

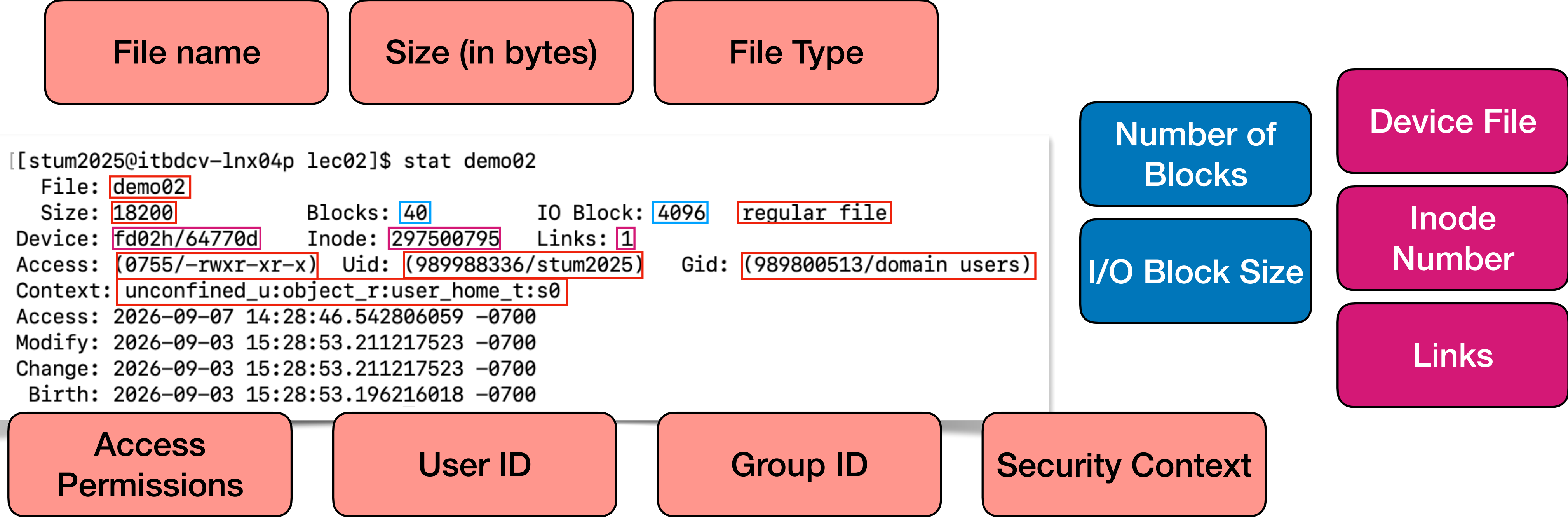
(Re-)Introducing File Systems

**Procs (programming) due tonight
at midnight!
Code Review period starts
tomorrow!**

Outline

- Reviewing files and file types
- Revisiting file system structures
- OS file system considerations

Files and File Types



Files and File Types

- Files may be classified as “regular files” or “directories,” both of which have several properties regarding their name, size, type, backing device, dates of access, etc...
- File permissions are organized in terms of “owner” permissions, “group” permissions, and “world” permissions (with a set of special bits to indicate certain special privileges) ➡ can be changed with `chmod` (and appropriate access)
- Per-file inodes indicate where in the file system this file can be found (and which backing device should be used to access it)

Chat with your neighbor(s)!

Takeaway: there is a different conception of “file blocks” that live in memory versus “disk blocks” where the files live! It is up to the OS to manage both!

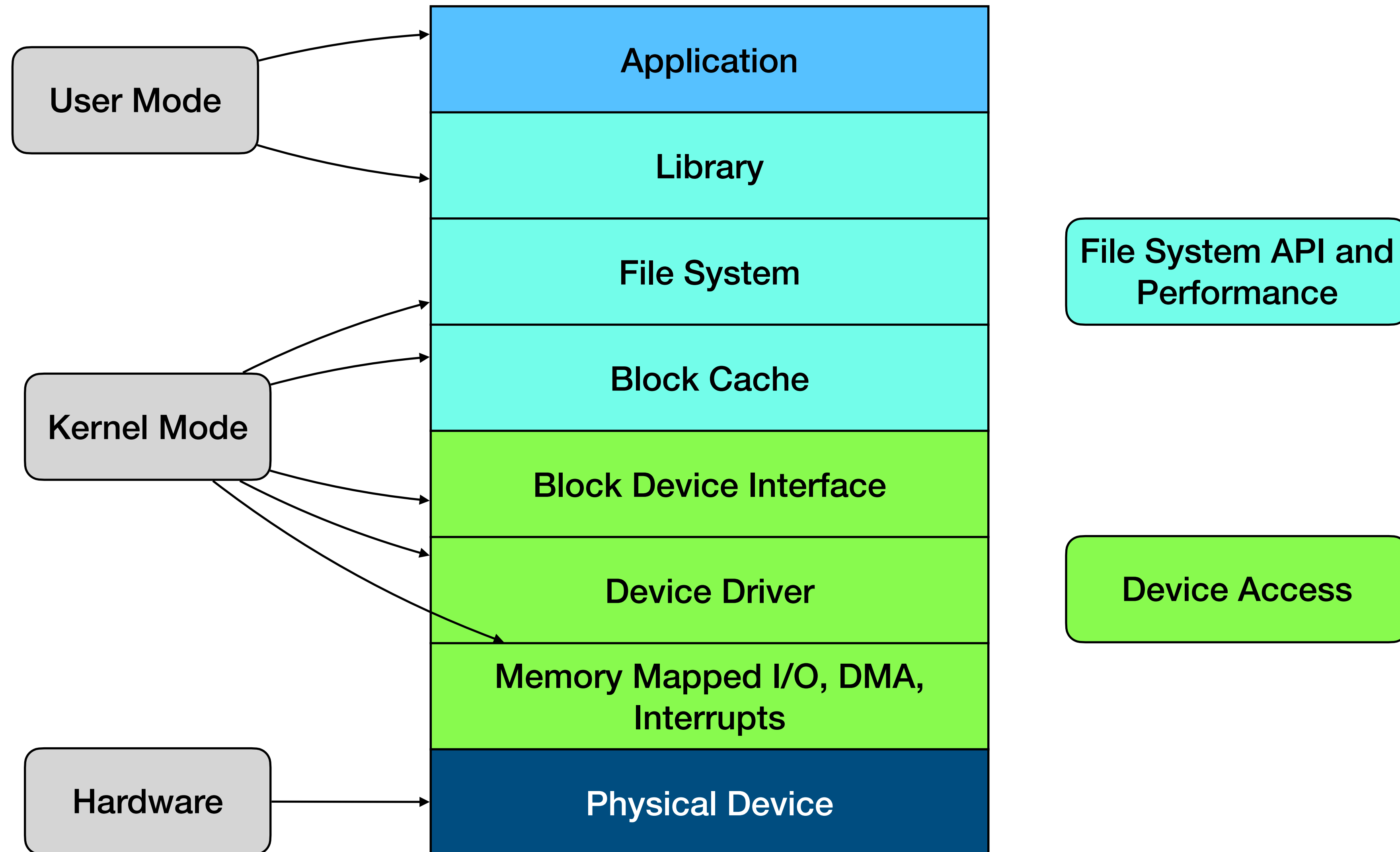
The math that describes the relationship between the file (as a size in terms of bytes), the number of blocks, and the I/O block size are not consistent. What might these values describe?

```
[[stum2025@itbdcv-lnx04p lec02]$ stat demo02
  File: demo02
  Size: 18200          Blocks: 40          IO Block: 4096   regular file
Device: fd02h/64770d  Inode: 297500795   Links: 1
Access: (0755/-rwxr-xr-x)  Uid: (989988336/stum2025)   Gid: (989800513/domain users)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2026-09-07 14:28:46.542806059 -0700
Modify: 2026-09-03 15:28:53.211217523 -0700
Change: 2026-09-03 15:28:53.211217523 -0700
 Birth: 2026-09-03 15:28:53.196216018 -0700
```

IO Block is a file system feature of how big should be for optimal transfer size

Blocks describes the number of 512B blocks used for this file in memory

The Operating System as a FS and Disk Manager

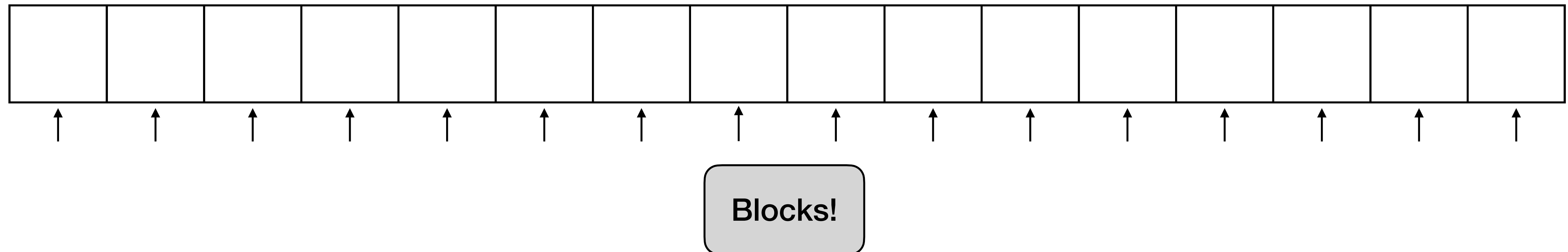


The Operating System as a FS and Disk Manager

- As the operating system is tasked with maintaining the file system, it must both create the file system subsystem (in the same way that it declares/creates the processes subsystem)
- The operating system also manages hardware ➡ because files and the file system should appear “persistent,” this means close management of the persistent storage devices (e.g., the disk)
- To read/write data from disk, the operating system uses a “driver” to create and handle disk interrupts ➡ a command is issued to the device driver to start the disk read, and the device will raise an interrupt when the read completes (e.g., the disk read is *asynchronous*)

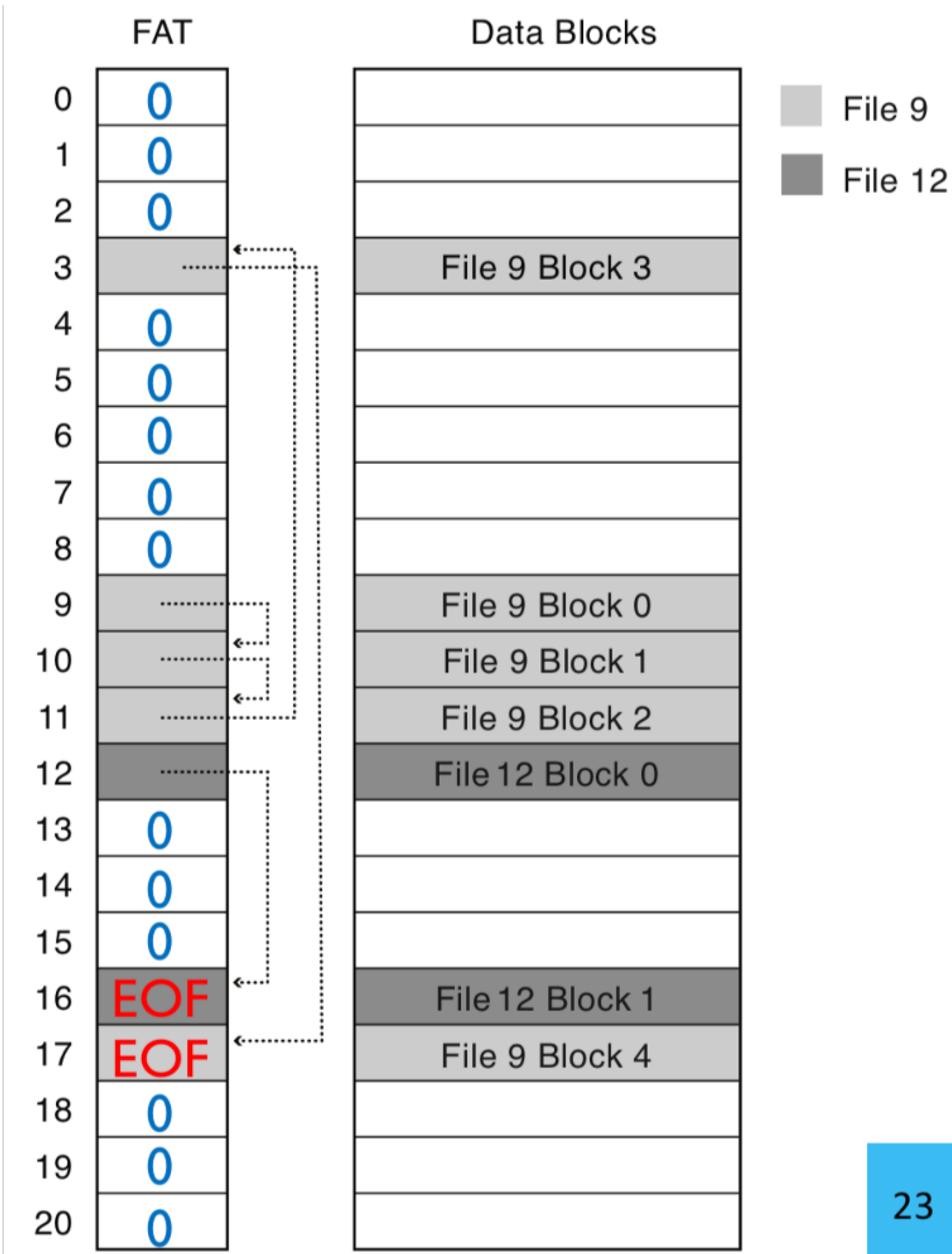
File System Disk Layouts

Disk (and other storage devices) are treated as “block devices” by the operating system (as they are typically accessed by block)



File System Disk Layouts

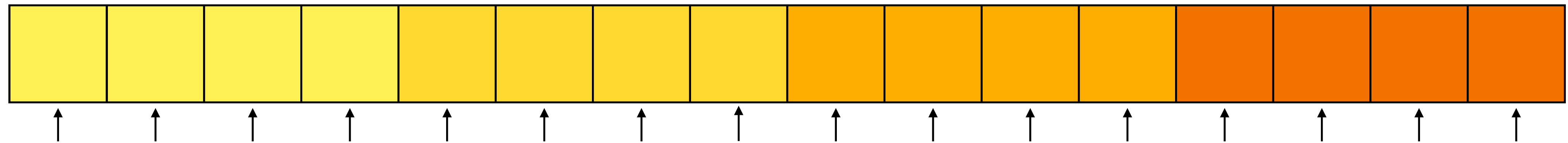
When writing to disk, the operating system must write an entire block (no random access), so most operating systems read in the entire block first to re-write just the small data to be written



FAT File System

File System Disk Layouts

Disk (and other storage devices) are treated as “block devices” by the operating system (as they are typically accessed by block)

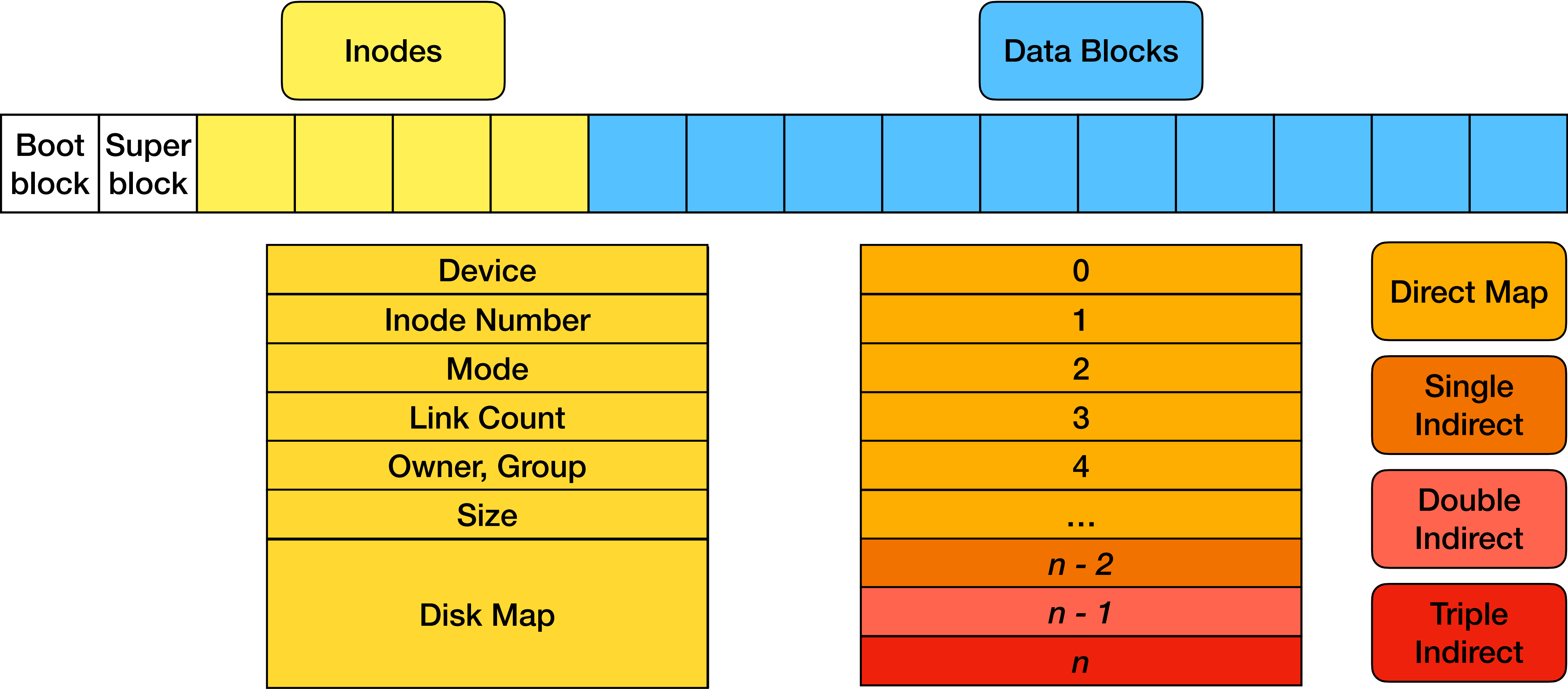


Modern operating systems organize groups of blocks into “segments” or “file system blocks” as storage technology now supports larger block sizes

Blocks!

To support backwards compatibility, many disks still support 512B operations, which are referred to as “logical” segment addresses

File System Disk Layouts (S5FS)



Chat with your neighbor(s)!

In practice, HDDs are becoming less common, but persistent storage devices remain slow to program and still require a special driver/OS kernel stack!

Recall that our disk blocks live on a real hardware device (e.g., a HDD as depicted below). What are some implications of implementing the S5FS file system on a HDD? You may want to think about the file system software stack in your answer!



Disk

File System Disk Layouts (S5FS)

- In practice, the S5FS was an early file system deployed in Unix in the 80s and 90s to increase the maximum file size of a system using single-, double-, and triple-indirect blocks in the node struct
- In S5FS, increasing the size of the file can be done flexibly, but requires the OS to maintain a list of all free blocks on the disk... this is inefficient, especially as the disk space starts to fill and the options of next blocks to allocate becomes more sparse!
- In general, pointers to arbitrary places in the disk is an inefficient way to read large files!
- Note, S5FS is introduced as FFS in CS105... we will contrast S5FS and FFS in more depth next week!

Operating Systems Considerations

- The role of the operating system is to perform several tasks when implementing the file system
 - (1) The OS needs to manage the placement and layout of data on a disk
 - (2) The OS implements the disk access interface
 - (3) The OS likely will want to maintain some “cached” copy of recently accessed file system data in memory to avoid expensive disk accesses
 - (4) The OS maintains the logic of mapping where data in the cached file system in memory resides and how/where to access this data
- Beyond the file system itself, other subsystems (e.g., the processes subsystem and the memory management subsystem) are effected by the OS’ design of the file system

Summary

- We introduced the idea of different file types, permissions, sizes, links, and other file attributes that can describe a file
- We overviewed a typical software stack that could be used to access the file system
- We reviewed the disk layout discussion from CS105 regarding files and file systems, and alluded to how disks may be thought of in terms of blocks and segments
- We described S5FS in more depth, and alluded to some of the challenges associated with implementing it and other file systems in practice