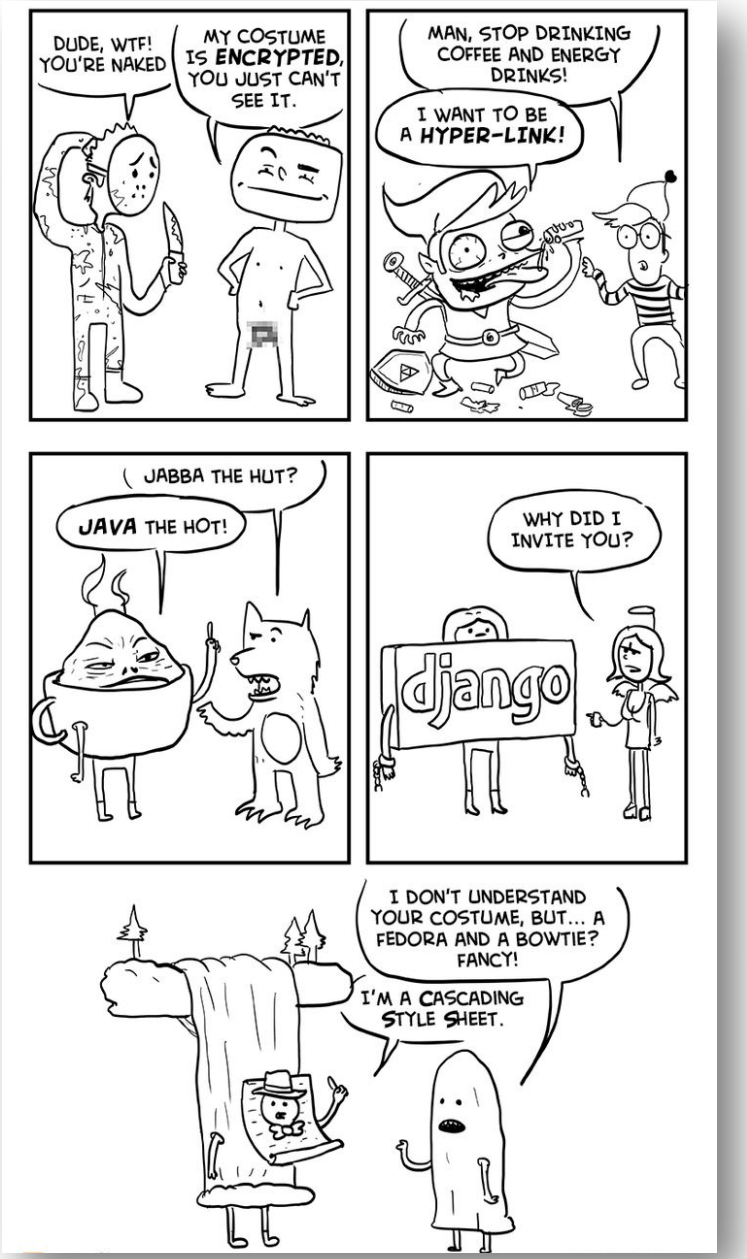
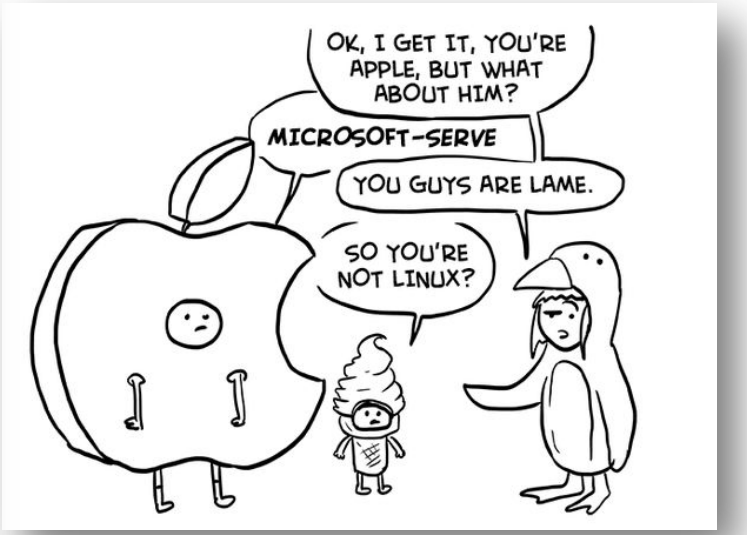
Halloween Costumes for Computer Scientists





https://i.pinimg.com/736x/7f/5a/5e/7f5a5e9349b15ac9506842c8ffc88585.jpg

Operating Systems!

Input/Output (I/O):
Files & File Systems

Prof. Travis Peters

Montana State University

CS 460 - Operating Systems

Fall 2020

https://www.cs.montana.edu/cs460

Some diagrams and notes used in this slide deck have been adapted from Sean Smith's OS courses @ Dartmouth. Thanks, Sean!

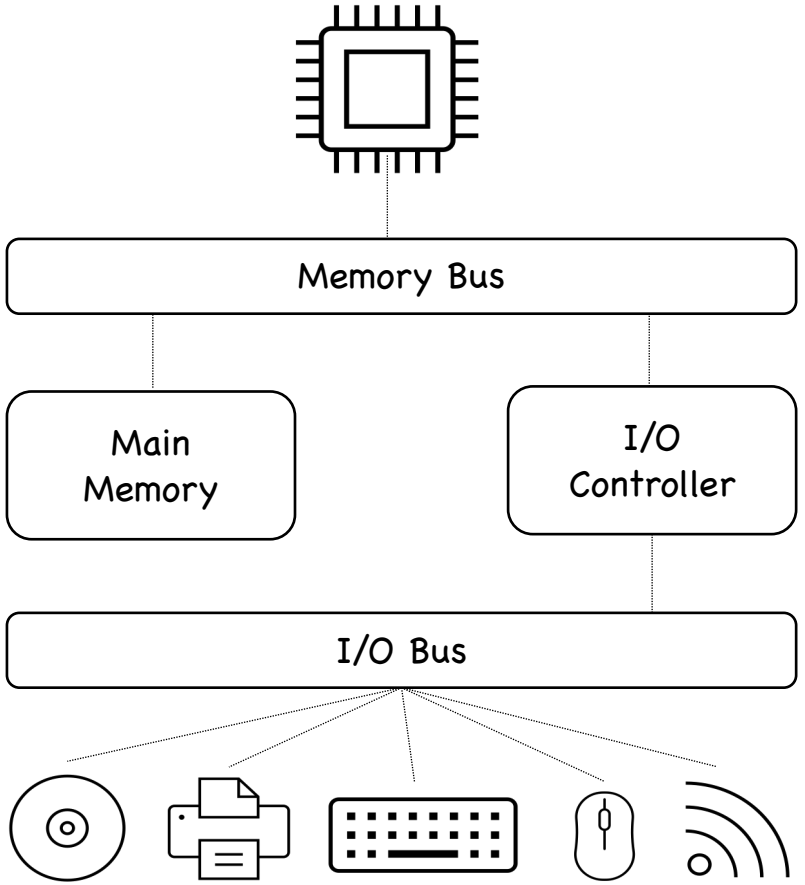
Today

- Announcements
  - *Almost* done grading PA3! ☺
  - Exam 2 details coming soon (11/20? 11/23?)

- Upcoming Deadlines
  - **Project Submission (HARD DEADLINE)**
    Sunday [11/15/2020] @ 11:59 PM (MST)
  - **Project Evaluations (HARD DEADLINE)**
    Wednesday [11/18/2020] @ 11:59 PM (MST)
  - **Exam #2**
    TBD

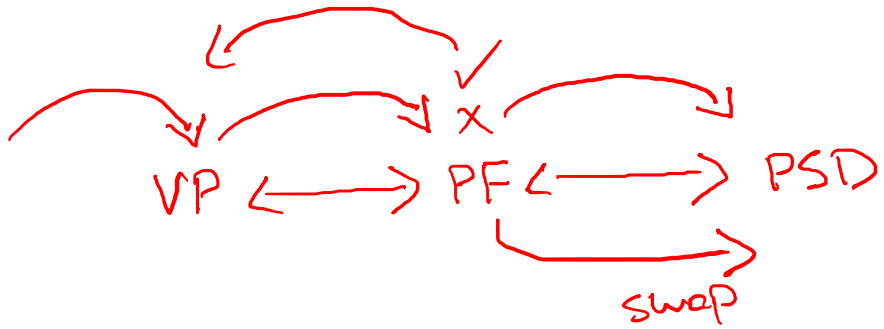
Today

- Agenda
  - ~~Key concepts behind I/O~~
    - ~~*What is I/O? I/O Categories. Memory access.*~~
    - *What is a file? What is a file system? etc.*
  - ~~Secondary Storage~~
    - ~~Emphasis on *disks* and *disk scheduling*~~
  - Files
    - Operations
    - Implementations
  - File System
    - Virtual File Systems
    - Implementations
  - Security



Reflection

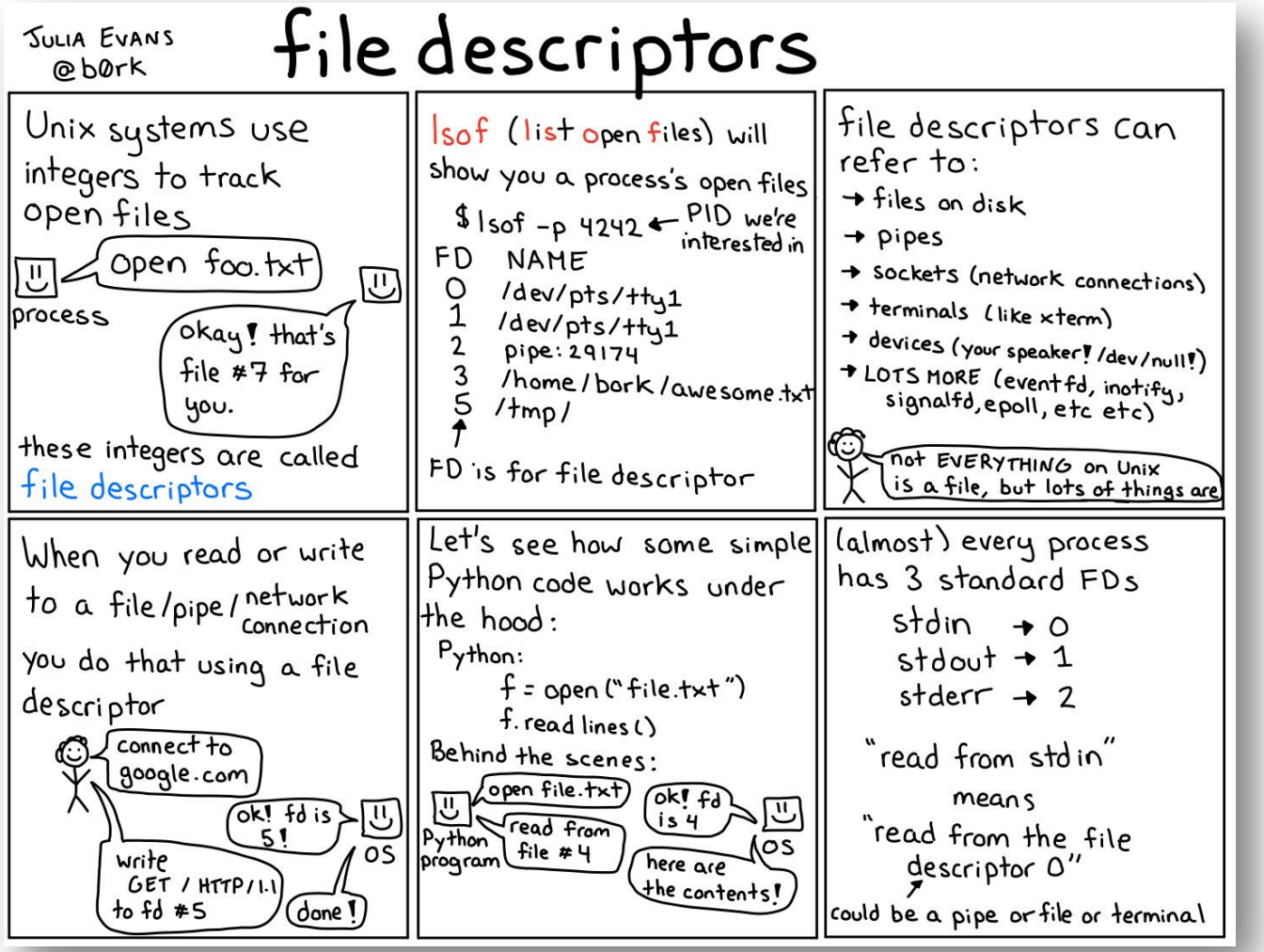
- What is a file? a place to put stuff
  persistent data (stored on disk)
- What is a file system? enables retrieval (operations)
  /tmp tmpfs
  an organization structure
- How do these concepts differ from the other storage concepts we've been discussing?
  E.g.,
  - physical memory?
  - virtual memory?
  - backing store for nonresident pages?



Files

Files & File Metadata

- A place to store stuff! (e.g., bytes)
  - Try: **xxd**
- Typical metadata attributes
  - name, size, identifier, location, protection, bookkeeping, type, etc.
  - Try: **stat**, **man 2 stat**



Files & File Types

- How do we know what is IN a file?
- How can we know the file type?
- Any issues?

Issues with Binding Applications/File Extensions

- File Extension Attacks:
  - Spoofed file types
    https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-14604
  - Games w/ parsing file extensions
    https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2009-4444
  - Visual truncation
    https://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-2530

File Systems

File System Operations

- ***What do we do with files?***
  - create
  - read
  - write
  - delete

- To do these things, we often need information…
  - sequential access
    vs.
  - direct/random access

current pos + len / # bytes

Addr

File System Operations

- ***What do we do with files?***
  - create
  - read
  - write
  - delete

- To do these things, we often need information…
  - sequential access **-> CURRENT LOCATION** seek
    vs.
  - direct/random access **-> need ADDRESS of specific thing to access**

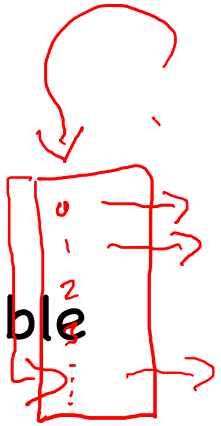
File Systems & Bookkeeping

- We need to store this information somewhere…
  - **Where?**
  - **When does it get set/reset?**

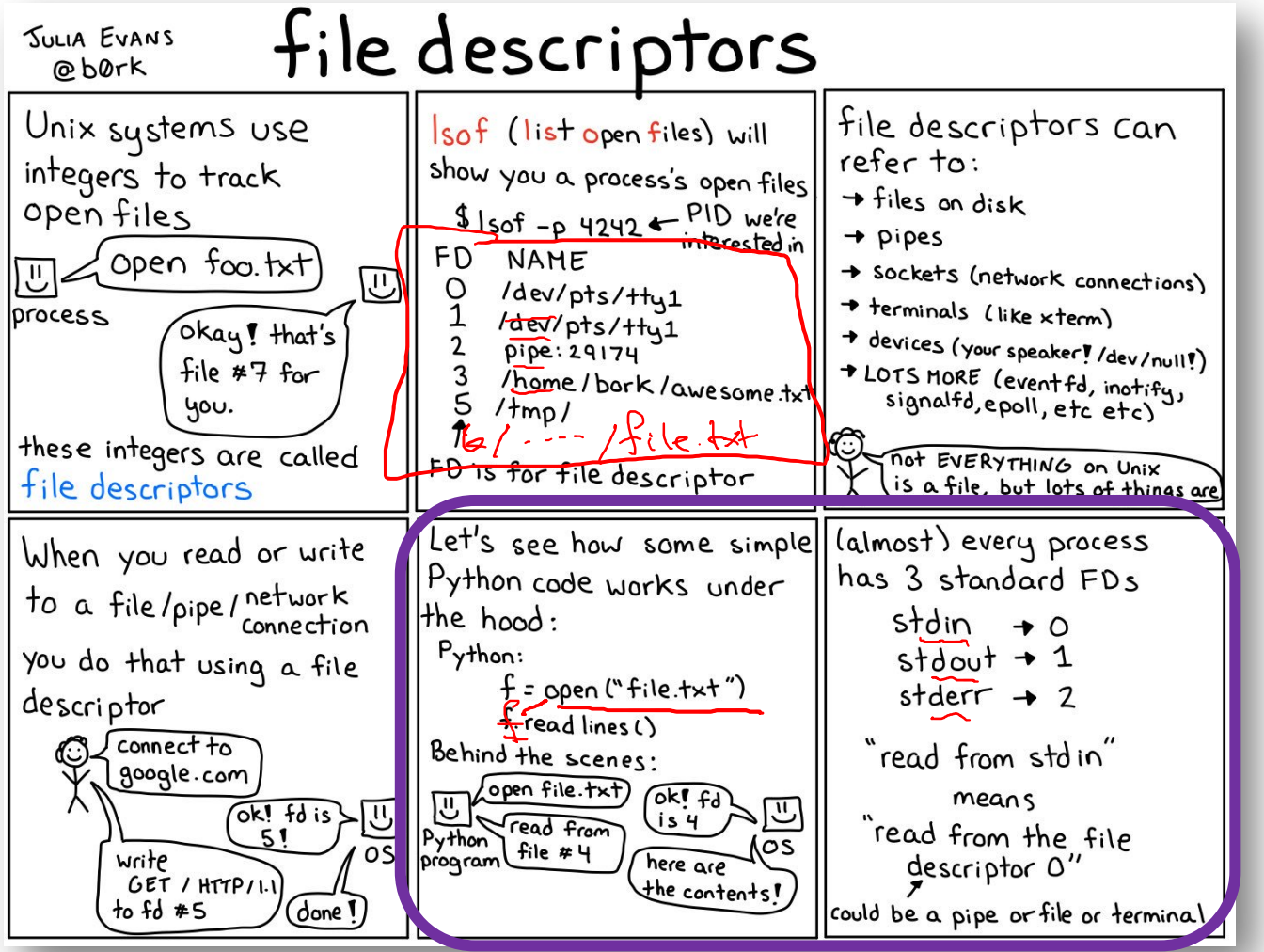
File Systems & Bookkeeping

- We need to store this information somewhere...
  - Where?
  - When does it get set/reset?

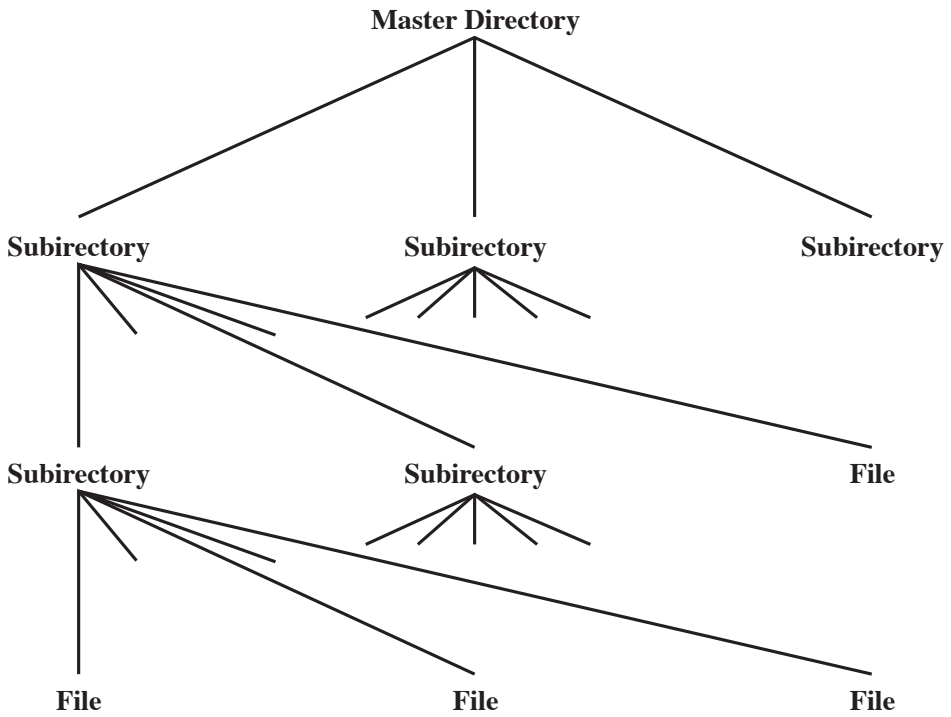
- On open/close
  -> open file table!



- I/O calls refer to this table
  - efficient references
  - easier sharing semantics
  - easier access control



Bookkeeping (detailed)

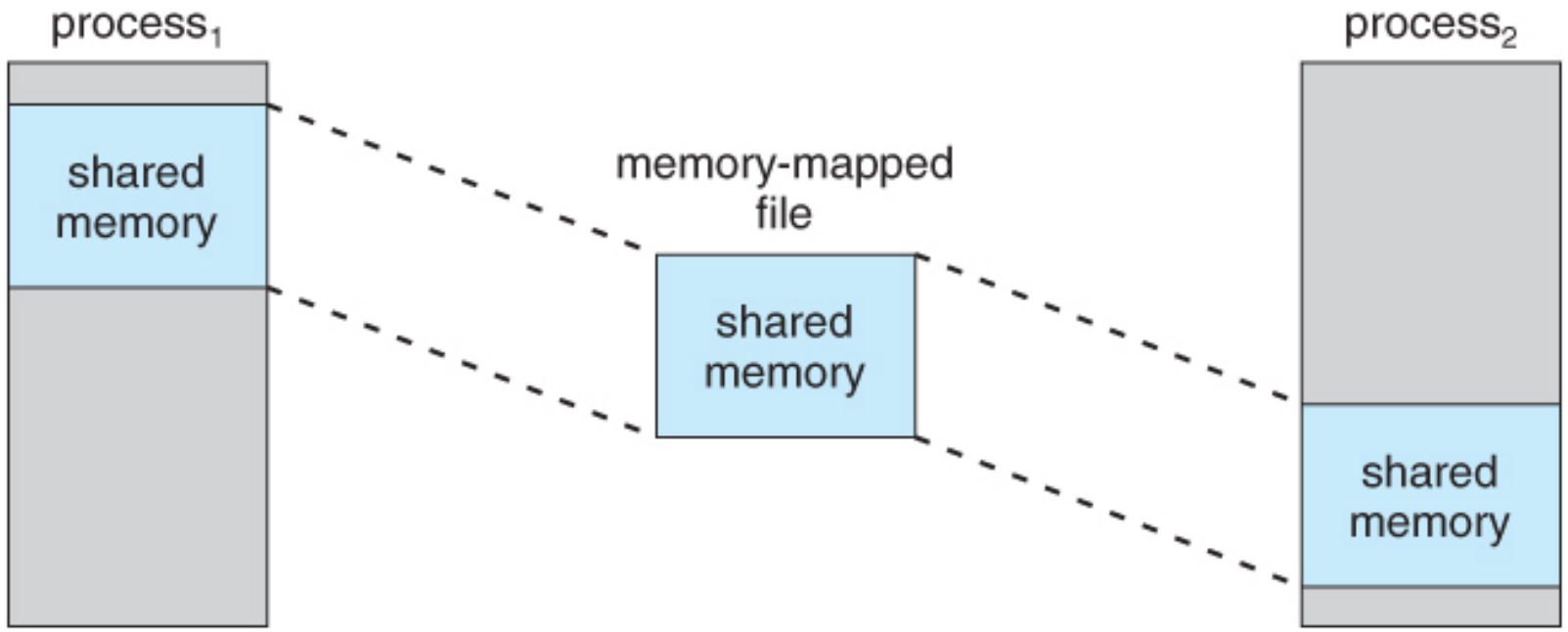


	Basic Information
File Name	Name as chosen by creator (user or program). Must be unique within a specific directory.
File Type	For example: text, binary, load module, etc.
File Organization	For systems that support different organizations
	Address Information
Volume	Indicates device on which file is stored
Starting Address	Starting physical address on secondary storage (e.g., cylinder, track, and block number on disk)
Size Used	Current size of the file in bytes, words, or blocks
Size Allocated	The maximum size of the file
	Access Control Information
Owner	User who is assigned control of this file. The owner may be able to grant/deny access to other users and to change these privileges.
Access Information	A simple version of this element would include the user's name and password for each authorized user.
Permitted Actions	Controls reading, writing, executing, transmitting over a network
	Usage Information
Date Created	When file was first placed in directory
Identity of Creator	Usually but not necessarily the current owner
Date Last Read Access	Date of the last time a record was read
Identity of Last Reader	User who did the reading
Date Last Modified	Date of the last update, insertion, or deletion
Identity of Last Modifier	User who did the modifying
Date of Last Backup	Date of the last time the file was backed up on another storage medium
Current Usage	Information about current activity on the file, such as process or processes that have the file open, whether it is locked by a process, and whether the file has been updated in main memory but not yet on disk

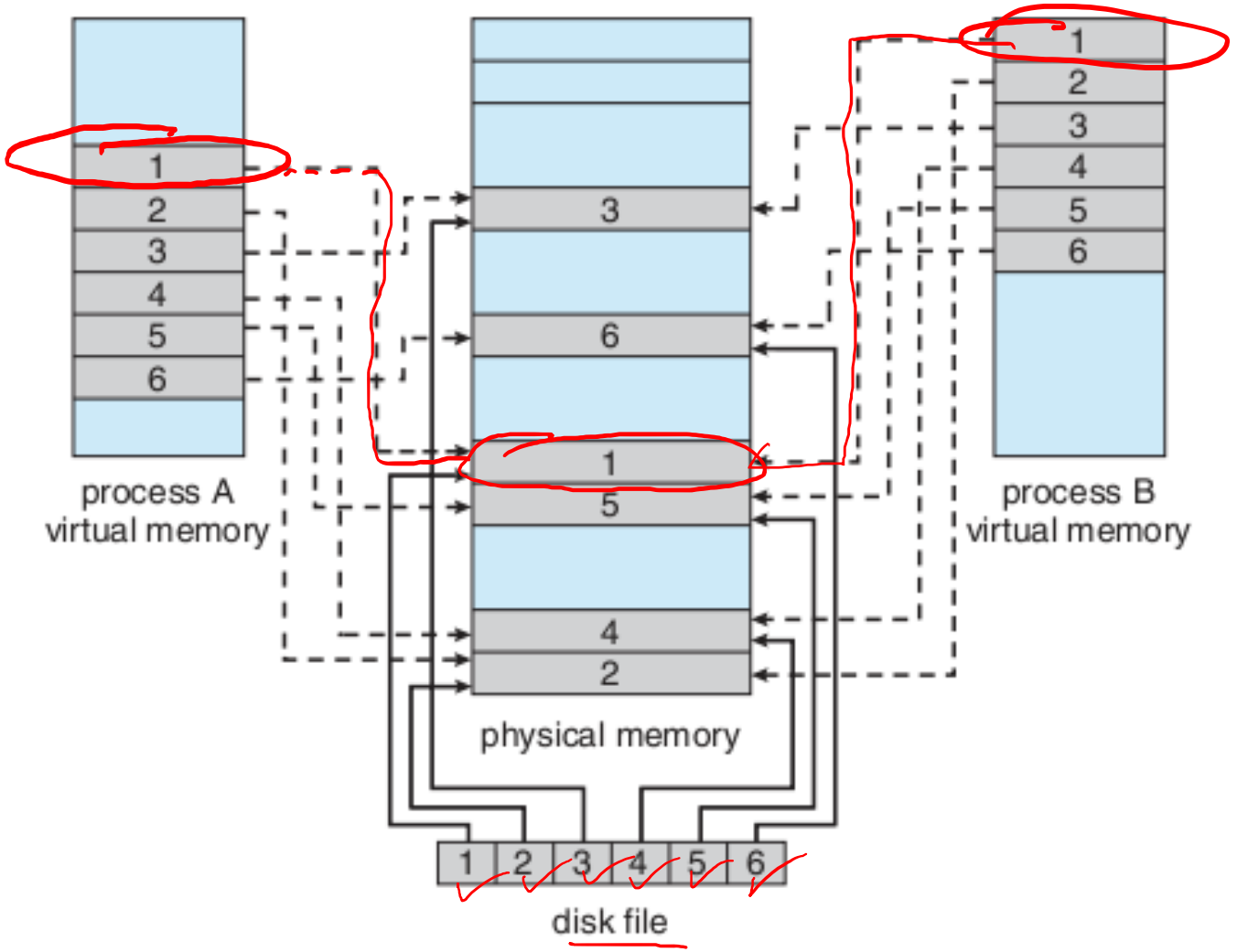
Memory Mapping

- How might we implement memory mapping?
- How might we implement memory mapped *files?*
- Advantages?
- Disadvantages?

Memory Mapping

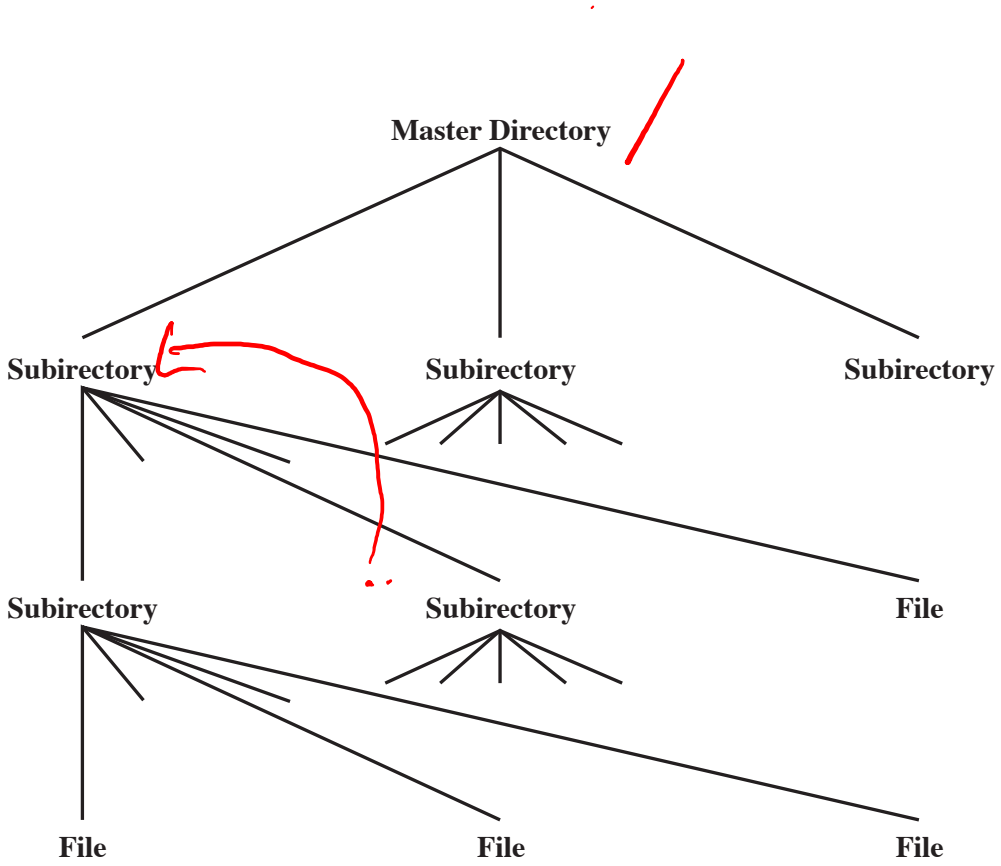


Memory Mapping



File System Organization

Basic File System Organization



- Put files in a **directory**!
- Put files and directories...
  ...inside directories!

(could there have ever been any other way?!)

- **Basic Operations:**
  search, create, delete, list, traverse
- **Basic Concepts:**
  current directory, absolute/relative paths, search path

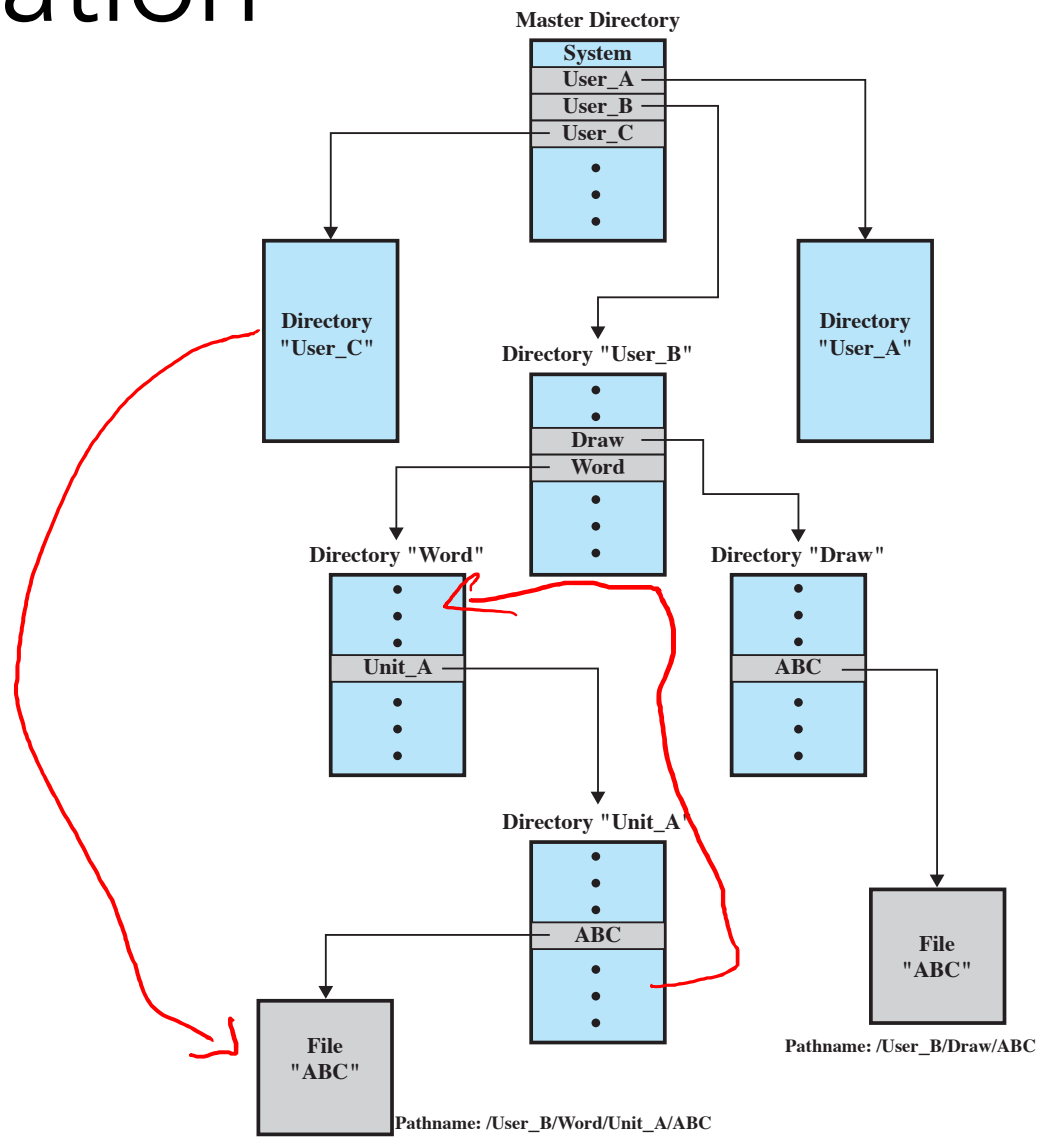
PATH

Basic File System Organization

- The file system is a tree, right? Maybe... maybe not...
  - single-level ("flat file system")
  - tree
  - DAG
  - graph

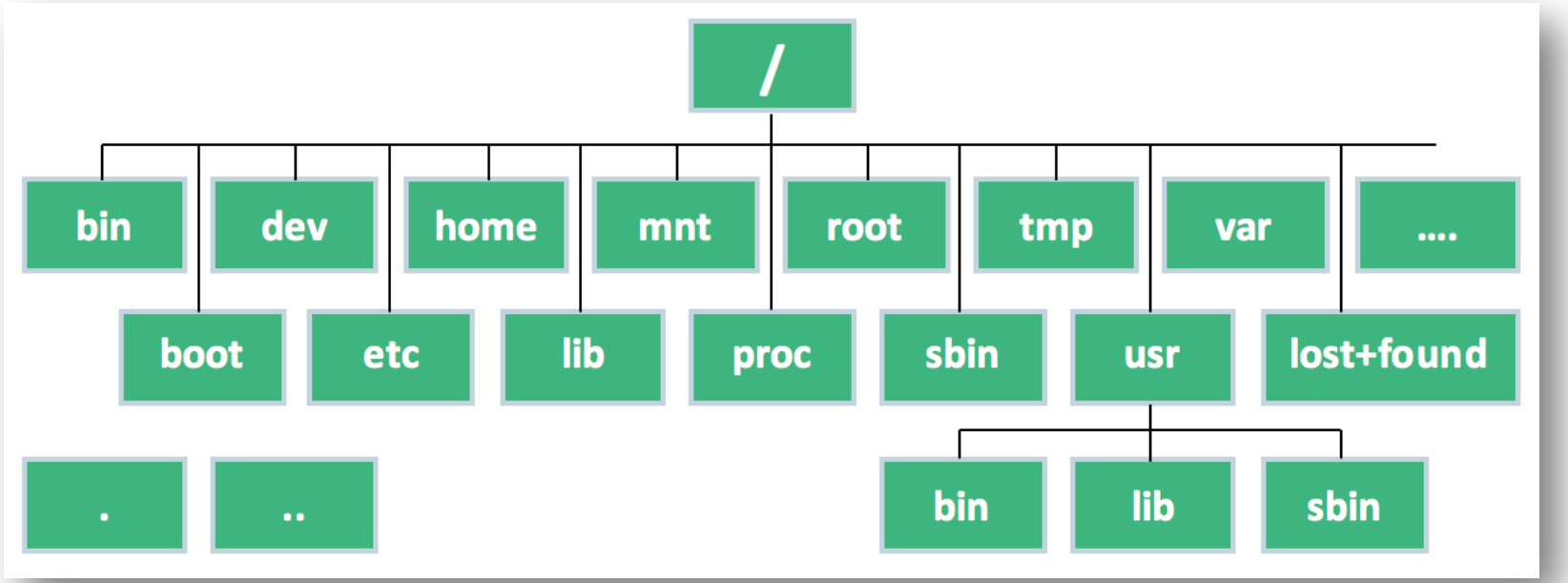
In UNIX/Linux:

- The FS is basically tree with **links**
  - **hard** links vs. **soft** links ln

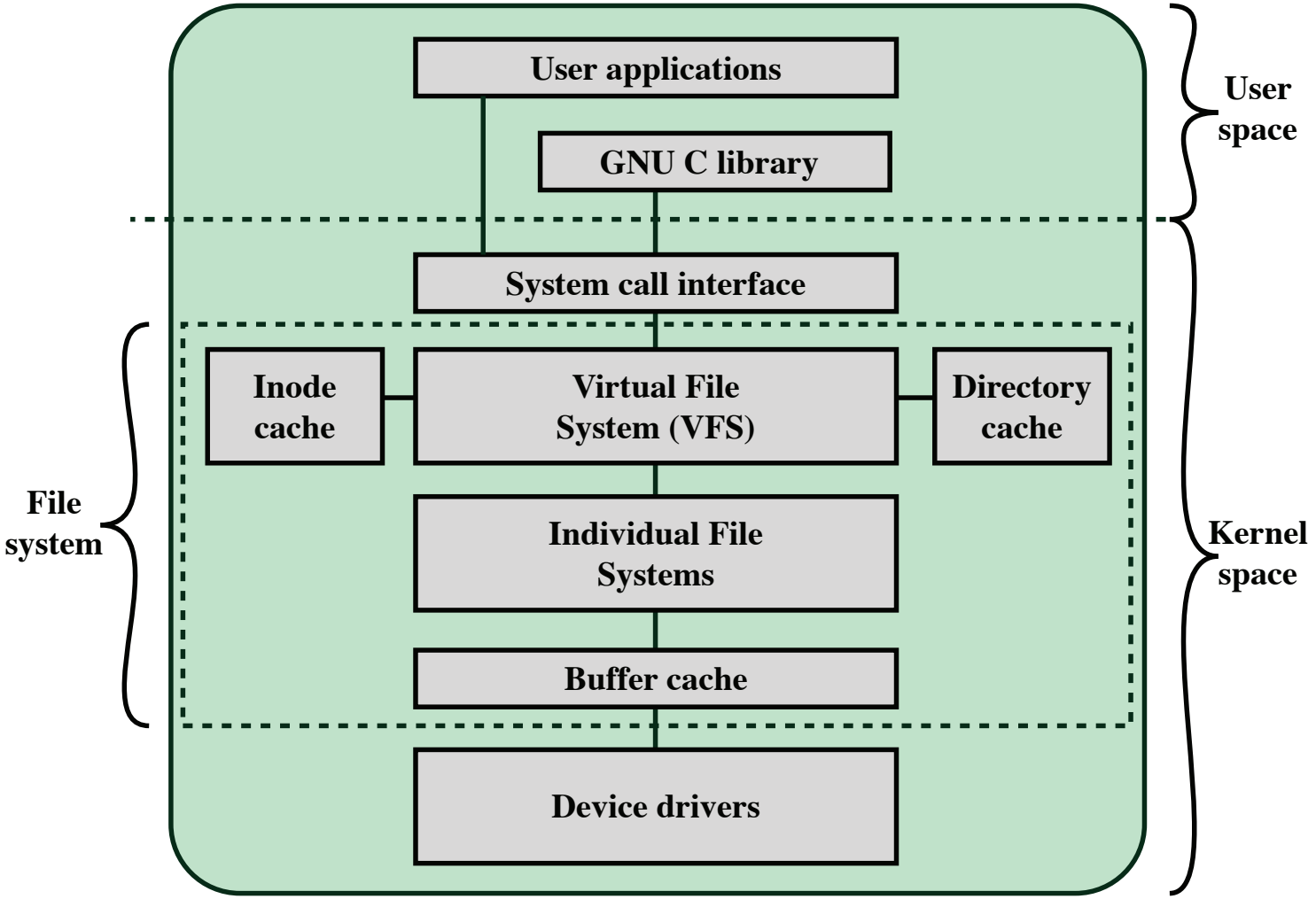


Kitchen Sink File Systems

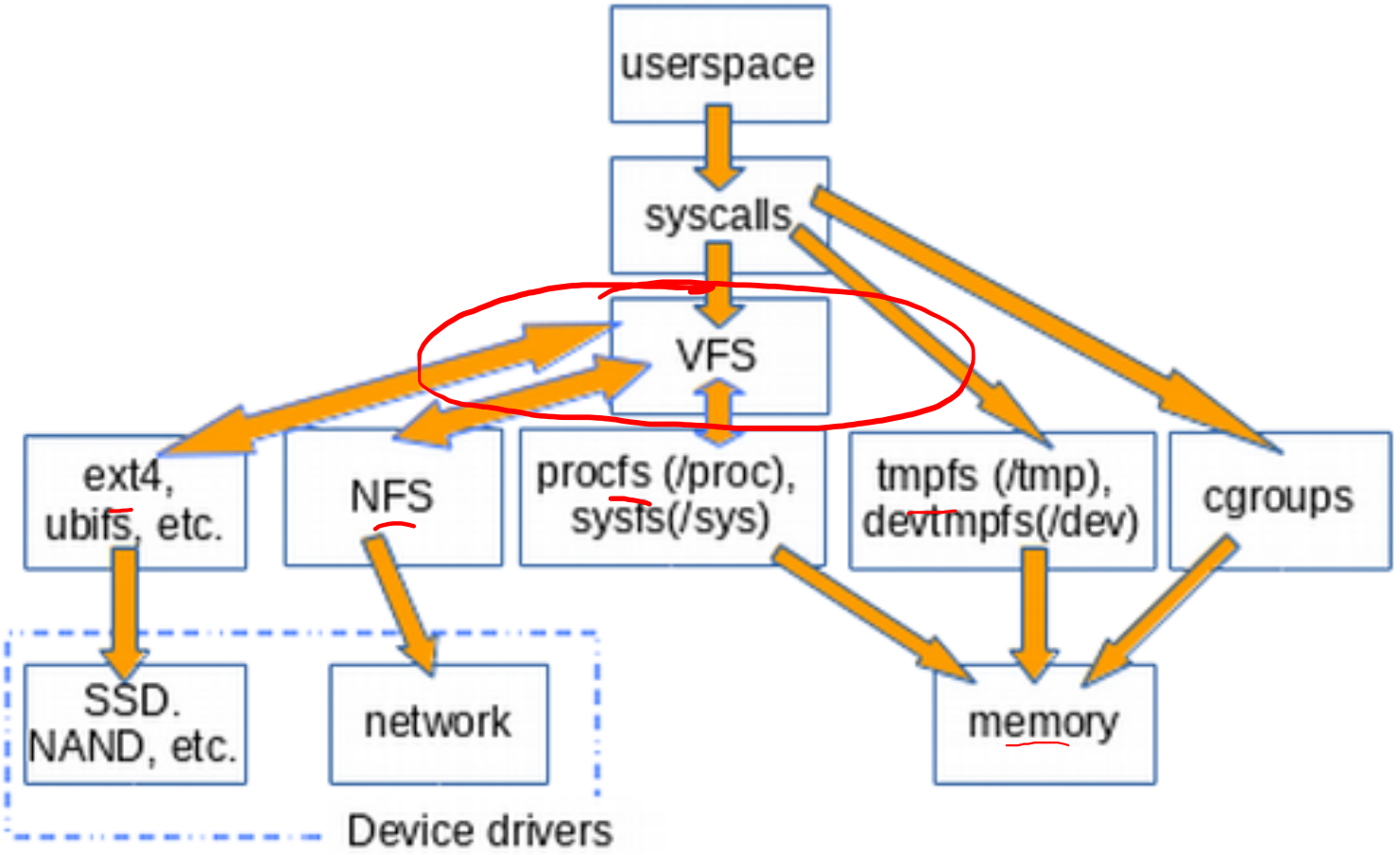
- Files can be more than just a bunch of stored bytes…



Linux VFS


User applications
GNU C library
System call interface
Inode cache
Virtual File System (VFS)
Directory cache
Individual File Systems
Buffer cache
Device drivers
User space
Kernel space
File system

Linux VFS


userspace
syscalls
VFS
ext4,
ubifs, etc.
NFS
procfs (/proc),
sysfs(/sys)
tmpfs (/tmp),
devtmpfs(/dev)
cgroups
SSD.
NAND, etc.
network
memory
Device drivers

Summary

Big Ideas

- Files and file systems are where you keep (and organize) stuff
  -> they have various attributes: **name**, **id**, **type**, other **bookkeeping**
  -> files have typical operations: **create**, **delete**, **read**, **write** (+memory map, mount)
  -> file systems have typical operations: **create**, **delete**, **read**, **write** (+memory map, mount)

- Virtual File System (VFS)
  -> the abstraction that makes "everything is a file" possible!

What next?

- *Digging deeper into implementation details (e.g., inodes,)*
- *File system types (e.g., FAT, HFS+, EXT*, ZFS)*
- *Security (some next week!)*

Security

File Modes & Permissions

- Every Unix file has a set of **permissions** that determine whether someone* can read, write, or run the file.
  - Try: `ls -l ~`
  - Try: `ls -l /dev`

- File mode (4 parts)
  -> [file type][user][group][other]
  - file type
  - ugo
  - rwx

```
$ls -l file
-rw-r--r-- owner group ... file
```

- How do we change these settings?
  - Ex: allow other members of my **group** to **write** "file"?

Getting Our Hands Dirty!

Some guided exploration of files & file systems - you try!

Fun with file/file system commands

- "." vs ".."
- ls
- chmod
- umask
- chown / chgrp
- ln
- du
- dup / dup2
- open, read, write, lseek, creat
- tar

Fun with File Descriptors

If two processes each open the same file:

- Do we expect their reads to be interleaved or independent?
- What would this say about whether these two openings of this file have the same file entry, or different file entries?

-> Try filefun2.c with the -2 -s options to see.

If a fork gives the child a copy of the parent's file descriptors:

- If the parent had a file open, is this file open in the child?
- If so, does the child have a different file entry or the same one?
- What would this say about whether their reads would be interleaved or independent?

-> Try filefun2.c with the -2 -i options to see.

What about dup?

-> Try filefun2.c with the -1 -i options to see.

Fun with Links

- On Linux, go to a fairly empty directory.
- Get a medium file in there (say, at least 20K).
- *(I couldn't figure out how to get cat or such to do N bytes, so I wrote **bytes.c** instead. Recall… stdin, stdout, stderr….) (**head -c** might also work)*
- Make a hard link to it.
  - Use **ls -lh** to look at the size of things in the directory.
  - Use **du -h** to look at the sizes of individual things.
  - Use **du -h .** to look at the size of the directory.

-> Do things add up?

-> Try making a soft link, and using **du** to look at sizes. ***What's going on?***

Fun with Permissions

- You might wonder, with group permissions, what if the requester in several groups?
- (Here, we'll step over to Linux…)
- I created a dir **permtest/** with various files. I, as user **vagrant**, can read, say, file **file_test**.
- (I'm in many groups - grep in /etc/group)
- But what happens if I try to read **file_test_2**?
- Suppose I also create **subdir/file**, and two hardlinks and softlinks to it.
- What happens if I chmod **hardlink1**?
- What happens if I chmod **softlink1**?
- What happens if I chmod **subdir** to remove userx permissions? Do the softlinks still work? Do the hardlinks?
- What might you conclude from all of this about potential implementation details?