Computer Systems

Web: cs107.stanford.edu

Cynthia Lee
Today’s Topics

1. Introductions
2. What is this class?
3. Course components and policies overview
4. Diving right into the content:
   › First look at UNIX
   › Your first C program
   › Your first command-line compilation

Next lecture (Friday):
   › More whirlwind tour of C and UNIX

First assignment (assign0):
   › Will be posted late tonight, due a week from today
Introductions

**CS107.STANFORD.EDU**

Cynthia Lee

**cs107@cs.stanford.edu** (all staff)
**cbl@cs.stanford.edu** (me)

Piazza!
What is CS107 all about?

Have you ever noticed how “Hacker” work looks in old movies?...
Independence Day (1996)

(Jeff Goldblum’s character uploading a virus to the alien mothership)

(So this guy can go divebomb it)

(TRINITY SAVING THE WORLD BY HACKING INTO THE POWER GRID)
The Girl with the Dragon Tattoo (2011)

(Lisbeth Salander researching murders)
TRON Legacy (2011)
What is CS107 about?

- The CS106 series teaches you how to solve problems as a programmer
- Many times CS106 instructors had to say “just don’t worry about that” or “it probably doesn’t make sense why that happens, but ignore it for now” or “just type this to fix it”

- **CS107 finally takes you behind the scenes**
- **How do things really work in there?**
  - It’s not quite down to hardware or physics/electromagnetism (those will have to stay even further behind the scenes for now!)
  - It’s how things work **inside Java/C++ (we will explore from C)**, and how your programs map onto the components of computer systems

  - *Who doesn’t love the idea of having an exclusive backstage pass?*
  - *Who doesn’t love being in on a secret?*
  - *It just feels good to know!*
What are the learning goals for the course?

- The goals for CS107 are for students to gain **mastery** of
  - writing C programs with complex use of memory and pointers
  - an accurate model of the address space and compile/runtime behavior of C programs
- to achieve **competence** in
  - translating C to/from assembly
  - writing programs that respect the limitations of computer arithmetic
  - identifying bottlenecks and improving runtime performance
  - writing code that correctly ports to other architectures
  - working effectively in UNIX development environment
- and have **exposure** to
  - a working understanding of the basics of computer architecture
How does CS107 work?

Course Information Sheet walkthrough
Textbooks

- You’ll want to get the Bryant and O’Hallaron book
- 3rd edition (for once in a college class, this actually matters)

- The suggested C reference is just one suggestion
  - You could do just as well with a different C book
  - You could do just as well with Google or websites like http://www.cplusplus.com/reference/clibrary/
  - Just need somewhere to turn when you have a question about C
Getting Started in UNIX

Welcome to the World of Green and Black!
TRON Legacy (2011)

https://www.youtube.com/watch?v=yQsaPVfze4s
TRON Legacy (2011) (screenshot #2)

https://www.youtube.com/watch?v=yQsaPVfze4s
TRON Legacy (closeup)
Navigating directories in UNIX (compared to Mac)
Navigating directories in UNIX (compared to Windows)
It’s ok to feel like this right now when it comes to Unix. We’ll continue to work on that while we introduce C. It’s important you spend as much time as you can building muscle memory with the tools.
History and background of C

- Birthdate around 1970
- Created to make writing Unix (the OS itself) and tools for Unix easier
- Part of the C/C++/Java family of languages
  - (with C++ and Java coming later)
- Design principles:
  - Small, simple abstractions of hardware
  - Minimalist aesthetic
  - C is much more concerned with efficiency and minimalism than safety (Java) or convenient high-level services and abstractions (Java, C++)
Comparison of C, Java, C++

- **Some things will be very familiar:**
  - Syntax
  - Basic data types
  - Arithmetic, relational, and logical operators

- **You may be sad about what’s missing:**
  - No power features of C++ (overloading operators, default arguments, pass by reference, classes/objects, fancy ADTs)
  - Thin standard libraries (no graphics, networking, etc)
  - Weak compiler checks, almost no runtime checks

- **Benefits:**
  - Small language footprint (not much to learn)

- **Philosophical difference:**
  - Procedural (C)
  - Procedural + Objects (C++)
  - Object-Oriented (Java)
Things to watch for in our num.c code:

- **First pass:**
  - Structure of C program
    - `#include`
    - `main`
    - `functions`

- Unix commands to compile, run program
  - `Makefile, “make” command, “make clean”`
  - `Data types: int, char, char* strings, arrays`

- Interactive output and input
  - `printf`
  - `scanf`

- CTRL-C to kill something in Unix
Basic anatomy of main()

```c
int main(int argc, char * argv[]) {
   // stuff
   return 0;
}
```

- Return value always int (just return 0 all the time and otherwise ignore it)
- argc is the size of the argv array
- argv array is a collection of the arguments that are typed on the command line in Unix when you run the program (captured as strings)
  - The 0\textsuperscript{th} argument is the name of the command itself
  - Args 1 and on are the command-line arguments