CS107, Lecture 1
Welcome to CS107!

readings:

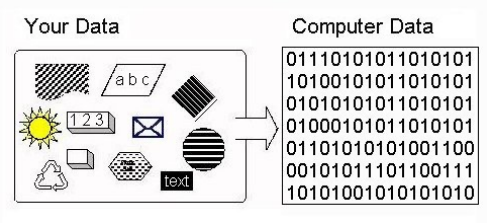
Course Syllabus

Bryant & O'Hallaron, Ch. 1 (skim)

Honor Code and Collaboration Page

CS107 Asks How and Why?

CS106B teaches you to solve problems using high-level programming languages. CS107 digs deeper into **how** and **why** programs truly work.



- How is data (e.g., an **`int`**, a string, a **`struct`**) represented on hardware?
- How does a computer execute code?
- How does an executable map onto memory and hardware?
- How does the heap work—and how is it implemented?
- Why is my code doing one thing when I expect it to do something else?

Studying programming at this level of detail reveals why systems work the way they do and helps us become better software engineers.

CS107 and Programming Experience

For the duration of the quarter, CS107 will:

- **further develop your programming ability** and give you some more coding mileage
- focus on **debugging strategies** that get to the crux of **why** code isn't working as expected
- emphasize how to **become a better debugger**, write **professional-grade code**, and refine your **software development skills**

CS107 Learning Goals

What do we hope you get out of the course? We'd love for students to:

attain **fluency** with

- pointers and memory, and how to use them effectively in C
- an executable's address space and runtime behavior

develop **competency** with

- the translation between C code and assembly
- implementing programs that respect the limits of computer arithmetic
- the ability to identify bottlenecks and improve runtime performance
- the ability to navigate your own Unix development environment
- ethical frameworks to consider when designing and implementing software

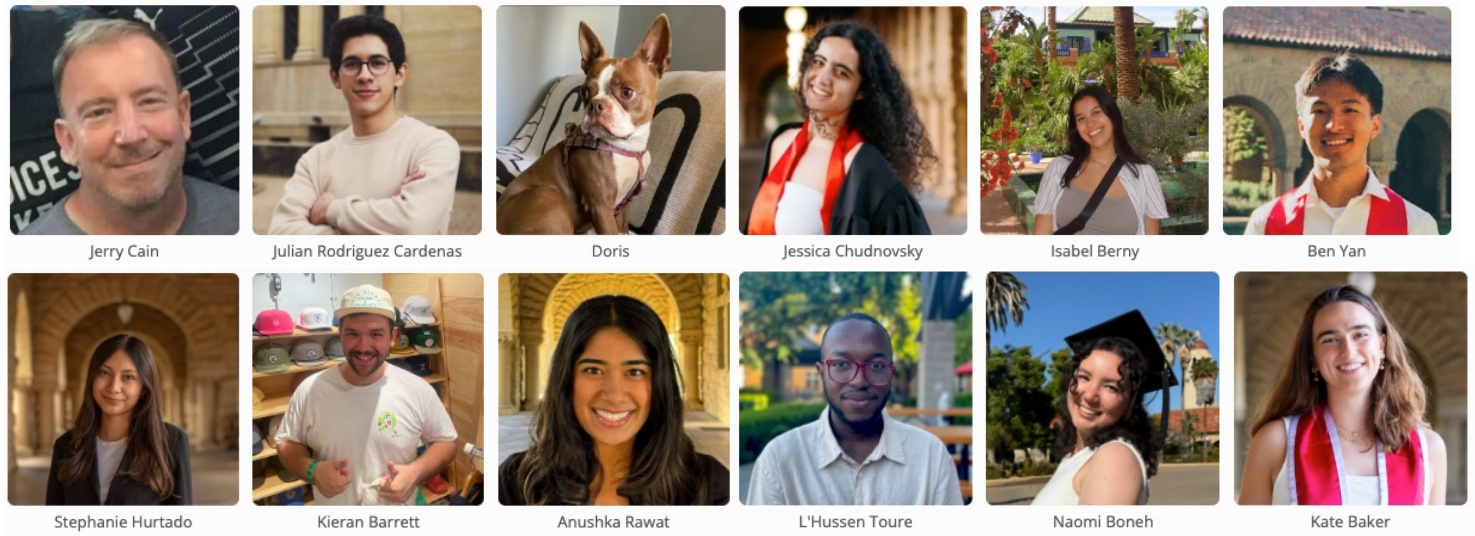
gain **exposure** to

- the basics of computer architecture
- compilers and assemblers and how they work

CS107's Topics and Big Questions

1. **Bits and Bytes**: How can a computer represent **`int`** values? or **`float`**s?
2. **Characters and C Strings**: How can a computer represent and manipulate more complex data, like text?
3. **Pointers, Stack and Heap Memory**: How can we effectively manage all forms of memory in our programs?
4. **Generics**: How can we leverage our understanding of memory and data representation to write code that works with **any** data type?
5. **Assembly**: How does a computer compile and execute C programs to assembly code? What does assembly code look like?
6. **Heap Allocators**: How do core memory allocation functions like **`malloc`** and **`free`** work? Are the built-in versions always good enough?

CS107 Teaching Team



Jerry Cain (**`jerry@cs.stanford.edu`**):

- Chemistry undergrad MIT, began a chemistry Ph.D. here, switched to CS in 1994
- Taught CS107 for the first time in 1999, have taught it nearly 30 times since

Companion Class: CS107ACE

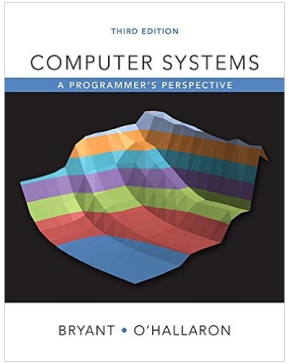
CS107ACE is an extra, single-unit discussion section providing additional support, practice, and instruction.

- Class meets on Tuesdays & Thursdays from 10:30 – 11:20am in Lathrop 017.
- Admission is by application, and everyone is welcome to apply. In past quarters, most students who've applied have been accepted.

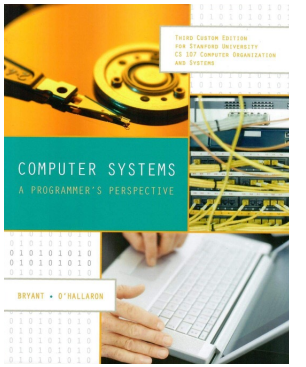
Isabel Berny

Textbooks

- *Computer Systems: A Programmer's Perspective* by Bryant & O'Hallaron, **3rd Edition**
  - **3rd edition matters** – important updates to content
  - Stanford Library has scanned **all** readings for CS107 under "fair use" (private study, scholarship, research). [**Canvas -> Files**].
  - If you want more, you can purchase a full copy.
- Any C programming reference of your choice
  - *The C Programming Language* by Kernighan and Ritchie (free link on course website Resources page)
  - Other C programming books, websites, or reference sheets



Full textbook



CS107 full chapters

canvas CS107-specific readings

The textbook and our C programming references are solid resources, and especially helpful post-midterm!

Course Structure

- **Lectures**: understand concepts, see demos
- **Labs**: learn tools, study code, discuss with peers  **Great preamble to homework!**
- **Assignments**: build programming skills, synthesize lecture/lab content

	Monday	Wednesday	Friday
Week N		Lecture: part A	Lecture: part B
Week N + 1	Lecture: part C		

CS107 labs will meet at various times on Thursdays, and you'll use that time to more fully digest and exercise the prior week's material

- **`assign0`**: goes out Wednesday and falls due on Wednesday, January 14th (requires today's, Wednesday's, and some of Friday's material)

CS107 Grading

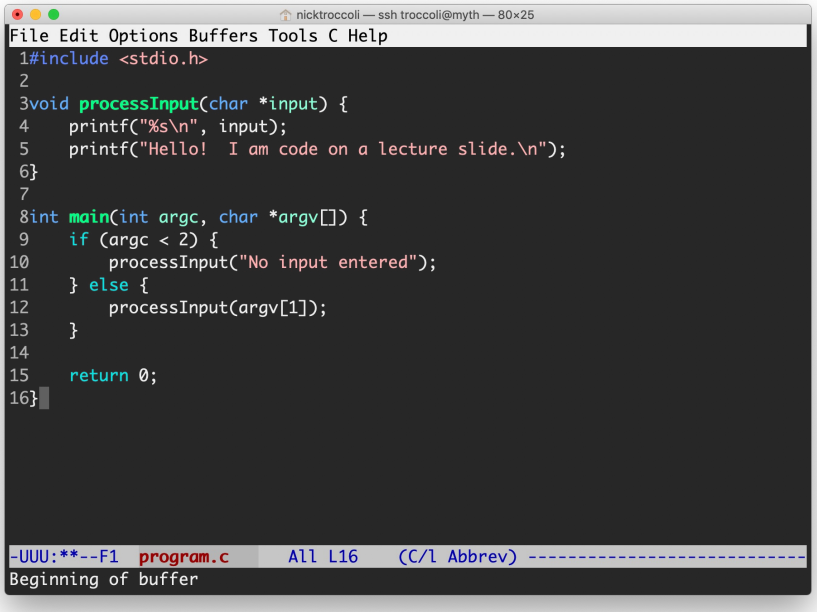
********	40%	Assignments
**	10%	Lab Participation
****	20%	Midterm Exam
******	30%	Final Exam

CS107 Grading: Assignments

********	40%	Assignments
**	10%	Lab Participation
****	20%	Midterm Exam
******	30%	Final Exam

CS107 Assignments

- 7 programming assignments completed individually using **Unix command line tools**
  - Free software, already installed on **`myth`** machines / available on course website
  - We supply starter projects for each assignment
- Graded on **functionality** and **style**
  - Functionality graded using **automated tools**, given as point score, with minimal CA review
  - Style graded via code review (with **occasional automated tests**) given as bucket score
  - Grades posted to website and shared via email



CS107 Style Bucket System

+	An outstanding job: rarely given, particularly on early assignments
ok	A solid job: strong overall, with opportunity for improvement.
–	Conveys some effort and understanding but has larger problems that would need to be addressed before submitting to a professional repo.
––	Suffers from many issues and represents minimally passing work.
0	No work submitted or barely changes any of the supplied starter code.

CS107 Late Assignment Policy

All programming assignments are due a minute before midnight, Stanford time, on whatever day they fall due.

- If you need to submit an assignment after the deadline, you still can. Doing so, however, imposes a cap on the maximum number of points you can get, depending on how late the submission.
  - Submit an assignment before the published deadline? You can get **100%**. Seems right.
  - Submit after the deadline, but within 24 hours? Functionality score is capped at 95%.
    - What does **capped** mean here? It means that all scores between 96% and 100% are demoted to 95%, but all other scores are left alone.
  - Submit an assignment between 24 and 48 hours post-deadline? Score capped at 90%.
  - Submit an assignment between 48 and 72 hours post-deadline? Score capped at 85%.
  - Unless you've made prior arrangements with us, no work is accepted after 72 hours.

CS107 Assignment Resubmission Policy

For all assignments except `assign6`, I will allow anyone who pulled less than 85% of the functionality points to resubmit to get to up to 85%.

You must, however, have received at least 25% of the functionality points in the first place. Otherwise, the maximum you can get is three times your original score.

- We only rerun the automated tests.
  - We don't review your code again or reread responses to short answer questions.
- Once grade reports for an assignment are shared with the class, you have seven calendar days to resubmit.
- Email Jerry when you'd like to resubmit and I'll let you know how.

CS107 Grading: Labs

********	40%	Assignments
**	10%	Lab Participation
****	20%	Midterm Exam
******	30%	Final Exam

CS107 Labs

- Weekly, in-person labs led by CAs, starting **next week**, offered on Wednesdays and Thursdays.
- Hands-on practice with lecture material and course concepts.
- Graded on attendance + participation. You get points for literally showing up on time, working through some exercises with everyone else, and staying for the duration.
- CGOE students complete lab work (and take exam exams) remotely. More info in CGOE Handout.
- Lab preference submissions are already open, but you take your time finalizing your schedule before submitting. You may submit your preferences anytime until **Sunday 01/11 at 5PM PST**.

CS107 Grading: Exams

********	40%	Assignments
**	10%	Lab Participation
****	20%	Midterm Exam
******	30%	Final Exam

CS107 Exams

- **Midterm exam**: Wednesday, February 11th, 7:00 - 9:00pm outside of class
  Wednesday, February 11th, 3:30 - 5:30pm if first time is bad
  - Email me by Friday, February 6th if you have an academic or extracurricular conflict with **both** times and absolutely cannot make either.
- **Final exam**: Monday, March 16th, 3:30 - 6:30pm
  - If and only if you hold a conflict with this time because of a competing final exam—that is, you're taking two MWF 10:30am classes and your other class also has a final—can you take the final on the same day, on March 16th, from 12:15 – 3:15pm instead.
- Both exams are **closed book, closed notes, and closed electronic device**. You'll be provided with a reference sheet with common function prototypes and other pertinent information not easily memorized.
- Both the midterm and final are administered as pencil-and-paper exams.

CS107 Grading

********	40%	Assignments
**	10%	Lab Participation
****	20%	Midterm Exam
******	30%	Final Exam

Read our full course policies document:

https://cs107.stanford.edu/syllabus.html

Getting Help

- Post on the **Ed Discussion Forum**
  - Online discussion forum for students, post questions, answer other student questions
  - Best for course material discussions, policy questions, debugging questions or general assignment concerns. **Never post assignment code to Ed.**
- Visit **Office Hours** in CoDa Basement
  - Chat about course topics or just hang out
  - Sign up in queues for 1:1 CA help, schedule to soon be posted on course website
  - Best for **group work**, **debugging questions** (with CAs, of course) or **longer discussions about course material**
- **Email** the Course Staff
  - `cs107-win2526-staff@lists.stanford.edu` or individual staff members
  - Best for **private concerns** (e.g., grading questions, academic accommodations, etc.)

Stanford Honor Code

- The **Honor Code** is an undertaking of the Stanford academic community, individually and collectively. Its purpose is to uphold a culture of academic honesty.
  - Students will support this culture of academic honesty by neither giving nor accepting unpermitted academic aid in any work that serves as a component of grading or evaluation, including assignments, examinations, and research.
  - Instructors will support this culture of academic honesty by providing clear guidance, both in their course syllabi and in response to student questions, on what constitutes permitted and unpermitted aid. Instructors will also not take unusual or unreasonable precautions to prevent academic dishonesty.
  - Students and instructors will also cultivate an environment conducive to academic integrity. While instructors alone set academic requirements, the Honor Code is a community undertaking that requires students and instructors to work together to ensure conditions that support academic integrity.

  (drawn from **https://communitystandards.stanford.edu/policies-guidance/honor-code**)

It is your responsibility to read and become familiar with the honor code guidelines on the CS107 course website. Please read them and come speak with us if you have questions.

https://cs107.stanford.edu/collaboration

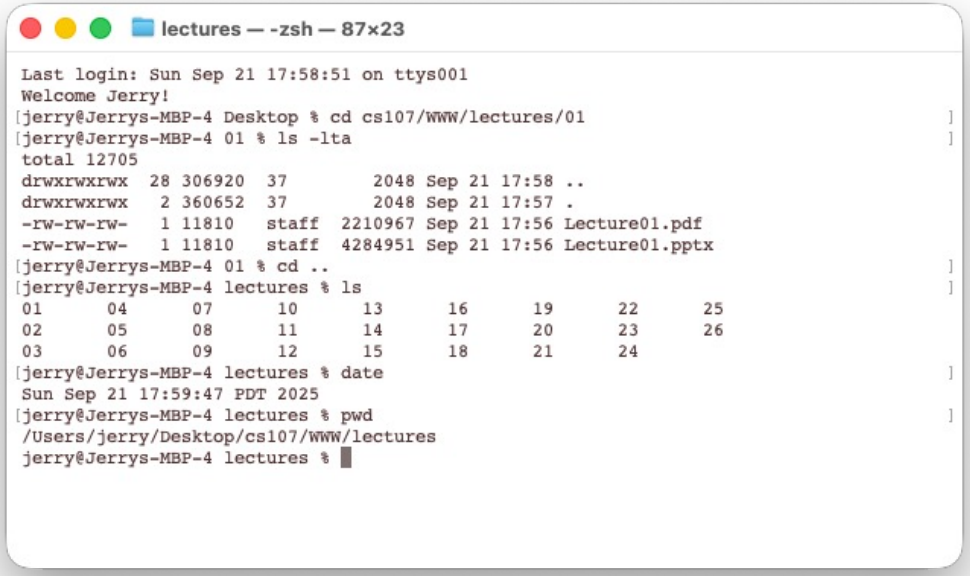
Stanford Honor Code and CS107

- Be fully transparent about how you get your work done
  - Indicate any assistance received on homework (books, friends, etc.).
  - Don’t look at other people's solution code or answers and never share solutions with others or post them to the web.
  - Never use LLMs to generate assignment code.
    - This replaces the intellectual work the assignment is designed to assess and is therefore considered one of the most egregious and indefensible forms of academic dishonesty.
- Assignment submissions are regularly examined using software tools.
- If you need help, contact us and we'll help you make good choices.
  - If you feel you’ve made a mistake, you may retract any assignment submission at any time before the final exam by simply emailing me and telling me.

https://cs107.stanford.edu/collaboration

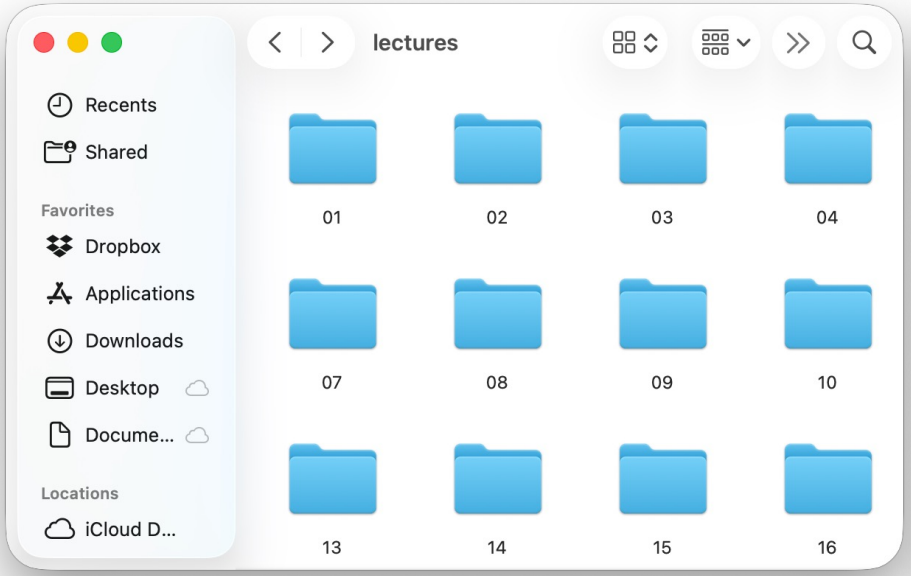
What is Unix?

- **Unix:** provides a set of standards and many tools used to write software.
  - **macOS** and **Linux** are operating systems built on top of Unix
- You navigate a Unix system using the **command line** (aka "the terminal")
- All Unix systems work using the same tools and commands

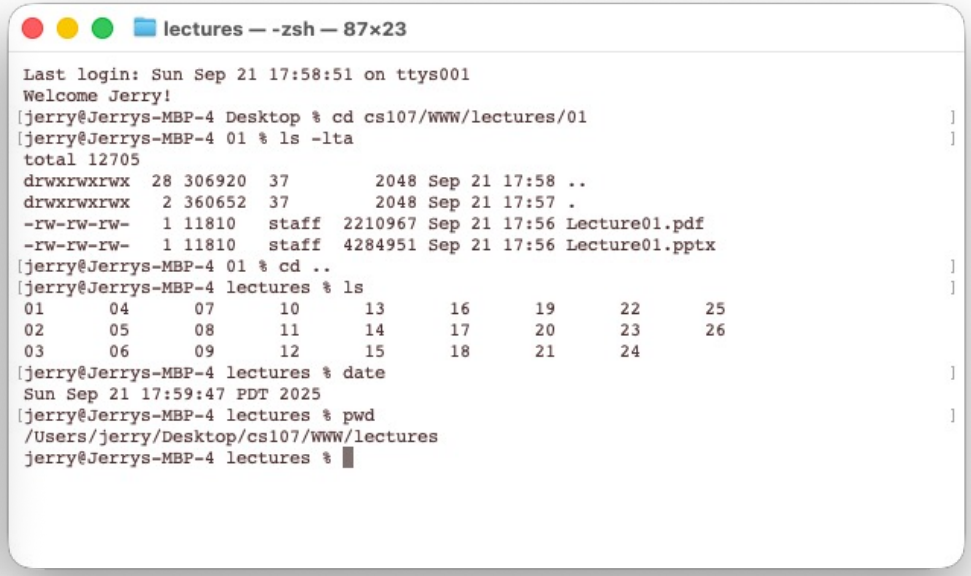


What is the Command Line?

The **command line** is a text-based interface to navigate a computer's file system and launch executables. It's a popular, lightweight alternative to the more traditional graphical user interfaces you've used before.



Graphical User Interface

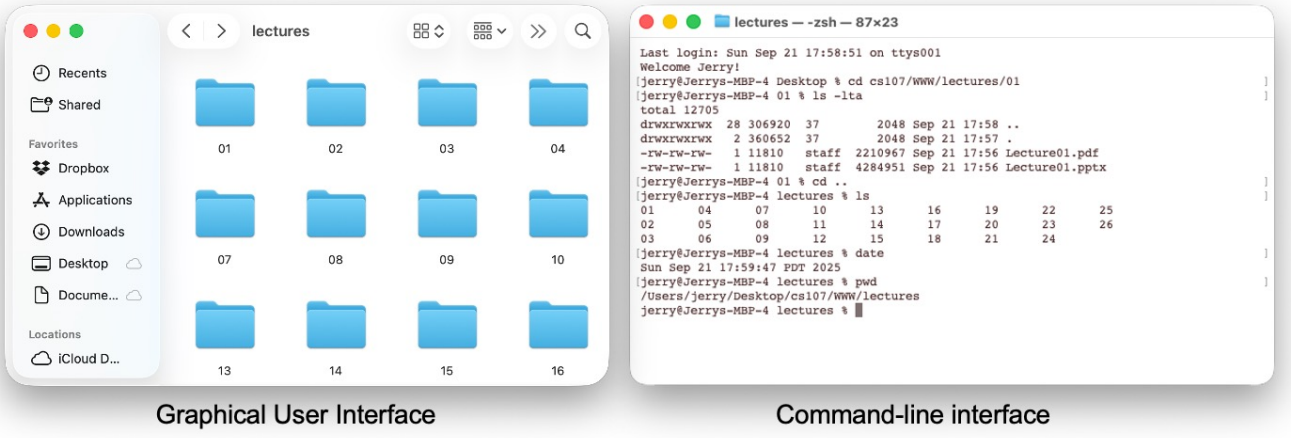


Command line interface

CLI vs. GUI

Just like a GUI file explorer, a terminal:

- shows you a **specific place** on your computer at any given time
- lets you descend **into folders** and **ascend out of them**
- lets you **create new** files and directories and **edit** existing ones
- lets you **execute programs**



Graphical User Interface

Command-line interface

Why Unix?

- You can navigate virtually any device using the same tools and commands:
  - laptops and desktops
  - embedded devices (Raspberry Pi, etc.)
  - mobile Devices (Android, etc.)
  - microwaves, dishwashers, air conditioners, cameras
- Used frequently by software engineers for:
  - **web development**: running servers and web tools on servers
  - **machine learning**: processing data, training models
  - **systems**: implementing operating systems, networks, compilers, and drivers
  - **mobile development**: running tools, managing libraries
- We'll use Unix and the command line to implement and test C programs.

Unix Commands To Try

- **`cd`**: change directories
- **`ls`**: list directory contents
- **`mkdir`**: create a new directory
- **`emacs`**: open text editor
- **`rm`**: irreversibly delete file or folder
- **`man`**: read documentation for standard C functions

See the course website for more commands and a complete reference.

Demo: Using Unix and the Command Line

Get up and running with our guide:

http://cs107.stanford.edu/getting-started.html

Learning Unix and the Command Line

- Using Unix and the command line can be very intimidating at first.
  - How do I know what to type?
  - It looks retro! It **is** retro!

- You're learning a new language.
  - You should repeatedly look things up (**resources** on course website)
  - Spend as much time as possible (during labs and assignments) building muscle memory with the tools

Question Break

Get up and running with our guide:

http://cs107.stanford.edu/getting-started.html

`assign0`

`assign0` (Intro to Unix and C) goes out on Wednesday and falls due a week later, on **Wednesday, January 14th at 11:59PM PDT**.

There are **5** parts to the assignment, which is meant to get you comfortable using the command line, and editing/compiling/running C programs:

- surveying the course website to learn a few different Unix commands
- **clone** the **`assign0`** starter project
- **answer** several questions in **`readme.txt`**
- **compile** a provided C program and **modify** it
- **submit** the assignment

Lecture Recap

CS107 is a programming class in C that teaches you about programming languages and software and what goes on under the hood.

- We'll use Unix and its tools to implement, debug and test our programs.
- Please visit the course website, **https://cs107.stanford.edu**, where you can read the lecture schedule, the syllabus, the information we provide about the Honor Code in CS107, and much, much more.

I'm looking forward to an awesome quarter!