CS 153
Design of Operating Systems

Winter 2016

Midterm Review

Midterm

- in class on Monday
- Covers material through scheduling and deadlock
- Based upon lecture material and Chapters 1 to 7
  - Closed book. No additional sheets of notes

Project 2 is out today

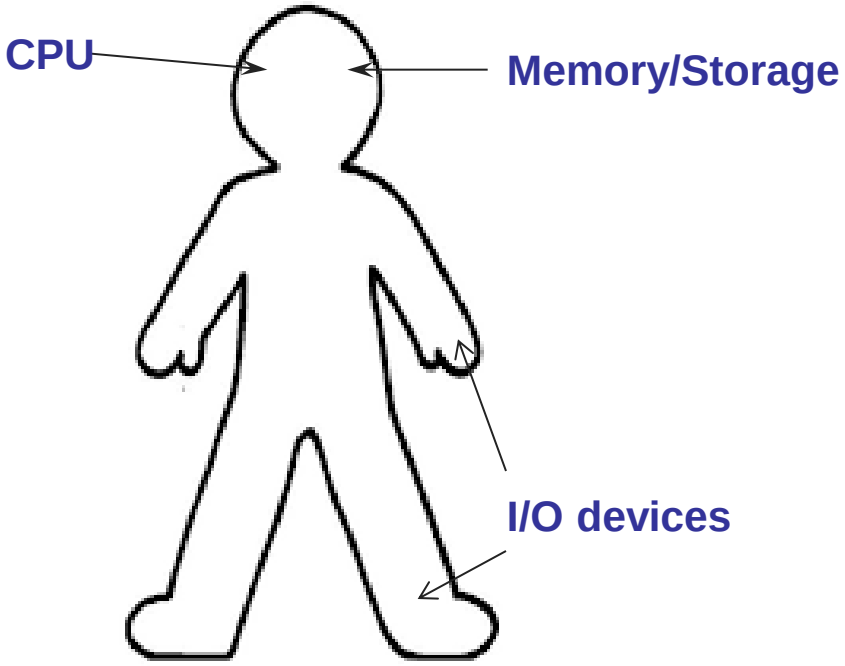
- Three weeks to complete
- Do not count on extensions!

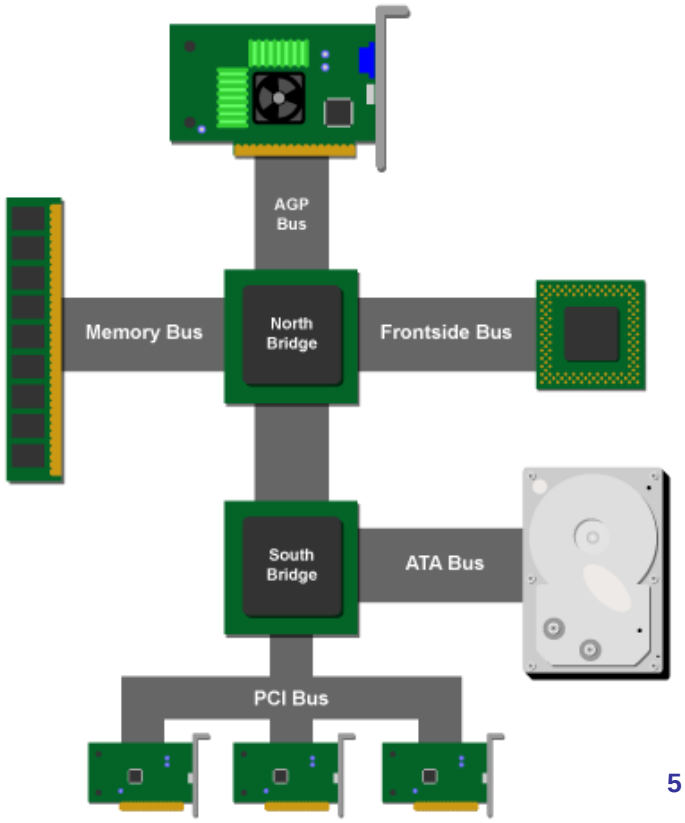
Overview

- Architectural support for Oses
- Processes
- Threads
- Synchronization
- Scheduling (including deadlock)

Arch Support for OSes

- Types of architecture support
  - Manipulating privileged machine state
  - Generating and handling events





Brain signals and controls

- https://www.youtube.com/watch?v=rSQNi5sAwuc

- One can control a different person's arm by forwarding brain signals!

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
- 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?

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?

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

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
  - 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?
  - Spinlocks
  - Disable/enable interrupts
- 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

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
  - 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?

Implementing Monitors

- What does the implementation of a monitor look like?
  - Shared data
  - Procedures
  - A lock for mutual exclusion to procedures (w/ a queue)
  - Queues for the condition variables
- What is the difference between Hoare and Mesa monitors?
  - Semantics of signal (whether the woken up waiter gets to run immediately or not)
  - What are their tradeoffs?
  - What does Java provide?

Scheduling

- What kinds of scheduling is there?
  - Long-term scheduling
  - Short-term scheduling
- Components
  - Scheduler (dispatcher)
- When does scheduling happen?
  - Job changes state (e.g., waiting to running)
  - 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

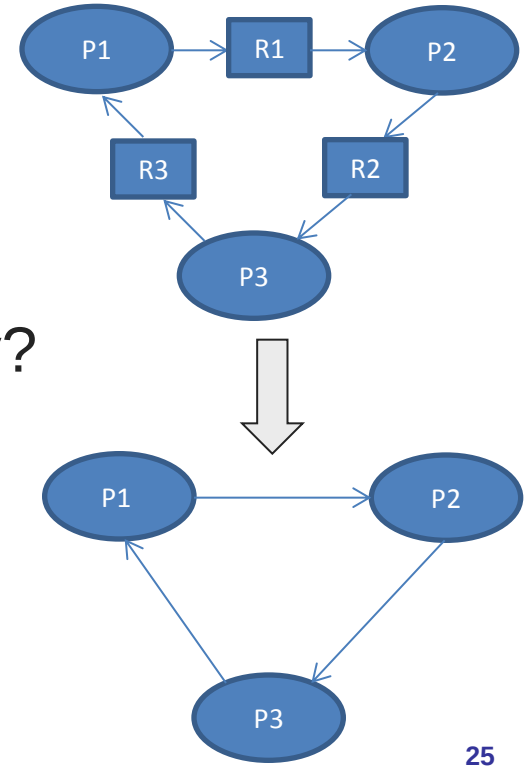
- Starvation
  - Indefinite denial of a resource (CPU, lock)
- Causes
  - Side effect of scheduling
  - Side effect of synchronization
- Operating systems try to prevent starvation

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?

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
  - Detect and recover from it
  - Prevent it (prevent one of the four conditions)
  - Avoid it (have tight control over resource allocation)

- What is the Banker's algorithm?
  - Which of the four approaches above does it implement?

Question 1

- Most operating systems are designed for general-purpose computation. A proposal has been put forth for an OS that is optimized for running math-intensive programs. In MathOS, the kernel includes system calls for many useful mathematical operations, such as matrix arithmetic, Bessel functions, Euclidean distance, etc. These system calls are written in highly optimized assembly language for maximum performance. Is this concept for MathOS a good idea? Explain why or why not.

- No! Math functions won't benefit from running in the kernel. They do not need privileged instructions or special facilities. You're putting crap in the kernel that has no reason to be there. Moreover: The overhead of calling them: mode switch + data copy is more time-consuming than a function call

Question 2

- What is the difference between a mode switch and a context switch?

- Mode switch: change CPU execution mode from one privilege level to another e.g., user → kernel via a trap or syscall.

- Context switch: save one process' execution context & restore that of another process

Question 3

- What is the similarity and difference between faults (exceptions) and interrupts?

- Both are unexpected, yet
- Faults are synchronous
- Interrupts are asynchronous

Question 4

- List two events that may take a process to a ready state.

- Startup: created → ready
- Preemption: running → ready
- I/O complete: blocked → ready

Question 5

- Given that we can create user-level code to control access to critical sections (e.g., Peterson's algorithm), why is it important for an operating system to provide synchronization facilities such as semaphores in the kernel?

- Question is about offering sync services via the kernel than via user-level code.
- To avoid busy waiting: the waiting thread can go to blocking mode
  - creates better CPU utilization; avoids priority inversion (if an issue)

Question 6

- What does a time-sharing system need that a multiprogramming system does not?

  (a) Trap mechanism

  (b) Kernel mode execution privileges

  (c) Shorter time slices

  (d) Timer interrupt

(d)

Question 7

- When does preemption take place?

  (a) When a quantum expires.
  (b) When a process issues an I/O request.
  (c) When a process exits.
  (d) All of the above.

(a)

Question 8

- What information is stored in a thread control block (TCB)?

  (a) List of open files.

  (b) Stack pointer.

  (c) Memory map.

  (d) Thread owner ID (e.g., Linux UID).

(b)

Question 9

- A test-and-set instruction allows you to:

(a) Modify a memory location only if its contents match a given value.

(b) Exchange the contents of two memory locations if their values are different.

(c) Exchange the contents of two memory locations if a lock is not set.

(d) Exchange the contents of two memory locations if a lock is set.

(a)

Question 10

- A thread that is blocked on a semaphore is awakened when another thread:

(a) Tries to decrement a semaphore's value below 0.

(b) Tries to increment the semaphore.

(c) Causes the semaphore's value to reach a specific number.

(d) Tries to block on the same semaphore

(b)

Question 11

- Switching among threads in the same process is more efficient than switching among processes. True of false?

- True

Question 12

- The value of a semaphore can never be negative.

True

Question 13

- Multilevel queues allow multiple processes to share the same priority level.

True