

**CS 153**  
**Design of Operating Systems**

**Spring 19**

**Midterm Review**

# Midterm

- in class on Friday (May 3)
- Covers material from arch support to deadlock
- Based upon lecture material and modules of the book indicated on the class schedule
  - ◆ Semaphore book: chapter 1-4
  - ◆ **Closed book. No additional sheets of notes**

# Overview

- Architectural Support
- Processes
- Scheduling
- Threads
- Synchronization

# Roles an OS

- **Abstraction:** defines a set of logical resources (**objects**) and well-defined operations on them (**interfaces**)
  - ◆ **Why?** Easier app programming
  - ◆ Humans are good at abstraction instead of details
- **Virtualization:** Isolates and multiplexes physical resources via spatial and temporal sharing
  - ◆ **Why?** Better hardware utilization, easier programming (don't need to consider co-existing programs)
- **Access Control:** who, when, how
  - ◆ **Why?** Fairness, performance, security, privacy, etc.

# Arch Support

- Purpose of architecture support
  - ◆ **Why we need them?** Easier OS implementation
- Types of architecture support
  - ◆ Manipulating privileged machine state
  - ◆ Generating and handling events

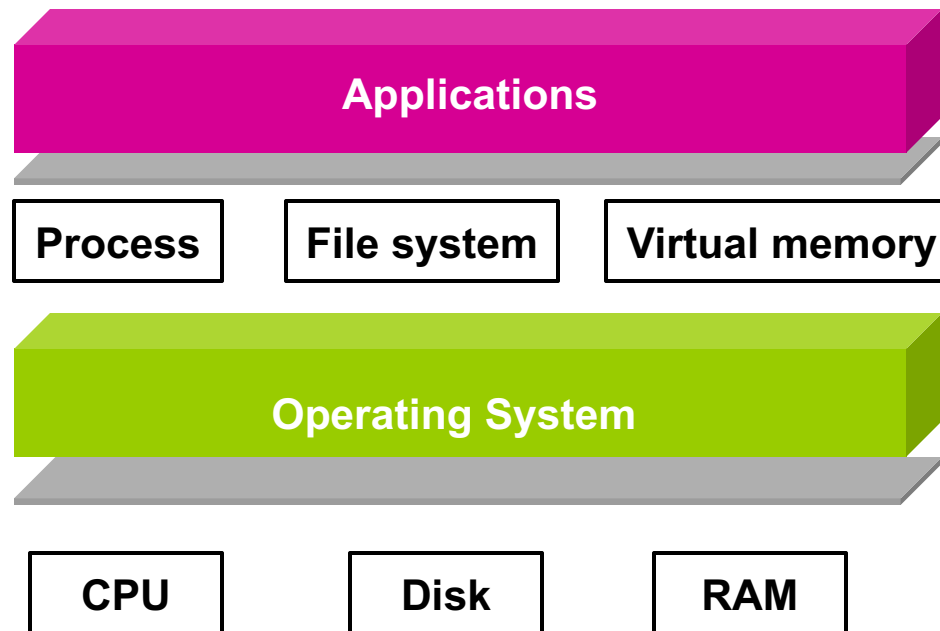
# Privileged Instructions

- What are privileged instructions?
  - ◆ Who gets to execute them?
  - ◆ How does the CPU know whether they can be executed?
  - ◆ Difference between user and kernel mode
- Why do they need to be privileged?
- What do they manipulate?
  - ◆ Protected control registers
  - ◆ Memory management
  - ◆ I/O devices

# Events

- Events
  - ◆ Synchronous: faults (exceptions), system calls
  - ◆ Asynchronous: interrupts, signals
- What are faults, and how are they handled?
- What are system calls, and how are they handled?
- What are interrupts, and how are they handled?
  - ◆ How do I/O devices use interrupts?
- What is the difference between exceptions and interrupts?

# OS Abstractions





# Processes

- What is a process?
- What resource does it **virtualize**?
- What is the difference between a process and a program?
- What is contained in a process?

# Process Data Structures

- Process Control Blocks (PCBs)
  - ◆ What information does it contain?
  - ◆ How is it used in a context switch?
- State queues
  - ◆ What are process states?
  - ◆ What is the process state graph?
  - ◆ When does a process change state?
  - ◆ How does the OS use queues to keep track of processes?

# Process Interface

- What **operations** can be performed on processes?
  - ◆ What does CreateProcess on Windows do?
  - ◆ What does fork() on Unix do?
    - » What does it mean for it to “return twice”?
  - ◆ What does exec() on Unix do?
    - » How is it different from fork?
- How are fork and exec used to implement shells?

# Threads

- What is a thread?
  - ◆ What is the difference between a thread and a process?
  - ◆ How are they related?
- **Why** are threads useful?
  - ◆ Concurrency → utilizing more CPU powers
    - » The thermal wall of single thread performance
  - ◆ Lightweight, fast communication
- What is the difference between user-level and kernel-level threads?
  - ◆ What are the advantages/disadvantages of one over another?

# Thread Implementation

- How are threads managed by the run-time system?
  - ◆ Thread control blocks, thread queues
  - ◆ How is this different from process management?
- What **operations** do threads support?
  - ◆ Fork, yield, sleep, etc.
  - ◆ What does thread yield do?
- What is a context switch?
- What is the difference between non-preemptive scheduling and preemptive thread scheduling?
  - ◆ Voluntary and involuntary context switches

# Scheduling

- **Why** we need scheduling?
  - ◆ Long-term scheduling / Goals
  - ◆ Short-term scheduling / Goals
- Components
  - ◆ Scheduler (dispatcher)
- When does scheduling happen?
  - ◆ Job changes state (e.g., running to waiting, waiting to ready)
  - ◆ Interrupt, exception
  - ◆ Job creation, termination

# Scheduling Goals

- Goals
  - ◆ Maximize CPU utilization
  - ◆ Maximize job throughput
  - ◆ Minimize turnaround time
  - ◆ Minimize waiting time
  - ◆ Minimize response time
- What is the goal of a batch system?
- What is the goal of an interactive system?

# Starvation

- Starvation
  - ◆ Indefinite denial of a resource (CPU, lock)
- Causes
  - ◆ Side effect of scheduling
  - ◆ Side effect of synchronization
- Operating systems try to **prevent starvation**



# Scheduling Algorithms

- What are the properties, advantages and disadvantages of the following scheduling algorithms?
  - ◆ First Come First Serve (FCFS)/First In First Out (FIFO)
  - ◆ Shortest Job First (SJF)
  - ◆ Priority
  - ◆ Round Robin
  - ◆ Multilevel feedback queues
- What scheduling algorithm does Unix use? Why?

# Synchronization

- **Why** do we need synchronization?
  - ◆ Coordinate access to shared data structures
  - ◆ Coordinate thread/process execution (**active scheduling**)
- What can happen to shared data structures if synchronization is not used?
  - ◆ Race condition
  - ◆ Corruption
  - ◆ Bank account example
- When are resources shared?
  - ◆ Global variables, static objects
  - ◆ Heap objects

# Mutual Exclusion

- What is mutual exclusion?
- What is a critical section?
  - ◆ What guarantees do critical sections provide?
  - ◆ What are the requirements of critical sections?
    - » Mutual exclusion (safety)
    - » Progress (liveness)
    - » Bounded waiting (no starvation: liveness)
    - » Performance
- How does mutual exclusion relate to critical sections?
- What are the mechanisms for building critical sections?
  - ◆ Locks, semaphores, monitors, condition variables

# Locks

- What does Acquire do?
- What does Release do?
- What does it mean for Acquire/Release to be atomic?
- How can locks be implemented?
  - ◆ Test-and-set spinlocks
  - ◆ Disable/enable interrupts
- What are the limitations of using spinlocks, interrupts?
  - ◆ Inefficient, interrupts turned off too long

# Semaphores

- What is a semaphore?
  - ◆ What does Wait/P/Decrement do?
  - ◆ What does Signal/V/Increment do?
  - ◆ How does a semaphore differ from a lock?
  - ◆ What is the difference between a binary semaphore and a counting semaphore?
- When do threads block on semaphores?
- When are they woken up again?
- Using semaphores to solve synchronization problems
  - ◆ Readers/Writers problem

# Deadlock

- Deadlock happens when processes are waiting on each other and cannot make progress
- What are the conditions for deadlock?
  - ◆ Mutual exclusion
  - ◆ Hold and wait
  - ◆ No preemption
  - ◆ Circular wait
- How to visualize, represent abstractly?
  - ◆ Resource allocation graph (RAG)
  - ◆ Waits for graph (WFG)

# Deadlock Approaches

- Dealing with deadlock
  - ◆ Ignore it
  - ◆ Prevent it (prevent one of the four conditions)
    - » Acquire recourses in the same order
  - ◆ Avoid it (have tight control over resource allocation)
  - ◆ Detect and recover from it

# Lets do some problems

- Check iLearn for keys to homework and previous exams