

Course Wrap-up

Computer Operating Systems, Spring 2024

Instructor: Travis McGaha

Head TAs: Nate Hoaglund & Seungmin Han

TAs:

Adam Gorka

Haoyun Qin

Kyrie Dowling

Ryoma Harris

Andy Jiang

Jeff Yang

Oliver Hendrych

Shyam Mehta

Charis Gao

Jerry Wang

Maxi Liu

Tom Holland

Daniel Da

Jinghao Zhang

Rohan Verma

Tina Kokoshvili

Emily Shen

Julius Snipes

Ryan Boyle

Zhiyan Lu



pollev.com/tqm

❖ How is PennOS going?

Administrivia

❖ PennOS

- Everyone should have already contacted their group, and should get started working on it.
- Full Thing due ~April 22nd (Yesterday)
 - Can still use late tokens, so late deadline is April 26th
 - After you submit, you need to schedule a meeting with your TA to demonstrate that it is working
- I am told some people are splitting the kernel into shell vs non-shell.
 - This is usually a terrible Idea. The Shell depends a lot on the Kernel and know how the kernel works will help A LOT. Shell can't be tested much until Kernel is implemented.

Administrivia

- ❖ Check-in released: due before lecture Thursday next week
 - Don't forget to do it!

- ❖ We released stress.c and stress.h for testing your PennOS kernel
 - Note: there was originally an error when first released. It should be calling the linux system call `usleep` in the provided code and NOT `s_sleep`


- ❖ CIS TA Application is out now!
 - Intro courses are due Tomorrow night @ midnight
 - 2400 is “due” April 26th @ midnight

Lecture Outline

- ❖ Course Wrap-up
- ❖ AMA & OH

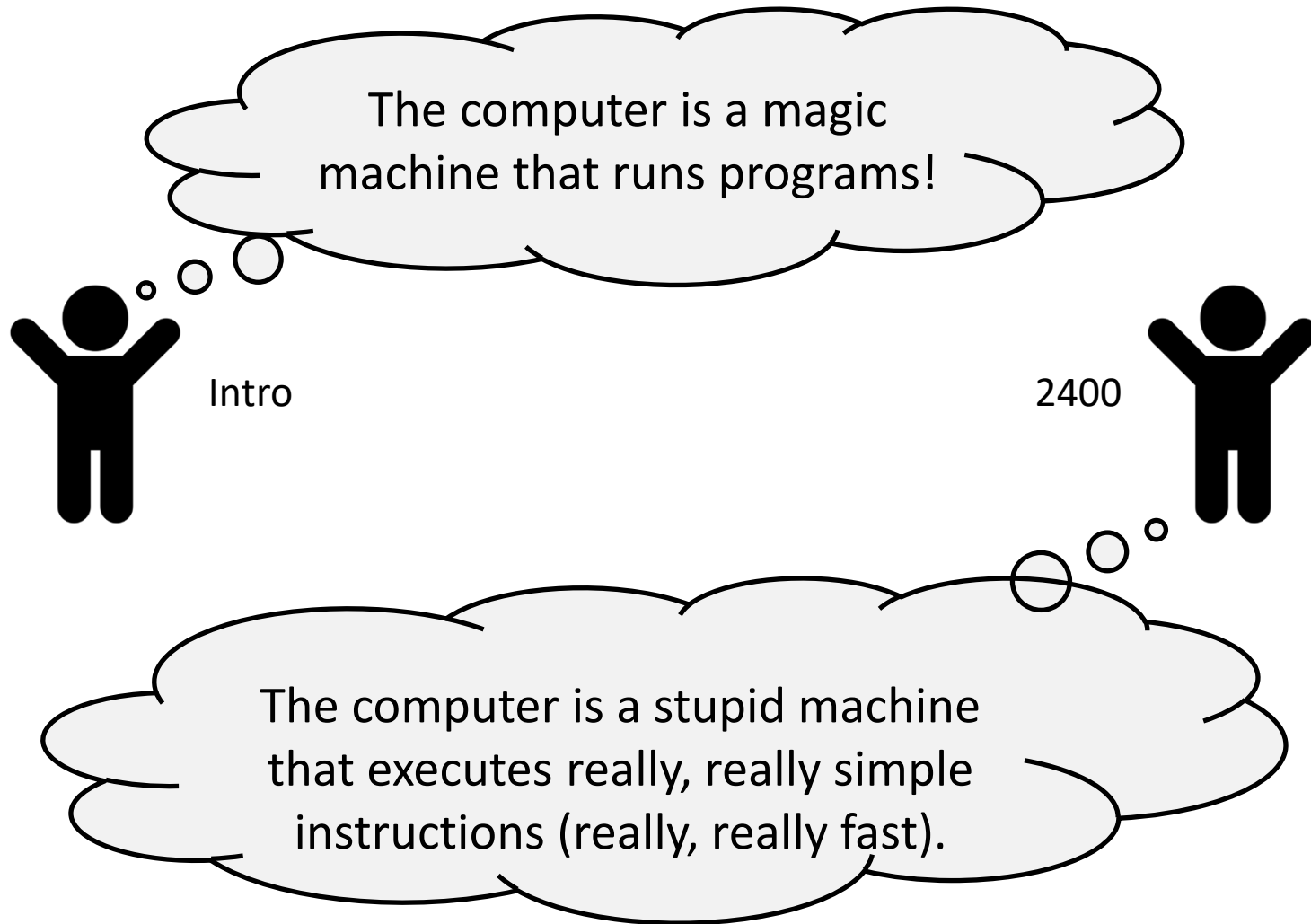


What have we been up to for the last 14 weeks?

- 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

Course Goals

- ❖ Explore the gap between:



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...?

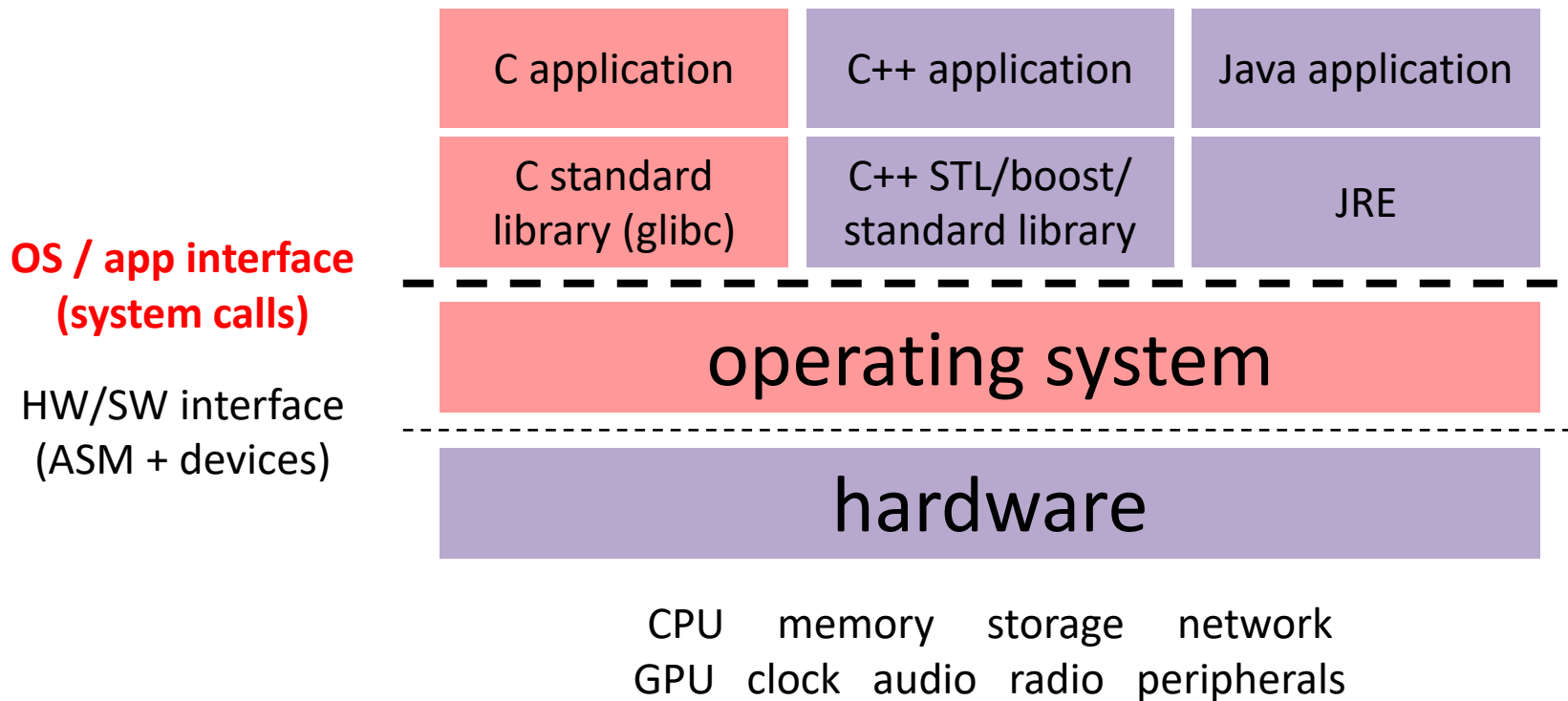
Software System

- ❖ 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!
 - threads, processes, pipes, files
 - Buffered vs. unbuffered I/O, blocking calls, caches, virtual memory



Systems Programming: The What

- ❖ The programming skills, engineering discipline, and knowledge you need to build a system

-  **Programming:** C (& other languages)

- **Discipline:** design, testing, debugging, performance analysis
- **Knowledge:** long list of interesting topics
 - Concurrency, OS interfaces and semantics, techniques for consistent data management, ...

-  Most important: a deep understanding of the “layer below”

Main Topics

- ❖ C
 - Low-level programming language
- ❖ Memory management & allocation
- ❖ System interfaces and services
- ❖ Concurrency basics – POSIX threads, synchronization
- ❖ Multi-processing Basics – Fork, Pipe, Exec
- ❖ Buffering, Caches, Locality
- ❖ Operating System Internals
 - File systems
 - Scheduling
 - Virtual Memory

Topic Theme: Abstraction

- ❖ C: **void*** to abstract away types for some functions (pthread_create, read, write, etc).
- ✳ abstract away details of interacting with system resources via system call interface (e.g. file descriptors and pids)
- ✳ The concept of processes and virtual memory to abstract away sharing hardware
- ✳ Read Write Locks and monitors abstract away their implementation of using a mutex & condition variable
- ❖ Nice abstractions minimize cognitive complexity and make it harder for users of the abstraction to fuck up.

Topic Theme: Data & Locality

- ✳️ C: Memory model (Stack vs Heap)
 - ❖ 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. non-blocking (and the sin that is spinning)

- ✳️ Buffers can be used to temporarily hold data

- Buffering can be used to reduce costly I/O accesses, depending on access pattern

- ✳️ Caching & Locality

- Some memory is quicker to access than others
- Hardware makes assumptions on your program's access patterns

Topic Theme: Allocating Resources

- ❖ It is often the tasks of a system to distribute/allocate a finite number of resources:
 - Scheduling algorithms allocate which threads can utilize the CPU
 - Memory allocation schemes (slab allocator, buddy algorithm)
 - Virtual Memory: allocating pages in physical memory
 - Caches: deciding what memory is in the cache.
 - File System: Allocating Blocks in file system

- ❖ These allocation schemes need to consider:
 - Efficient utilization of the resource that is being allocated
 - Fragmentation, fairness, minimize times we go to slower storage
 - Minimal overhead in the allocation scheme.
 - Time spent on the allocation is time not spent doing other things

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
 - Take Software Engineering (CIS 3500)
 - Join a community of people working on things that matter to you, (Unions or other organizations)
 - Join me as a TA for 2400 or 3800 next year. We will try to integrate ethics into those courses (still working out details).

Congratulations!

- ❖ Look how much we learned!

- ❖ Lots of effort and work, but lots of useful takeaways:
 - Debugging practice
 - Reading documentation
 - Tools (gdb, valgrind)
 - C familiarity
 - Concurrent Programming
 - Designing large systems
 - Working with others

- ❖ Go forth and build cool systems!

Future Courses

❖ Systems Courses

- CIS 3410: Compilers
- CIS 5050: Software Systems
- CIS 5530: Networked Systems
- CIS 5550: Internet and Web Systems
- CIS 5500: Database and Information Systems
- CIS 5470: Software Analysis

❖ Otherwise related courses

- CIS 5600 Interactive Computer Graphics
- CIS 5610 Advanced Computer Graphics
- CIS 5650 GPU Programming and Architecture
- CIS 5510 Security

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!



Nate Hoaglund

He/Him



nhoag@seas

Seungmin Han

he/him/his

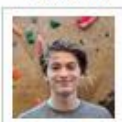


hanseun@seas

Teaching Assistants

Adam Gorka

He/Him



agorka@seas

Andy Jiang

he/him/his



jianga@wharton

Charis Gao

she/her



charisg@seas

Daniel Da

he/him/his



dadaniel@wharton

Emily Shen

she/her/hers



shenyit@seas

Haoyun Qin

He/him/his



qh@seas

Jeff Yang

he/him/his



yjeffrey@seas

Jerry Wang

He/His/Him



jwang01@seas

Jinghao Zhang

she/her



jinghaoz@seas

Julius Snipes

He/Him



jesnipes@seas

Kyrie Dowling

she/they



kyrdo@seas

Oliver Hendrych

he/him



hendrych@seas

Maxi Liu

she/her



maxiliu@seas

Rohan Verma

He/Him/His



rover@seas

Ryan Boyle

she/her



ryboyle@seas

Ryoma Harris

he/him



ryomah@seas

Shyam Mehta

he/him



smehta1@seas

Tom Holland

he/him



tomhol@seas

Tina Kokoshvili

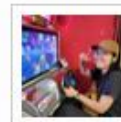
she/her/hers



tinatin@seas

Zhiyan Lu

she/her



zhiyanlu@sas

Thanks for a great semester!

- ❖ Thanks to you!
 - It has been another tough semester. Still not completely out of the pandemic, Zoom fatigue, faltering motivation, etc
 - My Second offering of the course, things are still a bit rough
 - 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**

Ask Me Anything

