Welcome to CS110: Principles of Computer Systems

- **I'm Jerry Cain** ([jerry@cs.stanford.edu](mailto:jerry@cs.stanford.edu))
  - Chemistry undergrad MIT, originally Ph.D. student in chemistry here, defected to CS
  - Senior Lecturer in CS, teaching CS106AX, CS106X, CS107, and CS110
  - Taught CS110 for the first time in Spring 2013, and I absolutely love teaching it!
    - 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 still an evolving system—particularly given the upheaval that is 2020—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
  ○ 220 students (!) as of September 14th at 9:00 am (only expected 125, since 508 students enrolled last spring and we thought there were no students left!)
  ○ 8 graduate student CA's at the moment
    ■ Ella, Ryan, Raejoon, Robbie, Semir, Chris, Matt, Patrick
    ■ We’re working on getting at least one more CA. We can never have enough!

• Soft Prerequisites
  ○ 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.
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: Ed (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, also TBD
  - Office hours are not for debugging your assignments, and the CA's have been instructed to never look at code, save for Assignment 1, which is designed to get everyone from varying backgrounds up to speed so that Assignments 2 through 6 all go brilliantly.
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, and that custom version can be purchased online from the Bookstore, and they’ll ship it right to you.
  - Second textbook is more about systems-in-the-large, less about implementation details
    - 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 Tuesdays and Thursdays by 8:30am, which is our traditional lecture time.
  - 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).
CS110 Class Resources

- CS110A: Problem Solving Lab for CS110
  - CS110A is a one-unit supplementary discussion section designed to foster additional community and establish a bedrock foundation in systems programming.
    - CS110A is new this quarter, and is based on one of the School of Engineering's Equity and Inclusion Initiatives.
    - CS110A meets twice a week, Tuesdays and Thursdays, 2:30pm (and the Thursday meeting satisfies the general CS110 discussion section requirement discussed later).
  - We limit enrollment--that limit is currently 28--so that CS110A students can more readily interact with a supportive and empathetic cohort and more easily access the time and attention of the course staff (particularly me and the CS110A CA, Ryan Eberhardt).
  - We're particularly eager to accept students who come from educationally disadvantaged backgrounds and students who question their ability to do well in CS110 unless they have this added support.
  - Students interested in taking CS110A must submit an application by September 18th, and CS110A formally begins on Tuesday, September 25th. You’ll know by Sunday, September 20th whether you’ve been accepted and offered a spot under the CS110A wing.
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 (8 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
  - Ports, sockets, socket descriptors, `socket`, `connect`, `bind`, `accept`, `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

- Additional Topics
  - 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
  - 85% of final grade, expect six assignments
  - Some assignments are a 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 80%
      - If you need to submit more than 48 hours late, you can, but you need to message me saying you’re submitting that late. Your overall score will also be capped at a 50%.
    - Assignment 1, which goes out today, must be turned in on time.
Expectations

- Discussion Sections
  - Everyone is expected to sign up for a weekly, 50-minute discussion section
  - 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, September 21st.
  - As opposed to last spring, we will not require discussion section attendance this quarter, though you are very much encouraged to attend.
    - 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.
    - We’re hoping to limit the size of discussion sections to be just 14 or so students.
    - We provide discussion section handouts every week, but those attending ultimately decide what gets covered.
Expectations

● Self-Assessments
  ■ There are no traditional exams at all this quarter
    □ The university cancelled traditional finals for the entire academic year, citing a
      distaste for anything high-stakes that might cause undue stress for those who are
      positionally and situationally disadvantaged by the state of the world right now.
    □ My takeaway from the cancellation: The traditional programming exam is on leave
      until further notice.
  ■ My compromise: low-stakes self-assessments as opportunities to take a pulse on how
    well you understand the accumulation of topics taught so far.
    □ 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 should be able to be completed in 45 minutes by those who are fully
      on top of the material.
    □ I will, however, give you a 72 hour stretch of time—during which no assignments will
      be assigned—to work through the questions and submit answers.
    □ Exams will be graded using a square-rooting scheme as described in Handout 01.
  ■ 15% of final grade: 5%, 5%, and %5
Honor Code

- Please take it seriously, because the CS Department does
  - 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 private hosting
    - Provide detailed citations of all sources and collaboration
  - Self-assessments are open book, open notes, open WWW, though you may not actively collaborate with others in person or online. In general, you can only rely on materials that are, in general, discoverable by all students via search engines.