

What did we do?

- OS theory
 - Usual OS topics: Concurrency, Synchronization, System calls, Interrupts, Run queues & wait queues, Scheduling, Virtual memory, File systems
 - But in the context of current Linux implementations
 - Skimmed/skipped the following:
 - Deadlock theory
 - I/O systems
 - Network file system (NFS); SSD
- Advanced UNIX programming
 - APUE book & multi-server assignments
 - Many advanced topics including:
 - Signal handling
 - Multi-threaded programming, concurrency, locking
 - Non-blocking I/O, `select()`, `mmap()`
 - IPC – pipes, shared memory, domain sockets

What else did we do?

- Linux kernel programming
 - HW1: intro to crazy OS-level C
 - HW5, *aka Fridge*: wait queues and kernel locking
 - HW6, *aka Freezer*: simple new scheduler for Linux
 - HW7, *aka Cabinet*: Linux page table structure
 - HW8, *aka Pantry*: simple file system from scratch!
 - We skimmed/skipped:
 - Interrupt handlers and bottom half
 - Kernel synchronization using RCU
 - Kernel memory management & block I/O layer
 - Virtualization

Please

- Fill out CourseWorks evaluation
- Remember your pledge
 - Don't share class materials with friends
 - Don't post any class-related code to GitHub
 - Don't post any class materials to Chegg, CourseHero, etc.

The most important thing I learned was not be afraid.

That's a harder lesson to learn than it sounds, because the only way to really learn it is to do the things you think sound hard. . . . this was the biggest takeaway for me from the kernel development work in OS.

- Andrew Kiluk