CS107, Lecture 1
Welcome to CS107!

reading:

General Information handout

Bryant & O'Hallaron, Ch. 1

Plan For Today

- Introduction
- CS107 Course Policies
- Unix and the Command Line
- Getting Started With C

Plan For Today

- Introduction
- CS107 Course Policies
- Unix and the Command Line
- Getting Started With C

What is CS107?

Computer Organization and Systems

- How languages like C++ and Java **represent data** under the hood
- How programming structures are encoded in **bits and bytes**
- How to efficiently **manipulate and manage memory**
- How computers **compile** programs
- Uses the **C** programming language
- Programming **style** and software development practices

CS107 Learning Goals

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 a Unix development environment

and have **exposure** to

- a working understanding of the basics of computer architecture

Course Overview

- Representing data with **bits** and **bytes**

- C-strings

- Pointers, memory allocation and management

- Generic C programs

- Floating-point numbers

- Assembly language

- The Heap

You'll be able to...

- Manipulate bits and bytes and work within the limits of computer arithmetic
- Implement complex algorithms using C-strings
- Re-implement existing Unix tools yourself
- Write generic C code that works with a variety of data types
- Defuse a "bomb" program without ever seeing its source code
- Find security vulnerabilities in programs
- Create your own memory allocator to manage heap memory

Companion Class: CS107A

- **CS107A** is an extra 1-unit "Pathfinders" or "ACE" section with additional course support, practice and instruction.

- Meets for an additional weekly section and has additional review sessions

- Entry by application – see the FAQ on the course website for details
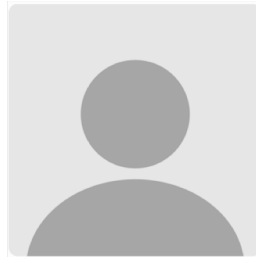

Reynis Vazquez-Guzman

Course Website

cs107.stanford.edu

Nice to meet you!
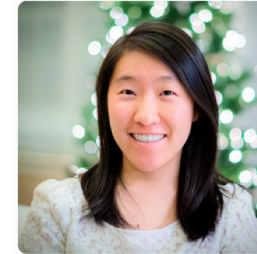
Lecturer: Nick Troccoli

Michal Wegrzynski   Nolan Handali   Petar Penkov   Annie Shi   Cooper Raterink   Emily Ling
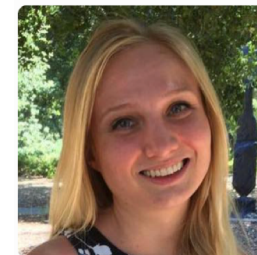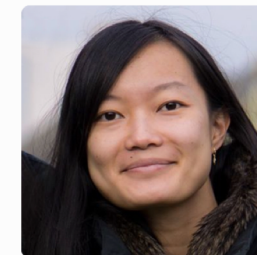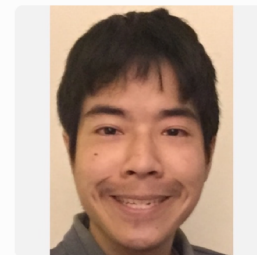
Rifath Rashid   Sushain Cherivirala   Timothy Le   Emily Marx   Jennifer Tao   Max Lam

Course Assistants (CAs)

The course staff will lead labs, grade coursework, help you when you have questions, and much more!

Plan For Today

- Introduction

- CS107 Course Policies

- Unix and the Command Line

- Getting Started With C

Textbooks

- *Computer Systems: A Programmer's Perspective* by Bryant & O'Hallaron, **3rd Edition**
  - **3rd edition matters** – important updates to course materials
- A 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
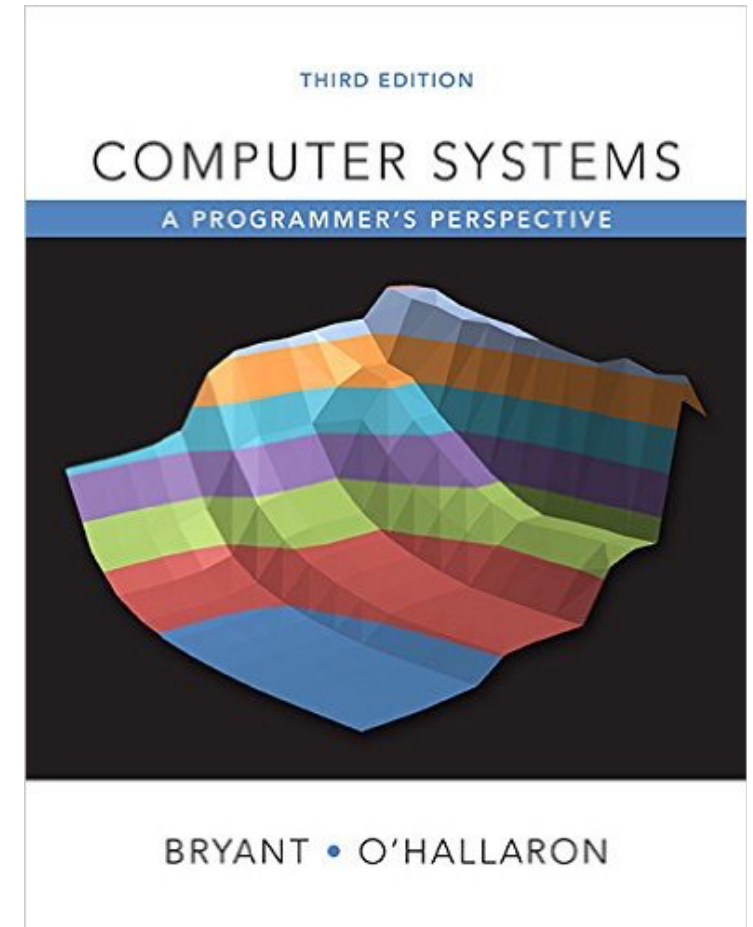
Grading

**** 35% Assignments
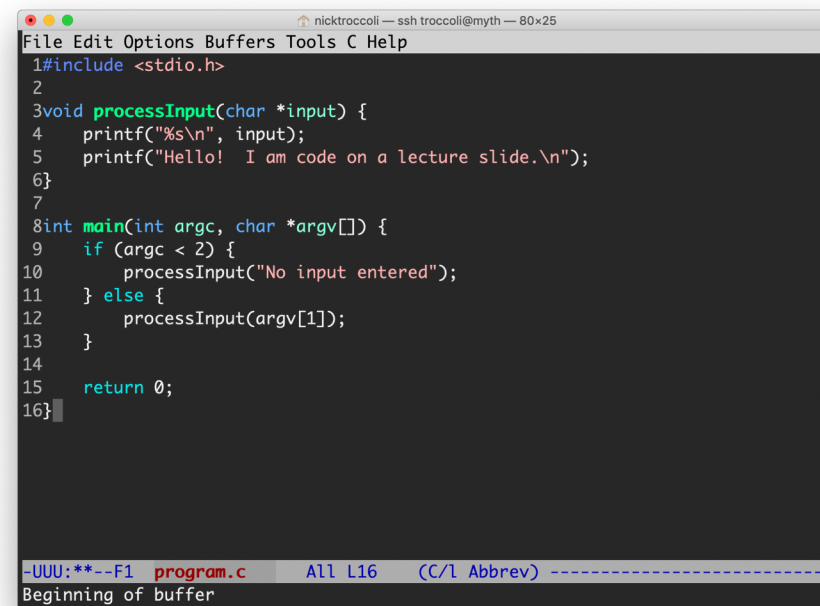
** 15% Lab Participation

** 17% Midterm Exam

**** 33% Final Exam

Grading

****   35%   Assignments

**   15%   Lab Participation

**   17%   Midterm Exam

****   33%   Final Exam

Assignments

- 8 programming assignments completed individually using **Unix command line tools**
  - Free software, pre-installed on Myth machines / available on course website
  - We will give out starter projects for each assignment
- Graded on **functionality** (behavior) and **style** (elegance)
  - Functionality graded using *automated tools,* given as point score
  - Style graded via *automated tests* and TA code review, given as bucket score
  - Grades returned via course website

The Style Bucket System

+	An outstanding job; could be used as course example code for good style.
ok	A good job; solid effort, but also opportunities for improvement.
-	Shows some effort and understanding, but has larger problems that should be focused on.
- -	Little effort; incomplete or mostly non-functional.
0	No work submitted, or barely any changes from the starter assignment.

Getting Help

- Visit Nick and the CAs at **office hours**
  - Scheduled throughout the week; schedule will be posted on course website tomorrow
  - Best for **most coding/debugging questions**, or **longer course material discussions**
- Post on **Piazza**
  - Online discussion forum for students; post questions, answer other students' questions
  - Best for course material discussions, course policy questions or general assignment questions (DON'T POST ASSIGNMENT CODE!)
- **Email** the Course Staff
  - cs107@cs.stanford.edu – please do not email CAs individually
  - Best for **private matters** or **brief** coding/debugging questions.

Late Policy

- Submitting by the assignment deadline typically earns an **extra-credit on-time bonus, usually ~5%**.

- If you miss the deadline, there may be a grace period for late submissions, typically 48 hours, but submitting during the grace period **does not earn any on-time bonus**.

- "Pre-granted grace period" – additional extensions granted only in *very special* circumstances. **Instructor** must approve extensions.

Grading

**** 35% Assignments

** 15% Lab Participation

** 17% Midterm Exam

**** 33% Final Exam

Lab Sections

- Weekly 1 hour 50 minute labs led by a CA, starting *next* week, offered on Tuesdays, Wednesdays and Thursdays.

- Hands-on practice in pairs at computers with lecture material and course concepts.

- Graded on attendance + participation (verified by submitting work at the end)

- Lab signups open **Wednesday 1/9 at 5:55PM**, and **are first-come first-serve**. A link will be posted on the course website.

Grading

****   35%   Assignments

**   15%   Lab Participation

**   17%   Midterm Exam

****   33%   Final Exam

Exams

- **Midterm exam** – Friday, February 15th, 12:30-2:20PM (during *full class period*)
- **Final exam** – Friday, March 22nd, 12:15-3:15PM

- You *MUST* be able to take both exams at the scheduled time, except for university athletics or OAE accommodations.
- Both exams are closed-book, closed-notes, but you may bring in 1 double-sided page of notes for the midterm, and 2 double-sided pages of notes for the final. You will also be provided with a syntax reference sheet.

Grading

****   35%   Assignments

**   15%   Lab Participation

**   17%   Midterm Exam

****   33%   Final Exam

Stanford Honor Code

- The **Honor Code** is an undertaking of the students, individually and collectively:
  - that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
  - that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

- The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

- While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

  see also: http://honorcode.stanford.edu/

It is your responsibility to ensure you have read and are familiar with the honor code guidelines posted on the main page of the CS107 course website. Please read them, and come talk to us if you have any questions or concerns.
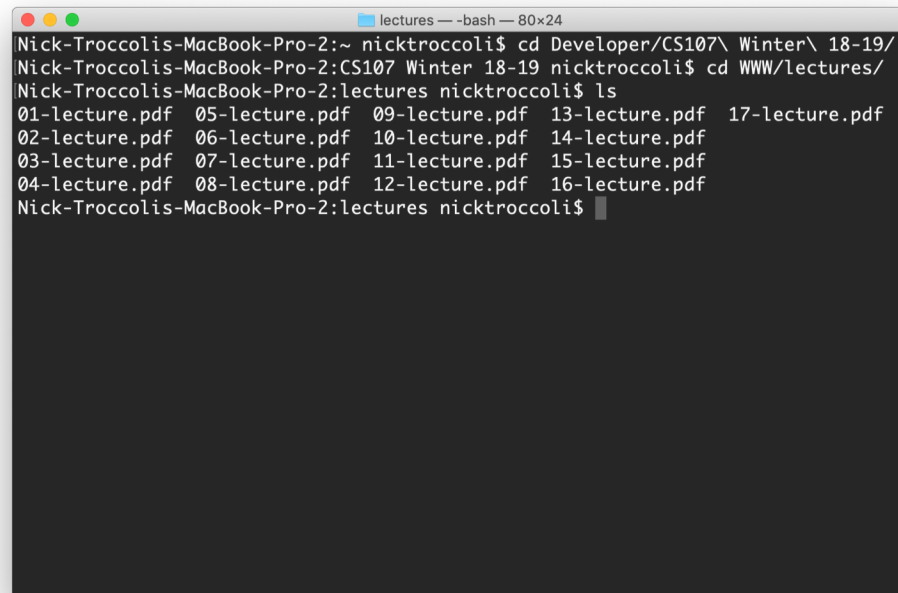
Honor Code and CS107

- Please help us ensure academic integrity:
  - Indicate any assistance received on HW (books, friends, etc.).
  - Do not look at other people's solution code
  - Do not give your solution code to others, or post it on the web or our Piazza forum.
  - Report any inappropriate activity you see performed by others.

- Assignments are checked regularly for similarity with help of software tools.

- If you realize that you have made a mistake, you may retract your submission to any assignment at any time, no questions asked.

- If you need help, please contact us and we will help you.
  - We do not want you to feel any pressure to violate the Honor Code in order to succeed in this course.

Plan For Today

- Introduction
- CS107 Course Policies
- Unix and the Command Line
- Getting Started With C

What is Unix?

- **Unix:** a set of standards and tools commonly used in software development.
    - **Macs** are built on top of Unix
    - **Linux** is built on top of Unix
- You can navigate a Unix system using the **command line** ("terminal")
- Every Unix system works with the same tools and commands

What is the Command Line?

- The **command-line** is a text-based interface to navigate a computer, instead of a Graphical User Interface (GUI).
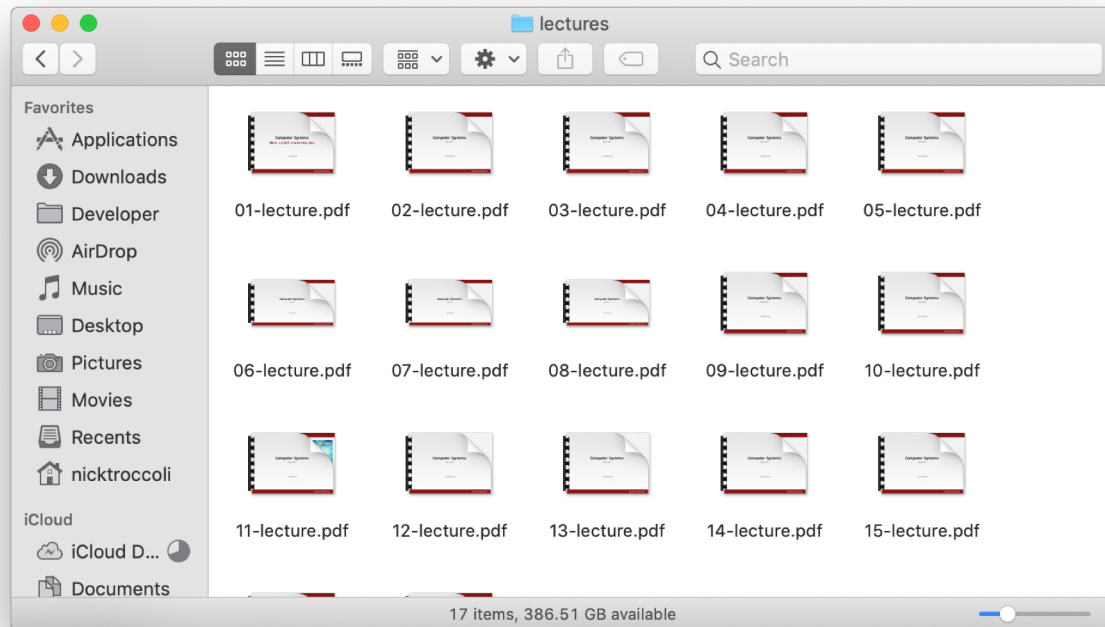
Graphical User Interface

Text-based interface

Why Use Unix / the Command Line?

- You can navigate almost any device using the same tools and commands:
  - Servers
  - Laptops and desktops
  - Embedded devices (Raspberry Pi, etc.)
  - Mobile Devices (Android, etc.)

- Used frequently by software engineers:
  - **Web development**: running servers and web tools on servers
  - **Machine learning**: processing data on servers, running algorithms
  - **Systems**: writing operating systems, networking code and embedded software
  - **Mobile Development**: running tools, managing libraries
  - And more…

- We are going to use Unix and the command line to write, debug, and run our programs.

Learning Unix and the Command Line

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

- We are going to teach you how!
    - Live lecture demos
    - Lab activities
    - Assignments
    - The **Resources** page of the course website

Command Line Vs. GUI

Just like a GUI file explorer interface, a terminal interface:

- shows you a **specific place** on your computer at any given time.

- lets you go **into folders** and **out of folders**.

- lets you **create new** files and **edit** files.

- lets you **run programs**.
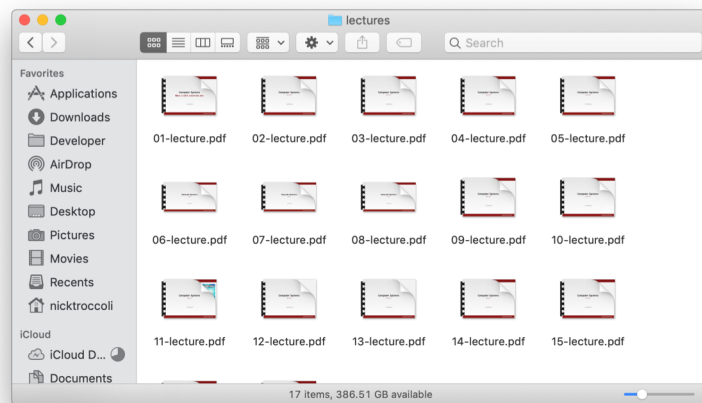
Graphical User Interface

Command-line interface

Demo: Using Unix and the Command Line

Unix Commands Recap

- **cd** – change directories
- **ls** – list directory contents
- **mkdir** – make directory
- **emacs** – open text editor
- **rm** – remove file or folder
- **man** – view manual pages

See the Resources page of the course website for more commands, and a complete reference.

Plan For Today

- Introduction
- CS107 Course Policies
- Unix and the Command Line
- Getting Started With C

The C Language

C was created around 1970 to make writing Unix and Unix tools easier.

- Part of the C/C++/Java family of languages (C++ and Java were created later)
- Design principles:
  - Small, simple abstractions of hardware
  - Minimalist aesthetic
  - Prioritizes efficiency and minimalism over safety and high-level abstractions

C vs. C++ and Java

They all share:

- Syntax
- Basic data types
- Arithmetic, relational, and logical operators

C doesn't have:

- More advanced features like operator overloading, default arguments, pass by reference, classes and objects, ADTs, etc.
- Extensive libraries (no graphics, networking, etc.) – this means not much to learn C!
- many compiler and runtime checks (this may cause security vulnerabilities!)

Programming Language Philosophies

C is procedural: you write functions, rather than define new variable types with classes and call methods on objects. C is small, fast and efficient.

C++ is procedural, with objects: you write functions, and also define new variable types with classes, and call methods on objects.

Java is object-oriented: it is even more centered around defining new variable types with classes, and calling methods on objects.

Why C?

- Many tools (and even other languages, like Python!) are built with C.
- C is the language of choice for fast, highly efficient programs.
- C is popular for systems programming (operating systems, networking, etc.)
- C lets you work at a lower level to manipulate and understand the underlying system.

Programming Language Popularity
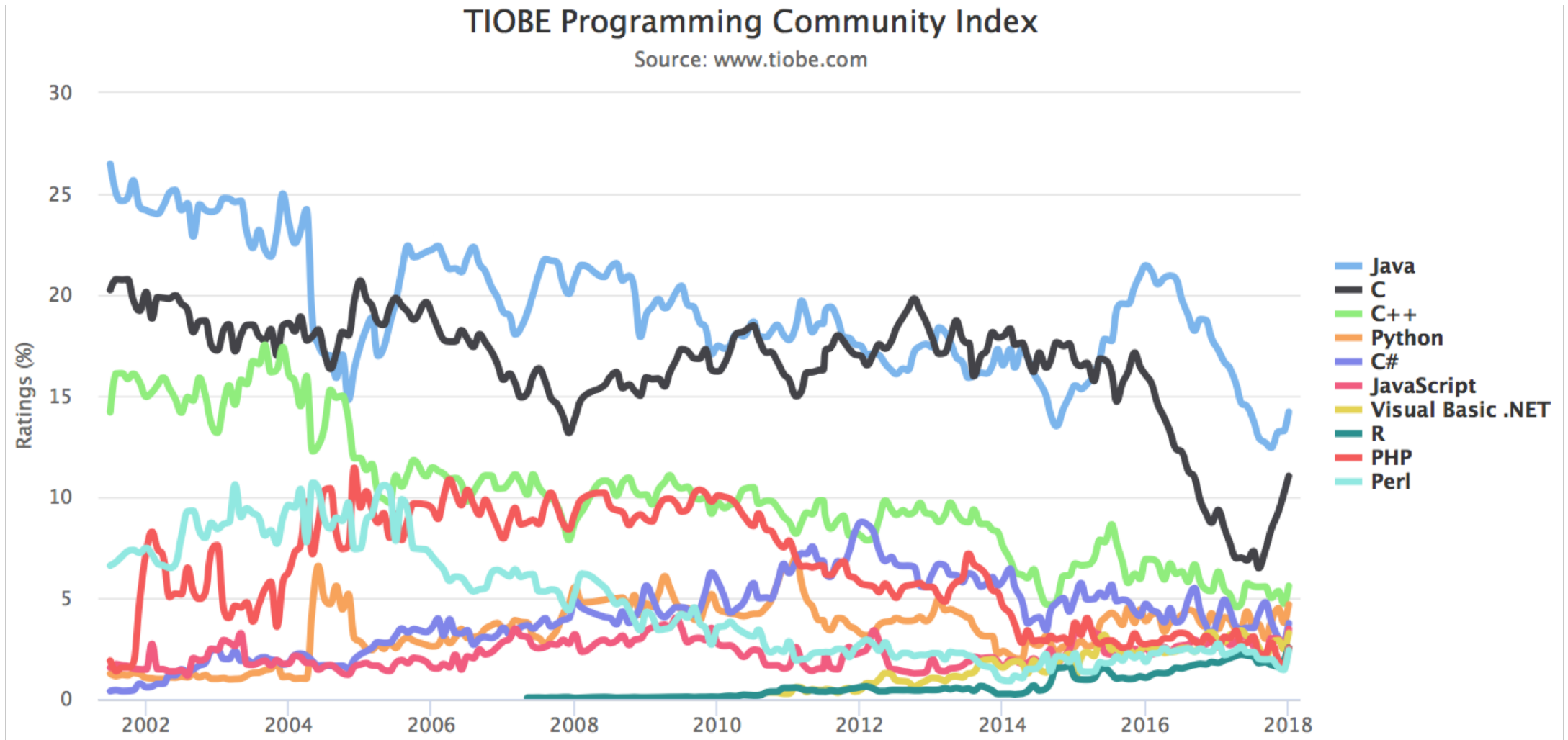
TIOBE Programming Community Index
Source: www.tiobe.com

Legend: Java, C, C++, Python, C#, JavaScript, Visual Basic .NET, R, PHP, Perl

Our First C Program

```c
/*
 * hello.c
 * This program prints a welcome message
 * to the user.
 */
#include <stdio.h>   // for printf

int main(int argc, char *argv[]) {
    printf("Hello, world!\n");
    return 0;
}
```

Our First C Program

```c
/*
 * hello.c
 * This program prints a welcome message
 * to the user.
 */
#include <stdio.h>    // for printf


int main(int argc, char *argv[]) {
    printf("Hello, world!\n");
    return 0;
}
```

Program comments
Inline comments can be written as:
`// comment`

```c
/*
 * hello.c
 * This program prints a welcome message
 * to the user.
 */
#include <stdio.h>    // for printf

int main(int argc, char *argv[]) {
    printf("Hello, world!\n");
    return 0;
}
```

Import statements
C libraries are written with angle brackets.
Local libraries have quotes:
`#include "lib.h"`

Our First C Program

```c
/*
 * hello.c
 * This program prints a welcome message
 * to the user.
 */
#include <stdio.h>   // for printf


int main(int argc, char *argv[]) {
    printf("Hello, world!\n");
    return 0;
}
```

Main function – entry point for the program
Should always return an integer (0 = success)

Our First C Program

```c
/*
 * hello.c
 * This program prints a welcome message
 * to the user.
 */
#include <stdio.h>   // for printf

int main(int argc, char *argv[]) {
    printf("Hello, world!\n");
    return 0;
}
```

Main parameters – **main** takes two parameters, both relating to the *command line arguments* used to execute the program.

argc is the *number* of arguments in **argv**
argv is an *array of arguments (char * is C string)*

44

Our First C Program

```c
/*
 * hello.c
 * This program prints a welcome message
 * to the user.
 */
#include <stdio.h>   // for printf

int main(int argc, char *argv[]) {
    printf("Hello, world!\n");
    return 0;
}
```

printf – prints output to the screen

```
int x = 42 + 7 * -5;                    // variables, types
double pi = 3.14159;
char c = 'Q';                           /* two comment styles */


for (int i = 0; i < 10; i++) {          // for loops
    if (i % 2 == 0) {                   // if statements
        x += i;
    }
}

while (x > 0 && c == 'Q' || b) {        // while loops, logic
    x = x / 2;
    if (x == 42) { return 0; }
}

fooBar(x, 17, c);                       // function call
```

Console Output: printf

```
printf(text, arg1, arg2, arg3);


        // Example
        char *classPrefix = "CS";
        int classNumber = 107;
        printf("You are in %s%d", classPrefix, classNumber);     // You are in CS107
```

printf makes it easy to print out the values of variables or expressions.

If you include *placeholders* in your printed text, printf will replace each placeholder *in order* with the values of the parameters passed after the text.

```
        %s (string)     %d (integer)     %f (double)
```

Writing, Debugging and Compiling

We will use:

- the **emacs** text editor to write our C programs.

- the **gcc** and **make** tools to compile our C programs.

- the **gdb** debugger to debug our programs.

- the **valgrind** tools to debug memory errors, and measure the efficiency of our programs.

Text Editor

- **Emacs** is the editor we recommend and will use this quarter in lectures and labs to write and edit C programs.

- There are helpful **emacs** tips and guides on the Resources page of the course website.

- We will only support the **emacs** text editor this quarter for working on programs. In particular, we **do not recommend** using a local text editor such as Sublime Text due to risks of data loss.

Demo: Compiling And Running A C Program

Working On C Programs Recap

- **ssh** – remotely log in to Myth computers
- **Emacs** – text editor to write and edit C programs
  - Use the mouse to position cursor, scroll, and highlight text
  - Ctl-x Ctl-s to save, Ctl-x Ctl-c to quit
- **make** – compile program using provided Makefile
- **./myprogram** – run executable program (optionally with arguments)
- **make clean** – remove executables and other compiler files
- Lecture code is accessible at **/afs/ir/class/cs107/samples/lectN**
  - Hint: try looking up the **cp** command for how to make your own copy to edit/run!

See the Resources page of the course website for more commands, and a complete reference.

assign0

Assignment 0 (Introduction to Unix and C) will be released tomorrow morning, and is due in one week on **Mon. 1/14 at 11:59PM PST**.

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

- Visit the **Resources** page to become familiar with 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

Recap

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

- We'll use Unix and command line tools to write, debug and run our programs.

- Please visit the course website, cs107.stanford.edu, where you can read the General Information Handout, information about the Honor Code in CS107, and more about CS107 course policies and logistics.

We're looking forward to an awesome quarter!

Next time: Bits and Bytes