CS110 Final Practice Final Exams

Final Exam

- Monday, June 12th, 2017
- 3:30 p.m. - 6:30 p.m.
- Locations:
  - Students with last names beginning with A through G: Gates B01
  - Students with last names beginning with H through Z: Nvidia Auditorium

The final exam is closed book, closed note, closed electronic device. I will provide you with all of the prototypes of C and C++ functions/methods that might be relevant to a particular problem, and you can always come to me or a CS110 CA during the exam if you want the interface for some C function or C++ class. (**Caveat**: You are permitted to populate both sides of **two** 8.5" x 11" sheets of paper with as much material as you can jam into it).

The exam will focus on multithreading and networking, although all material taught in lecture is fair game (although you won't be expected to write any code using nonblocking I/O).

Material

Here's the impressive list of topics you should be familiar with:

- You should understand how `open`, `read`, `write`, `close`, `stat`, and `lstat` all work.
- You should be familiar with the basic concepts of layering and naming in computer systems.
- You should understand the UNIX v6 file system concepts, data structures, and layers you coded against for Assignment 1.
- You should be familiar with `fork`, `waitpid`, all of the various status macros, `execvp`, `signal`, signal handlers, signal blocking and unblocking, `kill`, process ids, process groups, and `pipe`.
- You should be familiar with the various **inter**-process concurrency issues that can come up as a result of a single code base controlling multiple processes.
- You should have a basic understanding of how virtual-to-physical memory mapping works and how it's used to allow all processes to operate under the illusion that each owns its full virtual address space.
- You should understand the basics of `pthread_create` and `pthreads_join` as illustrated in all of the examples in `/usr/class/cs110/lecture-examples/spring-2017/threads-c`.
- You should understand what race conditions and critical regions are and how to identify them.

- You should understand the C++11 `thread` class, how to instantiate one, and how to use its `join` method.
- You should be familiar with the `mutex`, `condition_variable_any`, and `lock_guard` classes.
- You should understand the notion of a semaphore and how we implement it in terms of `mutex`es, `condition_variable_any`s, and `lock_guard`s.
- You should understand how to effectively use `mutex`es and `semaphore`s to protect against race conditions, protect against senseless busy waiting, protect against deadlock, and minimize the amount of processor time wasted by a `thread` unable to do any meaningful work.
- You should understand all of the various synchronization patterns discussed in Handout 07 and all of the examples that appear in `/usr/class/cs110/lecture-examples/spring-2017/threads-cpp`.
- You should understand the client-server model as discussed in lecture.
- You should be familiar with the `sock_addr` family of socket address records (which will be provided where needed on the exam).
- You should know all of the socket API calls we've used during our study of networking: `socket`, `bind`, `connect`, `listen`, `accept`, and what each of them does.
- You should be familiar with all of the examples presented in lecture and in `/usr/class/cs110/lecture-examples/spring-2017/networking`.
- You should be familiar with the basic structure of an HTTP request—the request line and the header—and how they were manipulated by your `proxy` solution.
- You should be able to address questions pertaining to the material presented during my Principles of System Design lecture (click here for the slides).
- You should be familiar with the MapReduce programming model and your own `mr`/`mrw` implementation of it.
- You should have a surface understanding of nonblocking I/O and be able to speak of its benefits and shortcomings when compared to multithreading and multiprocessing approaches to processes I/O.
- You should understand all of the material in the lab handouts (although I will not test you only anything regarding tools —i.e. `g++-5`, `gdb`, `valgrind`, `info`, the `/sys/proc` pseudo filesystem, etc. will not come up). Some of the problems from the practice exams below have been cannibalized to contribute to your lab handouts. But that's fine, because I should be able to ask you any of those questions again.

Practice Exams

I'm including links to the three CS110 final exams I gave in 2014, and I'm also including links to the companion final exam solutions, which include my own answers and our grading criteria. The course CA's and I will hold office hours during finals week (though the times may change, so check the schedule) so you can come by with any questions. **Understand, however, that I'm under no obligation to imitate the structure of these exams when writing yours.**

- Practice Final Exam 1 [Solution]
- Practice Final Exam 2 [Solution]
- Practice Final Exam 3 [Solution]

Good luck studying for final exams, everyone, and enjoy the summer break! Come back and CA for me next year, please! :)