Administrivia

• Midterm
  • In class on Friday (May 5)
  • Covers introduction to deadlock
  • Based on lecture, textbook, and supplementary materials
    • Indicated as Preparation on the class schedule
  • Closed book. No additional sheets of notes
Overview

• Introduction to Operating Systems
• Architecture support
• Processes
• Scheduling
• Threads
• Synchronization
Operating Systems

• What is an OS?
  • "All the code that you didn’t have to write"

• Roles of an OS
  • **Abstraction**: defines a set of logical resources (**objects**) and a set of well-defined operations on those objects (**interfaces**)
  • **Virtualization**: isolates and multiplexes physical resources via spacial and temporal sharing
  • **Access control**: who, when, how
Architecture support

• Isolation
  • Protects hardware configuration and direct access
• Virtualization
  • Efficient multiplexing
• Events generation and dispatch
  • Improves resources utilization
• Atomic operations
  • Concurrency and parallelism
Privileged instructions

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

• Synchronous vs. asynchronous
• Expected vs. unexpected

• 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?
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 manipulation

- 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?
- What happens during process termination?
Scheduling

• What kinds of scheduling is there?
  • Long-term scheduling
  • Short-term scheduling

• Components

• When does scheduling happen?
  • Job changes state (e.g., waiting to running)
  • Interrupt, exception
  • Job creation, termination
Scheduling goals

• Good service metrics
  • Maximize CPU utilization
  • Maximize job throughput
  • Minimize turnaround time
  • Minimize waiting time
  • Minimize response time
  • Guarantee fairness
  • etc.
Schedule goals (cont.)

• What is the goal of a batch system?
• What is the goal of an interactive system?
• What is the goal of an mobile system?
Starvation

- Starvation
  - Indefinite denial of a resource (CPU, lock)
- Causes
  - Side effect of scheduling
  - Side effect of synchronization
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)
- Round Robin
- Multilevel feedback queues
- Fair-share
Thread

• What is a thread?
  • What is the difference between a thread and a process?
  • How are they related?
• Why are threads useful?
• 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?
  • Create, yield, exist, join, etc.

• What is a context switch?

• What is the difference between non-preemptive scheduling and preemptive thread scheduling?
Synchronization

- Why do we need synchronization?
  - Coordinate access to shared data structures
  - Coordinate thread/process execution
- What can happen to shared data structures if synchronization is not used?
  - Race condition
  - Corruption
- 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?
• How does mutual exclusion relate to critical sections?
• What are the mechanisms for building critical sections?
Locks

• What does `acquire()`/`release()` do?
• What does it mean for `acquire()`/`release()` to be atomic?
• How can locks be implemented?
  • Spinlocks
  • Disable/enable interrupts
• How does test-and-set work?
  • What kind of lock does it implement?
• What are the limitations of using spinlocks, disabling interrupts?
Semaphores

- What is a semaphore?
  - What does `wait()`/`signal()` 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
Monitors

- What is a monitor?
  - Shared data, procedures, synchronization
- In what way does a monitor provide mutual exclusion?
  - To what extent is it provided?
- How does a monitor differ from a semaphore?
- How does a monitor differ from a lock?
- What kind of support do monitors require?
  - Language, run-time support
Condition variables

• What is a condition variable used for?
  • Coordinating the execution of threads
  • Not mutual exclusion

• Operations
  • What are the semantics of \texttt{wait()}/\texttt{signal()}/\texttt{broadcast()}?

• How are condition variables different from semaphores?
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
Deadlock visualization

• How to represent abstractly?
  • Resource allocation graph (RAG)
  • Waits for graph (WFG)
Deadlock approaches

• How to deal with deadlock?
  • Ignore it
  • Prevent it (prevent one of the four conditions)
  • Avoid it (have tight control over resource allocation)
  • Detect and recover from it

• What is the Banker's algorithm?
  • Which of the four approaches above does it implement?
For next class ...

- Example problems