Welcome to CS110: Principles of Computer Systems

- **I'm Jerry Cain** (jerry@cs.stanford.edu)
  - Chemistry undergrad MIT, originally Ph.D. student in chemistry here, defected to CS
  - Lecturer in CS, teaching CS106AX, CS106X, CS107, and CS110
  - Taught CS110 for the first time in Spring 2013
    - Leveraged much of Mendel Rosenblum's CS110 materials from prior offerings
    - Introduced my own materials since then, and will introduce even more this time
    - CS110 is an evolving system, but hopefully you don't notice one bit
  - Started working at Facebook in 2008, still formally associated with them in emeritus role
    - Learned web programming, PHP, CSS, JavaScript. Old CS107 student of mine (class of 2004) is my manager
    - Have grown to understand and appreciate large systems much better as a result of working there

Welcome to CS110: Principles of Computer Systems

- Staff and Students
  - 445 students as of April 4th at 7:00pm (projected enrollment was 175!)
  - Each of you should know C and C++ reasonably well so that you can…
    - write moderately complex programs
    - read and understand portions of large code bases
    - trace memory diagrams and always win
  - Each of you should be fluent with Unix, `gcc`, `valgrind`, and `make` to the extent they're covered in CS107 or its equivalent.
  - 15 graduate student CA's at the moment
    - Caroline, Ryan, Kristine, Anirudh, Wantong, Robbie, Raejoon, Clara, Luke, Armin, Matt, Anand, Shreya, Karey, Frits
    - We're working on getting more CAs. We can never have enough!

CS110 Class Resources

- Course Web Site: https://cs110.stanford.edu
  - Very simple, optimized to surface exactly what you need and nothing else
  - Check the website for information about upcoming lectures, assignment handouts, discussion sections, and links to slide decks like those you're working through right now
- Online Student Support
  - Peer-collaborative forum I: Piazza (for the questions that require staff response)
  - Peer-collaborative forum II: Slack (for the questions that needn't involve course staff)
- Office Hours
  - Jerry's office hours are still TBD, but I expect to have at least four hours per week. I'll decide shortly.
  - CA's will provide a full matrix of office hours, soon to be determined
  - Office hours are not for debugging your assignments, and the CA's have been instructed to not look at code. Ever.

CS110 Class Resources

- Two Textbooks
  - First textbook is other half of CS107 textbook
    - "Computer Systems: A Programmer's Perspective", by Bryant and O'Hallaron
    - Stanford Bookstore stocks custom version of just the four chapters needed for CS110
  - Second textbook is more about systems-in-the-large, less about implementation details
    - "Principles of Computer System Design: An Introduction", by Jerome H. Saltzer and M. Frans Kaashoek
    - Provided free-of-charge online, chapter by chapter. Not stocked at Stanford Bookstore by design. You can buy a copy of it from Amazon if you want.

CS110 Class Resources

- Lecture Examples
  - By default, we'll rely on lecture recordings from Winter 2019, when I co-taught CS110 with Chris Gregg. Those lectures were 80 minutes, so I'll post them on Mondays and Wednesday by 1:30pm.
  - Lectures are generally driven by coding examples, and all coding examples can be copied/cloned into local space so you can play and confirm they work properly
  - Code examples are developed and tested on the `myth` machines, which is where you'll complete all of your CS110 assignments
  - The accumulation of all lecture examples will be housed in a git repository at `/usr/class/cs110/lecture-examples`, which you can initially `git clone`, and then subsequently `git pull` to get the newer examples as I check them in
- Lecture Slides
  - I rely on slides like these when we need to press through lots of information not driven by coding examples
  - All lectures will have them. When provided, they'll be organic, in that we'll inject updates and clarifications (and be clear we added stuff when it really impacts you).

Course Syllabus

- Overview of Linux Filesystems
  - Linux and C libraries for file manipulation: `stat`, `struct stat`, `open`, `close`, `read`, `write`, `readdir`, `struct dirent`, file descriptors, regular files, directories, soft and hard links, programmatic manipulation of them, implementation of `ls`, `cp`, `find`, and other core Unix utilities you probably never realized were plain old C programs
  - Naming, abstraction and layering concepts in systems as a means for managing complexity, blocks, inodes, inode pointer structure, inode as abstraction over blocks, direct blocks, indirect blocks, doubly indirect blocks, design and implementation of a file system
- Exceptional Control Flow
  - Introduction to multiprocessing, `fork`, `waitpid`, `execvp`, process ids, interprocess communication, context switches, user versus kernel mode, system calls and how their calling convention differs from those of normal functions
  - Protected address spaces, virtual memory, virtual to physical address mapping, scheduling
  - Concurrency versus parallelism, multiple cores versus multiple processors, concurrency issues with multiprocessing, signal masks

Course Syllabus

- Threading and Concurrency
  - Sequential programming, desire to emulate the real world within a single process using parallel threads, free-of-charge exploitation of multiple cores (two per `myth` machine, 12-16 per `wheat` machine, 16 per `oat` machine), pros and cons of threading versus forking
  - C++ threads, `thread` construction using function pointers, blocks, functors, `join`, `detach`, race conditions, `mutex`, IA32 implementation of `lock` and `unlock`, spinlock, busy waiting, preemptive versus cooperative multithreading, `yield`, `sleep_for`
  - Condition variables, `condition_variable_any`, rendezvous and thread communication, `wait`, `notify_one`, `notify_all`, deadlock, thread starvation
  - Semaphore concept and `semaphore` implementation, generalized counters, pros and cons of `semaphore` versus exposed `condition_variable_any`, thread pools, cost of threads versus processes
  - Active threads, blocked threads, ready threads, high-level implementation details of a thread manager, `mutex`, and `condition_variable_any`
  - Pure C alternatives via `pthreads`, pros and cons of `pthreads` versus C++'s `thread` package

Course Syllabus

- Networking and Distributed Systems
  - Client-server model, peer-to-peer model, telnet, protocols, request, response, stateless versus keep-alive connections, latency and throughput issues, `gethostbyname`, `gethostbyaddr`, IPv4 versus IPv6, `struct sockaddr` hierarchy of records, network-byte order
  - Ports, sockets, socket descriptors, `socket`, `connect`, `bind`, `accept`, `read`, `read`, simple echo server, time server, concurrency issues, spawning threads to isolate and manage single conversations
  - C++ layer over raw C I/O file descriptors, introduction to `sockbuf` and `sockstream` C++ classes (via socket++ open source project)
  - HTTP 1.0 and 1.1, header fields, `GET`, `HEAD`, `POST`, response codes, caching
  - MapReduce programming model, implementation strategies using multiple threads and multiprocessing
  - Nonblocking I/O, where normally slow system calls like accept, `read`, and `write` return immediately instead of blocking
    - `select`, `epoll`, and `libev` libraries all provide nonblocking I/O alternatives to maximize CPU time using a single thread of execution within a single process

Expectations

- Programming Assignments
  - 80% of final grade, expect six assignments
  - Some assignments are single file, others are significant code bases to which you'll contribute. If CS107 is about mastering the periodic table and understanding the chemistry of every single element, CS110 is about building rich, durable polymers
  - Regardless of your overall average, you must get 50% of the points on each of the six assignments in order to pass the class.
  - Late policy is different than it is for many other CS classes
    - Every late day *potentially* costs you (read below why it's *potentially*)
      - If you submit on time, you can get 100% of the points. Woo.
      - If you can't meet the deadline, you can still submit up to 24 hours later, but your overall score is capped at 90%
      - If you need more than 24 additional hours to submit, you can submit up to 48 hours later, but overall score is capped at 70%
      - Assignments are only ever accepted more than 48 hours after the deadline by arrangement.

Expectations

- Discussion Sections
  - In addition to lectures, you'll also sign up for an 50-minute section to meet each week
  - We introduced the CS110 discussion section for the first time some three years ago, and the general consensus is that they've substantially improved the course
  - Section will be a mix of theoretical work, coding exercises, and advanced software engineering etudes using `gdb` and `valgrind`
  - Discussion section signups will go live a week from today, on Monday, April 13th.
  - 20% of final grade
    - If you participate in all sections and submit all lab sheets, your section grade is 100%
    - Your overall section grade is a simple percentage of the sections you fully engage in and submit lab sheets for.
    - We're relying on the online discussion section to foster a sense of community that will otherwise be conspicuously missing unless we work on creating one.
    - We're planning on offering a huge array of times—mostly on Thursday, and some on Friday—that will work for everyone across all times zones.

Expectations

- Optional Self-Assessments
  - There are no exams at all this quarter
    - The university cancelled traditional final exams, citing a distaste for any form of high-stakes exam that would cause undue stress for those who are positionally and situationally disadvantaged by the state of the world right now.
    - My takeaway from the cancellation: It's a horrific quarter to be giving traditional exams of any form.
  - I'm assuming everyone is still interested in learning the material and welcomes the opportunity to take a pulse on how well he or she understands the material.
  - My compromise to those who understandably crave some feedback: **optional** self-assessments
    - I plan on creating three very short, targeted quizzes, and I'll make them available the Fridays of Week 3, 6, and 8.
    - These quizzes—again, **completely optional**—should be able to be completed in 45 minutes by those who are fully on top of the material.
    - I will, however, give you the entire week—during which no assignments will be assigned—to work through the questions before I distribute answers.

Honor Code

- Please take it seriously, because the CS Department does
  - Everything you submit for a grade is expected to be original work
  - Provide detailed citations of all sources and collaborations
  - The following are clear no-no's for CS110 assignments
    - Looking at another person’s code
    - Showing another student your code
    - Discussing assignments in such detail that you duplicate a portion of someone else's code in your own program
    - Uploading your code to a public repository (e.g. github) so others can find it
      - If you'd like to upload your code to a **private** repository, you can do so on github or some other hosting service that provides free-of-charge private hosting