Operating Systems W13L1 - File Systems II (ctd)

Class	Operating Systems
Created	@Nov 23, 2020 7:12 PM
Materials	
Reviewed	☐
Type	Lecture

Implementing File Systems

▼ #1 - Contiguous Allocation

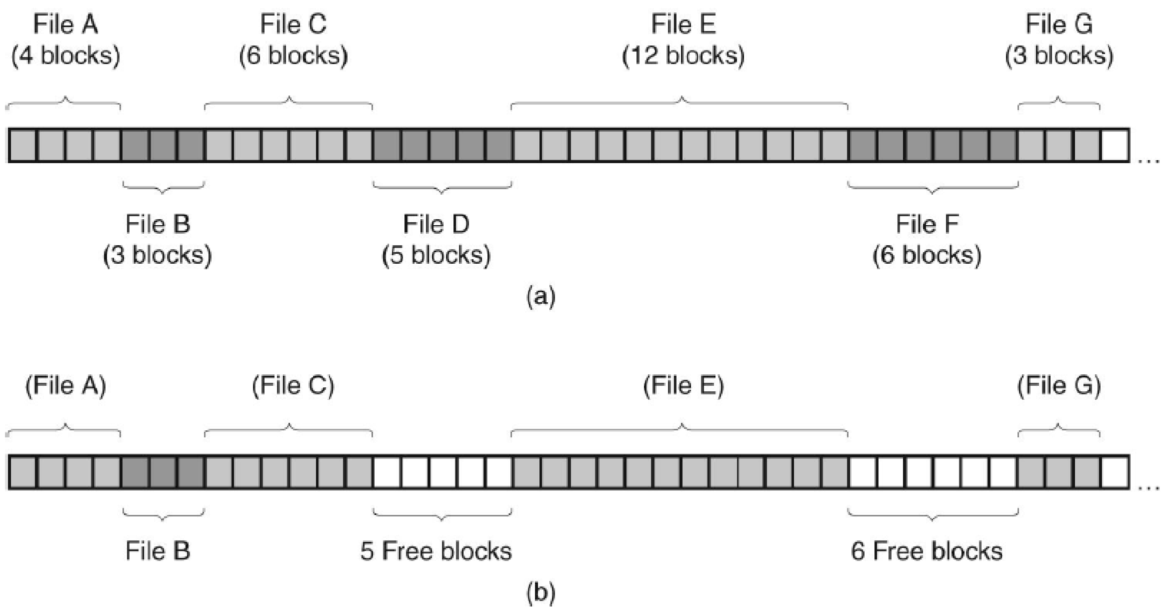
Each file is stored as a contiguous run of disk blocks

Advantages include...

- Simple implementation
- Great read speed

Disadvantages include...

- Disk could become fragmented
- Need to know size of file when created



After files D and F were deleted

▼ #2 - LinkedList Allocation

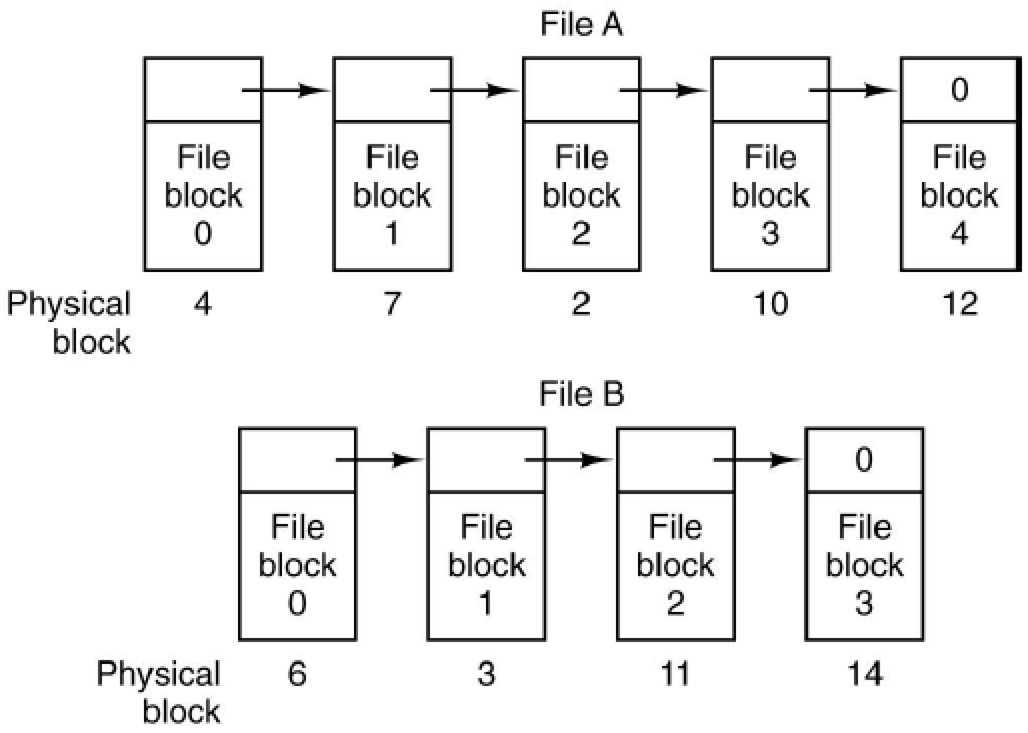
The first 'word' of each block is a pointer to the next

Advantages include...

- No (external) fragentation because empty blocks can be added to the next
- Directory only needs to store disk address of first block

Disadvantages include...

- Random access is extremely slow
- Data storage is no longer a power of two since pointer takes up a few bytes (Some binary numbers won't map to anything as a result)

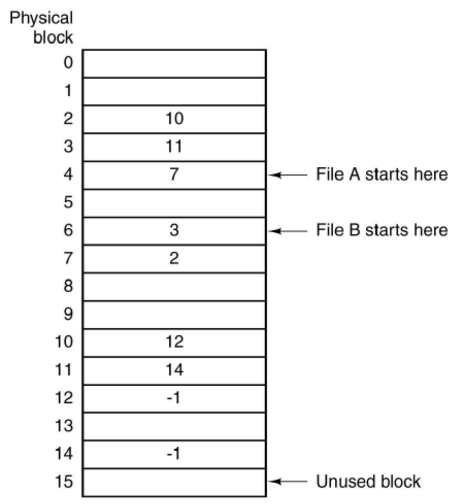


▼ #3 - Linked List Using a Table in Memory

Put pointer for each word into a table in memory. The table is called the File Allocation Table (or FAT)

Advantage is that it allows for an entire block to be used, versus #2 where part of the block was used to point to the next block

The main disadvantage is that this doesn't scale, since the table needs to be in memory all the time



•This table is called:
File Allocation Table (FAT)
•Directory entry needs to keep
a single integer:
(the start block number)

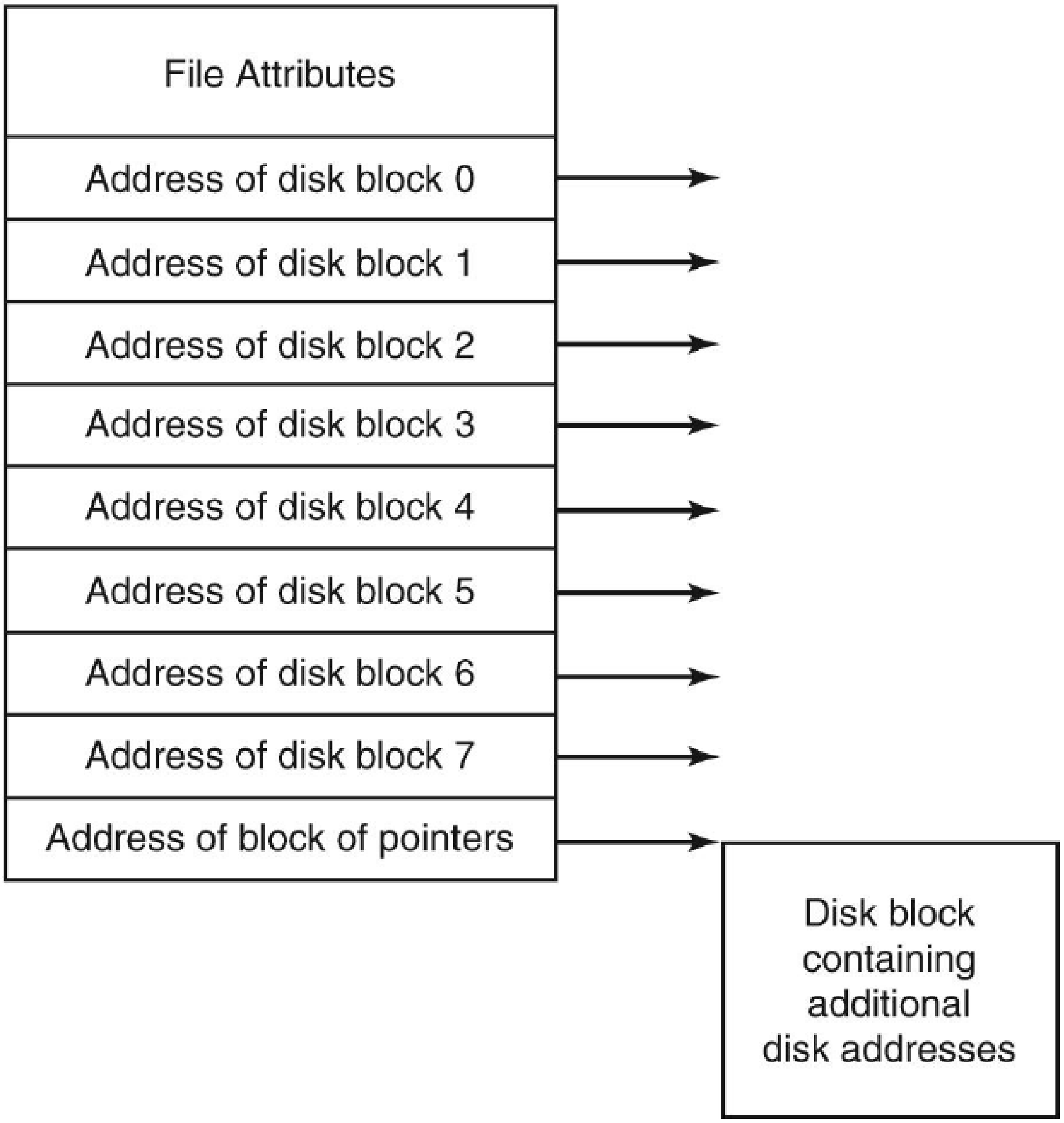
- The figure shows File A occupying blocks: 4, 7, 2,10, and 12.
- (-1) indicates the end of a file

Main drawback: Does not scale to large disks because the table needs to be in memory all the time.

[-1 indicates end of file]

▼ #4 - I Nodes

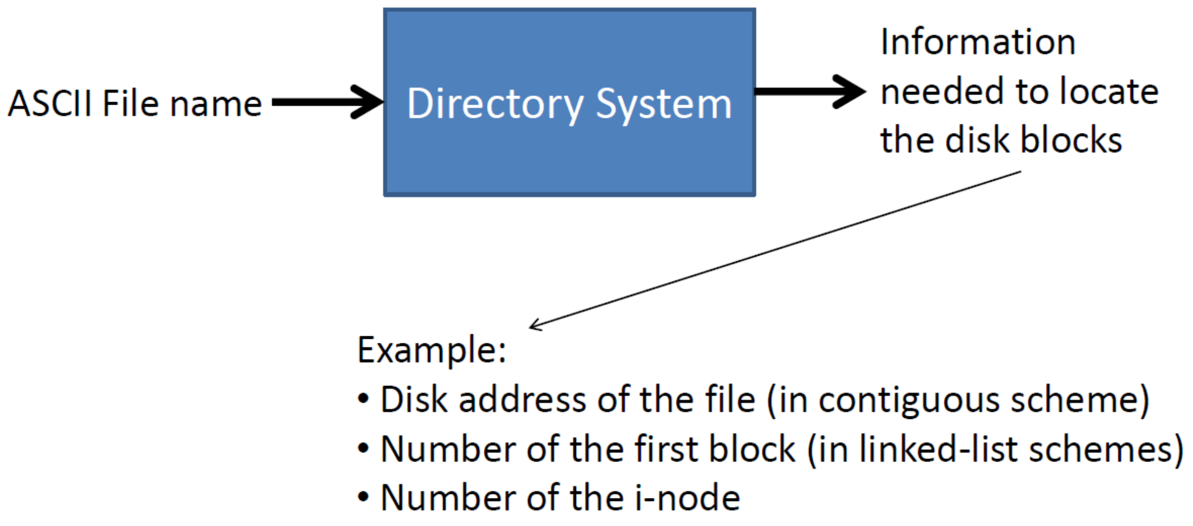
- I-Nodes are data strctures associated with each file and stores attributes and disk addresses of the file blocks
- It only needs to be in memory when a file is opened



How does the OS now where specific I-Nodes are? **Directories!**

Directories

▼ Implementation diagram

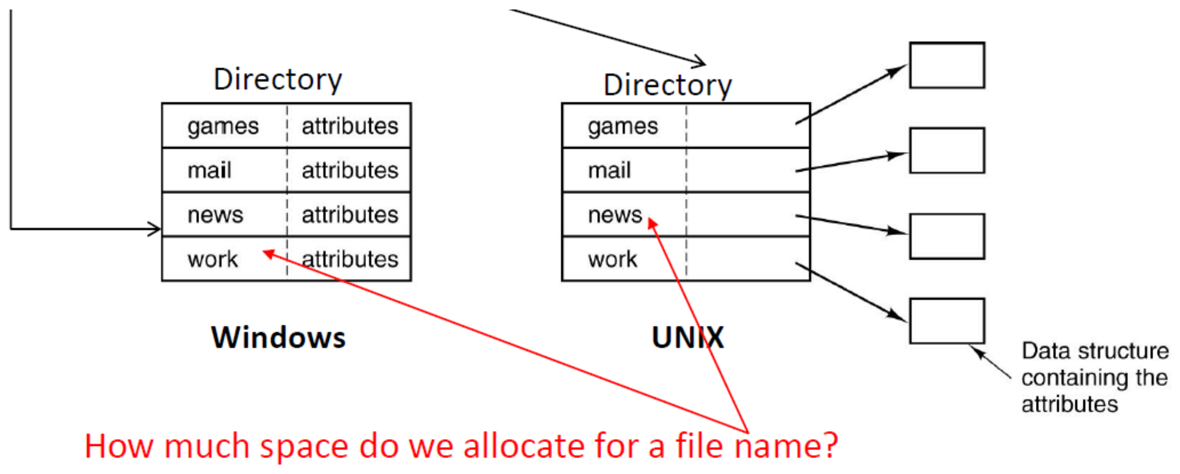

ASCII File name
Directory System
Information needed to locate the disk blocks
Example:
• Disk address of the file (in contiguous scheme)
• Number of the first block (in linked-list schemes)
• Number of the i-node

- With the use of I-Nodes, the OS knows where to go

Implementing Directories

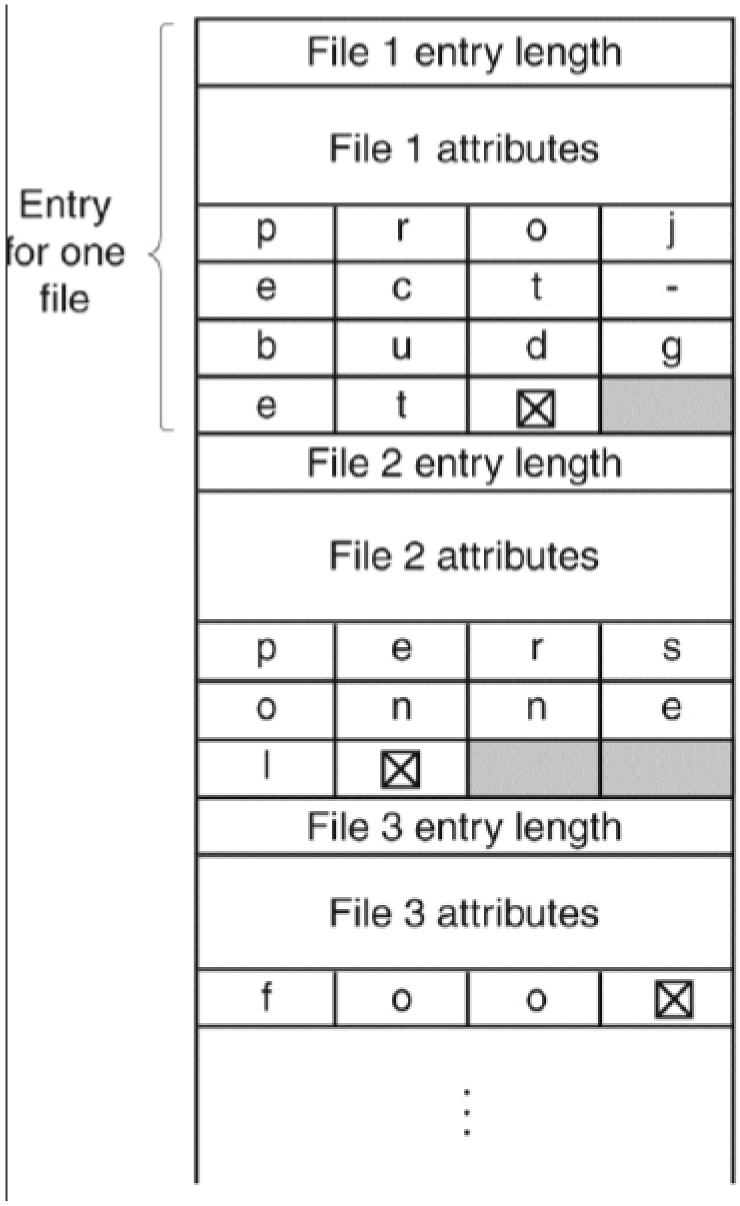
- Need to know where to store the attributes of a file / multiple files

▼ Diagram

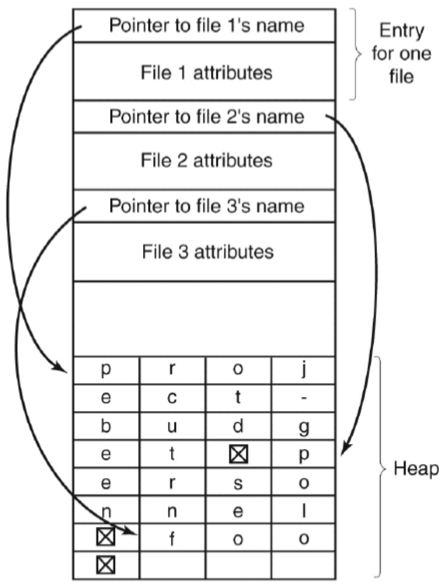


▼ Variable-Length Filenames

The disavantage to this is that entries are not the same length, varying size gaps upon file removal, and page faults can occur since a directory might span several pages



▼ Fixed-Length Diagram

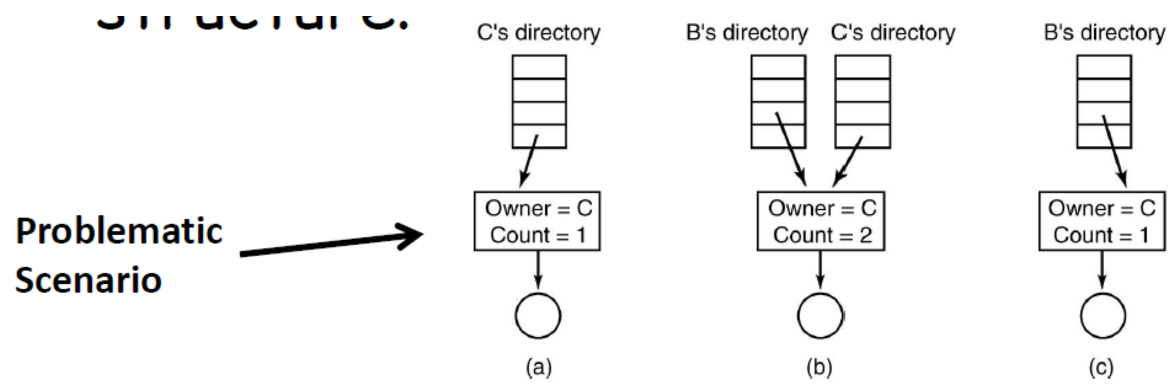


•Keep directory entries fixed length
•Keep filenames in a heap at the end of the directory.
• Page faults can still occur while accessing filenames.

- For long directories, linear search is slow → We can use hasing and caching (method 1)
- **Shared Files** is a more efficient approach — Files that apear simultaneously in different directories

▼ Problematic Scenario

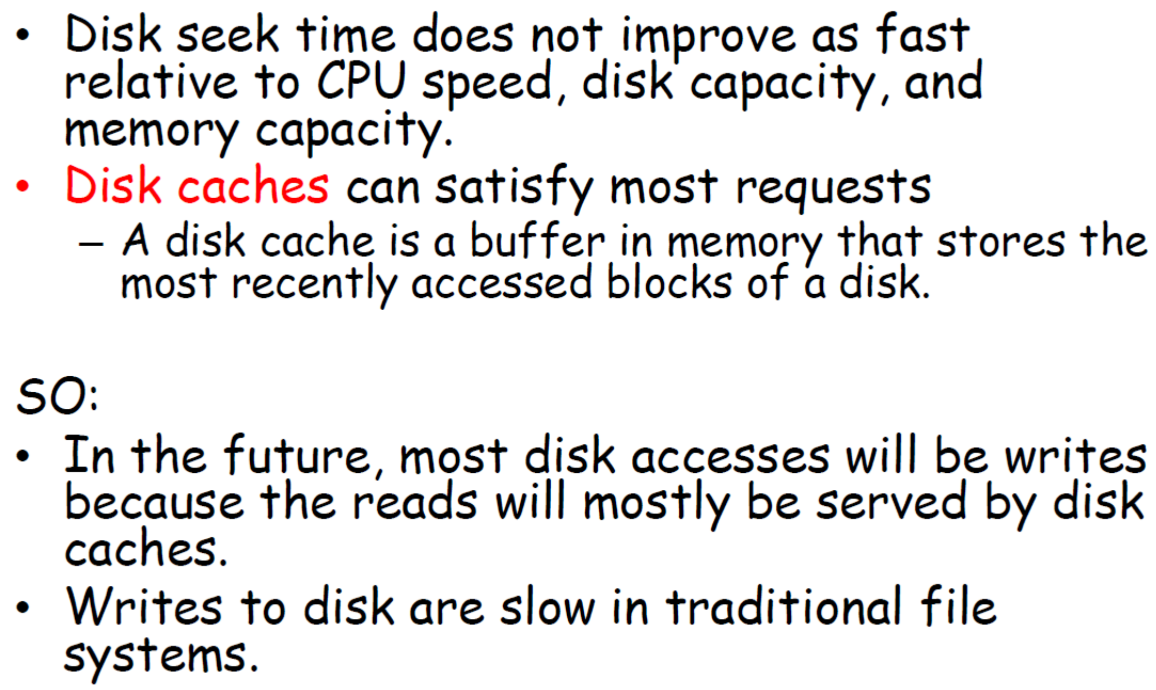
The issue is that two things are accessing the same file at once



- Another approach to shared files is **symbolic linking** (method 2)
  - Create a LINK file that contains pathname to a linked file
  - Main drawback is that there is a significant amount of extra overhead required

Special File Systems

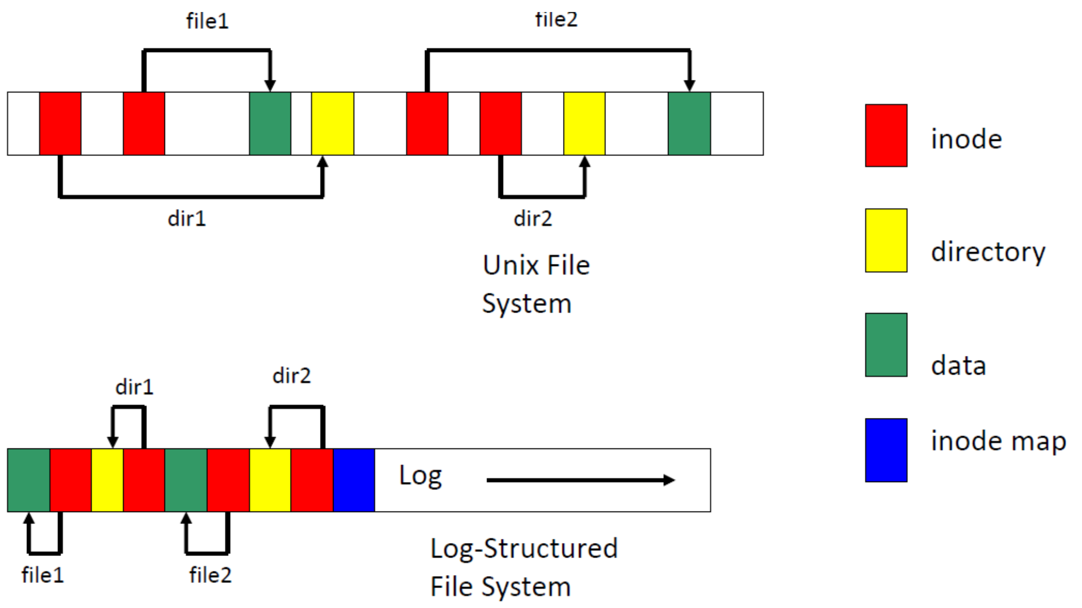
▼ Log-Structured File Systems


- Disk seek time does not improve as fast relative to CPU speed, disk capacity, and memory capacity.
- Disk caches can satisfy most requests
  - A disk cache is a buffer in memory that stores the most recently accessed blocks of a disk.

SO:
- In the future, most disk accesses will be writes because the reads will mostly be served by disk caches.
- Writes to disk are slow in traditional file systems.

- Log as a data structure → Data is *always added to the **end***
- Must be edited periodically or necessarily; must be careful when editing
- Need an I-Node map to keep track of things
  - This map is kept on disk and also cached

▼ Log Structured vs. Traditional Unix


file1
file2
dir1
dir2
Unix File System
dir1
dir2
Log
file1
file2
Log-Structured File System
inode
directory
data
inode map