Course Wrap-up

Computer Systems Programming, Spring 2025

Instructor: Travis McGaha

Teaching Assistants:

Andrew Lukashchuk	Ashwin Alaparthi	Lobi Zhao
Angie Cao	Austin Lin	Pearl Liu
Aniket Ghorpade	Hassan Rizwan	Perrie Quek

Poll Everywhere

pollev.com/tqm

- What did you learn in this course?

Administrivia

- Final Project Autograder Posted
  - SOME of it is auto graded. There is a lot of functionality that is not autograded that you will need to implement
  - Extended to Midnight on Thursday
- This lecture: Course wrap-up. Next lecture: Exam Review
- Last Check-in posted (Due tomorrow night at midnight)
- End of semester survey posted, due Tuesday the 6th
- Exam logistics & Practice exam questions posted!

Logistics

- Project released
  - Due May 1st at midnight, please get started if you haven't already
  - Autograder to be posted soon
  - NOTE: part of it is manually checked, not auto-graded

- HW4
  - Due this Friday
  - Autograder posted

- Last Checkin to be released soon
  - Due May1st at midnight (late deadline over reading days)
  - (Post Semester Survey)

Poll Everywhere

pollev.com/tqm

- What did you learn in this course?

What have we been up to for the last 14 weeks?

- Ideally, you would have "learned" everything in this course, but we'll use red stars ☆ today to highlight the ideas that we hope stick with you beyond this course

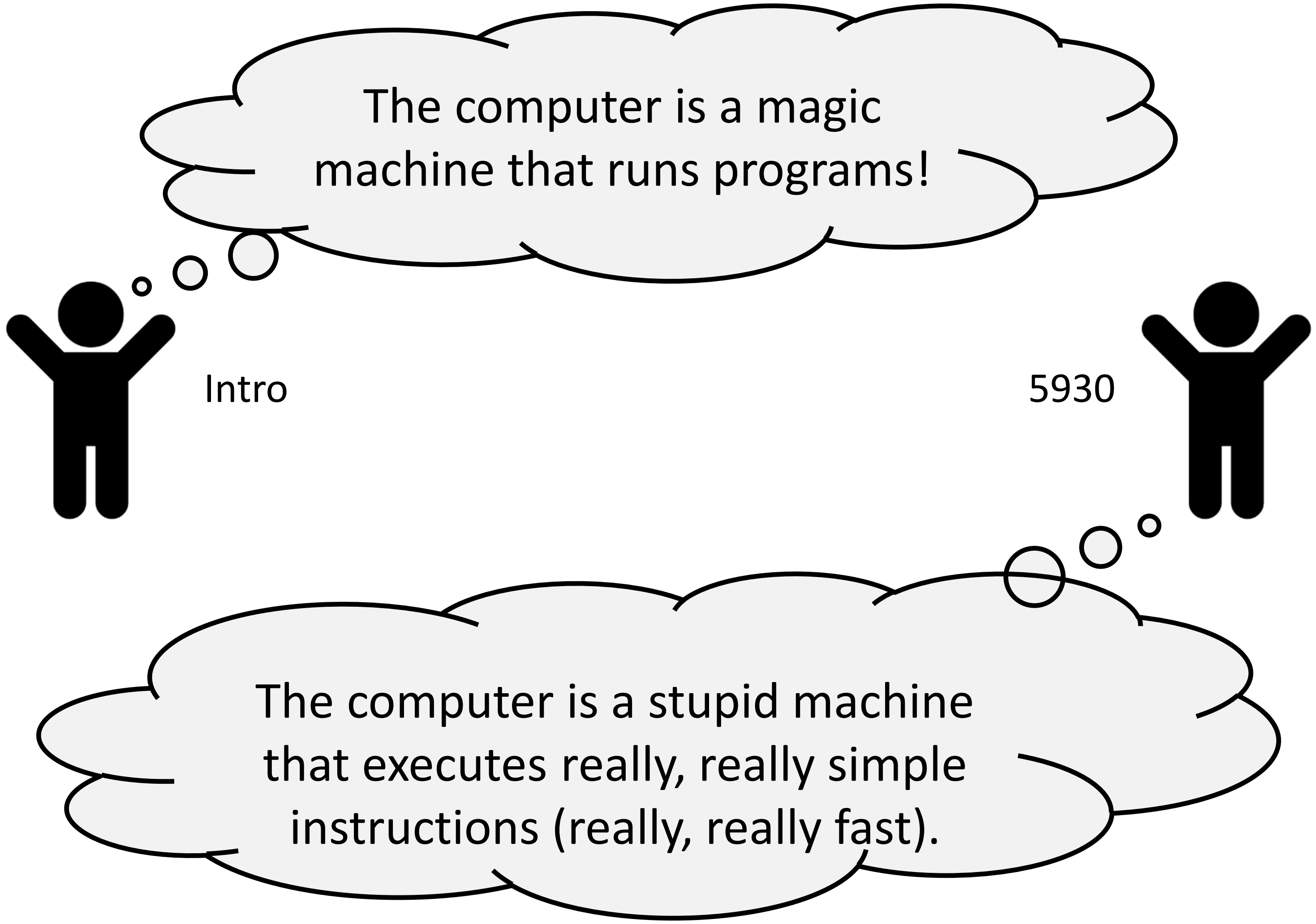
Poll Everywhere

pollev.com/tqm

- What did you learn in this course?

Course Goals

- Explore the gap between:


The computer is a magic machine that runs programs!
Intro
5930
The computer is a stupid machine that executes really, really simple instructions (really, really fast).

Systems Programming: The Why

- The programming skills, engineering discipline, and knowledge you need to build a system
  1) Understanding the “layer below” makes you a better programmer at the layer above
  2) Gain experience with working with and designing more complex “systems”
  3) Learning how to handle the unique challenges of low-level programming allows you to work directly with the countless “systems” that take advantage of it

So What is a System?

- "A **system** is a group of interacting or interrelated entities that form a unified whole. A system is delineated by its spatial and temporal boundaries, surrounded and influenced by its environment, described by its structure and purpose and expressed in its functioning."
  - https://en.wikipedia.org/wiki/System
  - Still vague, maybe still confusing

- But hopefully you have a better idea of what a system in CS is now
  - What kinds of systems have we seen…?

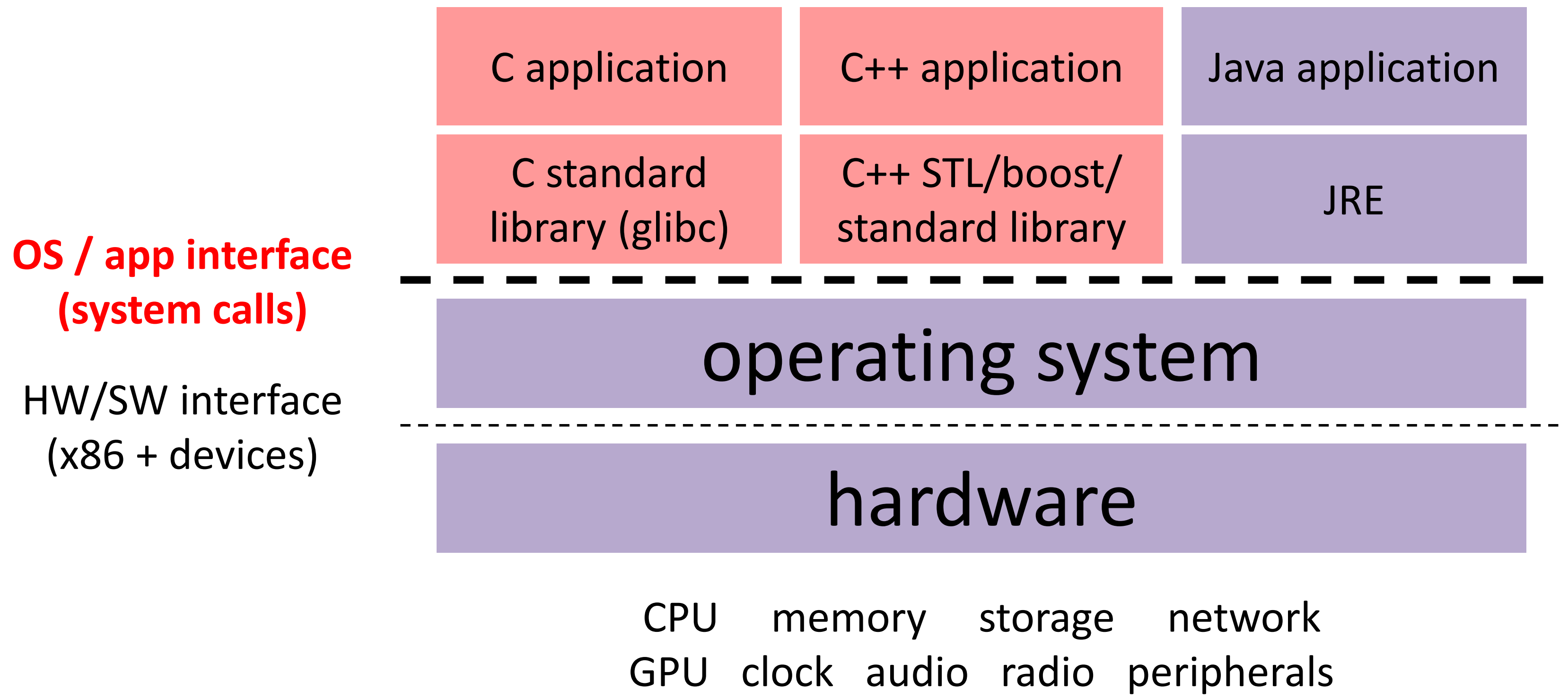
Software System

- Writing complex software systems is *difficult*!
  - Modularization and encapsulation of code
  - Resource management
  - Documentation and specification are critical
  - Robustness and error handling
  - Must be user-friendly and maintained (not write-once, read-never)

- **Discipline:** cultivate good habits, encourage clean code
  - Coding style conventions
  - Unit testing, code coverage testing, regression testing
  - Documentation (code comments, design docs)

The Computer as a System

- Modern computer systems are increasingly complex!
  - Networking, threads, processes, pipes, files
  - Buffered vs. unbuffered I/O, blocking calls, latency


C application
C++ application
Java application
C standard library (glibc)
C++ STL/boost/ standard library
JRE
OS / app interface (system calls)
operating system
HW/SW interface (x86 + devices)
hardware
CPU memory storage network
GPU clock audio radio peripherals

A Network as a System

- A networked system relies heavily on its connectivity
  - Depends on materials, physical distance, network topology, protocols
- Conceptual abstraction layers
  - Physical, data link, network, transport, session, presentation, application
  - Layered *protocol* model
    - We focused on IP (network), TCP (transport), and HTTP (application)
- Network addressing
  - MAC addresses, IP addresses (IPv4/IPv6), DNS (name servers)
- Routing
  - Layered packet payloads, security, and reliability

Systems Programming: The What

- The programming skills, engineering discipline, and knowledge you need to build a system
  - **Programming:** C & C++
  - **Discipline:** design, testing, debugging, performance analysis
  - **Knowledge:** long list of interesting topics
    - Concurrency, OS interfaces and semantics, techniques for consistent data management, distributed systems algorithms, …
    - Most important: a deep understanding of the "layer below"

Main Topics

- C
  - Low-level programming language
- C++
  - "better C" + classes + RAII + STL + …
- Memory management
- System interfaces and services
- Multi-processing Basics – Fork, Pipe, Exec
- Concurrency basics – POSIX threads, synchronization
- Networking basics – TCP/IP, sockets, …

Topic Theme: Abstraction

- C: `void*` as a generic data type
- C++: hide execution complexity
  - *e.g.*, operator overloading, dispatch, containers & algorithms
- C++: standard templates to generalize code
- OS: abstract away details of interacting with system resources via system call interface
- Networking: 7-layer OSI model hides details of lower layers
  - *e.g.*, DNS abtracts away IP addresses, IP addresses abstract away MAC addresses

Topic Theme: Using Memory

- Variables, scope, and lifetime
  - *Static*, *automatic*, and *dynamic* allocation / lifetime
  - C++ objects and destructors; C++ containers and copying
- Pointers and associated operators (`&`, `*`, `->`, `[]`)
  - Can be used to link data or fake "call-by-reference"
- Dynamic memory allocation
  - **`malloc`**/**`free`** (C), **`new`**/**`delete`** (C++), smart pointers (C++)
  - Who is responsible? Who owns the data? What happens when (not if) you mess this up? (dangling pointers, memory leaks, …)
- Tools
  - Debuggers (`gdb`), monitors (`valgrind`)
  - Most important tool: thinking!

Topic Theme: Data Passing

- C: output parameters
- C++: Copy constructors, and copy vs move semantics
- Threads: return values or shared memory/resources
  - Leads to synchronization concerns
- I/O to send and receive data from outside of your program (*e.g.*, disk/files, network, streams)
  - Linux/POSIX treats all I/O similarly
  - Takes a LONG time relative to other operations
  - Blocking vs. polling
- Buffers can be used to temporarily hold passed data
  - Buffering can be used to reduce costly I/O accesses, depending on access pattern. Similar thing for caches.

Topic Theme: Concurrency

- Processes
  - Exec
  - Process Groups
    - Terminal Control
  - IPC
    - Pipe
    - Signals
- Threads
  - Synchronization
    - mutex
    - Condition variables
  - Deadlock
- Concurrency vs parallelism

MISSING Topic Theme: Society

- One flaw (among others) of this course is how we don't talk about how this relates to the rest of the world
  - These systems we build do not have to necessarily be "evil", but can often be used in those ways
  - We need to work and communicate with other people, even in CS.

- Actions:
  - Take Algorithmic Justice (CIS 7000) with Danaë Metaxa
  - Join a community of people working on things that matter to you, (Unions or other organizations)
  - Join me as a TA for next year. We will try to integrate ethics into those courses (still working out details).

This stuff is not C/C++ Exclusive

- These topics apply to other programming languages:
  - Python subprocess: https://docs.python.org/3/library/subprocess.html
  - Java threadpool: https://docs.oracle.com/javase/tutorial/essential/concurrency/pools.html
  - C# TCP Socket: https://learn.microsoft.com/en-us/dotnet/api/system.net.sockets.tcpclient?view=net-9.0
  - Python system call wrappers: https://docs.python.org/3/library/fcntl.html
  - Etc.

- These features are supported by almost all programming languages

Poll Everywhere

pollev.com/tqm

- Is there anything you wish we talked about (more) in this course?

Congratulations!

- Look how much we learned!

- Lots of effort and work, but lots of useful takeaways:
  - Debugging practice
  - Reading documentation
  - Tools (gdb, `valgrind`, `helgrind`)
  - C and C++ familiarity, including multithreaded and networked code

- Go forth and build cool systems!

Impost Syndrome

- **Impostor syndrome**, also known as **impostor phenomenon** or **impostorism**, is a psychological experience in which a person suffers from feelings of intellectual and/or professional fraudulence.

- Don’t just look at how others are doing. Look at the progress you have made. It may be more gradual, but progress is progress. You get better with time and practice

- It is ok to not have an internship, things may still work out.

Future Courses

- Systems Courses
  - CIS 5050: Software Systems
  - CIS 5480: Operating Systems Design and Implementation
  - CIS 5521 Compilers
  - CIS 5470: Software Analysis
  - CIS 5530: Networked Systems
  - CIS 5550 Internet and Web Systems
  - CIS 5500: Database and Information Systems

- Otherwise related courses
  - CIS 5600 Interactive Computer Graphics
  - CIS 5650 GPU Programming and Architecture

Thanks for a great semester!

- Special thanks to all the instructors before me (Both at UPenn and UW) who have influenced me to make the course what it is

- Huge thanks to the course TA's for helping with the course!

Thanks for a great semester!

- Thanks to you!
  - It has been another tough semester. Look at the state of society ☺
  - Relatively “new” version of the course. Many of the assignments and infrastructure are recently developed.
  - You’ve made it through so far, be proud that you’ve made it and what you’ve accomplished!

- **Please take care of yourselves, your friends, and your community**

Poll Everywhere

pollev.com/tqm

- Ask Me Anything!