- Kernel / Protection
    - Kernel vs User Space
    - Avoiding boundary crossings
    - Page faults
- Virtual Address Space
    - Different Regions
    - Growing Stack on demand.
    - Heap allocation.
- Executing Kernel Code
    - Synchronous vs. Asynchronous transfer of control
    - Entry point
    - Interrupts
        - Timer
    - Traps
        - System Call Process
        - Exceptions
    - System call process
- Processes
    - What is a process?
        - Process vs. thread
    - CreateProcess() vs. fork()
    - fork(), wait(), exec()
    - exec() user stack layout
    - COW fork() vs Deep Copy fork()
    - Process Cleanup
        - Orphans / Zombies
    - Context switching
        - Pre-emptive (timer) vs. non-preemptive (yield(), sleep())
    - Process States and Transitions
- File Descriptors
    - Open
    - Read/Write
    - fork() and the Process Open File Table
- Synchronization
    - Spinning vs. Sleeping
        - When to use which.
        - Spinlocks
            - Disabling interrupts - when does it work?
        - Sleeplocks
            - Cost of changing to sleep state
        - Atomic Operations
            - Test-and-set and Compare-and-swap
        - Non-blocking sync

        - Condition Variables
        - Memory Barriers
            - __sync_synchronize()
- Inter Process Communication (IPC) / Pipes
    - Pipes vs Files for communication.
    - Pipe Open/Close.
- CPU Scheduling
    - FIFO
    - Round Robin
- File System
    - Layers of the file system calls
    - Extent Management implementations (pre-alloc vs. array vs. Unix style indirection using blocks).
    - Transactions and Atomicity
    - inodes
- Persistent Disk
    - Disk data structures
        - Bitmap
        - Log region
        - inodes
        - Superblock
        - Boot block
    - Disk and Cache consistency
- Security
    - Spectre and Meltdown