Semester Review

Will Hedgecock
Sandeep Neema
Bryan Ward

ISIS
Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu

VANDERBILT UNIVERSITY

Operating System: Definition

- Definition: A program that shares a computer among multiple programs and provides a more useful set of services than the hardware alone
  - Sharing: we discussed several approaches that are useful to facilitate sharing a platform
    - Virtual memory, scheduling, security and privileges, concurrency and synchronization, etc.
  - Useful services
    - Filesystems, IPC, networks, etc.
- We have only covered the fundamentals of common design approaches. In practice, there are OSes with alternative designs
  - e.g., microkernels are super small and may not implement something like a filesystem
  - Linux has many features and performance optimizations that xv6 doesn't
- These core ideas are useful to understand what is going on under the hood, what OS services may be available, and how to think about low-level systems issues

Topics Covered

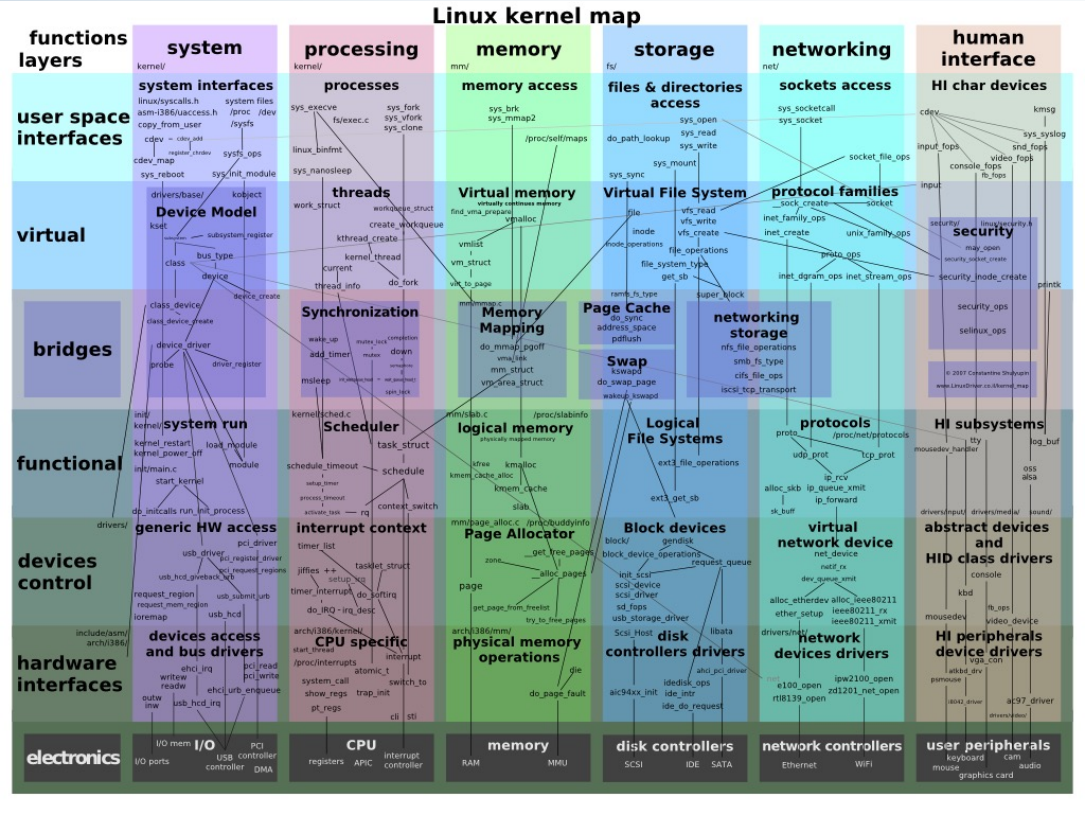
- OS Architecture and Fundamentals
  - syscalls, privilege rings
- Exceptional Control Flow
  - syscalls, interrupts, faults
- Process Creation and Termination
  - fork(), wait(), exit(), process graphs
- Virtual memory
  - Page tables, copy on write, page faults, MMU
- Concurrent Programming
  - Threads, locks, semaphores
  - Race conditions, concurrency
- Scheduling
  - Round Robin, MLFQ, real-time scheduling
- Interprocess Communication (IPC)
  - Pipes, shared memory, sockets
- Networking
  - Sockets, TCP/UDP, network stack, etc.
- I/O Devices
  - Polling vs. Interrupts
- Filesystems
  - inodes, direct vs. indirect blocks, journaling
- Virtualization
  - Native virtualization, paravirtualization, virtual, physical, and machine memory
- Security
  - Threat models, memory corruption, address randomization,

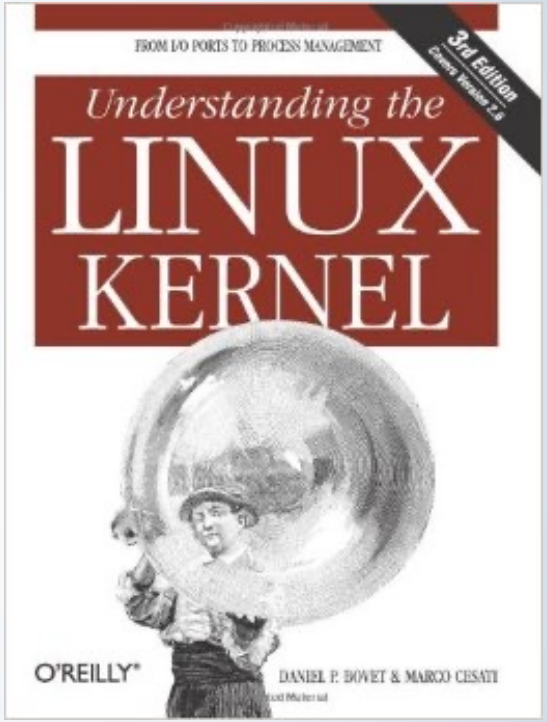
Dependencies

- While this class is taught linearly, in point of fact there are many dependencies and relationships between the concepts we covered
- We have seen some of this already, e.g., implemented a system call for symlink
- Another example: consider exec() system call that we use to run new program
  - Need system call to communicate what process to run
  - Need to update the address space – therefore interact with virtual memory
  - Need to load in new process – must fetch the data off disk
    - Need to request data from an I/O device
    - Need to interpret data stored on disk to know where and what to read and load

We hadn't covered these topics when we discussed exec early in the semester!

Dependencies





Some Common Themes

- Virtualization
  - CPU is virtualized to allow processes to run independently of one another
  - Operating systems themselves can even be virtualized
  - Must think carefully about control flow, e.g., context switching and exceptional control
- Resource management
  - Virtual memory and page tables (managing memory, shared memory, CoW)
  - Scheduling (managing time)
  - Lazy allocation (e.g., CoW) and caching (e.g., using virtual memory)
- Persistence
  - Filesystems and disk storage
- Abstraction
  - Abstraction makes programming easier (e.g., fork/exec, pipes)
  - Many abstractions enable efficient resource usage (e.g., shared memory, CoW)

So What?

- OS and low-level systems programming is important in many industrial applications
- Even high-level software development requires interfacing with the OS, so important to know features and concepts
- How software interfaces with the OS and hardware can significantly affect performance, e.g.,
  - Synchronization
  - I/O bound applications
  - Memory management