CS107 Lecture 2
Unix and C

Announcements

- Remember to input your section preferences through 5PM Sat! Link is on the course website (under “Sections”).
- Helper Hours scheduled and starting this week!
- assign0 released, due Mon 11:59PM PDT
- Please email Megan (Head TA) about OAE accommodations and midterm conflicts as soon as you can

CS 106L: Standard C++ Programming

1 unit, TTh 4:30-5:50, Thornton 110
Course Info: cs106l.stanford.edu

Containers

```
Vector<T> → std::vector<T>
```

Algorithms

```
std::binary_search
```

Streams

```
std::iostream
```

Lambdas

```
[&]() { return val; }
```

Operator Overloading

```
T operator+(T& rhs);
```

Special Member Functions

```
T(const T& copy);
```

rvalues

```
int&& a = 5 + 2;
```

Move Semantics

```
T(T&& tempval);
```

RAII

```
std::make_unique<Type>(args…);
```

Learning Goals

- Learn how to navigate a computer and edit/run programs using the terminal
- Understand the differences between C and other languages and how to write C programs

Lecture Plan

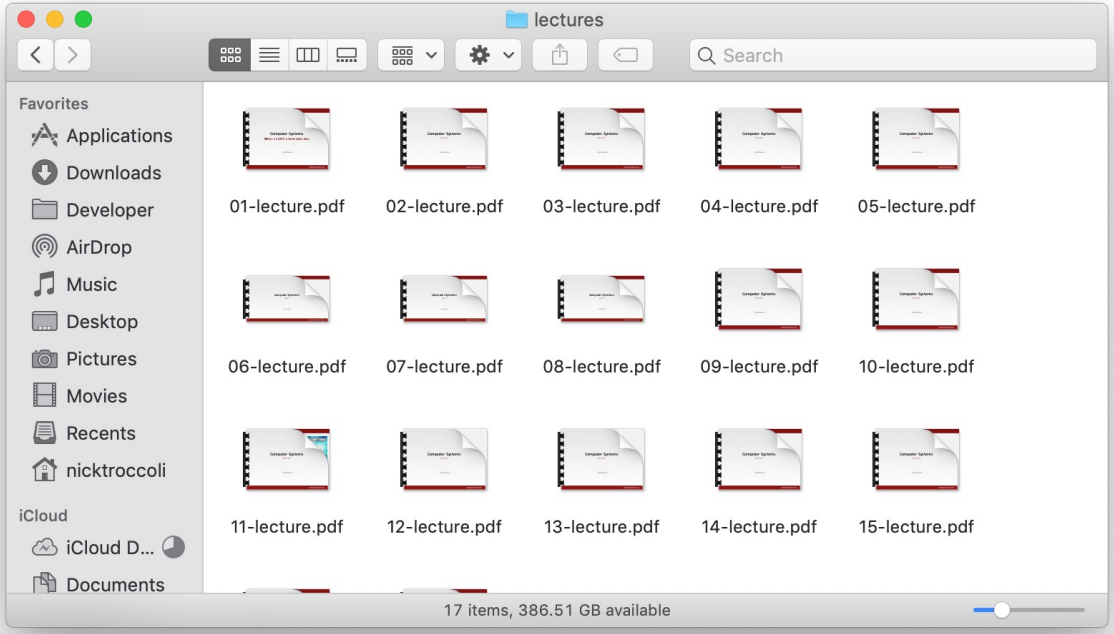
- Unix and the Command Line
- Getting Started With C

Lecture Plan

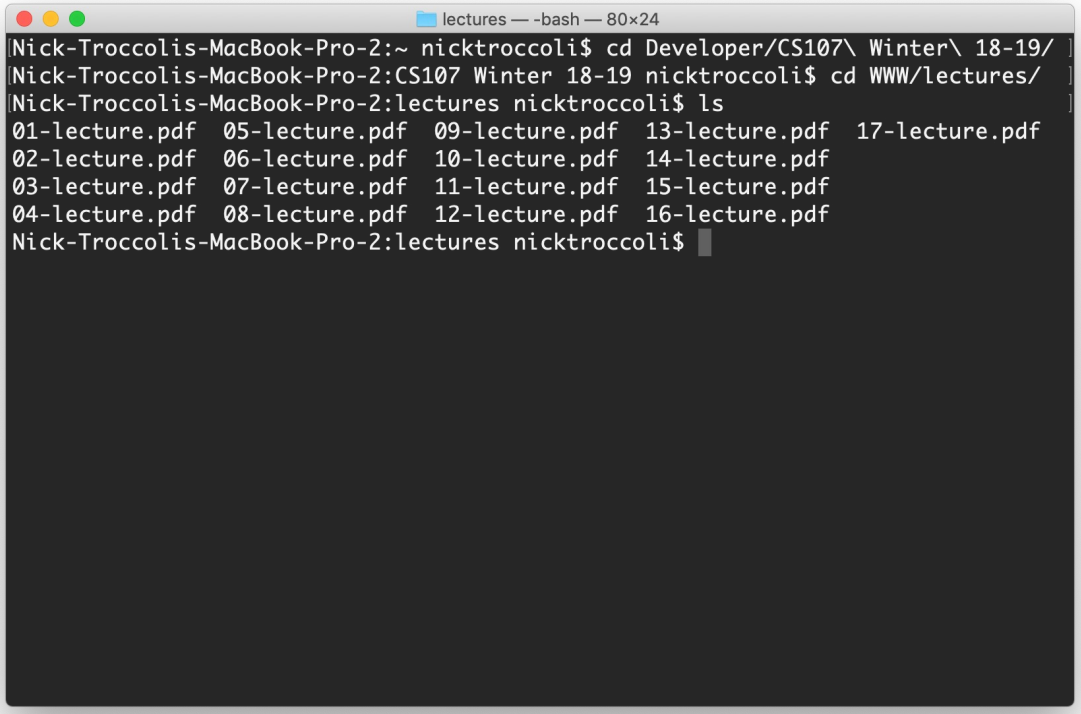
- **Unix and the Command Line**
- Getting Started With C

What is the Command Line?

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



Graphical User Interface



Text-based interface

Unix Commands To Try

- **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 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/resources/getting-started.html

Lecture Plan

- Unix and the Command Line
- **Getting Started With C**
- Integer Representations
- Bits and Bytes
- Unsigned Integers

The C Language

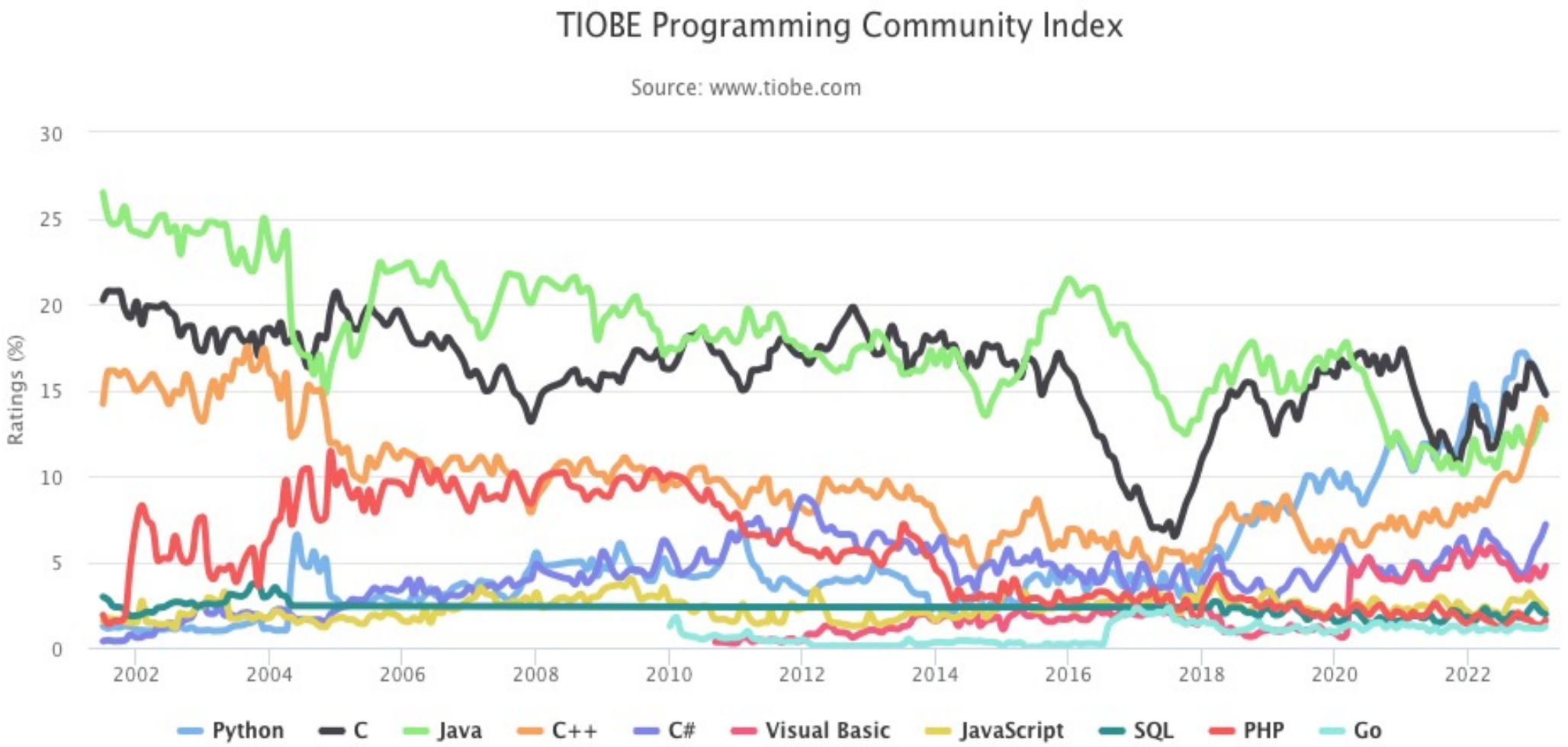
C was created around 1970 to make writing Unix and Unix tools easier.

- 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
- Procedural (you write functions, no classes or methods) – vs. C++ or Python where you can write functions but also classes with methods
- Doesn't have all features you may know from other languages (e.g., no pass by reference, no classes and objects, no ADTs, no extensive libraries, weak compiler and almost no runtime checks – which can cause security vulnerabilities!)

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



https://www.tiobe.com/tiobe-index/

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
You can write block or inline comments.

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;
}
```

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)***

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

Console Output: printf

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

`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)`

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

Familiar Syntax

```
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;
    }
}

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

Boolean Variables

To declare Booleans, (e.g. `bool b = ____`), you must include `stdbool.h`:

```
#include <stdio.h>     // for printf
#include <stdbool.h>   // for bool

int main(int argc, char *argv[]) {
    bool x = 5 > 2 && binky(argc) > 0;
    if (x) {
        printf("Hello, world!\n");
    } else {
        printf("Howdy, world!\n");
    }
    return 0;
}
```

Boolean Expressions

C treats a nonzero value as true, and a zero value as false:

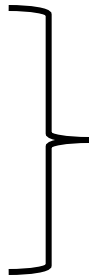
```
#include <stdio.h>

int main(int argc, char *argv[]) {
    int x = 5;
    if (x) {   // true
       printf("Hello, world!\n");
    } else {
       printf("Howdy, world!\n");
    }
    return 0;
}
```

Writing, Debugging and Compiling

We will use:

- the **emacs** text editor to write our C programs
- the **make** tool to compile our C programs
- the **gdb** debugger to debug our programs
- the **valgrind** tools to debug memory errors and measure program efficiency

Now



Next week

Working On C Programs

- **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/lecture-code/lect[N]**
  - Make your own copy: **cp -r /afs/ir/class/cs107/lecture-code/lect[N] lect[N]**
  - See the website for even more commands, and a complete reference.

Demo: Compiling And Running A C Program

Get up and running with our guide:
http://cs107.stanford.edu/resources/getting-started.html

Assign0

Assignment 0 (Intro to Unix and C) is due on **Mon. 4/10 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:

- Visit the website resources 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

Preview: Next Time

- Make sure to reboot Boeing Dreamliners every 248 days
- Comair/Delta airline had to cancel thousands of flights days before Christmas
- Many operating systems may have issues storing timestamp values beginning on Jan 19, 2038
- Reported vulnerability CVE-2019-3857 in libssh2 may allow a hacker to remotely execute code

Next time: *How can a computer represent integer numbers? What are the limitations?*