CS110 Course Information

Instructors: Chris Gregg and Nick Troccoli
Email: cgregg@stanford.edu / troccoli@stanford.edu
Office hours (Chris): Check course website
Office hours (Nick): Check course website

Lectures: Mondays and Wednesdays from 1pm – 2:20pm, on Zoom (see Canvas for Zoom link). **During the *first week only*, we will hold class on Friday from 1pm-2:20pm.**

Units: 5 units. Only matriculated graduate students may register for fewer than five. The requirements are the same for all students, including those who take the course for 3 or 4 units. *Anyone registered with Stanford as an undergraduate **must register** for the **full 5 units**, or you will not be assigned a grade in the course.*

Course Assistants: Nick Comly, Patrick DeMichele, Ella Hofmann-Coyle, Raejoon Jung, Thea Rossman, Semir Shafi, Arjun Sawhney

CS110 CAs attend lectures, lead discussion sections, hold office hours, evaluate homework submissions, monitor the online forums, and grade assessments. All of them have either completed the CS110 material before, CA'ed CS110 before, or both. They know the material so well they already know what your questions are going to be.

CS 110A: CS110A is part of ACE (Additional Calculus for Engineers), a supplementary instruction program that includes weekly sections, office hours, and ACE-specific review sessions. CS110A is done in addition to all the normal requirements for CS110. CS110A is application-only; please see the course homepage for more information.

CS110A CA: Ryan Eberhardt

Prerequisites: Formally, the prerequisite for the course is CS107. Informally, you need to be familiar with the C and C++ programming languages, Unix/Linux, **make**, **Makefile**s, **gcc**/**g++**, **valgrind**, **gdb**, and have some experience with basic computer architecture (x86 as it's taught in CS107, or exposure to some other architecture with the confidence and ability to pick up x86 as we reference it).

We'll be coding in a mixture of C and C++ throughout the quarter. We rely on C, because the libraries needed to interface with system resources are written in C. We rely on C++, because the projects become large enough that it's useful to use a language that supports encapsulation and generic programming better than C does. You should understand pointers, dynamic memory allocation (**malloc**/**realloc**/**free**), and C strings well enough that you're not intimidated by them. You should understand C++ classes, methods, references, templates, and C++'s **new** and **delete** operators. There are C++ features you're not expected to know, but you should have enough programming maturity to pick those features up and search the web for reference materials as needed.

The first assignment, which goes out this week, is a systems programming assignment that should bring all relevant CS107 and software development skills back into rotation. If you haven't taken CS107 and/or programmed in C and C++ before, but you're able to work through this first assignment without drama, then you're more than qualified to take CS110.

Readings:

- The first required textbook is Computer Systems: A Programmer's Perspective by Bryant and O'Hallaron, either the 2nd or 3rd edition. Both CS107 and CS110 teach from a subset of the B&O textbook, so a custom reader—with just the chapters we need—is available at the Stanford Bookstore. Of course, if you want to purchase the entire textbook, you're free to do that as well, though you'll need to buy from Amazon or some other online retailer.
- The second required textbook is Principles of Computer System Design: An Introduction by Jerome H. Saltzer and M. Frans Kaashoek. Stanford has university-wide digital access to the textbook, so you are welcome to access it online. Again, you're welcome to purchase a hard copy if you'd like. It's available for purchase on Amazon.

Website: **http://cs110.stanford.edu** is your new favorite website. There you'll find all reading assignments, lecture slides, homework assignment specifications, and the full list of office hours. If you have any suggestions on how to make the course website even more useful, then drop Chris or Nick an email and we'll talk.

Software: The shared UNIX workstations (**myth** machines in Gates B08, available via **ssh**) provide all of the development tools needed for lecture examples and assignments (although we may occasionally reference the more powerful **rice**, **wheat**, and **oat** machines to clarify the impact that more processors and larger caches have on execution).

It's true that we live in a laptop world, and we suspect you'd like to code on your own machines and eventually port everything over to the **myth**s (where you submit your work and we grade your assignments). However, we strongly urge you to code, test, and debug directly on the **myth**s via **ssh**. The **myth**s are outfitted with a hip version of **g++** that supports the advanced C++ features we'll be relying on almost immediately, and it's better to incrementally develop there instead of porting everything over ten minutes before a deadline.

Student Forums: We're using Ed Stem for the class forum. If you have a question that might be of interest to other students, please post there for a speedy response. Note, however, that you should **never include snippets of code directly from your own homework submissions**, since that's code sharing and a huge no-no.

Canvas: All enrolled students should have access to CS110's page on Canvas (https://canvas.stanford.edu). Canvas will be used to access course lecture videos, Zoom links and lecture check-in quizzes.

Grading: You can take the course for a letter grade, or **CR/NC.** The course grading is divided between several programming assignments, discussion section participation, short lecture check-in quizzes, and 3 short assessments. The grade breakdown is:

- Programming Assignments: 65%
- Discussion Section Participation: 10%
- Assessments: 15%
- Lecture Quizzes: 10%

One very large caveat: regardless of what your overall average is, you need to score 50% or higher on functionality on **every single assignment** in order to receive a passing grade—that is, either a C- or a CR. Restated, if you get lower than a 50% on any single assignment, then that assignment counts for 100% of your grade. This policy is in place to ensure you meet the post-conditions of the course by establishing fluency in all subjects. If you encounter any concerns about this policy, please reach out to us!

If you're taking the course **CR/NC**, your final grade is computed precisely the same way as it is for those taking a letter grade, and you need a C- or better to get the **CR**.

Discussion Sections: In addition to the weekly lecture videos, each and every one of you needs to sign up for a weekly discussion section and participate to demonstrate minimal fluency with the week's material. Each 50-minute section will have at most 20-25 or so students, as we're relying on them to establish community in a world that's otherwise demonstrably isolating. Everyone is expected to choose a discussion section they can tune in live via Zoom, as we won't be recording them for general consumption.

Sections will be a combination of written problems, coding exercises, and software engineering tips to ensure that you understand the material and are outfitted to complete the assignments with minimal drama. We recognize that enrolled students are living everywhere on earth, so we'll offer sections at various different times on Thursdays and Fridays. Discussion sections will begin in Week 2.

Discussion section signups will go live on **Thursday, January 14th,** and are *not* first-come first-serve. Instead, you can submit (and resubmit) your preferences for times anytime between **1/14 and Sun. 1/17**. Once that deadline passes, we'll place people in sections as best we can based on your preferences.

Late Policy: We understand everyone here is busy, and we want to provide flexibility as best we can, while also acknowledging that falling behind on assignments can lead to more problems, and it interferes with our ability to review them and turn around grades in a timely manner.

All programming assignments are due at the stroke of midnight. If you need to submit an assignment after the deadline, you still can. But doing so places a cap on the maximum number of points you can get, depending on how late you submit.

- If you submit an assignment before the deadline, then you can potentially get 100% of the points. Seems right.
- If you submit an assignment after the deadline, but within 24 hours, you can get at most 90% of the points. This doesn't mean we impose a 10% penalty regardless of your final score. It means that all scores between 91% and 100% are demoted to 90%, but all other scores are left alone. If your assignment is severely broken at the time you would normally need to submit, then you have a good reason to take an additional 24 hours to increase your score, as it can only go up. If your program is pretty much working with no obvious flaws, then you probably should submit it by the published deadline.
- If you submit an assignment between 24 hours and 48 hours after the deadline, you can get at most 60% of the points.
- Submissions won't go through more than 48 hours after the deadline.
- **The first assignment must be turned in by the published deadline, without exception.** We want to grade your first assignment as quickly as possible so you can get feedback well before your second assignment falls due.

Note that requests for extensions will be denied unless something extenuating—certainly anything stemming from the pandemic, but also any other family emergencies or other serious illnesses—presents itself, in which case you can send Nick and Chris an email and we'll do what we can to make your life easier while being fair to everyone else. Please don't hesitate to reach out to us!

Assignment Grading: Each assignment has the same weight towards your final grade. Assignments are graded on both functionality *and* style, with functionality worth ⅚ and style worth ⅙ of the grade. You will see a numerical functionality score when you look at your assignment grades, which are available via the course website (direct link). The style portion of your grade will show as a set of English descriptions (**0, multiple-major-issues, major-issue, minor-issues, great**) for important aspects of each assignment. For example, on your second assignment, one of the style blocks might be (don't worry if the details don't make sense yet):

> **lift properly sized payload out of identified block (clean references to inode functions, simple math for identifying how many bytes of final block should count)**

Here are descriptions for each bucket:

- **great** - An outstanding job; reflects code that is notably clean, elegant and readable, with no issues present.
- **minor-issues** - A good job; reflects code that demonstrates solid effort and is fairly successful at meeting expectations, but also has opportunities for improvement.
- **major-issue** - Has more problems, but shows some effort and understanding. There was either a large concern, or several smaller concerns, in the submission.
- **multiple-major-issues** - Has significant issues, either several large issues or a multitude of smaller ones, that together constitute very poor style work.
- **0** - No work submitted, or barely any changes from the starter assignment.

We purposely make the grading for style a bit fuzzy (i.e., we don't give a numerical grade), as it is not something that we want students to get too concerned about. If you do have a question about either the functionality grade or the style grade, please reach out to your grader, to Chris or Nick, or stop by office hours and we can take a look.

Self-Assessment Quizzes: We are removing high-stakes assessments from the course, and instead administering three low-stakes, open-book self-assessments the weekends after Weeks 3, 7 and 8. The quizzes will be written so that they could be completed in a short period of time (much less than any traditional examination) by someone completely on top of the material, and you'll complete it under a timed setting anywhere during the specified exam window over the weekend. Each of the assessments counts for 5% of your overall grade. These self assessments aren't meant to be like traditional exams - they are meant to take a pulse on how well you've learned the material.

Lecture Check-In Quizzes
The main lecture material is pre-recorded as a collection of shorter videos that are posted on Canvas in advance of that actual lecture day. Each video or set of videos will have an associated "lecture check-in quiz" – a very short understanding check that will permit multiple attempts. You will have until the date of the live lecture session to do these quizzes. The sum of lecture quizzes for each day is weighted equally (in other words, all quizzes combined for lecture 2 are weighted the same as all quizzes combined for lecture 3). Live lecture sessions will be optional (but highly encouraged!) and will be devoted to reviewing concepts further, answering questions, and doing additional exercises.

Video Notice
Lecture sessions will be recorded on video and posted to the course Canvas site only. Lab sessions and Helper Hours sessions will not be recorded. If you have questions, please contact a member of the teaching team.

Students with Documented Disabilities: We are happy to provide accommodations recommended by the OAE to ensure you can be comfortable in the course. Students who may need an academic accommodation based on the impact of a disability must initiate the request with the Office of Accessible Education (OAE). Professional staff will evaluate the request with required documentation, recommend reasonable accommodations, and prepare an Accommodation Letter for faculty. Unless the student has a temporary disability, Accommodation letters are issued for the entire academic year. Students should contact the OAE as soon as possible since timely notice is needed to coordinate accommodations. The OAE is located at 563 Salvatierra Walk (phone: 723-1066, URL: https://oae.stanford.edu).

Honor Code: Although you are encouraged to discuss ideas with others, your programs are to be completed independently and should represent fully original work. Whenever you obtain help you should credit those who helped directly in your program, e.g. in a program comment.

Any assistance that is not given proper citation is considered plagiarism, and a violation of the Stanford Honor Code. To be even more specific, you are not allowed to collaborate on the coding of your programs, nor are you allowed to copy even minute snippets of programs from other students, past or present. The following activities are among the many we consider to be Honor Code violations:

1. Looking at another student's code.
2. Showing another student your code.
3. Discussing assignments in such detail that you duplicate a portion of someone else's code in your own program.
4. Uploading your code to a public repository (e.g. **github.com**) 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 **github** or some other hosting service that provides free-of-charge private hosting.

Unfortunately, the CS department sees more than its fair share of Honor Code violations. Because it's important that all cases of academic dishonesty be identified for the sake of those playing by the rules, we exercise our right to use software tools to compare your submissions against those of all other current and past CS110 students, including any we might find online. While we certainly don't want to create some Big Brother environment, we do need to be clear how far we'll go to make sure the consistently honest feel their honesty is valued.

If the thought of copying code has never crossed your mind, then you needn't worry, because we've never seen a false accusation go beyond a single conversation. But if you're ever tempted to share code—whether it's because you don't understand the material, or because you do but just don't have enough time to get the work done—then you need to remember these paragraphs are here.

Course Expenses

All students should retain receipts for books and other course-related expenses, as these may be qualified educational expenses for tax purposes. If you are an undergraduate receiving financial aid, you may be eligible for additional financial aid for required books and course materials if these expenses exceed the aid amount in your award letter. For more information, review your award letter or visit the Student Budget website (https://financialaid.stanford.edu/undergrad/budget/index.html).

Final Notes. We want you to succeed in the course, and we will do everything in our power to help you do well. The course is *rigorous* and requires a significant time commitment. Plan on starting the assignments early and working diligently on them. Come to office hours, and do the readings. If you have any concerns about your progress, email Chris or Nick as soon as you recognize that you are falling behind. This is even more important if you are a graduating senior who needs CS110 to graduate -- Spring seniors have failed the course numerous times before and we do not make exceptions for students simply because they are going to graduate.

This course has been carefully cultivated by Jerry Cain for many quarters, and the general outline and almost all of the assignments, labs, and course material is credited to him.