Course Review

Module 14

Class Topics

- Module 1 – Introduction
- Module 2 – Architectural Support for OS
- Module 3 – OS Components and Structure
- Module 4 – Processes
- Module 5 – Threads
- Module 6 – Synchronization
- Module 7 – Sync: Performance and Multi-Object
- Module 8 – Storage Systems
- Module 9 – File Systems
- Module 10 – Journaling File Systems
- Module 11 – Log Structured File System
- Module 12 – RAID
- Module 13 – (CPU) Scheduling
- Module 14 – Course Review

What comprises an OS?

User-level software

- Compilers / linkers / debuggers
- Shells

Kernel Software

- Syscall architecture / API
- Process memory model
- Processes / threads
- File Systems / Networking Stack / etc.
- Resource allocators
- Protection model
- IPC mechanisms

Hardware

- Interrupt mechanism
- Timer
- Memory addressing architecture
- IO architecture
- Privilege levels
- Privileged/unprivileged instructions
- Atomic instructions
- Memory consistency model / mechanisms

Avoid Boundary Crossings

- Hardware user/privileged mode
- Monolithic vs. More Modularized OS Structures (e.g., micro-kernel)
- File system
  - caches: block, directory, i-nodes
  - delayed writes
  - open/close semantics
- Runtime library app level caches: FILE*
- Shared mapped memory regions
- Threads (vs. processes)
- Cache aware locks
- Per-core run queues

Optimize the Common Case

- vfork
  - COW fork
- FFS inode: data block map
- Log structured file system
- MCS locks
- RCU locks

Policies: Beware Starvation

- [Beware deadlock!]
- Any policy that prioritizes is at risk
  - Disk scheduling algorithms (SCAN, CSCAN)
  - SJF scheduling
  - CFS CPU scheduling

Atomicity Simplifies, at a Cost

- Hardware: atomic instructions
- Software: synchronization primitives
  - locks: spin and blocking
  - condition variables
  - lock-free synchronization
- Persistent storage: journaling
  - Alternative reliability approach: redundancy (e.g., RAID)

Isolation Simplifies, at a Cost

- Processes / address spaces
  - pipes
  - files
  - shared memory
  - signals
  - threads
- Processes / memory & CPU
- File system protection mechanisms

Simpler is Faster

- This is a variation of “the end-to-end principle” from networking.
  - Put minimal functionality in the base
  - If you need more, layer it on top
- Flat address spaces, not segments
- Processes, not objects
- Flat files, not structured files

Namespaces / Scope

- File system namespace / system
- Process namespace / system
- Open file namespace / process
- Disk block namespace / disk
- Syscall namespace / static global

Protection by Naming

- Virtual memory
- syscall number, not target code address
- chroot
  - we didn’t talk about it
- containers
  - we didn’t talk about them...