CS370 Operating Systems

Colorado State University

Yashwant K Malaiya

Fall 2021 L21

File-system Interface

Slides based on

- Text by Silberschatz, Galvin, Gagne
- Various sources

FAQ

- TLB vs Cache? Caches contains instructions and data, TLB contains only page-to-frame mapping
- Can the page table be accessed by the user programs? Kernel space
- Working set can mean
  - Pages accessed in a specified time window tools available
  - Pages currently allocated to a process
- Reference bit: set to one if frame accessed. Minimal info needed for LRU
- What page replacement algorithms are currently in use [variations of LRU/Clock]
- Second chance/Clock: combination of LRU approx. and sequential search

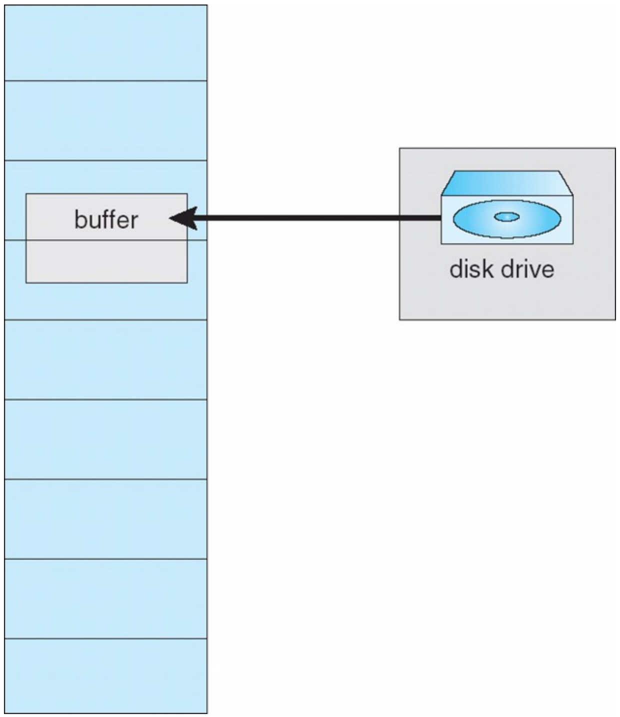
Colorado State University

Please be considerate

- Allow other students to concentrate
- No talking, humming, etc ..

Other Issues – I/O interlock

- **I/O Interlock** – Pages must sometimes be locked into memory
- Consider I/O - Pages that are used for copying a file from a device must be locked from being selected for eviction by a page replacement algorithm
- **Pinning** of pages to lock into memory



Example: MS Windows

- Uses demand paging with **clustering**. Clustering brings in pages surrounding the faulting page
- Processes are assigned **working set minimum** and **working set maximum**
- Working set minimum is the minimum number of pages the process is guaranteed to have in memory
- A process may be assigned as pages up to its working set maximum
- When the amount of free memory in the system falls below a threshold, **automatic working set trimming** is performed to restore the amount of free memory
- Working set trimming removes pages from processes that have pages in excess of their working set minimum

CS370 Operating Systems

Colorado State University

Yashwant K Malaiya

Fall 2021

File-system

Slides based on

- Text by Silberschatz, Galvin, Gagne
- Various sources

File-Systems

Ch 13: File system interface

- File Concept, types
- Attributes, Access Methods, operations, Protection
- Directory Structure, namespace, File-System Mounting, File Sharing

Ch 14: File system implementation

Ch 15: File system internals

- **Storage abstraction**: File system metadata (size, free lists), File metadata(attributes, disk block maps), data blocks
- **Allocation of blocks to files**: contiguous, sequential, linked list allocation, indexed
- **In memory info**: Mount table, directory structure cache, open file table, buffers
- **Unix**: inode numbers for directories and files

Ch 11: Mass storage

File Systems

"MS. GRIMMETT, I SORT OF LIKED THE OLD FILING SYSTEM...IN THE FILE CABINETS."

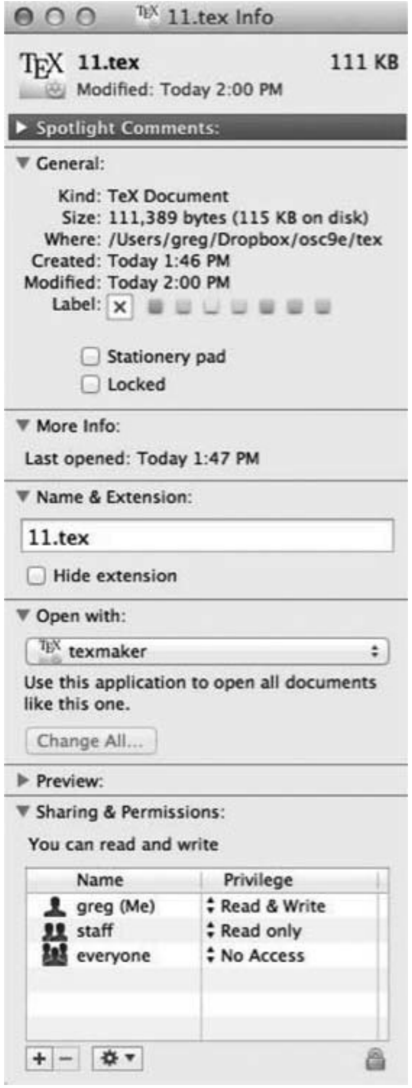
File types

Type used by programs *not OS*

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 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



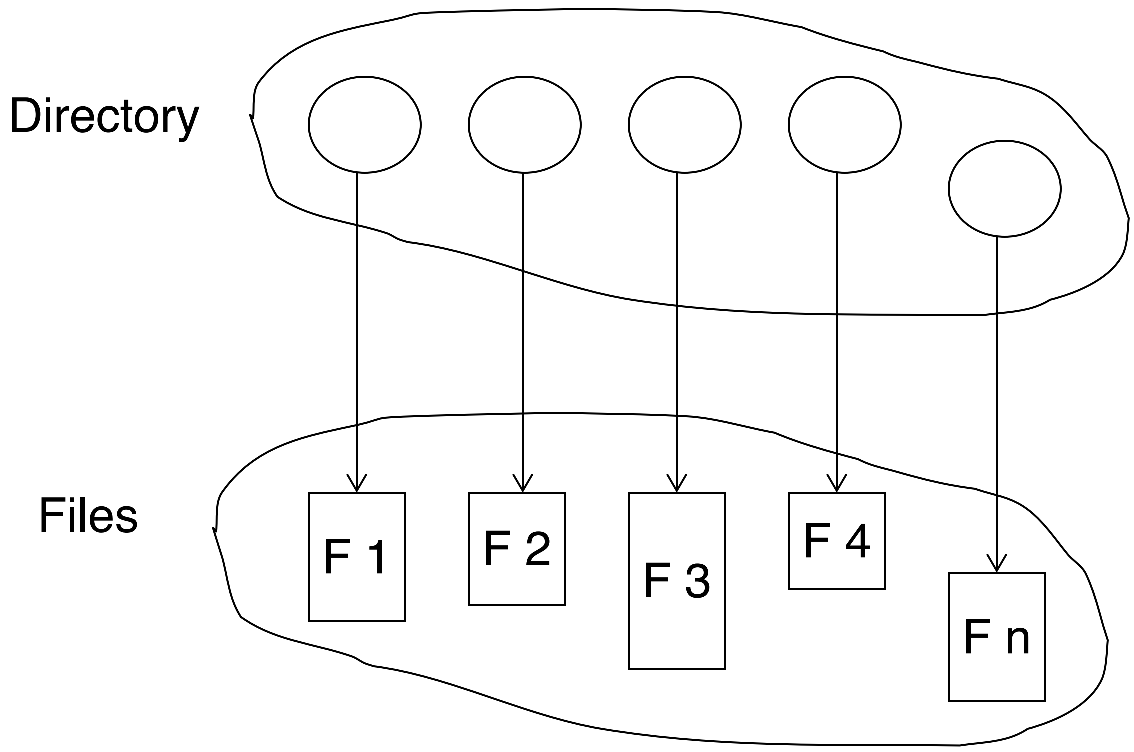
Colorado State University

Disk Structure

- Disk can be subdivided into **partitions**
- Disks or partitions can be **RAID** protected against failure
- Partition can be **formatted** with a file system
- 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

Directory Structure

- A collection of nodes containing information about all files



Both the directory structure and the files reside on disk

Colorado State University

Operations Performed on Directory

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

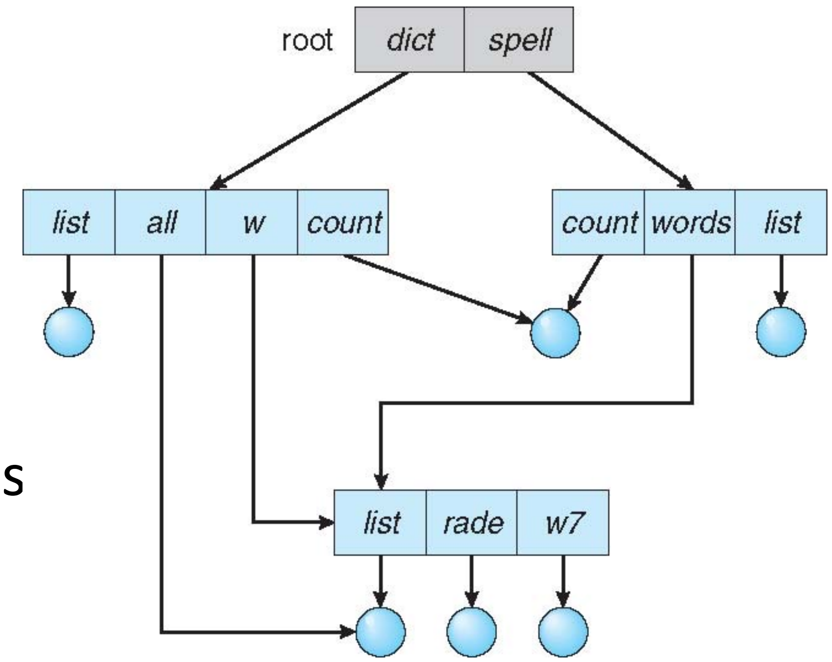
Directory Organization

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, …)

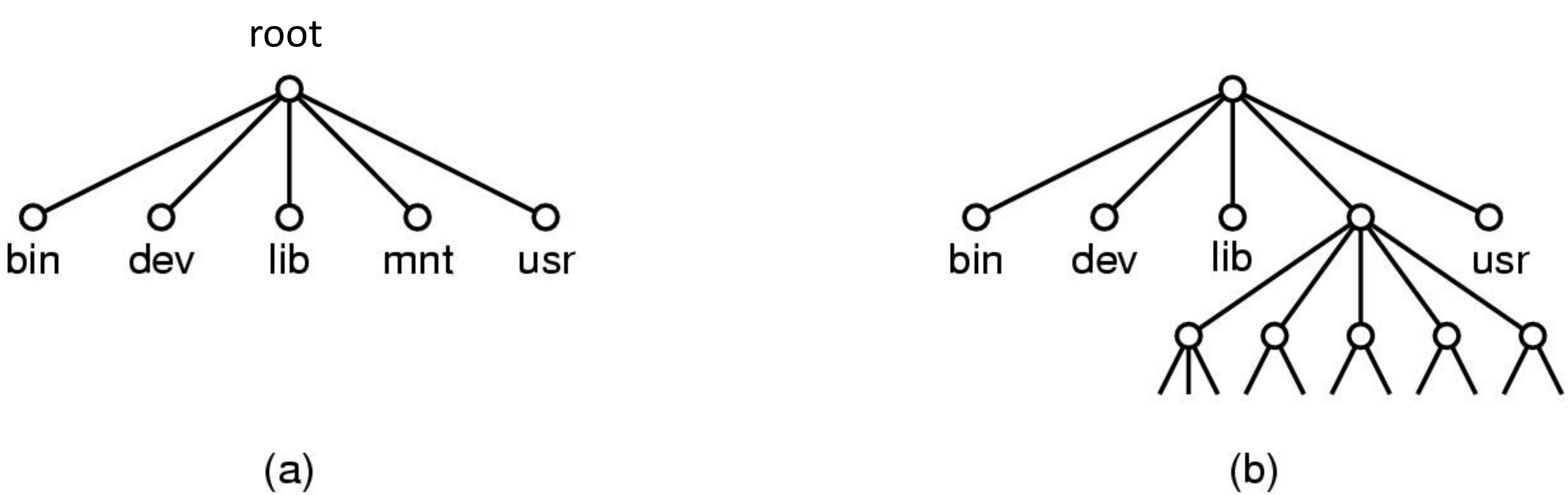
Directory Organization

- Single level directory
- Two-level directory
- Tree-structured directories:
  - efficient grouping, searching,
  - absolute or relative path names
- Acyclic graph directories
  - Shared sub-directory, files



File System Mounting

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



Colorado State University

File Sharing

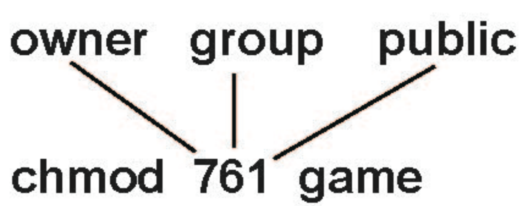
- Sharing of files on multi-user systems is desirable
- Sharing may be done through a **protection** scheme
- On distributed systems, files may be shared across a network
- Network File System (NFS) is a common distributed file-sharing method
- If multi-user system
  - **User IDs** identify users, allowing permissions and protections to be per-user
    Group IDs allow users to be in groups, permitting group access rights
  - Owner of a file / directory
  - Group of a file / directory

Protection: 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
			RWX
b) **group access**	6	⇒	1 1 0
			RWX
c) **public access**	1	⇒	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.

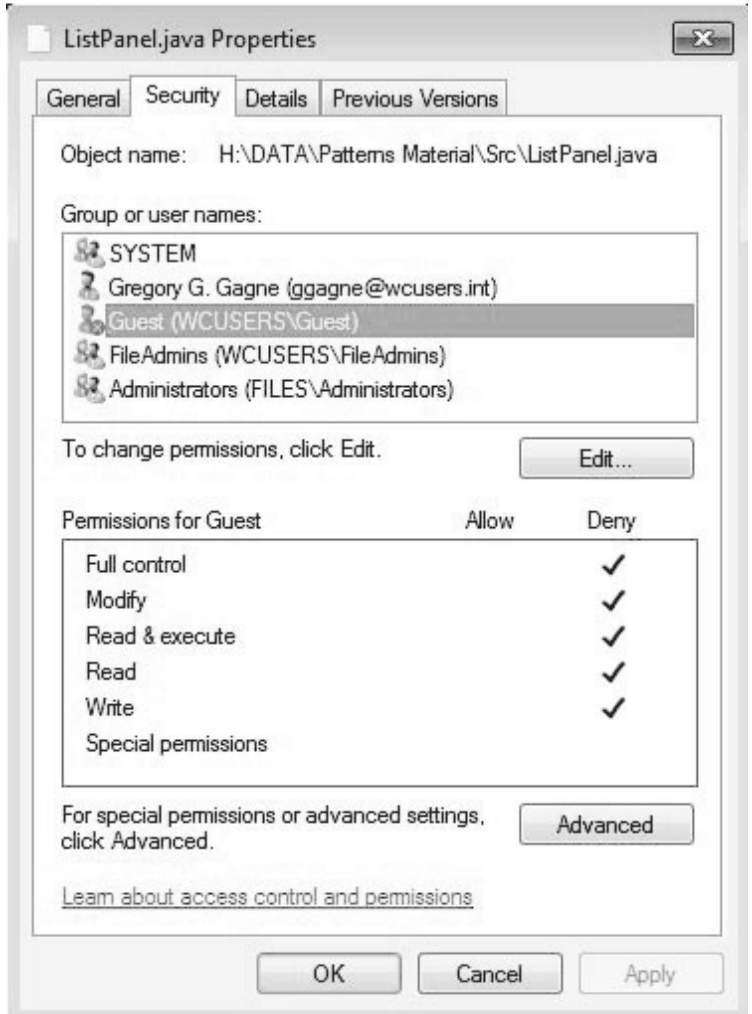


Attach a group to a file

```
chgrp     G     game
```

Colorado State University

Windows 7 Access-Control List Management



A Sample UNIX Directory Listing

-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4 pbg	faculty	512	Jul 31 10:31	lib/
drwx------	3 pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3 pbg	staff	512	Jul 8 09:35	test/

dir, access, links, owner, group owner, size, last modification time, name

CS370 Operating Systems

Colorado State University
Yashwant K Malaiya

File-system Implementation

Slides based on

- Text by Silberschatz, Galvin, Gagne
- Various sources

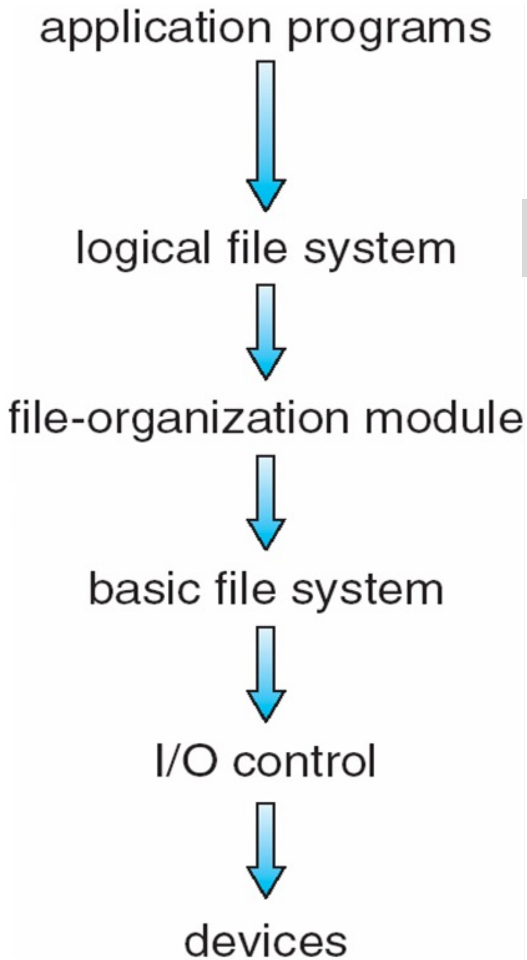
- File-System Structure
- File-System Implementation
- Directory Implementation
- Allocation Methods
- Free-Space Management
- Efficiency and Performance
- Recovery

File-System Structure

- **File** structure
  - Logical storage unit
  - Collection of related information
- **File system** resides on secondary storage (disks/SSD)
  - Provides user interface to storage, mapping logical to physical
  - Provides efficient and convenient access to disk by allowing data to be stored, located retrieved easily
  - Can be on other media (flash etc), with different file system
- Disk provides in-place rewrite and random access
  - I/O transfers performed in **blocks** of **sectors** (usually 512 bytes)
- **File control block** – storage structure -information about a file ("inode" in Linux) inc location of data
- **Device driver** controls the physical device

Layered File System

File system



Files, metadata

Logical blocks to physical blocks

Linear array of blocks

Device drivers

Layered File System

Processes

Logical File System Layer

Search dir, find file location, determine which file blocks will be used

File Organization Layer

Map file blocks (logical blocks) to disk blocks (physical blocks), disk allocation

"Basic File System" Layer

Commands to device driver, Buffering of disk data, caching of disk blocks

Disk Driver

Disk Controller

File System Layers (from bottom)

- **Device drivers** manage I/O devices at the I/O control layer
  - Given commands like "read drive1, cylinder 72, track 2, sector 10, into memory location 1060" outputs low-level hardware specific commands to hardware controller
- **"Basic file system"** given command like "retrieve block 123" translates to device driver
  - Also manages memory buffers and caches (allocation, freeing, replacement)
    - Buffers hold *data in transit*
    - Caches hold *frequently used data*
- **File organization module** understands files, logical address, and physical blocks
  - Translates logical block # to physical block #
  - Manages free space, disk allocation
- **Logical file system** manages metadata information
  - Translates file name into file number, file handle, location by maintaining *file control blocks* (**inodes** in UNIX)
  - Directory management
  - Protection

File Systems

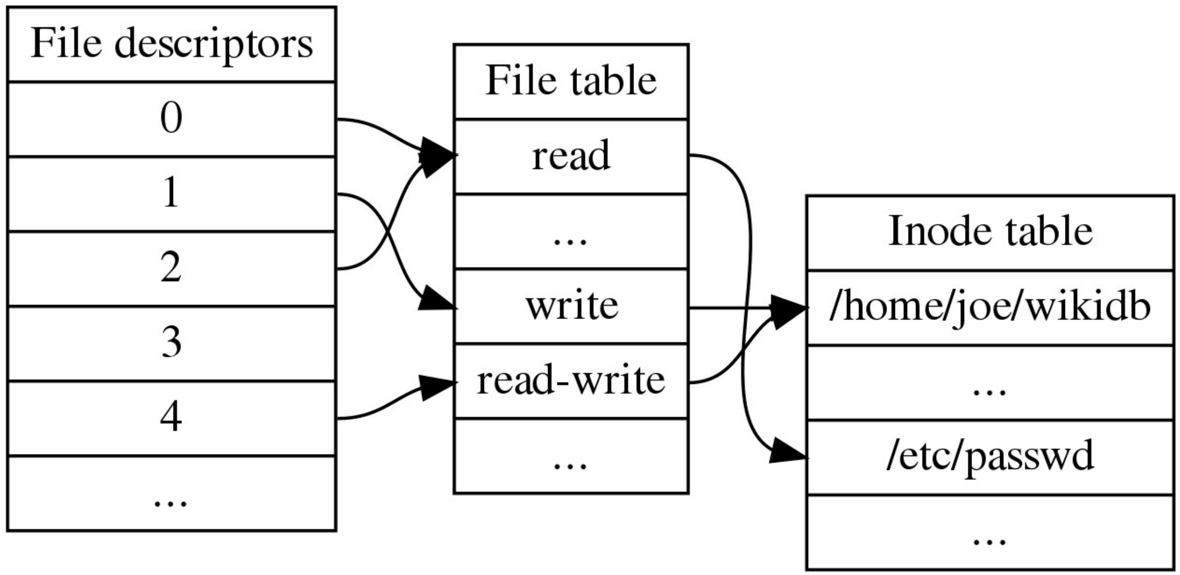
- Many file systems, sometimes several within an operating system
  - Each with its own format
    - Windows has FAT (1977), FAT32 (1996), NTFS (1993), xFAT (USB/SD cards 2006), ReFS (2012)
    - Linux has more than 40 types, with **extended file system** (1992) ext2 (1993), ext3 (2001), ext4 (2008);
    - distributed file systems, GoogleFS (2003), HDFS (2006)
    - floppy, CD, DVD Blu-ray ..
  - New ones still arriving..

Data and Metadata

Storage abstraction:

- File system metadata (size, free lists),
  - File metadata (attributes, disk block maps),
    - Data blocks

Process, System, Files



- **File descriptor table for a process:** File descriptor, pointer
- **System wide open File Table:** r/w status, offset, inode number
- **Inode table for all files/dirs:** indexed by inode numbers **(unix: ls –ia)**
  - **Inode for a file:** file/dir metadata, pointers to blocks

OS File Data Structures

- **Per-process file descriptor table** - for each file,
  - pointer to entry in the open file table
  - current position in file (offset)
  - mode in which the process will access the file (r, w, rw)
  - pointers to file buffer

FD: int

- **Open file table** - shared by all processes with an open file.
  - open count
  - Inode number
- **Inode table** – an inode contains
  - file attributes, including ownership, protection information, access times, ...
  - pointers to location(s) of file in memory

Common File Systems

Journaling: keeps track of changes not yet committed: allows recovery

File System	Max File Size	Max Partition Size	Journaling	Notes
Fat32	4 GiB	8 TiB	No	Commonly supported
ExFAT	128 PiB	128 PiB	No	Optimized for flash
NTFS	2 TiB	256 TiB	Yes	For Windows Compatibility
ext2	2 TiB	32 TiB	No	Legacy
ext3	2 TiB	32 TiB	Yes	Standard linux filesystem for many years.
ext4	16 TiB	1 EiB	Yes	Modern iteration of ext3.

File-System Implementation: Outline

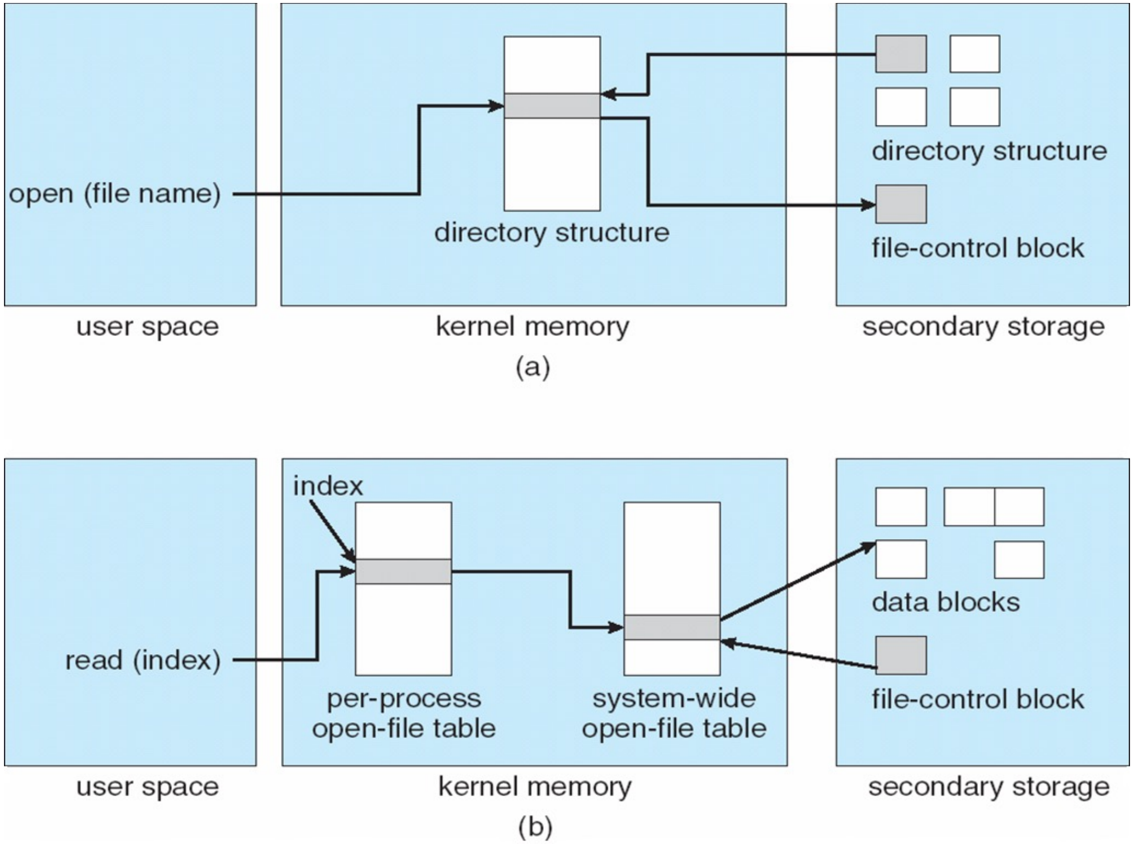
- In memory/On disk structures
- Partitions, mounting
- Disk Block allocation approaches

File-System Implementation

- Based on several on-disk and in-memory structures.
- On-disk
  - Boot control block (per volume) boot block in unix
  - Volume control block (per volume) master file table in UNIX
  - Directory structure (per file system) file names and pointers to corresponding FCBs
  - File control block (per file) inode in unix
- In-memory
  - Mount table about mounted volumes
  - The open-file tables (system-wide and per process)
  - Directory structure cache
  - Buffers of the file-system blocks

Volume: logical disk drive, perhaps a partition

In-Memory File System Structures



Opening a file
fopen( ) returns fid

Reading a file
Inode refers to an individual file

Colorado State University

On-disk File-System Structures

1. **Boot control block** contains info needed by system to boot OS from that volume
   – Needed if volume contains OS, usually first block of volume

Volume: logical disk drive, perhaps a partition

2. **Volume control block** (**superblock** $_{\text{ext}}$ **or master file table**$_{\text{NTFS}}$) contains volume details
   – Total # of blocks, # of free blocks, block size, free block pointers or array
3. Directory structure organizes the files
   – File Names and inode numbers $_{\text{UFS}}$, master file table $_{\text{NTFS}}$

Boot block	Super block	Directory, FCBs	File data blocks

File-System Implementation (Cont.)

4. Per-file **File Control Block** (**FCB or "inode"**) contains many details about the file

- Indexed using inode number; permissions, size, dates UFS (unix file system)
- master file table using relational DB structures NTFS

file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

Create a file

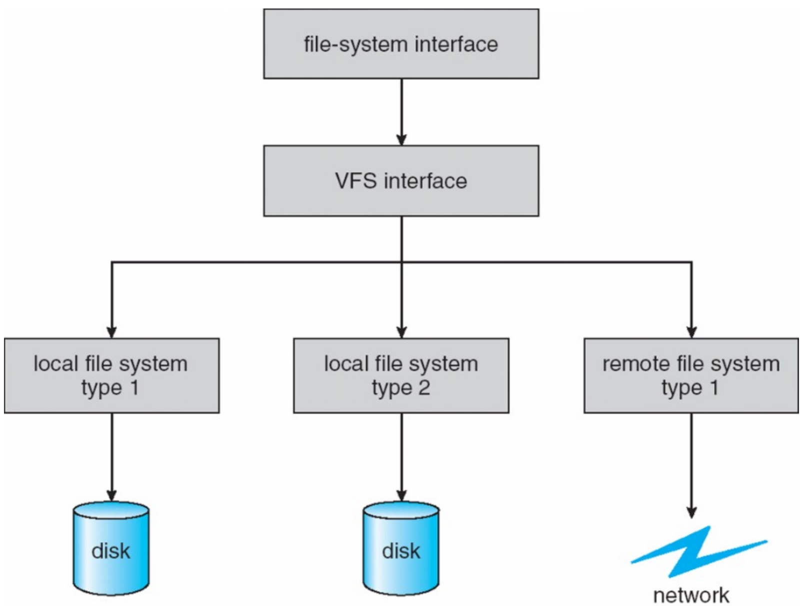
- Allocates a new FCB.
- Update directory
  - Reads the appropriate directory into memory, in unix a directory is a file with special type field
  - updates it with the new file name and FCB,
  - writes it back to the disk.

Partitions and Mounting

- Partition can be a volume containing a file system (*cooked*) or **raw** – just a sequence of blocks with no file system perhaps for swap space
- Boot block can point to boot volume or boot loader set of blocks that contain enough code to know how to load the kernel from the file system
- **Root partition** contains the OS, Mounted at boot time
  - other partitions can hold other OSes, other file systems, or be raw
  - Other partitions can mount automatically or manually
- At mount time, file system consistency checked

Virtual File Systems

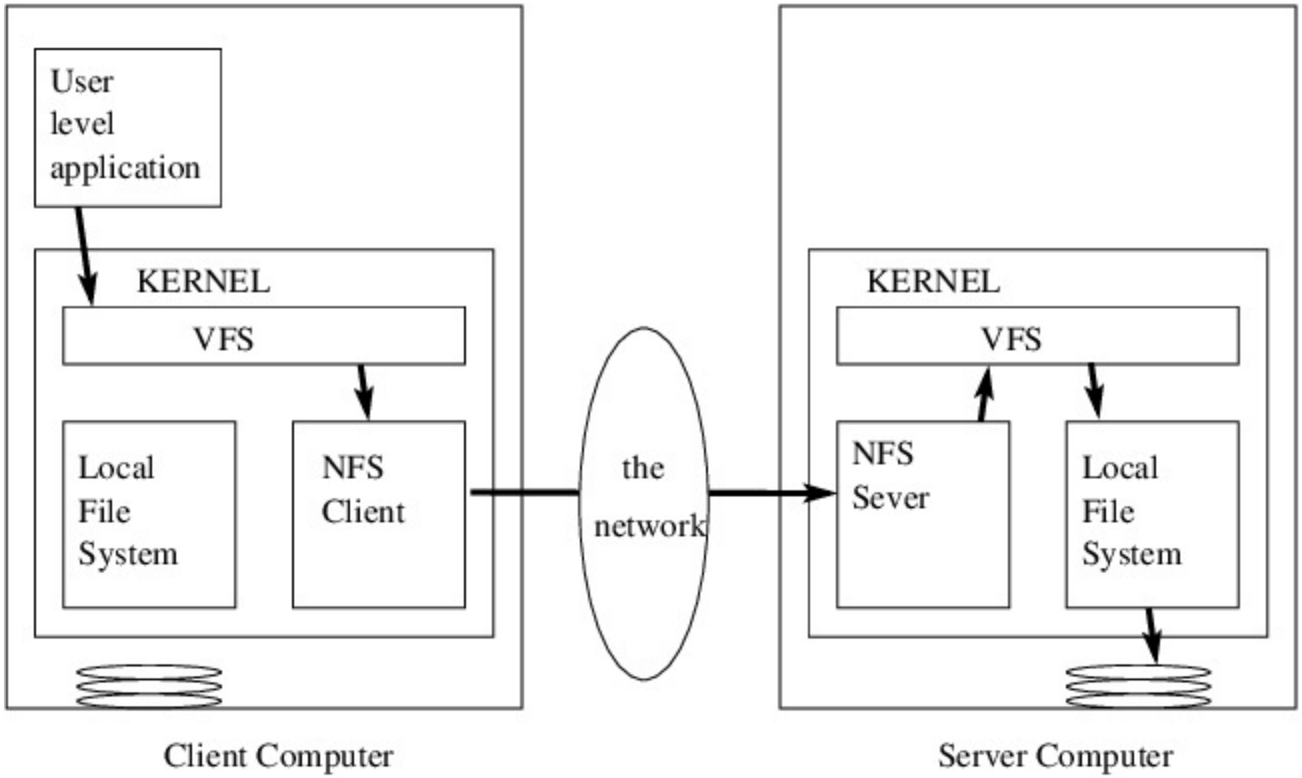
- **Virtual File Systems** (**VFS**) in Unix kernel is an abstraction layer on top of specific file systems.
- VFS allows the same system call interface (the API) to be used for different types of file systems
- The API (POSIX system calls) is to the VFS interface, rather than any specific type of file system



Virtual to specific FS interface

Colorado State University

NFS (Network File System)



[Source]

A distributed file system protocol uses the Open Network Computing Remote Procedure Call (ONC RPC) system (1984).

Colorado State University

File Sharing – Remote File Systems

- Uses networking to allow file system access between systems
  - Manually via programs like FTP/SFTP
  - Automatically, seamlessly using **distributed file systems**
  - Semi automatically via the **world wide web**
- **Client-server** model allows clients to mount remote file systems from servers
  - Server can serve multiple clients
  - Client and user-on-client identification is insecure or complicated
  - **NFS** is standard UNIX client-server file sharing protocol
  - **CIFS** is standard Windows protocol
  - Standard operating system file calls are translated into remote calls