Today

- File Systems
  - User interaction
  - Virtual File Systems

Review: Course Grade

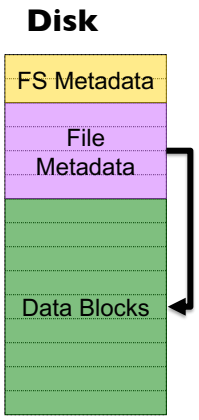
- (50%) OS Programming Projects
- (15%) Individual programming, reading, and writing assignments
- (15%) Midterm Exam
- (15%) Final Exam
- (5%) Participation and attendance

Review

- A disk is a bunch of blocks in which to store data
  - How does a file system (FS) give order and structure to those blocks?
- What is data? What is metadata?
- What metadata do we need to know about the file system?
  - Where is that data stored?
- What information does a FS store about a file?
  - How does it store that information?

Review: Disk

- File System Metadata
  - Format, size of blocks
  - Stored in *superblock*
  - Replicated
- File Metadata
  - Inode table
- Data Blocks

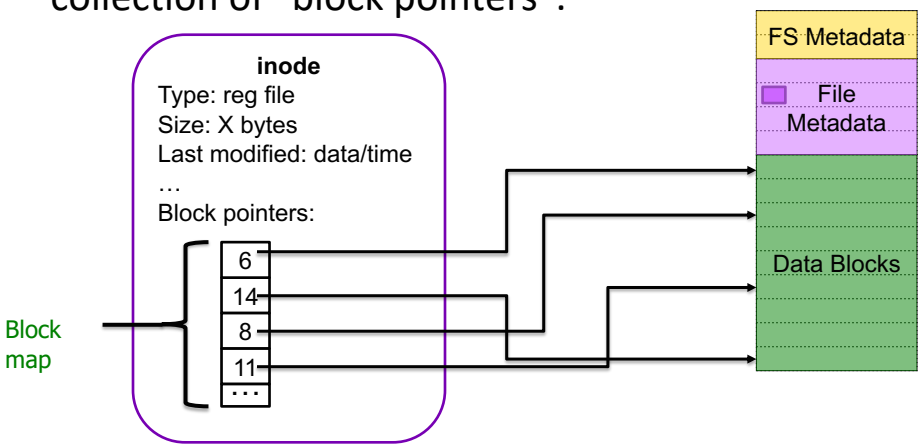


Review: Data vs. Metadata

- **Data**: the files, directories, and other stuff for the user
  - data for the user or programs
- **Metadata**: information stored *about* the data
  - For the OS to make the file system work
  - Examples for entire FS: What type of FS is it? How large is it?
  - Example for one file: Where are the file's blocks located on disk?
- Both data and metadata stored together on disk!

Review: inodes

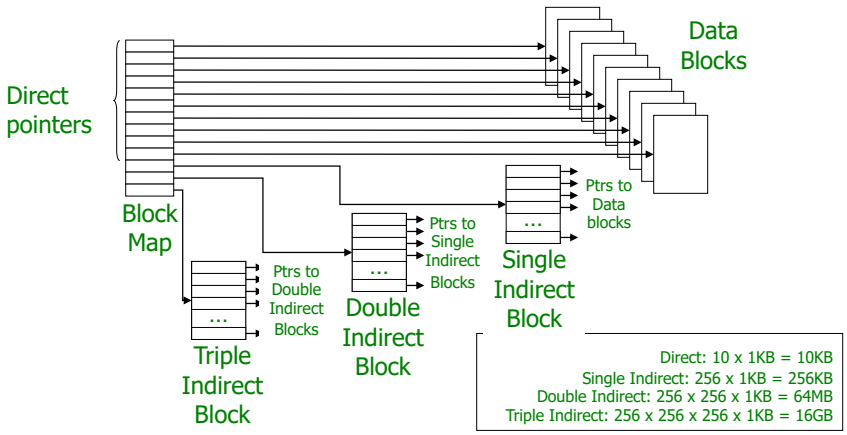
- In the inode structure for a file, there is a collection of "block pointers".



More Review

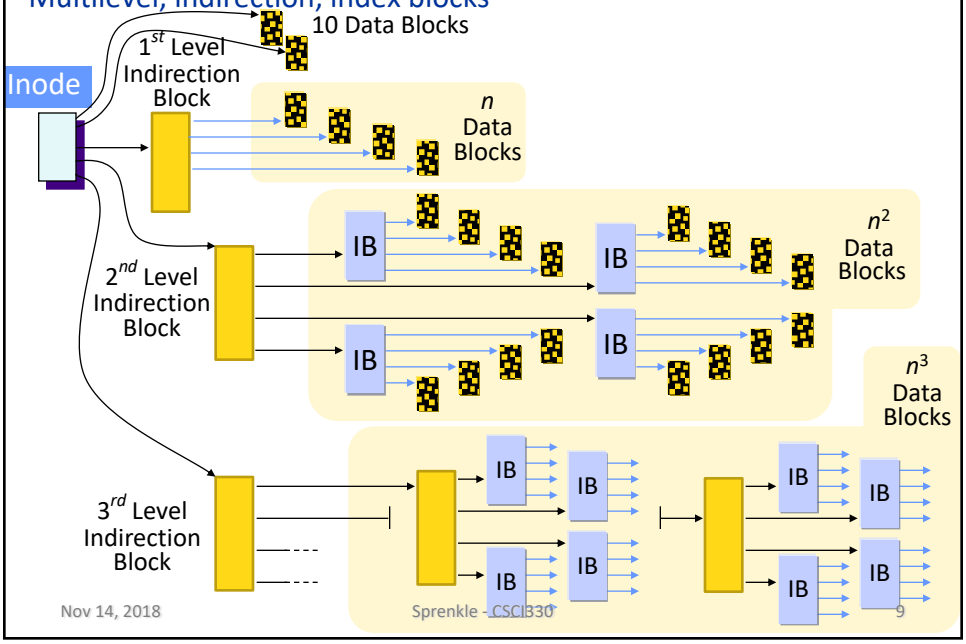
- How do inodes handle the wide variety of file sizes?
  - What are the benefits of this design?
- What file *types* does the file system distinguish?
- How are files within the file system structured?
  - How are they structured in our OS project? On most modern OSs?
- Given the full path/filename, how do we find that file (and its information)?

Review: Block Pointers: Multi-Level Table

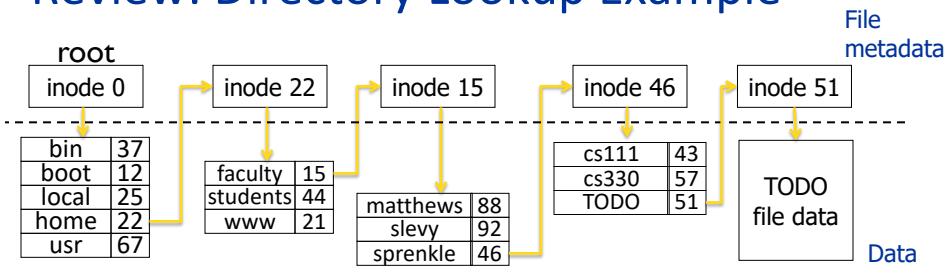


Indexed Allocation in UNIX

Multilevel, indirection, index blocks



Review: Directory Lookup Example



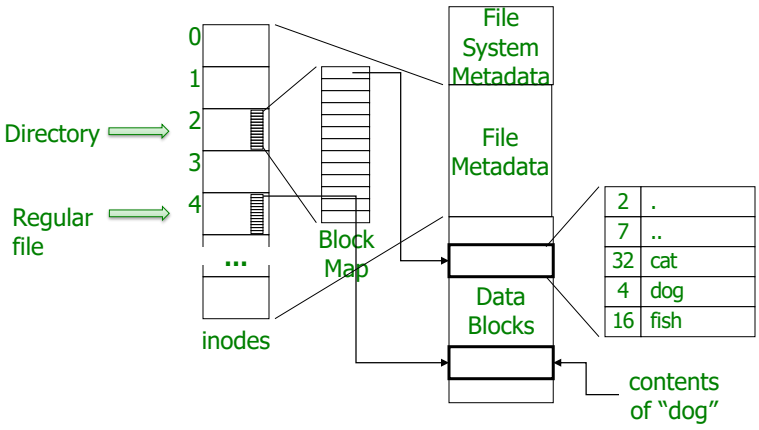
- To OS, file type: regular or directory?
- Given pathname: /home/faculty/sprenkle/TODO
  1. Inode 0 block map points to data block(s) of root directory
  2. Look up "home" in root directory to get inode 22
  3. Inode 22 block map points to data block(s) of home directory
  4. Look up "faculty" in home directory to get inode 15

  …

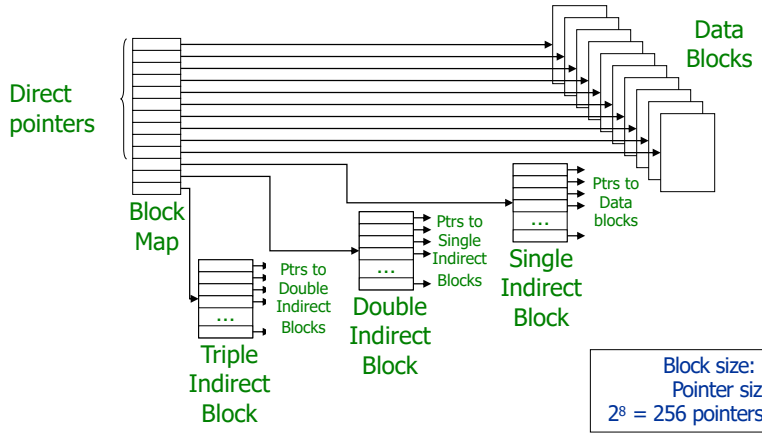
Optimizing Finding Files

- Starting at the root every time is expensive
- Maintain the notion of the *current working directory* (CWD)
- Users can now specify *relative* file names
- OS can cache the data blocks of CWD

Review: The Big Picture



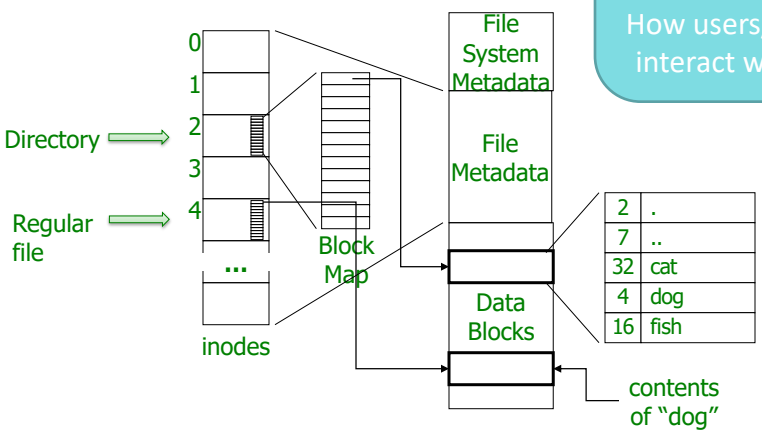
Suppose we want to store a 100 KB file. How many bytes of metadata will we need for our block map?



File Systems So Far

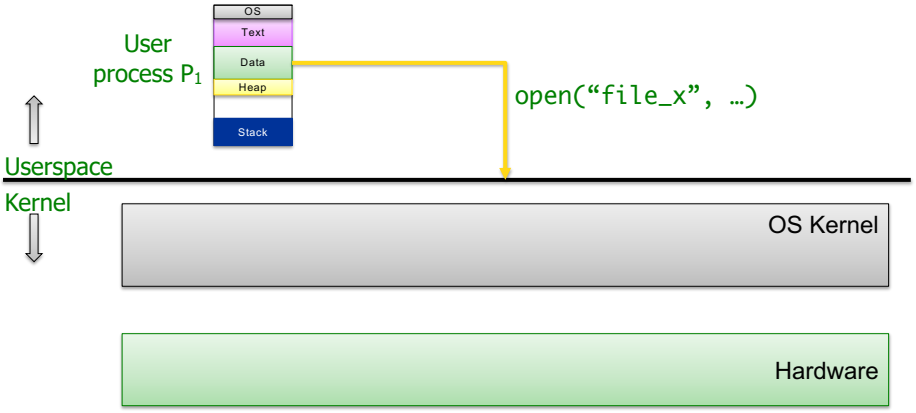
So far, how FS is structured on disk.

Next:
How users/programs interact with them!



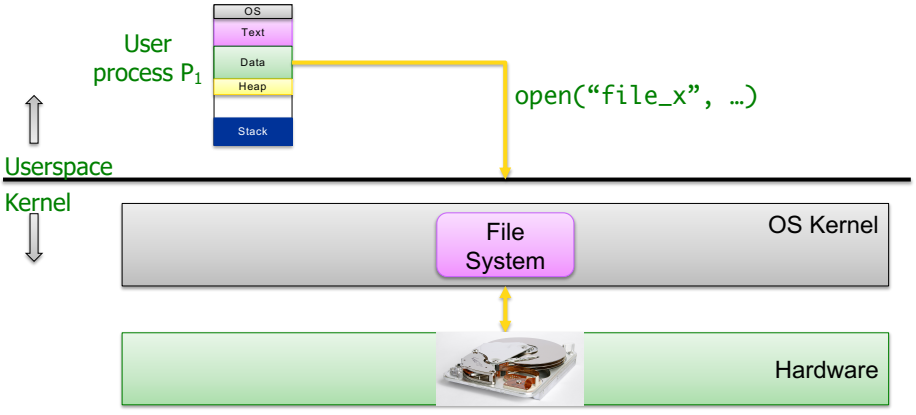
Userspace Perspective

- Userspace processes make system calls to interact with files:



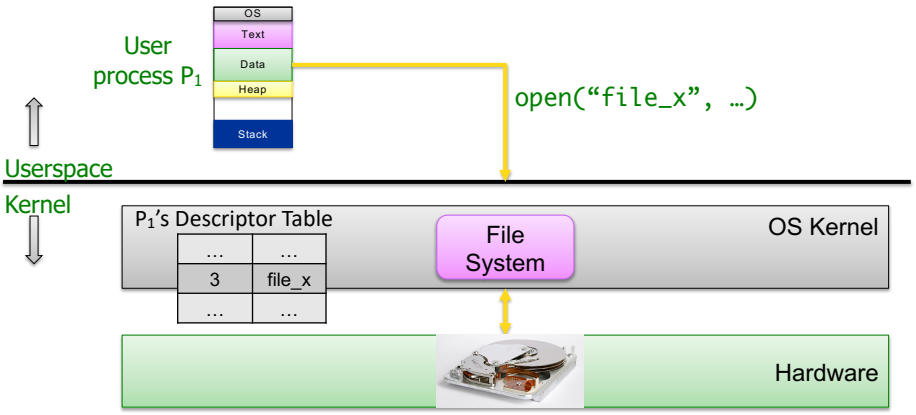
Userspace Perspective

- Userspace processes make system calls to interact with files:



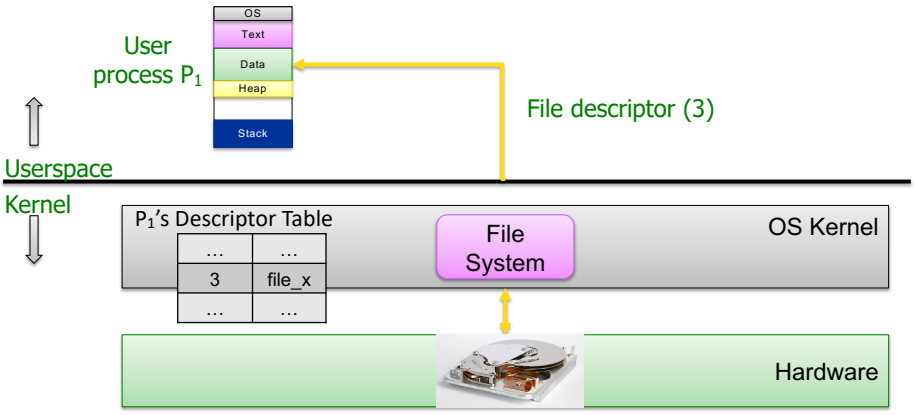
Userspace Perspective

- Userspace processes make system calls to interact with files:



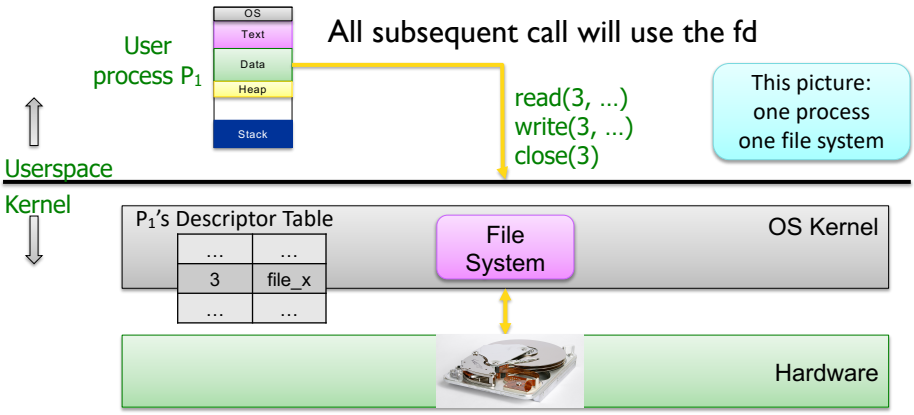
Userspace Perspective

- Userspace processes make system calls to interact with files:



Userspace Perspective

- Userspace processes make system calls to interact with files:



Challenges

- What if we have multiple disks?
  - Example: external hard drive for backups/long-term storage
- What if those disks use different file systems?
  - On the lab machines:
    - /home → mounted from hydros
    - /csdept → mounted from hydros
      - Files for courses, groups (shared)
    - Other directories are local to the lab machine

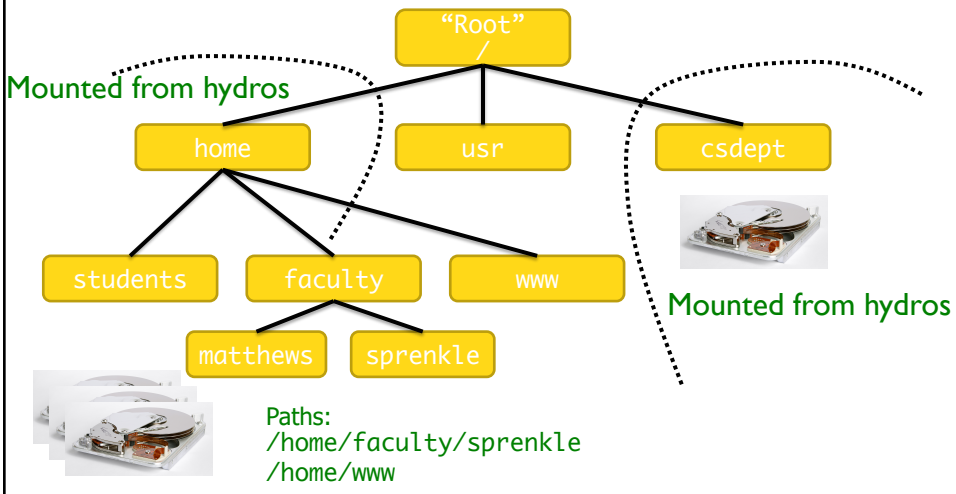
df –Th on knuth

```
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  7.7G     0  7.7G   0% /dev
tmpfs          tmpfs     7.8G     0  7.8G   0% /dev/shm
tmpfs          tmpfs     7.8G  1.5M  7.8G   1% /run
tmpfs          tmpfs     7.8G     0  7.8G   0% /sys/fs/cgroup
/dev/sda4      ext4      108G   20G   83G  20% /
tmpfs          tmpfs     7.8G   24K  7.8G   1% /tmp
/dev/sda2      ext4      976M  182M  728M  20% /boot
/dev/sda1      vfat      200M   18M  183M   9% /boot/efi
tmpfs          tmpfs     1.6G   16K  1.6G   1% /run/user/42
hydros:/home   nfs4      493G  318G  150G  68% /home
tmpfs          tmpfs     1.6G     0  1.6G   0% /run/user/1501
```


df –Th on knuth

```
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  7.7G     0  7.7G   0% /dev
tmpfs          tmpfs     7.8G     0  7.8G   0% /dev/shm
tmpfs          tmpfs     7.8G  1.5M  7.8G   1% /run
tmpfs          tmpfs     7.8G     0  7.8G   0% /sys/fs/cgroup
/dev/sda4      ext4      108G   20G   83G  20% /
tmpfs          tmpfs     7.8G   24K  7.8G   1% /tmp
/dev/sda2      ext4      976M  182M  728M  20% /boot
/dev/sda1      vfat      200M   18M  183M   9% /boot/efi
tmpfs          tmpfs     1.6G   16K  1.6G   1% /run/user/42
hydros:/home   nfs4      493G  318G  150G  68% /home
tmpfs          tmpfs     1.6G     0  1.6G   0% /run/user/1501
```

FS Abstraction: Hierarchical Name Space - Tree



Challenges

- What if we have multiple disks?
  - Example: external hard drive for backups/long-term storage
- What if those disks use different file systems?
  - On the lab machines:
    - /home → mounted from hydros
    - /csdept → mounted from hydros
    - Other directories are local to the lab machine
- **The path in the file tree does NOT say anything about which FS files live on.**

Discussion

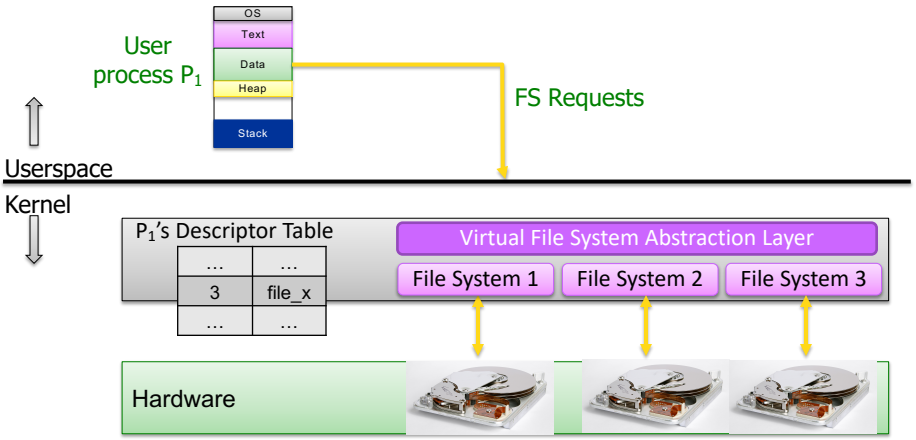
- How would you design the OS such that multiple file systems, possibly on different physical disks, can all share one name space?

This sounds like a job for …

Abstraction!

Virtual File System (VFS) Layer

- Userspace processes make system calls to interact with files:

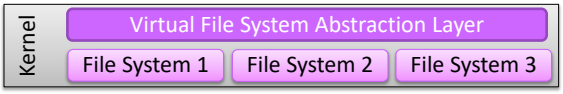


VFS Layer

- Unifies the file name space and paths
  - Paths all start from common root (/) and are passed to VFS layer
  - VFS layer records which paths correspond to which FS
- VFS translates application requests to appropriate low-level FS calls

Analyzing VFS Layer

- Consider how we evaluate OS
- What are the benefits and drawbacks of having a VFS layer?



Analyzing VFS Layer

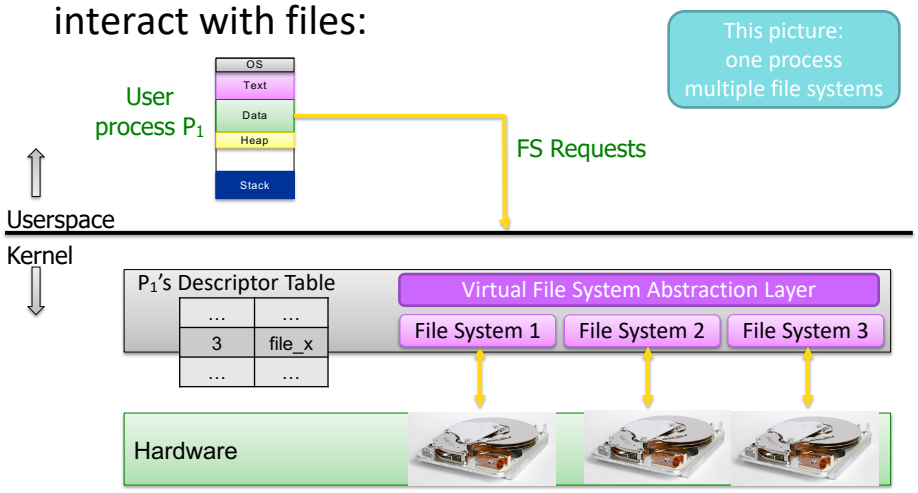
- Benefits
  - user doesn't need to know the details about file systems
  - easy expansion, removal of disks/file systems
- Drawback: layer adds overhead – could slow down performance

Analyzing VFS Layer

- How can we mitigate that performance hit?
  - Caching!
  - Inode Cache
    - Store recently accessed inodes (file/directory info)
  - Directory Cache
    - Full directory path → inode id

Virtual File System (VFS) Layer

- Userspace processes make system calls to interact with files:



Looking Ahead

- Project 4 due in two Mondays