CS 370: OPERATING SYSTEMS

[FILE SYSTEMS]

Computer Science
Colorado State University

Instructor: Louis-Noel Pouchet
Spring 2024

** Lecture slides created by: SHRIDEEP PALLICKARA

Topics covered in this lecture

- File System Structure
- File System Implementation
- Virtual file systems
- Allocations
  - Contiguous allocation
  - Linked allocations
  - Indexed allocations

File Systems

Rationale: Applications need to store and retrieve information

- A program can store a **limited** amount of information in its own address space
- Storage capacity is **restricted** to the size of virtual memory
  - Far too small for several applications
    - Airline reservations, banking, directory services etc

Rationale: Information in the address space of process is not persistent

- When process terminates, information is lost
- For many applications information must be **retained** for a much longer time
  - Weeks, Years, Forever

Rationale: Multiple processes often need to access (parts of) information at the same time

- Storing an online telephone directory in the address space of one process?
  - Only that process can access the info
  - Only one telephone number can be looked up at a time

Essential requirements for long-term storage

1. **Store** a very large amount of information
2. Information must **survive** process termination
3. Multiple processes must be able to **concurrently access** the information

- Store info on disk or external media
  - In units called **files**

Files are an abstraction mechanism

- Provide a way to store information and read it back later
- Do this is an way that **shields** the user from
  - How and where information is stored on disk
  - How disks really work

Naming files

- Important characteristic of the abstraction mechanism
- Strings 8-255 characters long
- Most OS support two-part file names separated by a period
  - Last part referred to as the **file extension**
    - Conventions: Easy to remember
    - Enforced in some cases e.g. the compiler

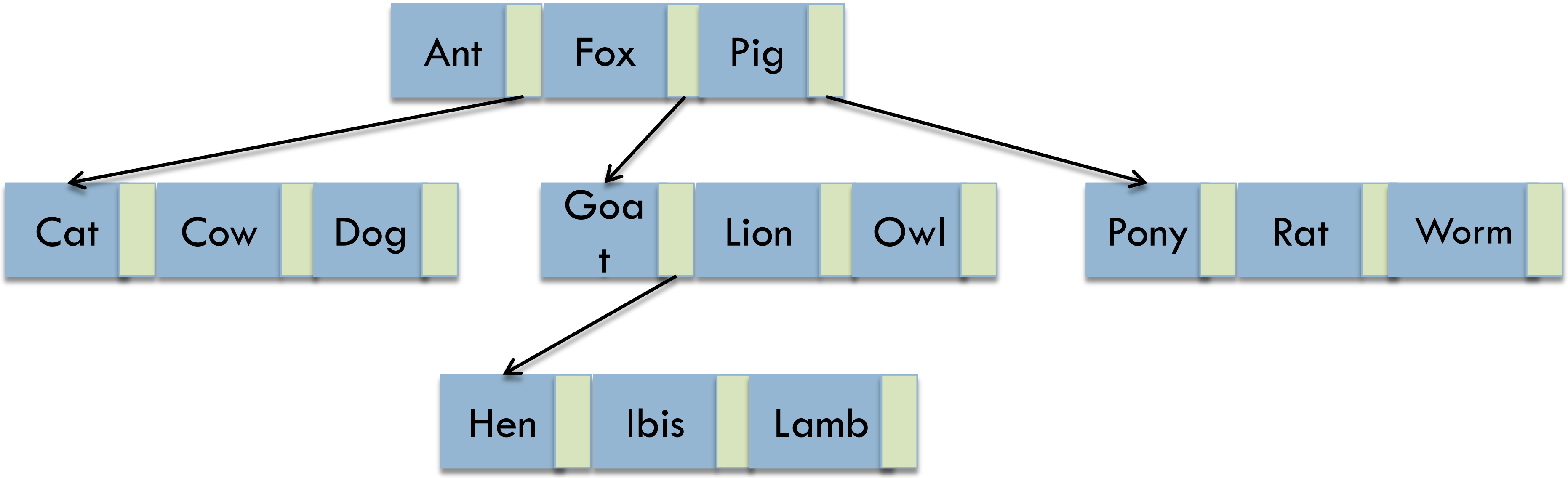
Files can be structured in many ways: Unstructured sequence of bytes

- The OS does not know or care what is in the file
  - Maximum **flexibility**
- OS does not help, but does not get in the way either
- Meaning is imposed by programs
- Most OS support this

File Structure: A sequence of records

- A file is a sequence of **fixed-length** records
- Read operation returns one record
  - Write operation overwrites/appends one record
- 80-column punch card used to be dominant
  - Files consisted of 80 character records

File structure: A tree of records



- Get record with specific key
- OS, <u>not user,</u> decides where to place new records

Directory and disk structure

- Typically, there are millions of files within a computer
- Storage device can be used in its entirety for a file system
- It could also be **partitioned**
  - Limit size of individual file systems
  - Put multiple file system types
  - Set aside for **swap space**

Directories are used to organize files

- Can be viewed as a **symbol table**
- In many systems directories themselves are files
- Supported operations
  1. Insert, delete, search, list and rename entries
  2. Traversal

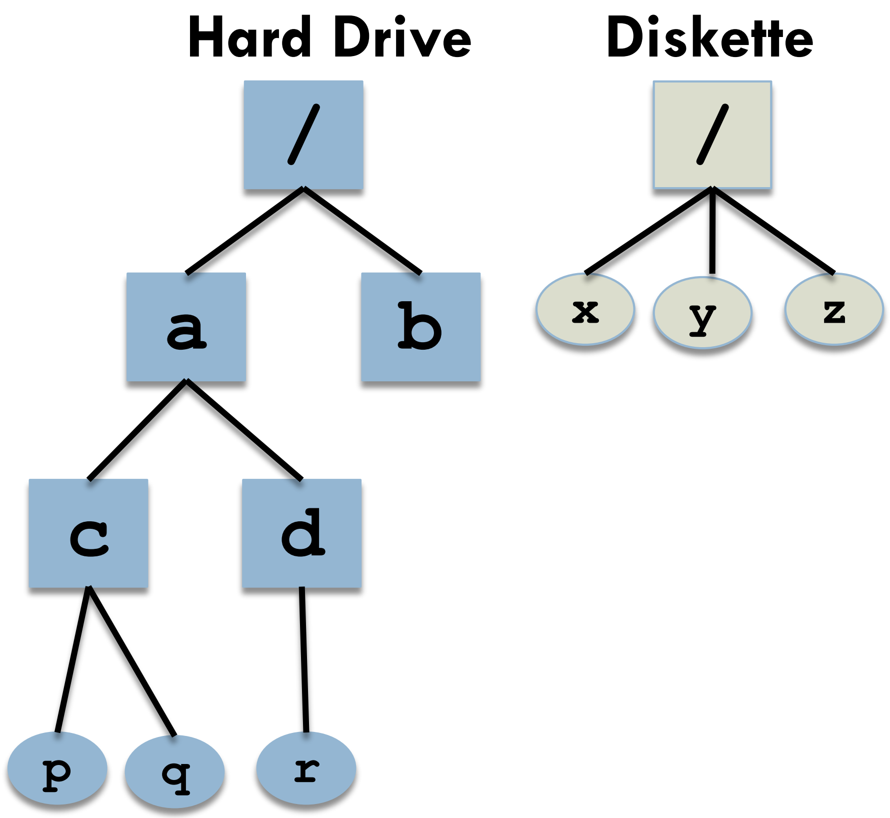
Organization of directories

- Single level directory
- Two-level directory
- Tree-structured directories
- Acyclic graph directories
  - Shared sub-directory

Mounting file systems

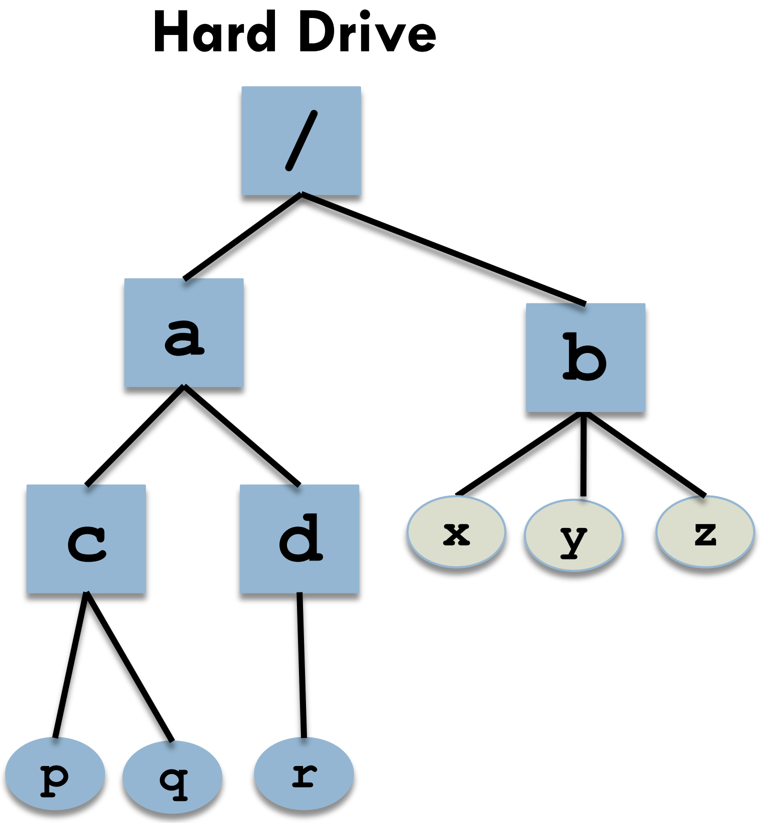
- Many systems have more than one disk
  - How do you handle them?
- **S1**:Keep self contained file system on each disk
  - And keep them separate
- **S2**: Allow one disk to be **mounted** in another disk's file tree

Mounting file systems



`cp D:/x /a/d/x`

H is default



`cp /b/x /a/d/x`

Checks performed during mounting

- OS **verifies** that the device contains a valid file system
- Read device directory
  - Make sure that the format is an expected one
- Windows mounting
  - Each device in a separate name space
  - {Letter followed by a colon e.g. **G:**}

FILE SYSTEM STRUCTURE

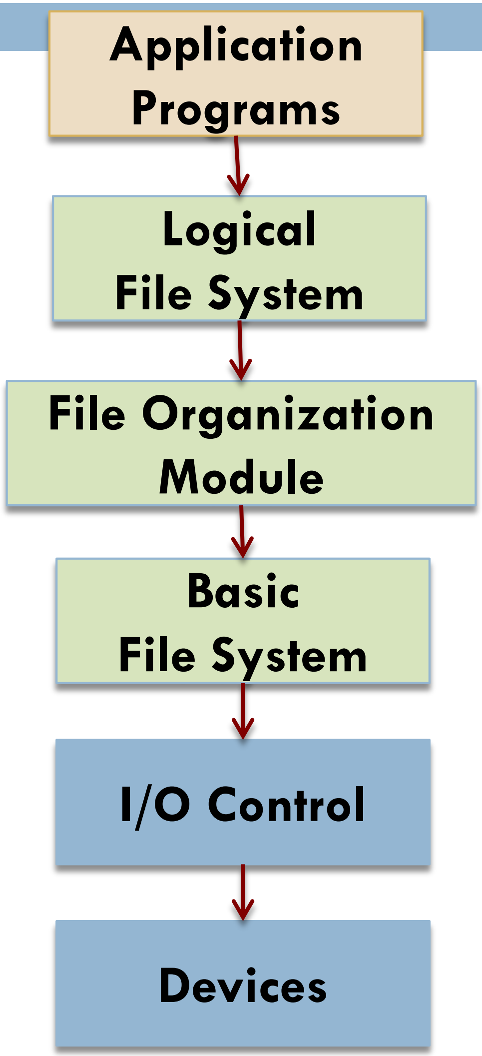
Disks provide the bulk of secondary storage

- A disk can be **rewritten** in place
  - Read, modify, and then write-back to same place
- Disks can **directly access** any block of information
- I/O transfers between memory and disk are performed in units of **blocks**

There are two core design problems in file systems

- Defining how the file system should **look** to the user
- Creating algorithms and data structures to **map** logical file system onto physical storage

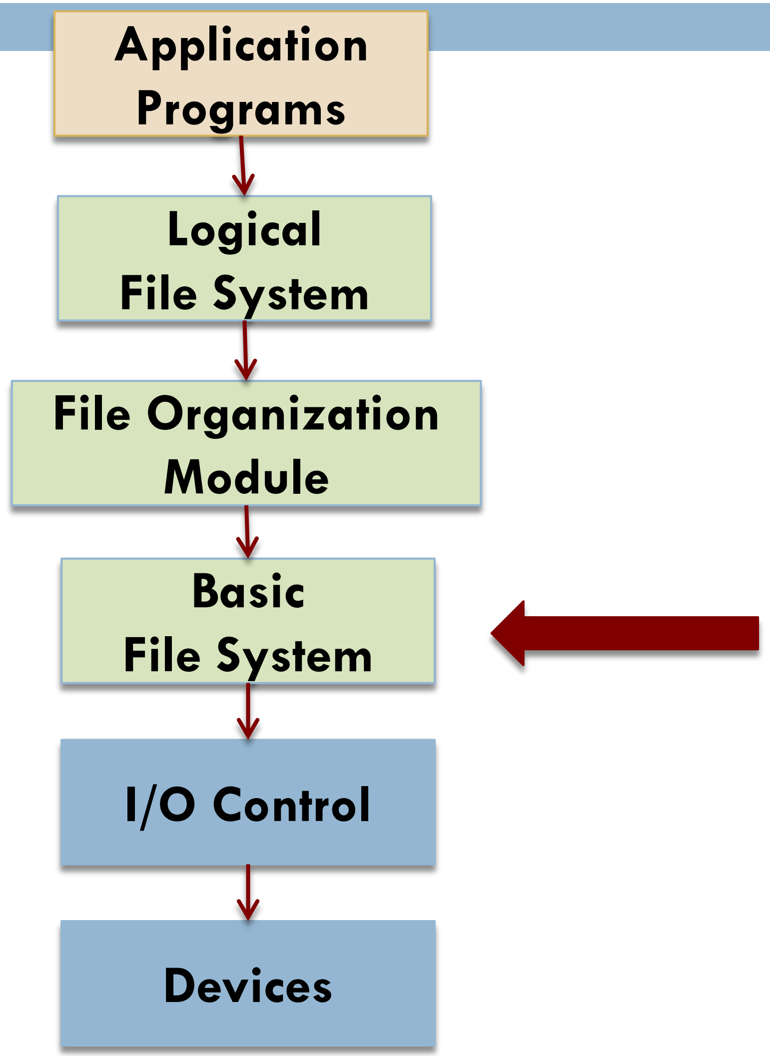
There are many levels that comprise a file system



I/O Control consists of device drivers

- Transfers information *between main memory and disk*
- Receives **high-level** commands
  - Retrieve block 123, etc
- Outputs low-level, hardware-specific instructions
  - Used by the hardware controller
  - Writes bit patterns into specific locations of the I/O controller

There are many levels that comprise a file system

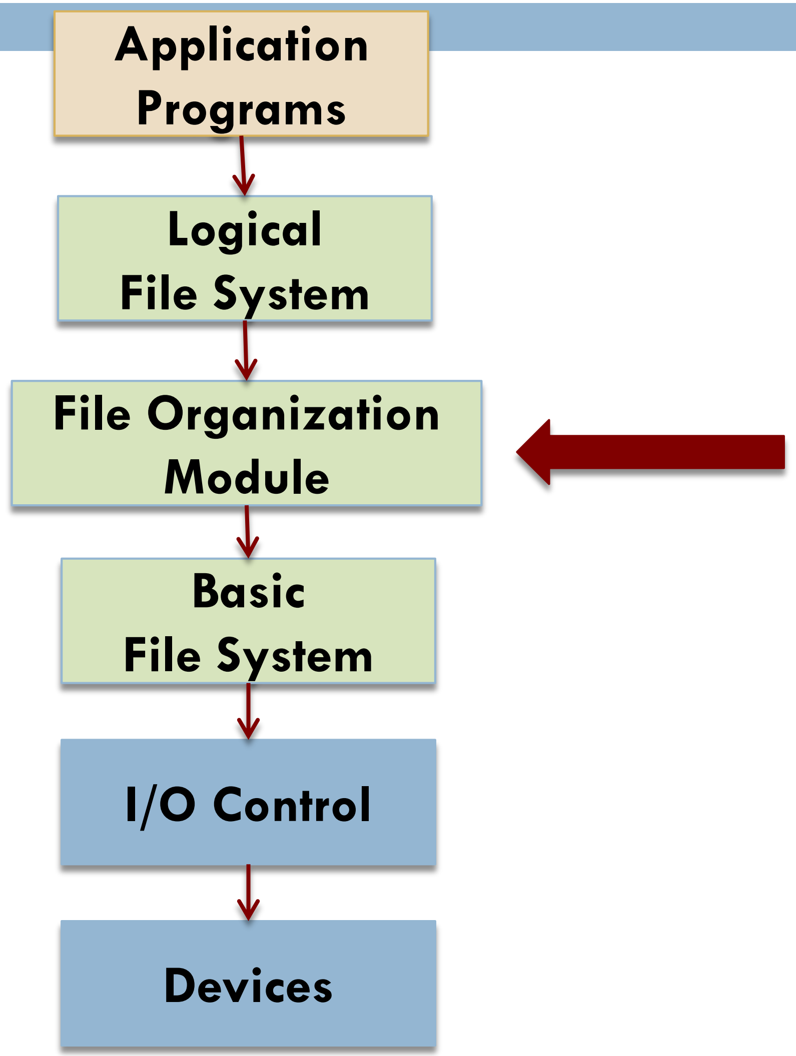


Basic file system issues commands to the device driver

- Read and write physical blocks on disk
  - E.g. Drive 1, cylinder 73, sector 10

- Manages **buffers and caches**
  1. To hold file system, directory and data blocks
  2. Improves performance

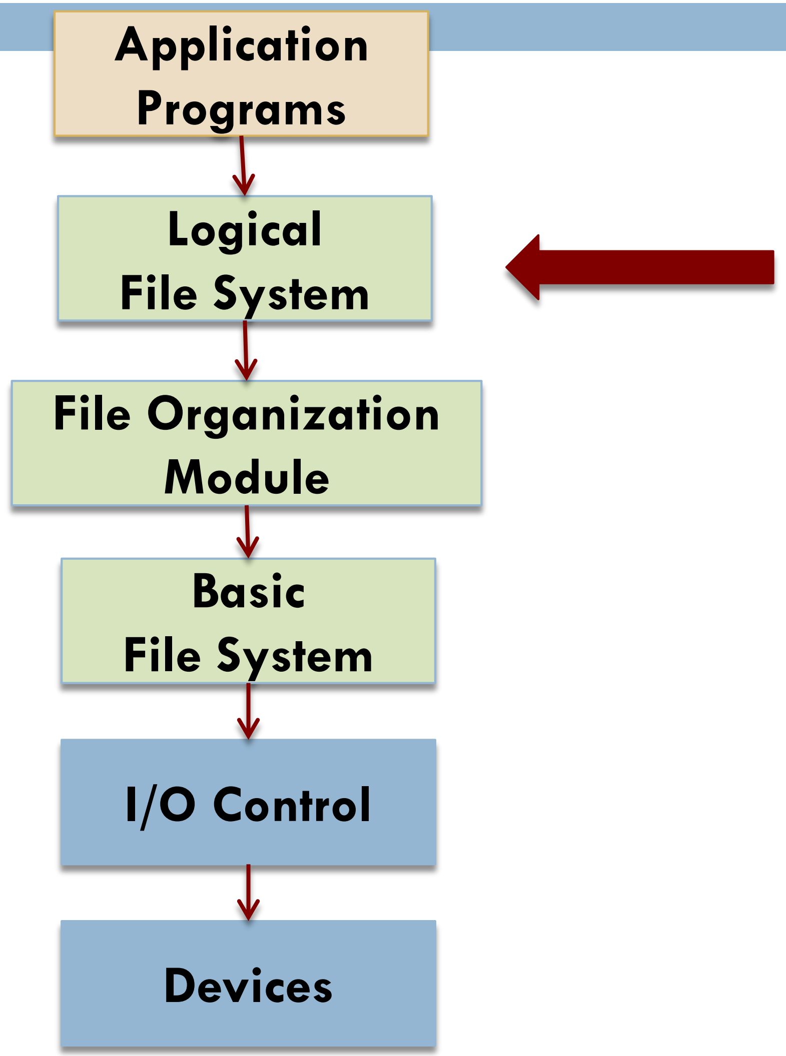
There are many levels that comprise a file system



File organization module

- Knows about files
  - Logical and physical blocks
- **Translate** logical addresses to physical ones
  - Needed for every block
- Includes a **free space manager**
  - Tracks unallocated blocks and allocates as needed

There are many levels that comprise a file system



The logical file system

- Manages **metadata** information
  - Metadata is *data describing the data*

- Maintains file structure via **file control blocks**
  - Info about the file
    - Ownership and permissions
    - Location of file contents
  - **inode** in UNIX file systems

Several file systems are in use

- CD-ROMs written in ISO 9660 format
  - Designed by CD manufacturers
- UNIX
  - Unix file system (**UFS**)
  - Berkley Fast File System (**FFS**)
- Windows: **FAT, FAT32** and **NTFS**
- Linux
  - Supports 40 different file systems
  - Extended file system: **ext2, ext3** and **ext4**

The anatomy of a disk

Using a magnet and a nail for instant messaging? (1)

- **Message**: See you later; or not

- Drop a nail in your friend's mailbox
  - If nail is magnetized? You'll see the friend
  - If nail is not magnetized? You won't

Using a magnet and a nail for instant messaging? (2)

- Your friend comes home and picks up the nail
  - Uses the nail to pick up a paper-clip
    - If it sticks? Friend will expect to see you

- Magnetism can be used to store information!

Using magnetism to store information

- Store information even when you turn power off!

- Storing …**10001**…?
  - Magnetize first bit
  - Demagnetize next 3
  - Magnetize the next bit

The anatomy of a disk

- A disk comprises a set of **platters**
  - These have a flat, circular shape
  - Usually made of glass or aluminum
- Both surfaces of a platter covered with **magnetic material**
  - Store information by recording it magnetically
- A platter is logically divided into circular **tracks**
  - These are subdivided into **sectors**

Rates and times associated with disks

- Rate of data movement between the disk and the memory
  - **Transfer rate**
- Positioning time
  - **Seek time**
    - Move disk arm to the necessary cylinder
  - **Rotational latency**
    - Time for the desired sector to rotate to the disk head

How about CD-ROMs, DVDs, and Blu-Rays? (1)

- Data written with the help of *high intensity* laser that makes "**pits**" on the reflecting surface
- During reads:
  - Use a lower intensity laser
  - Mirrors and a focusing lens are used to shine light on a specific portion of the disk
  - The amount of light that is *reflected back* depends on the presence or absence of a pit
    - Use this to interpret a 1 or 0

How about CD-ROMs, DVDs, and Blu-Rays? (2)

- The *shorter* the wavelength, the *smaller* the pit
  - And greater the density of what can be stored
- DVD uses a 650 nm wavelength laser diode
  - 780 nm for CD
  - Pit sizes: DVD = 0.74 μm and CD = 1.6 μm
- What about Blu-Ray?
  - 405 nm wavelength, 0.13 μm pit size
  - 50 GB storage possible on one disk
- What's next?
  - Archival Disc (Sony/Panasonic) 79.5 nm with 300 GB of data storage

File System Implementation

Boot: Etymology tidbit

Pull your self pull yourself up by your (own) bootstraps

- To improve your situation in life by your own efforts

- Bring at least a portion of the OS (kernel) into main memory
  - Then have a processor execute it

On-disk structures used to implement a file system (1)

- **Boot control block**
  - Information needed to boot an OS from that volume
- **Volume control block**: Volume information
  - Number of blocks in the partition
  - Size of the blocks
  - Free-block count/pointers
  - Free file-control-block count/pointers
  - UFS: **super-block** Windows: **Master file table**

On-disk structures used to implement a file system (2)

- Directory structure to organize files
  - One per file system
- Per file file-control-block
  - Contains details about individual files

In memory structures used to improve performance via caching

- **Mount** table
  - Information about each mounted volume
- Directory structure **cache**
  - Holds information about recently accessed directories
- System-wide **open file** table
  - File control block of each open file
- **Buffers** to hold file-system blocks
  - To read and write to storage

Creation of a new file

- **Allocate** a file-control block (FCB)
- Read appropriate directory into memory
  - Directory is just a file in UNIX
    - Special **type** field
- **Update** directory with new file name and FCB
- Write directory back to disk

Partitions: A disk can be sliced into multiple partitions

- **Cooked**
  - Has a file system
- **Raw**
  - No file system
  - UNIX swap space uses this
  - Hold information needed by disk RAID (***R****edundant* ***A****rray of* ***I****ndependent* ***D****isks*) systems

Boot information can be stored in a separate partition

- Usually a **sequential** series of blocks
  - Loaded as an image into memory
- Image execution starts at a predefined location

DIRECTORY IMPLEMENTATION

Directory Implementation

- Linear List
- Hash Table

Directory Implementation
Linear List

- File names with pointers to data blocks
- Simple to program
  - Inefficient and slow execution
- Finding a file requires a **linear search**
- Sorted list
  - Complicates creation and deletion
- Tree data structures might help here
  - B-Tree

Directory implementation: Hash table

- Linear list **and** a hash table is maintained
- Key computed from file name
  - Hash table value returns pointer to entry in linear list
- Things to consider
  1. Account for **collisions** in the hash space
  2. Need to **rehash** the hash table when the number of entries exceed the limit

Allocation methods: Objective and approaches

- How to allocate space for files such that:
  - Disk space is utilized effectively
  - File is accessed **quickly**

- Major Methods
  - Contiguous
  - Linked
  - Indexed

Contiguous Allocations

Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk
  - If file is of size **n** blocks and starts at location **b**
    - Occupies blocks **b**, **b+1**, …, **b+n-1**

- Disk head movements
  - None for moving from block **b** to (**b+1**)
  - Only when moving to a different track

Sequential and direct access in contiguous allocations

- Sequential accesses
  - Remember *disk address* of the last referenced block
  - When needed, read the next block

- **Direct access** to block `i` of file that starts at block `b`

  `b + i`

Contiguous allocations suffer from external fragmentation

- Free space is broken up into chunks
  - Space is **fragmented** into holes
- Largest continuous chunk may be insufficient for meeting request
- **Compaction** is very slow on large disks
  - Needs several hours

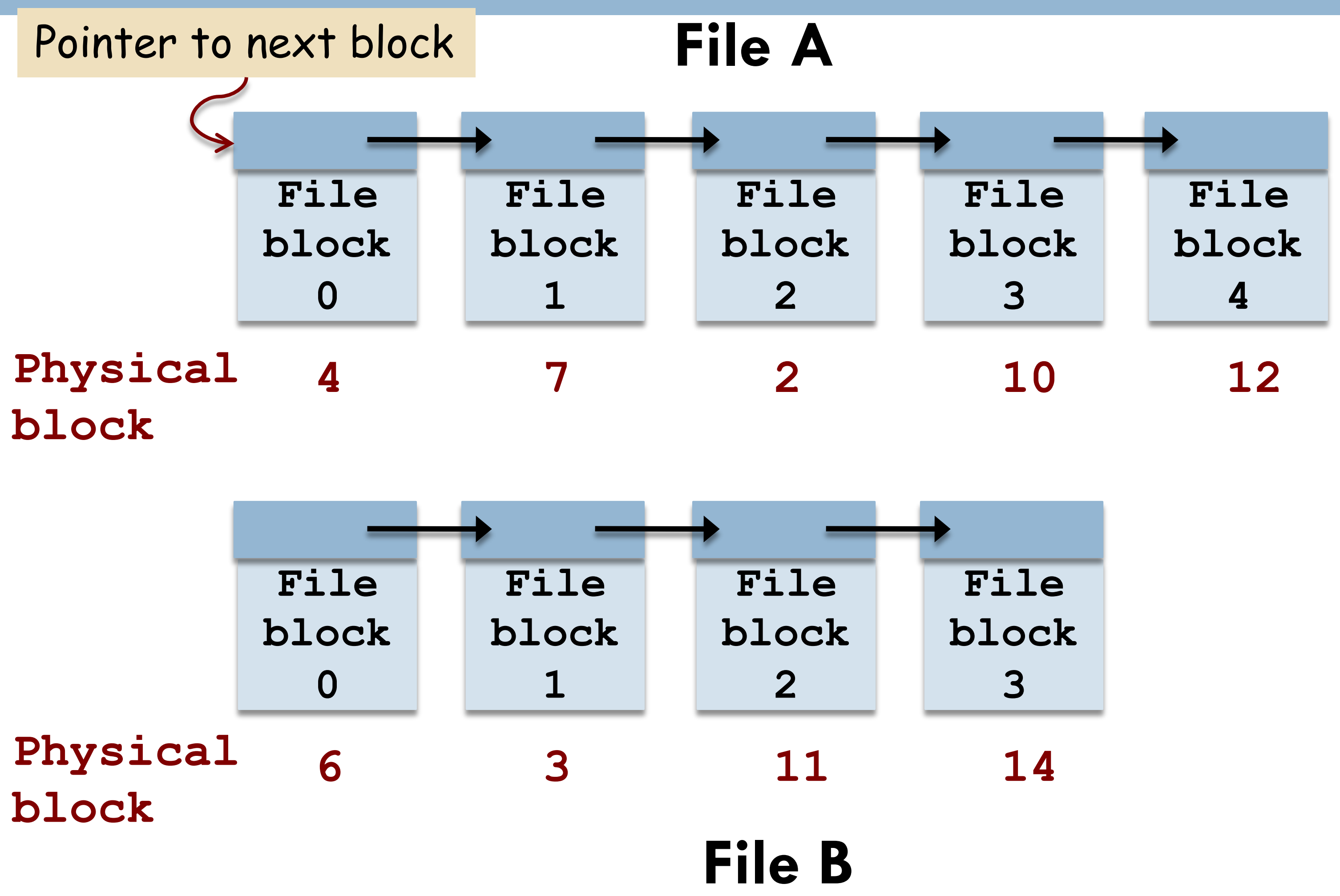
Determining how much space is needed for a file is another problem

- **Preallocate** if eventual size of file is known?
  - Inefficient if file grows very slowly
    - Much of the allocated space is unused for a long time

- Modified contiguous allocation scheme
  - Allocate space in a continuous chunk initially
  - When space runs out allocate another set of chunks (**extent**)

LINKED ALLOCATIONS

Linked Allocation: Each file is a linked list of disk blocks



Linked List Allocations: Advantages

- **Every** disk block can be used
  - No space is lost in external fragmentation
- Sufficient for directory entry to merely store *disk address of first block*
  - Rest can be found starting there

Linked List Allocation: Disadvantages

- Used effectively only for sequential accesses
  - Extremely **slow random access**
- Space in each block set aside for pointers
  - Each file requires *slightly more space*
- Reliability
  - What if a pointer is lost or damaged?

Linked List Allocations: Reading and writing files is much less efficient

- Amount of data storage in block is no longer a **power of two**
  - Pointer takes up some space
- **Peculiar size** is less efficient
  - Programs read/write in blocks that is a power of two

Linked list allocation: Take pointers from disk block and put in table

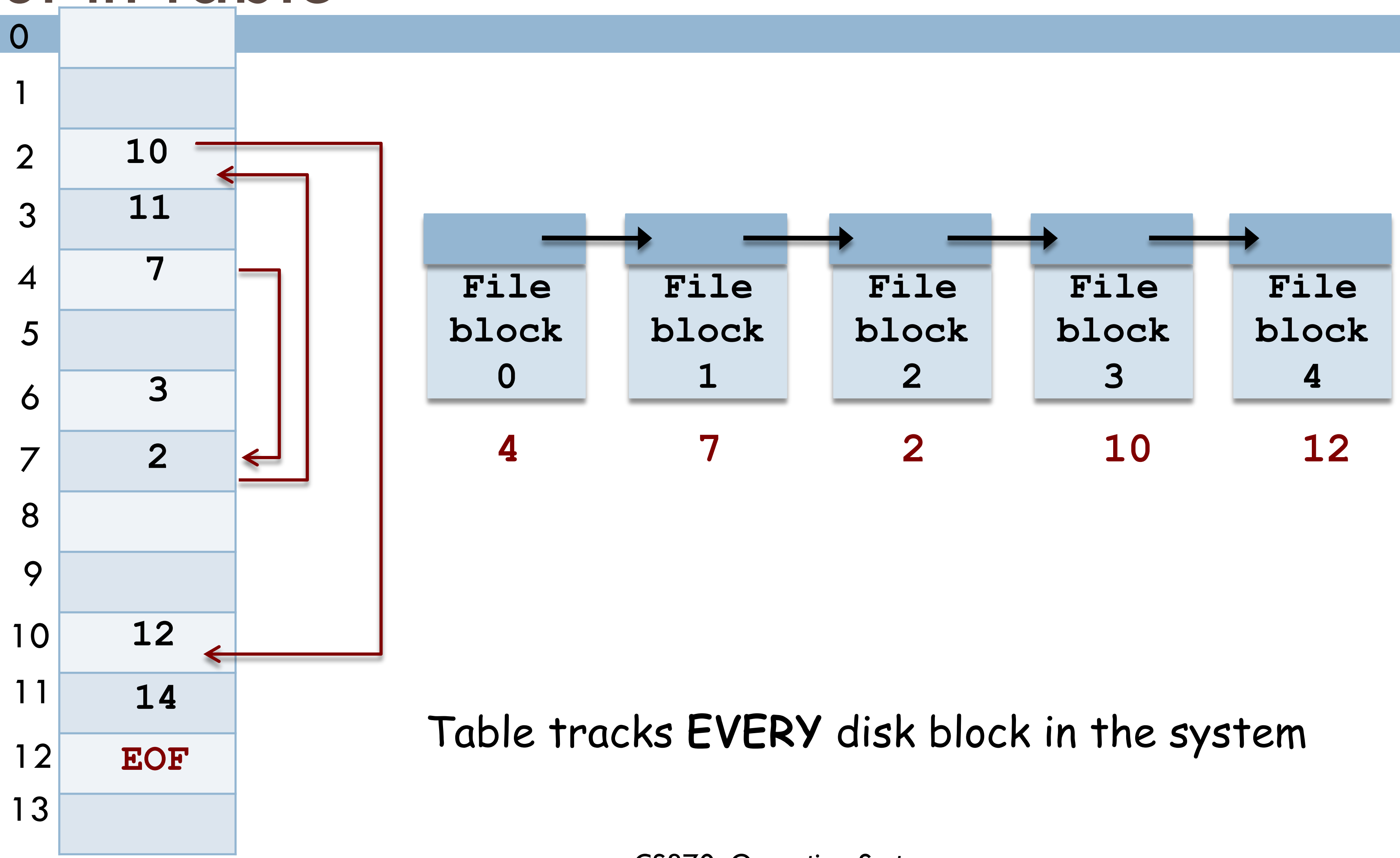


Table tracks **EVERY** disk block in the system

Linked list allocation using an index

- **Entire** disk block is available for data
- Random access is much easier
  - Chain must still be followed
    - But this chain could be cached in memory
- MS-DOS and OS/2 operating systems
  - Use such a file allocation table (FAT)

Linked list allocation using an index: Disadvantages

- Table must be cached **in memory** for efficient access
- A large disk will have a large number of data blocks
  - Table consumes a large amount of physical memory

The contents of this slide-set are based on the following references

- *Avi Silberschatz, Peter Galvin, Greg Gagne. Operating Systems Concepts, 9th edition. John Wiley & Sons, Inc. ISBN-13: 978-1118063330.* [Chapter 11]
- *Andrew S Tanenbaum. Modern Operating Systems. 4th Edition, 2014. Prentice Hall. ISBN: 013359162X/ 978-0133591620. [Chapter 4]*
- *Kay Robbins & Steve Robbins. Unix Systems Programming, 2nd edition, Prentice Hall ISBN-13: 978-0-13-042411-2.* [Chapter 4]
- Hard Drives. http://www.explainthatstuff.com/harddrive.html
- http://en.wikipedia.org/wiki/DVD