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 CS106AJ, CS106B, CS106X, CS107, CS110
  - Taught CS110 for the first time in Spring 2013
    - Leveraged much of Mendel Rosenblum's CS110 materials from prior offerings
    - Introduced some of my own material since then, and will be introducing 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 an emeritus role
    - Learned web programming, PHP, CSS, JavaScript. Old CS107 student (class of 2004) is still 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

- **I'm Chris Gregg** (cgregg@stanford.edu)
  - Electrical Engineering undergrad Johns Hopkins, Master's of Education, Harvard, Ph.D. in Computer Engineering, University of Virginia
  - Lecturer in CS, teaching CS 106B/X, CS 107/107E, CS208E, CS 110
  - At Stanford since 2016, at Tufts prior, and high school teaching prior to that.
  - First time teaching CS 110
    - Will be teaching it again in the Spring quarter
  - I love to tinker
    - Stop by my office (Gates 201) some time to see my musical typewriter project.
    - I'm always happy to chat about Arduino / Raspberry Pi / iOS apps you are working on

Welcome to CS110: Principles of Computer Systems

- Staff and Students
  - 248 students as of January 7th at 1:00pm
  - 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.
  - 9 graduate student CA's at the moment
    - Feross, Ikechi, Zoe, Ryan, Garrick, Divyahans, Jake, Sarah, Blanca
    - If student enrollment surpasses 250, we'll get more CA's

CS110 Class Resources

- Course Web Site: http://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 lecture slides like the one 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 Wednesdays from 3:15 until 5:00pm in Gates 192, and Chris's office hours are Tuesdays from 9:00 until 11:00am and then again on Thursdays from 10:00am until 12:00pm
  - 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
  - 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 will be 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/winter-2019`, which you can initially `git clone`, and then subsequently `git pull` to get the newer examples as I check them in

CS110 Class Resources

- Lecture Slides
  - Will rely on slides when we need to press through lots of information not driven by coding examples
  - Most 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)
  - They are not a substitute for attending lecture
    - We go off script quite a bit and discuss high-level concepts, and you're responsible for anything that comes up in lecture
    - Exams include short answer questions in addition to coding questions, so all aspects of the course are tested

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
  - 40% of final grade, expect eight 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
  - 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 60%
      - No assignments are ever accepted more than 48 hours after the deadline
    - Exception: first assignment must be submitted on time, no late days allowed
    - Requests for extensions are routinely denied, save for extenuating circumstances (e.g. family emergency, illness requiring medical intervention, and so forth)

Expectations

- Discussion Sections
  - In addition to our MW lectures, you'll also sign up for an 80-minute section to meet each week
  - We introduced the CS110 discussion section for the first time almost two years ago, and the general consensus is that they've substantially improved the course
  - If you have a laptop, bring it to discussion section. 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 later this week
  - 5% of final grade, provided you attend all of them
    - Everyone's discussion section grade is 100%
    - Every time you miss a discussion section, your discussion section grade counts a little less, and your final exam score counts a little more
    - Exact policy details are spelled out in the Course Information handout

Expectations

- Midterm
  - In-class midterm is Friday, February 15th at 1:30pm.
  - 20% of final grade, material drawn from first five or so weeks of lecture, mix of implementation and short answer questions
  - Closed-book, closed-notes, closed-electronics, one double-sided cheat sheet that you can prepare ahead of time
  - You must pass the midterm in order to pass the class
    - Passing score will be revealed on midterm solution set, which will be posted well before the withdrawal deadline
  - Multiple practice midterms will be provided
  - If you have a competing class and would prefer to take the midterm another time, we'll allow you to take it earlier that same day, provided you email Jerry ahead of time and explain why you need to take the exam earlier

Expectations

- Final Exam
  - Three-hour final is Wednesday, March 20th at 3:30pm
    - 35% of final grade, cumulative, mix of implementation and short answer questions
      - Counts even more with each discussion section absence
    - Closed-book, closed-notes, closed-electronics, two double-sided cheat sheets that you can prepare ahead of time
    - You must pass the final in order to pass the class
    - Multiple practice finals will be provided
  - The final exam will also be offered on Wednesday, March 20th at 7:00pm
    - Only students with another 3:30pm final on the 20th are permitted to take the final exam at the later time
    - Email Jerry directly if you need to take the final exam during the alternate time slot because of a competing final
  - The final exam isn't being offered any other times.

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
    - Looking at another student'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