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 in 1993
  - Lecturer in CS, teaching CS106J, CS106B, CS106X, CS107, CS110
  - Taught CS110 for the first time in Spring, 2013
    - Leveraged much of Mendel Rosemblum'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, have worked on the platform team the entire time
  - Learned web programming, PHP, CSS, JavaScript. Old CS107 student (class of 2004) is still my manager #karma
  - 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**
  - 232 enrolled students as of Saturday at 8:15pm
  - Each of you should know C and C++ reasonably well so that you can...
    - write moderately complex programs
    - read and understand portions of medium and even large code bases
    - calmly 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.
  - 10 CA's at the moment
    - Mahesh, Sona, Chase, Kushaagra, David, Audrey, Raunak, Michael I, Michael II, Pamela
    - If the student enrollment count surpasses 250, we'll get more staff

CS110 Class Resources I

- **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, lecture summaries, and links to lecture slides like the one you're working through right now
- **Online Student Support**
  - Peer-collaborative forum: Slack
  - My email: jerry@cs.stanford.edu
- **Office Hours**
  - My office hours are TBD, and they'll normally be held in my Gates 192 office
    - Walk-ins are almost always fine, and I'm happy to help if I'm in the office
    - MWF **before** lecture is not good. I'm a just-in-time kinda guy and I'm almost certainly prepping for class
  - 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
    - Reader drawn from either the 2nd or 3rd edition of the book is fine.
    - Examples in book are in C, though we'll migrate to C++. (Good to know pros and cons of both)
  - Second textbook is more about systems-in-the-large, less about implementation details
    - "Principles of Computer System Design: An Introduction", by by Jerome H. Saltzer and M. Frans Kaashoek
    - Provided free-of-charge online, chapter by chapter
    - Not stocked at Stanford Bookstore by design, since free is better than $60. You can buy a copy of it from Amazon if you want one
- **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 prove 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 mercurial repository at **`/usr/class/cs110/lecture-examples/spring-2017/`**, which you can initially **`hg clone`**, and then subsequently **`hg pull && hg update`** to get the newer and updated examples as I check them in

CS110 Class Resources

- **Lecture Slides**
  - Will rely on slides when I 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 I'll inject updates and clarifications (and be clear that I added stuff when it really impacts you)
  - They are not a substitute for attending lecture
    - I go off code 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
  - (Thanks go out to David Mazières, whose excellent CS240H slides inspired my decision to use markdown and pandoc, and further inspired my decision to, with attribution, steal his stylesheets)

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`**, etc.
  - 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
  - Additional systems examples that rely on naming, abstraction, modularity, and layering, including databases, DNS, TCP/IP, network packets, HTTP, REST, descriptors and pids
  - Building modular systems with simultaneous goals of simplicity of implementation, fault tolerance, and flexibility of interactions
- ## Exceptional Control Flow
  - Introduction to multiprocessing, **`fork`**, **`waitpid`**, **`execvp`**, process ids, interprocess communication, context switches, user versus supervisor mode, system calls and how their calling convention differs from those of normal functions
  - Protected address spaces, virtual memory, main memory as cache, virtual to physical address mapping, scheduling
  - Concurrency versus parallelism, multiple cores versus multiple processors, concurrency issues with multiprocessing, signal masks

- Interrupts, faults, systems calls, signals, design and implementation of a simple shell
- Virtualization as a general systems principle, with a discussion of processes, RAID, load balancers, AFS servers and clients

Course Syllabus (continued)

- **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, eight per **rye** machine, 12 per **corn** machine, 24 per **barley** 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, rendezvous and thread communication, **unique_lock**, **wait**, **notify_one**, **notify_all**, deadlock, thread starvation
  - Semaphore concept and **class 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
- **Networking and Distributed Systems**
  - Client-server model, peer-to-peer model, telnet, protocol as contract for clear communication between programs, 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`**, **`write`**, simple echo server, time server, concurrency issues, spawning threads to isolate and manage single conversation
- C++ layer over raw C I/O file descriptors, pros and cons, introduction to **`sockbuf`** and **`sockstream`** C++ classes (via **`socket++`** open source project)
- HTTP 1.0 and 1.1, header fields, GET, HEAD, POST, complete versus chunked payloads, response codes, caching
- IMAP protocol, custom protocols, Dropbox and iCloud reliance on HTTP
- 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_*`** set of functions, **`libev`** and **`libuv`** open source libraries all provide nonblocking I/O alternatives to ensure maximum use of CPU time using a minimum number of threads (ideally 1) and processes (ideally 1)

Student Expectations

- **Programming Assignments**
  - 40% of final grade
  - Expect seven or eight assignments (depends on how much as-time-permits material I get to)
  - 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
  - Lateness policy is different than it is for many other CS classes
  - There are no free late days
  - 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
  - Exceptions: 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 that requires you leave the area, illness that requires medical intervention, and so forth)

Student Expectations

- ## Discussion Sections
  - 5% of final grade.
  - Discussion sections will be held at various times on Wednesdays.
  - I'm introducing the CS110 discussion section for the first time ever this quarter.
  - 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.
    - We'll lecture this coming Wednesday from 1:30pm until 2:50pm.
    - Discussion sections will fire up in Week 2, at which point we'll no longer hold Wednesday lecture.

Student Expectations

- **Midterm**
  - Midterm is Friday, May 12th during normal class time.
    - 20% of final grade, material drawn from first five 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 (and only if) you're taking another class at the same time will I allow you to take the midterm some time earlier that same day. Email me directly if you need to take the exam before everyone else does.

Student Expectations

- ## Final Exam

  - Three-hour final is Monday, June 12th at 3:30pm
    - 35% of final grade, cumulative, mix of implementation and short answer questions
    - 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 Monday, June 12th at 7:00pm.
    - But only to those who are leveraging the SCPD system and taking another class whose final exam competes with mine.
    - Email me directly if you need to take the final exam during the alternate time slot because of a competing final.
  - You must take the final exam at one of these two 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 or bitbucket) so that others can easily discover it via word of mouth or search engines
    - If you'd like to upload your code to a private repository, you can do so on bitbucket or some other hosting service that provides free-of-charge private hosting

Reading

- **Skim Chapter 2, Sections 1 - 4**
- **Be prepared to consult Chapter 2, Section 5 when Assignment 1 goes live, as it provides the theory that backs what you'll be implementing**
- **Live off campus? Read this so you still can get access to the free version of the online textbook**