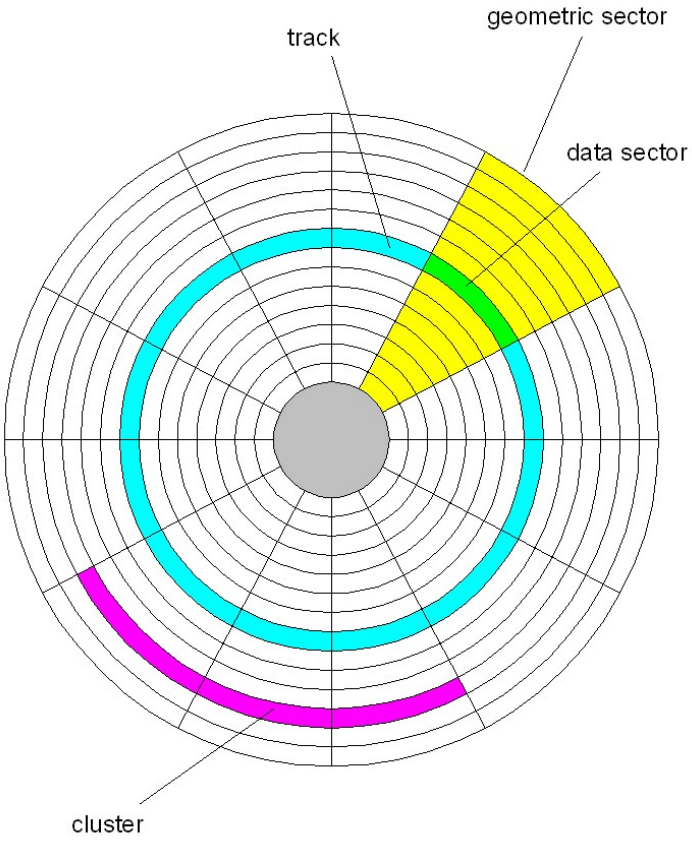
- Main memory is expensive, volatile.
- Unit of disk storage and retrieval – disk block or page
  - Disk block/page is a contiguous sequence of bytes
  - Size is 4KB or 8KB

- Like RAM, disks support direct access to a page
- Unlike RAM, time to retrieve a page varies
  - Depends upon the location on disk
  - Relative placement of pages on disk has major impact on performance

Physical Disk Structure

- Disk components
  - Platters
  - Surfaces
  - Tracks
  - Sectors
  - Cylinders
  - Arms
  - Heads





Accessing a Disk Page

- Time to access (read/write) a disk block:
  - seek time (moving arms to position disk head on track)
  - rotational delay (waiting for block to rotate under head)
  - transfer time (actually moving data to/from disk surface)
- Seek time and rotational delay dominate.
  - Seek time varies from about 1 to 20msec
  - Rotational delay varies from 0 to 10msec
  - Transfer rate is less than 1msec for a 4KB page
- Key to lower I/O cost: reduce seek/rotation
  - delays!

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**
  - **Need to find space in the FS**
  - **Create entry in the directory**
- **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

```
SYSCALL_DEFINE3(read, unsigned int, fd, char __user *, buf, size_t, count)
{
        struct fd f = fdget_pos(fd);
        ssize_t ret = -EBADF;

        if (f.file) {
                loff_t pos = file_pos_read(f.file);
                ret = vfs_read(f.file, buf, count, &pos);
                if (ret >= 0)
                        file_pos_write(f.file, pos);
                fdput_pos(f);
        }
        return ret;
}

SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,
                size_t, count)
{
        struct fd f = fdget_pos(fd);
        ssize_t ret = -EBADF;

        if (f.file) {
                loff_t pos = file_pos_read(f.file);
                ret = vfs_write(f.file, buf, count, &pos);
```

```
SYSCALL_DEFINE3(open, const char __user *, filename, int, flags, umode_t, mode)
{
        if (force_o_largefile())
                flags |= O_LARGEFILE;

        return do_sys_open(AT_FDCWD, filename, flags, mode);
```

Open Files

- Several pieces of data are needed to manage open files:
  - **Open-file table**: tracks 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
  - 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** – once a process acquires an exclusive lock OS will prevent other process from accessing the lock
  - **Advisory** – OS will not prevent process from accessing
    - if any write operation need to be made it should manually acquire the lock before accessing the file

File Types – Name, Extension

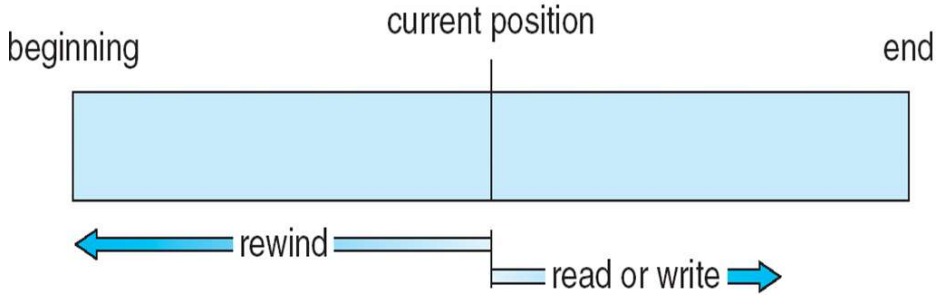
file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual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

- Indicate internal structure of a file
- Files must conform to a required structure understood by the operating system.
- Supporting multiple different file structures make operating system large and cumbersome.
- Some operating systems impose minimal number of file structures.
  - Eg: UNIX – consider each file to be a sequence of 8-bit bytes. This provides maximum flexibility but little support.

Access Methods

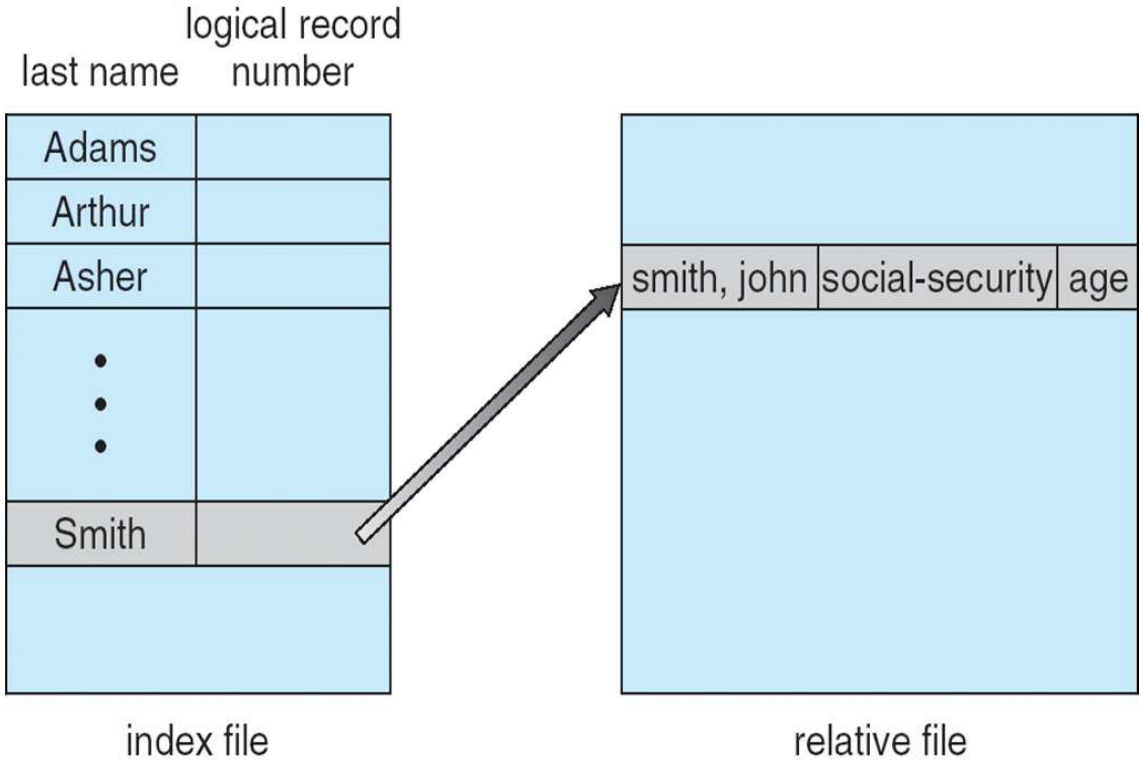
- Accessing helps to use information in a file.
- Access methods
  - Sequential Access
    - read next
    - write next
    - reset
    - no read after last write (rewrite)
  - Direct Access- file is fixed length logical records
    - Read n
    - Write n
    - Position to n , read next, write next
    - Rewrite n , #n is relative block number (Relative block numbers allow OS to decide where file should be placed)



Other Access Methods

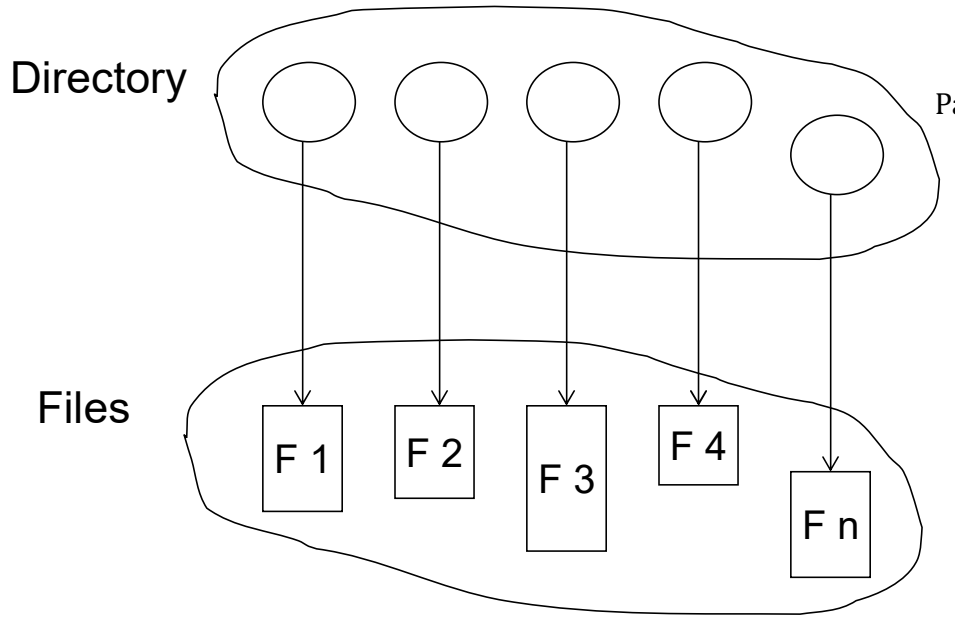
- Can be built on top of base methods
- General involve creation of an index for the file
- Keep index in memory for fast determination of location of data to be operated on
- If too large, index (in memory) of the index (on disk)
- IBM indexed sequential-access method (ISAM)
  - Small master index, points to disk blocks of secondary index
  - Secondary index blocks point to the actual file blocks.
  - File kept sorted on a defined key
  - To find a particular item using 2 direct access reads –
    - Make a binary search of the master index -> provides block number of the secondary index. Then binary search is used to find block containing desired record.

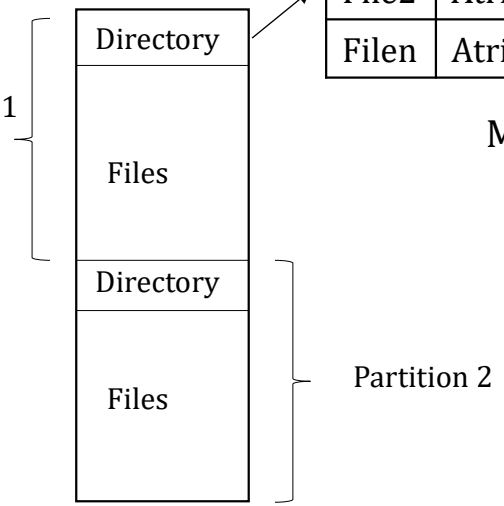
Example of Index and Relative Files



Directory Structure

- A collection of nodes containing information about all files
- Symbol table that translates file names into their file control blocks





File1	Atri1	Atri2	Atrin
File2	Atri1	Atri2	Atrin
Filen	Atri1	Atri2	Atrin

Metadata

Both the directory structure and the files reside on disk

Operations Performed on a Directory

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

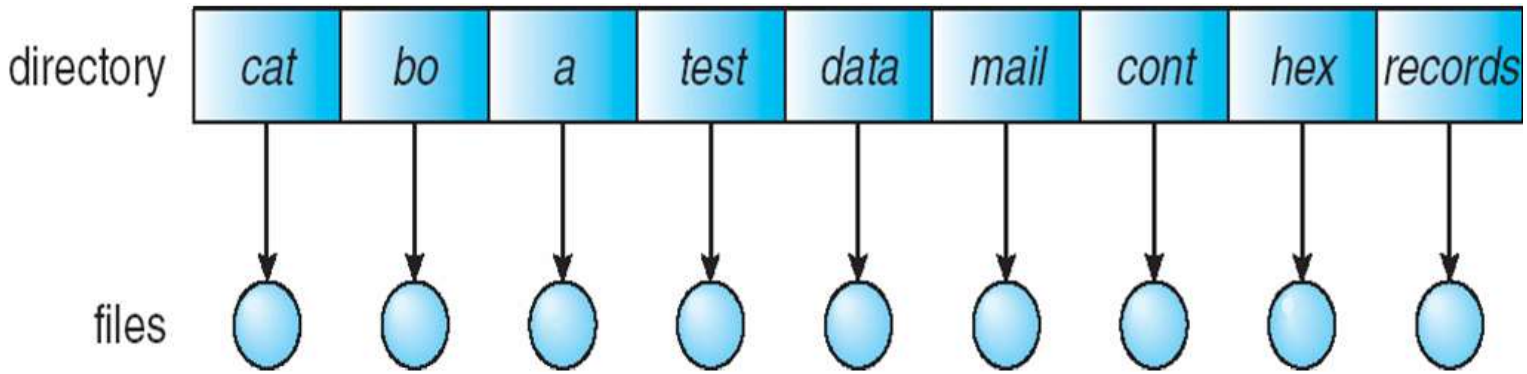
Directory Organization

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

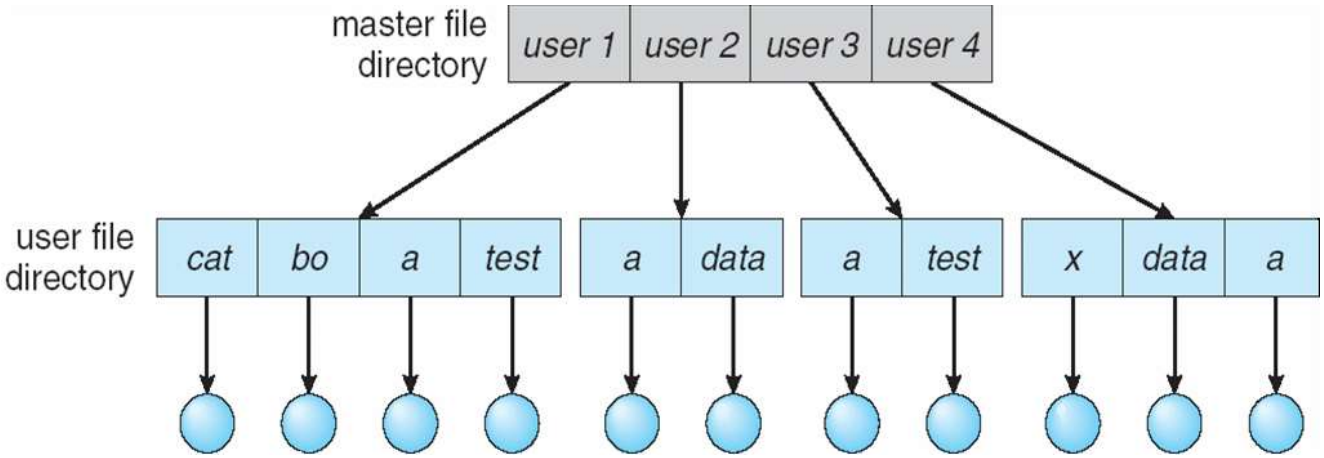


Limitations

- Naming problem
- Grouping problem

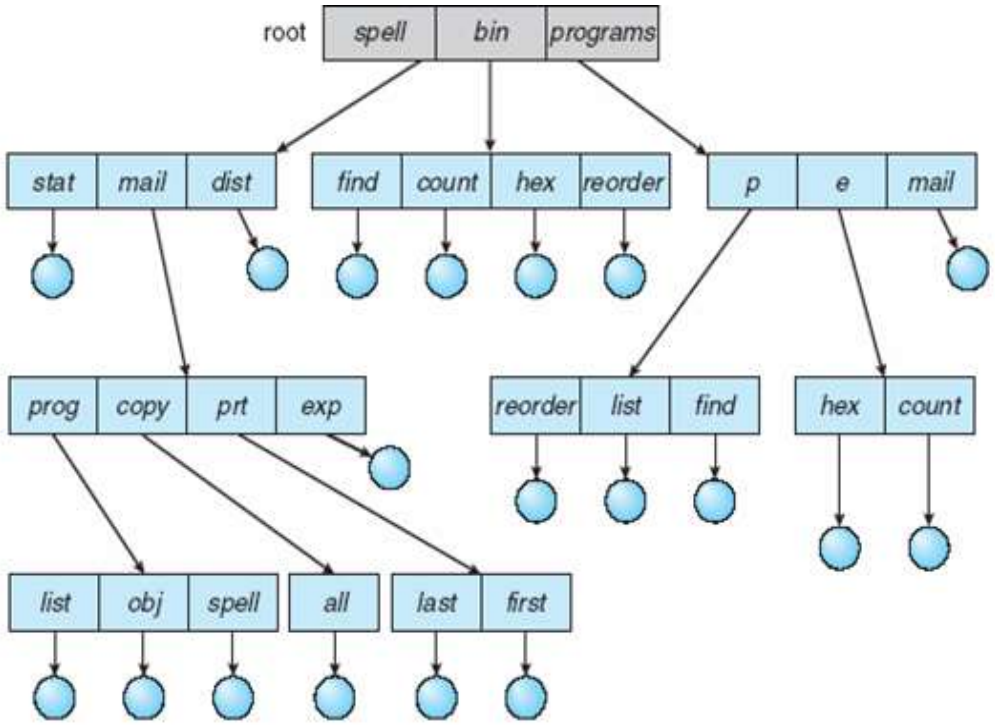
Two-Level Directory

- Separate directory for each user



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

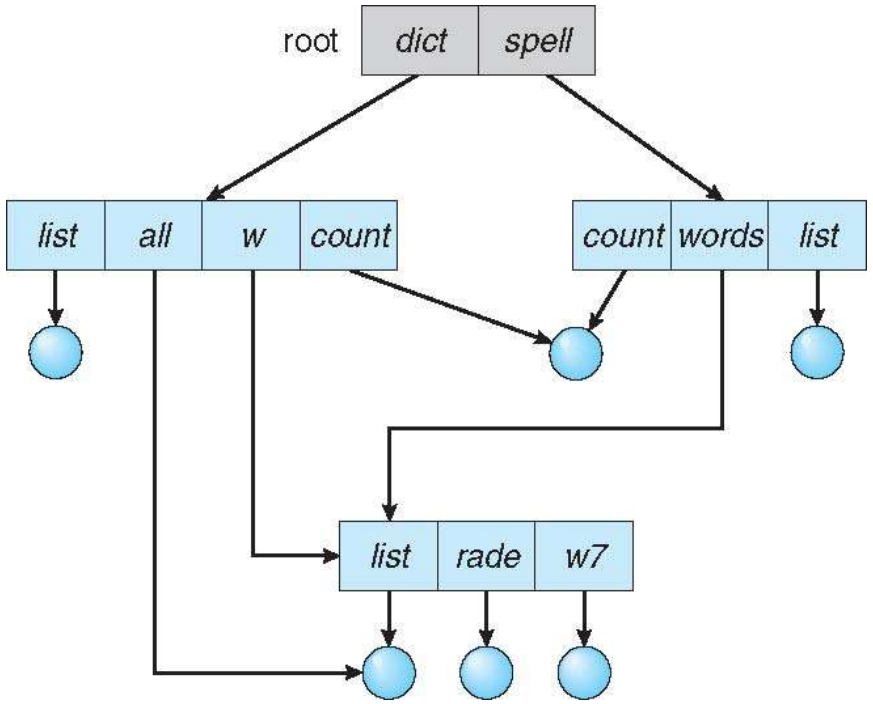
Tree-Structured Directories



- Efficient searching
- Grouping Capability
- Current directory (working directory)
  - `cd /spell/mail/prog`
- **Absolute** or **relative** path name
- Sharing is not possible

Acyclic-Graph Directories

- Have shared subdirectories and files

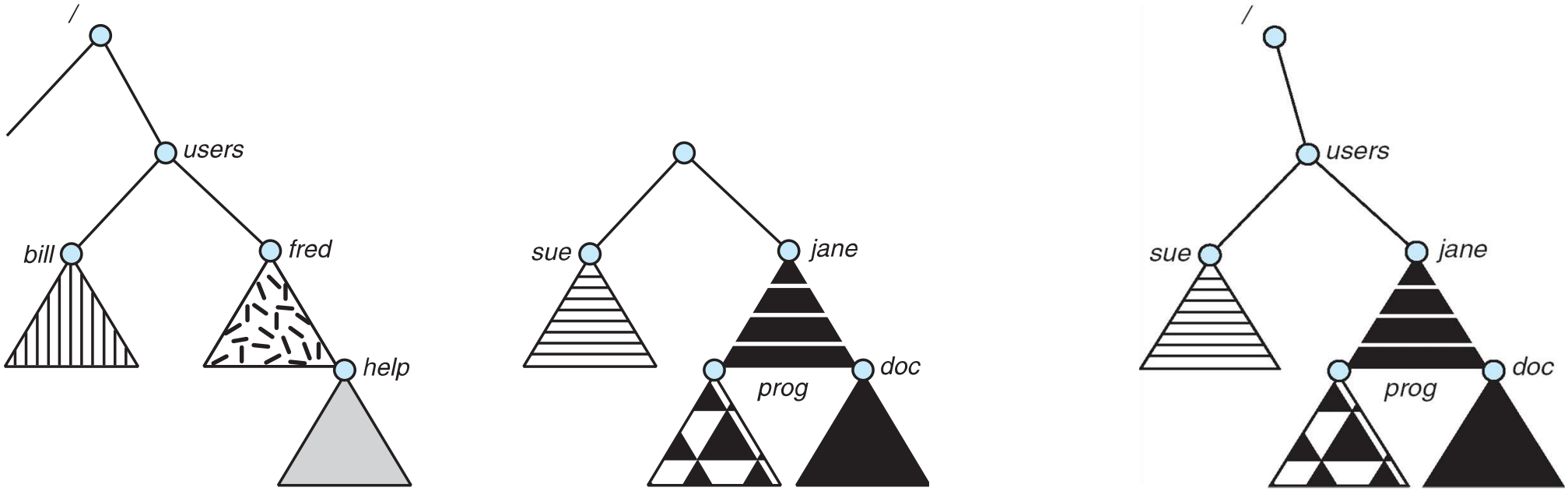


What If **a file gets deleted?**

1. Soft link : the file just gets deleted, and we are left with a dangling pointer.
2. Hard link : the actual file will be deleted only if all the references to it gets deleted.

File System Mounting

- A file system must be **mounted** before it can be accessed
- An unmounted file system is mounted at a **mount point**



Protection

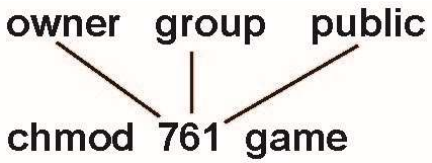
- File owner/creator should be able to control:
  - what can be done
  - by whom
- Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

Access Lists and Groups

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



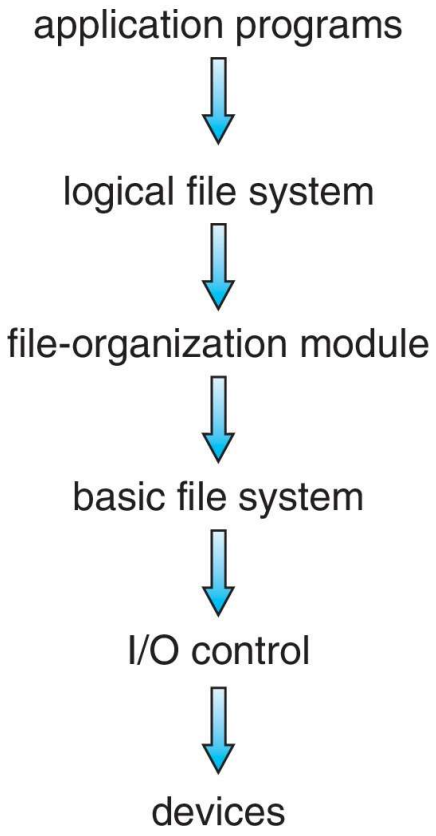
Attach a group to a file

```
chgrp      G      game
```

File System Structure

- File structure
  - Logical storage unit
  - Collection of related information
- **File system** resides on secondary storage (disks)
  - Provided user interface to storage, mapping logical to physical
  - Provides efficient and convenient access to disk by allowing data to be stored, located retrieved easily
- Disk provides in-place rewrite and random access
  - I/O transfers performed in **blocks** of **sectors** (usually 512 bytes)
- **File control block** (**FCB**) – storage structure consisting of information about a file
- **Device driver** controls the physical device
- File system organized into layers

Layered File System



Logical file system manages metadata information

File organization module understands files, logical address, and physical blocks
Translates logical to physical blocks

Basic file system given command like "retrieve block 123" translates to device driver
Managers memory buffers and caches

Device drivers manage I/O devices at the I/O control layer

On Disk Data Structures

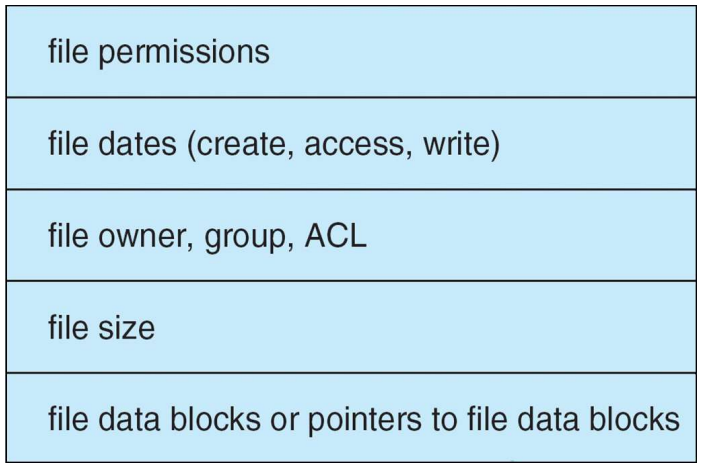
- Boot Control Block – contain information needed to boot an OS from volume.
  - Eg: Unix – boot control block or boot block, NTFS – partition boot sector.
- Volume Control Block - contains volume details
  - Total # of blocks, # of free blocks, block size, free block pointers or array
- Directory Structure (per file system) – contains file names and pointers to corresponding FCB's.
  - Eg: Unix – includes inode numbers associated to file names.
- File control block – contains all details about the file

When creating a new file

- To create a new file
- A process calls the logical file system
  - Logical file system is aware of the format of the directory structures.
- Allocates a new FCB
- System reads the appropriate directory into memory
  - Updates it with new file name and FCB
- Writes it back to the file system

On Disk Data Structures (Cont.)

- Per-file **File Control Block** (**FCB**) contains many details about the file
  - typically inode number, permissions, size, dates



In memory data structures

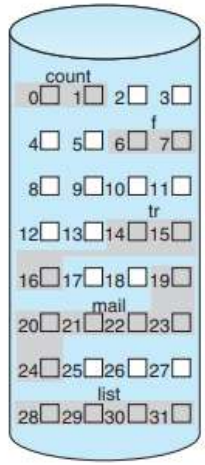
Used for file system management and to improve performance via caching.

- In-memory mount table : contains list of all the devices mounted to the system.
- In-memory directory structure cache : list of directories recently accessed by the CPU.
- System-wide open file table : list of all open files
- Per-process open file table : list of open files per process

Allocation Methods

- To allocate disk space to files.
- Selection will affect the performance and efficiency of the system.
- Allocation methods
  - Contiguous
  - Extent based
  - Linked
  - Indexed
  - FAT
  - Multilevel Indexed
  - Inode

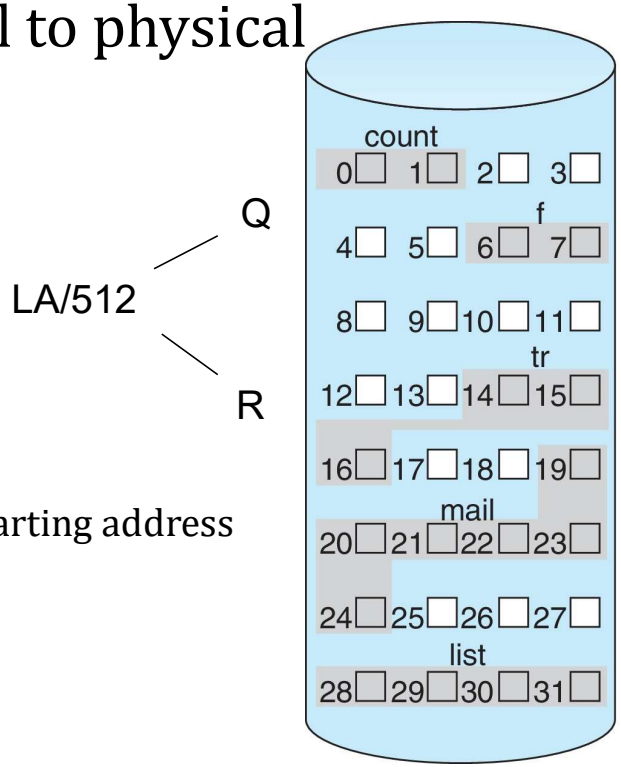
Allocation Methods – Contiguous



- An allocation method refers to how disk blocks are allocated for files:
- **Contiguous allocation** – each file occupies set of contiguous blocks
  - Best performance in most cases
  - Simple – only starting location (block #) and length (number of blocks) are required
  - Problems include finding space for file, knowing file size, external fragmentation, need for **compaction off-line** (**downtime**) or **on-line**

Contiguous Allocation

- Mapping from logical to physical



directory

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

Block to be accessed = Q + starting address
Displacement into block = R

Extent-Based Systems

- Many newer file systems (i.e., Veritas File System) use a modified contiguous allocation scheme

- Extent-based file systems allocate disk blocks in extents

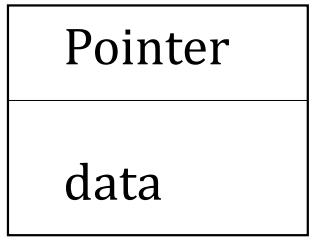
- An **extent** is a contiguous block of disks
  - Extents are allocated for file allocation
  - A file consists of one or more extents

Allocation Methods - Linked

- **Linked allocation** – each file is a linked list of blocks
  - File ends at nil pointer
  - No external fragmentation
  - Each block contains pointer to next block
  - No compaction, external fragmentation
  - Free space management system called when new block needed
  - Improve efficiency by clustering blocks into groups but increases internal fragmentation
  - Reliability can be a problem
  - Locating a block can take many I/Os and disk seeks

Linked Allocation

Block =



Each file is a linked list of disk blocks(L) blocks may be scattered anywhere on the disk

Mapping

- Let LA be the logical address and let the block size be 512 bytes.
- Each block uses its first four bytes to hold a pointer to the next block.
- Q == LA /508 (integer division), R == LA %508

- Block to be accessed is the Q^{th} block in the linked chain of blocks representing the file.
- Displacement into block = R+4

Allocation Methods - Linked