Quick notes

- Check Piazza
- Reminders:
  - Assign5 is due friday
  - C debug + sort
  - Weekly quiz
- Assignment 10 (only applies to students enrolled in Lab 1)
  - Email me your topic
  - Create presentation (<10 min, semi-strict) + brief summary (<200 words)
    - Must present to class (email me if you're in a different timezone)
    - Talk about topic of your choice, goal is for everyone else to understand what you researched. Example/interesting video https://youtu.be/VVdmmN0su6E
  - Will start week 8 (or sooner!?), email me if you want to volunteer
  - Most likely will record these presentations. If you do not want to be recorded then let me know!

Feedback / Office Hours / Other

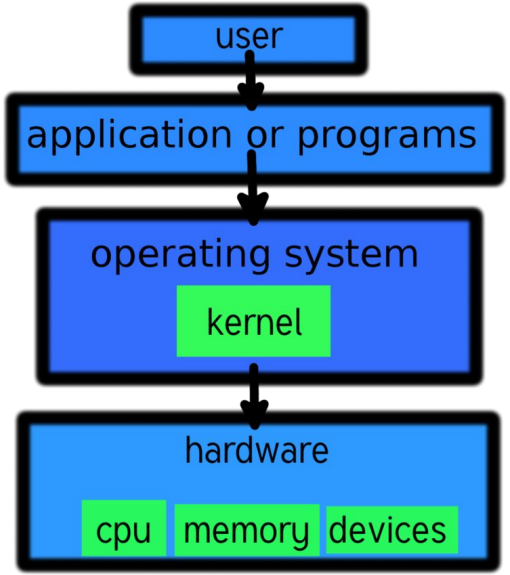
- Tameez Latib
  - tameezlatib@gmail.com, please add “CS35L” to the subject line
  - Office Hours: Monday **4pm-6pm** (or by appointment)
  - Feedback: https://forms.gle/6kcJ2aJtzAzFMhHQ7 (anonymous google form)
- If you guys are stressed out:
  - CAPS (https://www.counseling.ucla.edu/)
    - Free with UC ship
- Assignment grades (for all assignments):
  - If you think there is a mistake, ask for re-grade request
  - I think you only have 1 week (?) to do this after you receive grade

Kernel

- Core of operating system. Processes <-> Kernel <-> Hardware
- Memory Management
  - Keeps track of memory locations, and how much memory is being used to store what
- Process Management and Scheduling
  - Determine what processes can use the CPU, when, and for how long
- Device Drivers
  - Acts as the interpreter between hardware and software
- Systems Calls and Security
  - Protects sensitive information and access while still allowing applications to perform as necessary
- Kernel loaded first at bootup
  - If it crashes, everything crashes.

Kernel

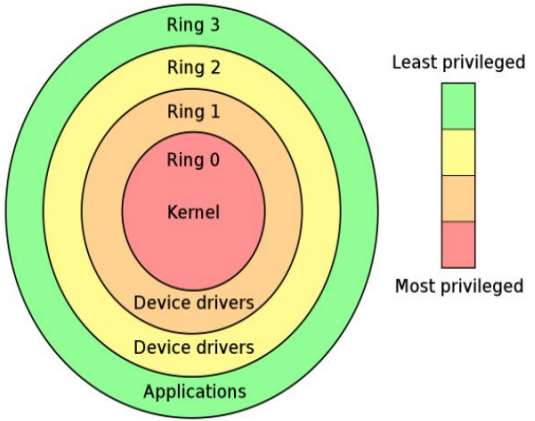
- user/kernel mode/bit
- Similar to file permissions, there is a bit to designate user mode / kernel mode
- If a process has kernel mode:
  - Complete access to hardware. Issue any command, access any memory, etc
- If a process has user mode:
  - Limited, controlled access. Generally more secure and error-proof.
  - All hardware requests go through system APIs
- Mainly for security reasons, but having a controlled environment helps prevent failure cases



System calls

- Users can access kernel level instructions using system calls
- User (sys call) -> interrupt + swap to kernel -> kernel exec function -> return to user mode with output
  - Example: I/O + disk operations
- Note that this user->kernel switch is expensive

Remember: Kernel has most privileges, user/apps have least



System calls vs library calls

- putchar/printf/etc are library calls
  - These work by making system calls. System calls are the ONLY way to get kernel privileges
- read/write are system calls
  - In general they are faster, but require more ‘work’ to do the same as what library functions do
- Example sys calls you may need:
  - Read, write, getpid, dup, fstat, open, close, syscall(this one is most general, least useful)
  - Look them up (e.g. https://man7.org/linux/man-pages//man2/read.2.html)
  - They use file descriptors
    - 0,1,2 stdin,stdout,stderr
    - Create more with open

Example

- Notes:
  - File descriptors 0, 1, 2 are stdin, stdout, stderr
  - When you open a new file, you start from 3
- Strace __
  - Get all sys calls
- time ___
  - Get time

Buffered vs unbuffered I/O

- Syscalls expensive
  - Instead of syscall for each character, use one syscall to read all of it and place in buffer
  - Read from our buffer, parse locally
- Difference between:
  - 1. Check file
  - 2. Read
  - 3. Parse
  - 4. Repeat 1,2,3
- And
  - 1. Check file
  - 2. Read + store in buffer
  - 3. Read buffer
  - 4. Parse
  - 5. Repeat 3,4

Buffered vs unbuffered I/O

- Stdout is (line) buffered
- Stderr is unbuffered
- Why?
  - With stdout, lots of data. Don’t care if we wait slightly longer
  - With stderr, less data. Want errors immediately. It matters!

Questions??