Operating Systems
ECE344

Midterm review

Ding Yuan
Overview

• The midterm
• Architectural support for Oses
• Processes
• Threads
• Synchronization
Midterm

- Date: Feb 24th, 2-4 PM
- Location: EX300
- Covers material through synchronization
- Based upon lecture material and project
- No cheat sheets

- Please, do not cheat
  - Do not copy from your neighbor
    - You will be noticed
  - No one involved will be happy, particularly the teaching staff
Arch Support for OSes

• Why OS needs Arch support?
• Types of architecture support
  – Manipulating privileged machine state
  – Generating and handling events
    • Events: Interrupts, exceptions, system calls, etc.
  – Concurrency
  – ...

Ding Yuan, ECE344 Operating Systems
Protected (privileged) Instructions

• What are protected 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
Q1 from midterm@2013W

For each instruction below, is it a protected instruction?

(A) load instruction (read a value from memory into a register) (2 marks)

(B) modify the PC register (program counter) (2 marks)

(C) modify the SP register (stack pointer) (2 marks)

(D) modify the register that controls kernel/user mode (2 marks)

(E) direct access I/O device (2 marks)
Events

• Events
  – Synchronous: fault (exceptions), system calls
  – Asynchronous: interrupts, software interrupt

• 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?
  – Timer interrupt, why?
Processes

• What is a process?
• 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 NT 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?
Q4 from midterm@2013W

You write a UNIX shell, but instead of calling fork() then exec() to launch a new job, you instead insert a subtle difference: the code first calls exec() and then calls fork() like the following:

```c
shell (..) {
  . . .
  exec (cmd, args);
  fork();
  . . .
}
```

Does it work? What is the impact of this change to the shell, if any? (Explain)
Threads

• 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?
Q6 from midterm@2013W

Assume you want to implement a web-server for YouTube by using multithreading, where each thread serves one incoming request by loading a video file from the disk. Assume the OS only provides the normal blocking read system call for disk reads, do you think user-level threads or kernel-level threads should be used? Why?
Thread Implementation

• How are threads managed by the run-time system?
  – Thread control blocks, thread queues
• 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
Synchronization

• Why do we need synchronization?
  – Coordinate concurrent access to shared data structures
  – Coordinate thread/process execution

• What can happen to shared data structures if synchronization is not used?
  – Race condition
  – Bank account example

• When are resources shared?
  – Global variables, static objects
  – Heap objects
Mutual Exclusion

• What is mutual exclusion?
• What is a critical region?
  – What guarantees do critical region provide?
  – What are the requirements of critical regions?
    • Mutual exclusion (safety)
    • Progress (liveness)
    • Bounded waiting (no starvation: liveness)
    • No assumption of the CPU speed/number
• What are the mechanisms for building critical regions?
  – 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?
  – Spinlocks
  – Disable/enable interrupts
  – Blocking (OS161)
• How does test-and-set work?
  – What kind of lock does it implement?
• What are the limitations of using spinlocks, interrupts?
  – Inefficient, interrupts turned off too long
Q7 from midterm@2013W

A spin lock acquire() can be implemented with a test-and-set instruction as follows:

while (test-and-set(&lock->held) == 1)
  ; // spin

Now a new lock acquire() is implemented as follows:

1: while (1) {
2:   while (lock->held > 0)
3:     ; // spin
4:   if (test-and-set(&lock->held) == 0)
5:     return;
6: }

Does it work? How does it change the behavior of the lock compared to the first implementation?
Semaphores

• What is a semaphore?
  – What does P do?
  – What does V 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
  – Bounded Buffers problem
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 Wait?
  – What are the semantics of Signal?
  – What are the semantics of Broadcast?

• How are condition variables different from semaphores?
GOOD LUCK!