CS107, Lecture 1
Welcome to CS107!

reading:

General Information page

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

Honor Code and Collaboration Page

Plan For Today

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

CS107 on Zoom

- You are encouraged to share video!
- Post questions/comments/followups on the discussion forum thread for that day's lecture. We'll take periodic "question breaks" to address them.
- Everyone is muted by default.

Join: Through Canvas, or try
https://edstem.org/us/courses/6868

Today's thread:
https://edstem.org/us/courses/6868/discussion/491803

Introduce yourself!
https://edstem.org/us/courses/6868/discussion/491795

Plan For Today

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

What is CS107?

The CS106 series:

- Taught 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”

CS107 finally takes you **behind the scenes**:

- Not quite down to hardware or physics/electromagnetism (that’s for later...)
- It’s how things work **inside C++/Python/Java**, and how your programs map onto the components of computer systems
- Not only does it just feel good to know how these work, it can also inform projects you work on in the future.

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

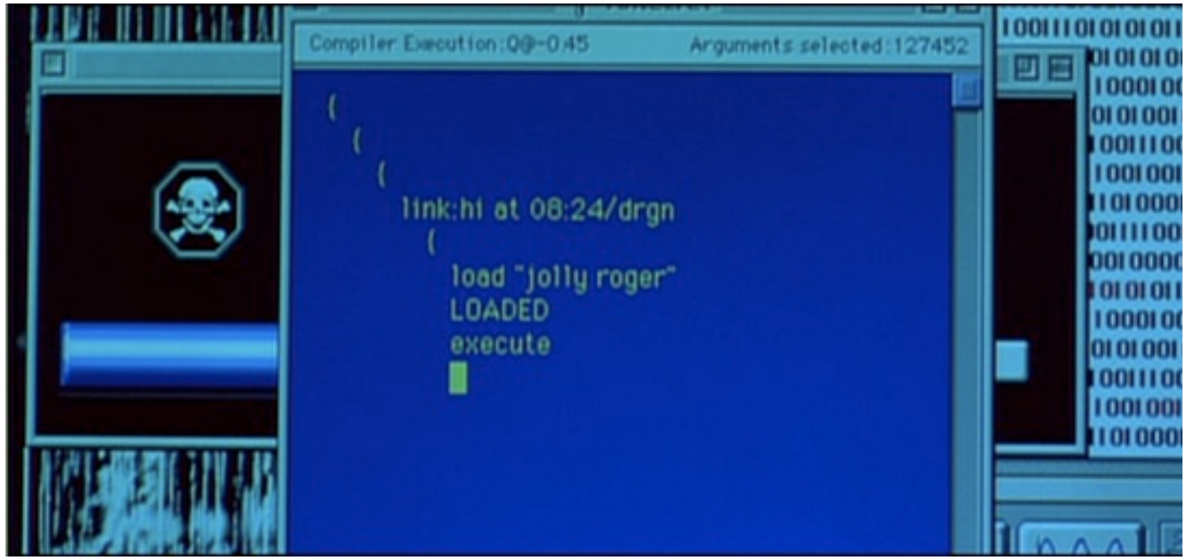
- translating C to/from assembly
- writing programs that respect the limitations of computer arithmetic
- identifying bottlenecks and improving runtime performance
- working effectively in a Unix development environment

and have **exposure** to

- a working understanding of the basics of computer architecture

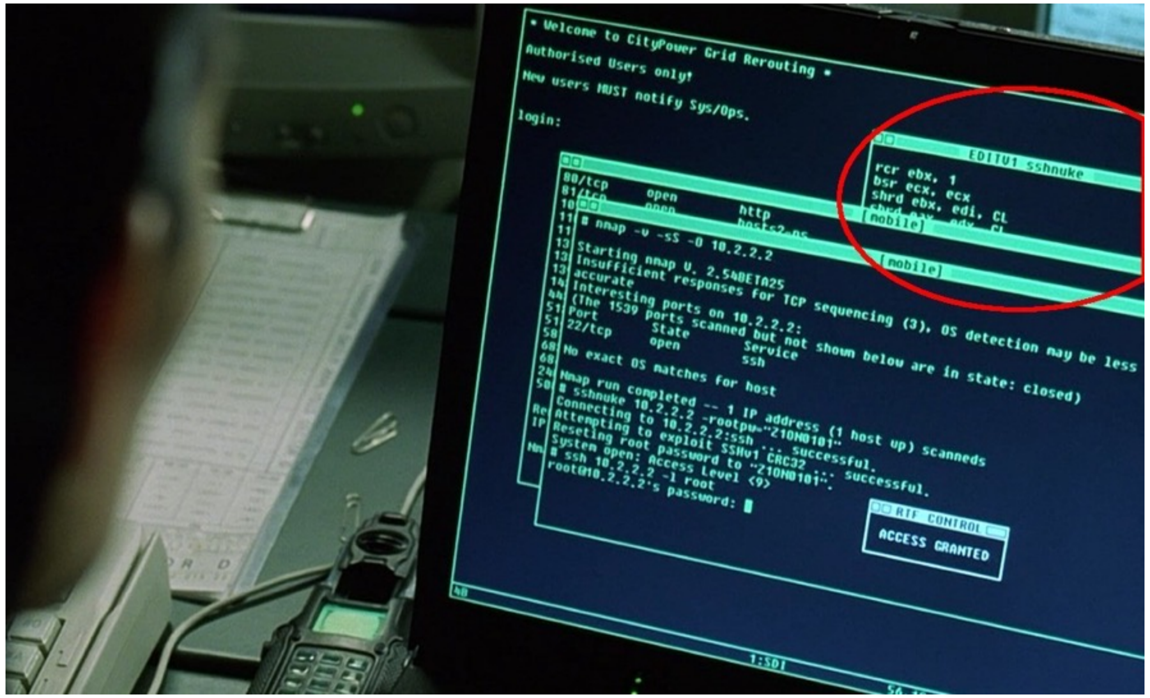
CS107 Learning Goals

(also learn to identify legitimate programmer scenes in old movies)



```
Compiler Execution:Q@-0:45          Arguments selected:127452
{
  {
    {
      link:hi at 08:24/drgn
        {
          load "jolly roger"
          LOADED
          execute
```

Jeff Goldblum's character saving the world by uploading a virus to the alien mothership
Independence Day, 1996



```
* Welcome to CityPower Grid Rerouting *
Authorised Users only!
New users MUST notify Sys/Ops.

login:
```

```
EDITV1 sshnuke
rcr ebx, 1
bsr ecx, ecx
shrd ebx, edi, CL
```

```
# nmap -v -sS -O 10.2.2.2
Starting nmap V. 2.54BETA25
Insufficient responses for TCP sequencing (3), OS detection may be less
accurate
Interesting ports on 10.2.2.2:
(The 1539 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh

No exact OS matches for host

Nmap run completed -- 1 IP address (1 host up) scanneds
# sshnuke 10.2.2.2 -rootpw="Z1ON0101"
Connecting to 10.2.2.2:ssh ... successful.
Attempting to exploit SSHv1 CRC32 ... successful.
Reseting root password to "Z1ON0101".
System open: Access Level <9>
# ssh 10.2.2.2 -l root
root@10.2.2.2's password: █
```

Trinity saving the world by hacking into the power grid using Nmap Network Scanning
The Matrix Reloaded, 2003

Course Overview

1. **Bits and Bytes -** *How can a computer represent integer numbers?*
2. **Chars and C-Strings -** *How can a computer represent and manipulate more complex data like text?*
3. **Pointers, Stack and Heap –** *How can we effectively manage all types of memory in our programs?*
4. **Generics -** *How can we use our knowledge of memory and data representation to write code that works with any data type?*
5. **Assembly -** *How does a computer interpret and execute C programs?*
6. **Heap Allocators -** *How do core memory-allocation operations like malloc and free work?*

CS107 Online

- This quarter, we are making many adjustments and changes to make CS107 the best it can be in an online format.
- We are working to emphasize community and connection.
- We understand the unprecedented situation this quarter presents for everyone involved.
- We will constantly evaluate and listen to ensure the class is going as smoothly as possible for everyone.
- Please communicate with us if any personal circumstances or issues arise! We are here to support you.

CS107 Online

- This quarter, we are making many adjustments and changes to make CS107 the best it can be in an online format.
- **We are working to emphasize community and connection.**
- We understand the unprecedented situation this quarter presents for everyone involved.
- We will constantly evaluate and listen to ensure the class is going as smoothly as possible for everyone.
- Please communicate with us if any personal circumstances or issues arise! We are here to support you.

A Wonderful Community ❤️

I hope we can find a way to connect with each other!

Really hope that this helps be be a better software engineer!

I love being outdoors and in nature!

I am learning how to spearfish

research the extraction of rare earth metals!

I like to make song remixes/mashups

It turns out that I am very picky about rice

I aspire to be a pirate.

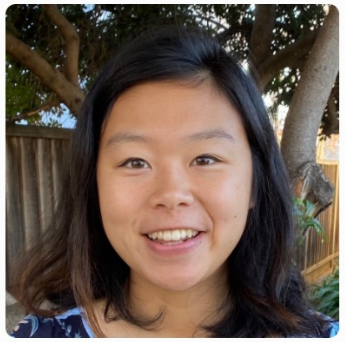
I'm scared of weeks 8-10

*Quotes taken from CS107 welcome survey, please fill it out!

Teaching Team

Andrew Benson

Cem Gokmen


Maddy Yip

Ofure Ebhomielen

Seiji Eicher

Nick Troccoli

About Andrew Benson (adbenson@stanford.edu):

- Current Stanford MS student in CS, Software Theory track
- Native of Seattle, but have been in the Bay for the last 4 years
- Systems has played a key part in my discovery of CS. I hope it will for you too ☺

Course Website

cs107.stanford.edu

*lecture videos on Canvas

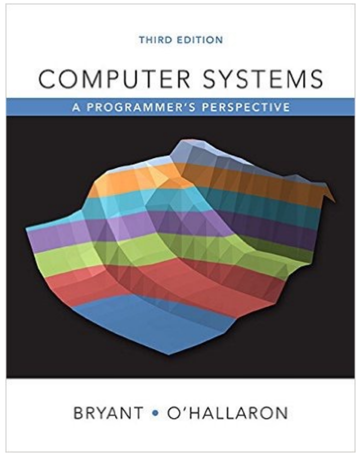
Question Break!

Plan For Today

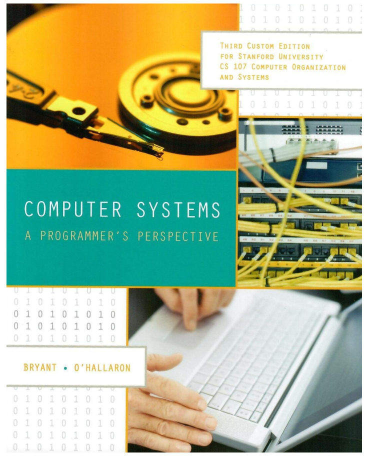
- Introduction
- **CS107 Course Policies**
- Unix and the Command Line
- Getting Started With C

Textbook(s)

- *Computer Systems: A Programmer's Perspective* by Bryant & O'Hallaron, **3rd Edition**
  - **3rd edition matters** – important updates to content
  - Stanford Library has generously scanned **all** readings for CS107 under "fair use" (private study, scholarship, research). [**Canvas -> Files**]. Please do not distribute.
  - If you want more context, or if you plan to continue onto CS110, you may want to purchase a full copy
- 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



Full textbook


Third Custom Edition for Stanford University CS 107 Computer Organization and Systems
COMPUTER SYSTEMS
A PROGRAMMER'S PERSPECTIVE
BRYANT • O'HALLARON

CS107 full chapters

canvas CS107-specific readings

The textbook (and C programming references) are **very** good resources in this course, especially post-midterm!

Course Structure

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

The assignment release cadence for assignments (except assign0 and the last two)

	Friday	Monday	Tues-Thurs	Friday	Monday	Tues-Thurs
assign N	Lecture: part A	Lecture: part B	Lab; assignN released			assign N due
assign N+1				Lecture: part A	Lecture: partB	Lab; assign N released

- **assign0**: already out, due next Monday (covers today's lecture)

*Lectures are "flipped" (pre-recorded) on Canvas; live lecture is optional review and Q&A

Grading

*	10%	Lecture Check-in Quizzes
*****	45%	Assignments
**	20%	Final Project
*	10%	Mid-quarter Assessment
**	15%	Lab Participation

Read our full course policies document:
https://cs107.stanford.edu/syllabus.html

Grading

*	10%	Lecture Check-in Quizzes
*****	45%	Assignments
**	20%	Final Project
*	10%	Mid-quarter Assessment
**	15%	Lab Participation

Read our full course policies document:
https://cs107.stanford.edu/syllabus.html

Lecture Check-in Quizzes

- Main lecture material is **pre-recorded** in short video “bytes” (get it?) and posted in advance of live lecture.
- Short “lecture check-in quizzes” after every ~1-2 videos, permitting 3 attempts, due by that live lecture.
- Each day’s worth of lecture quizzes are weighted the same in aggregate.
- **Live lecture** times are optional, will be 30-45 minutes max, and we instead review concepts further, answer questions, and do additional exercises.
- You can submit questions in advance of the live lectures or bring questions!

Experiment: Lecture Watch Parties 🎉

- All lecture content (including live sessions) is recorded and available on Canvas. Main lecture material is pre-recorded; live sessions are posted afterwards.
- But the extreme flexibility of pre-recorded content can make it hard to keep yourself accountable…
- Experiment: we will designate optional twice a week *lecture watch parties* to encourage you and your classmates to keep up with lectures
- Lecture watch parties will be on Nooks (same as Helper Hours)
- I or the CAs may drop in to do work as well, though we won't be actively answering questions
- Starting *next week* - more info to come soon

Question Break!

Read our full course policies document:
https://cs107.stanford.edu/syllabus.html

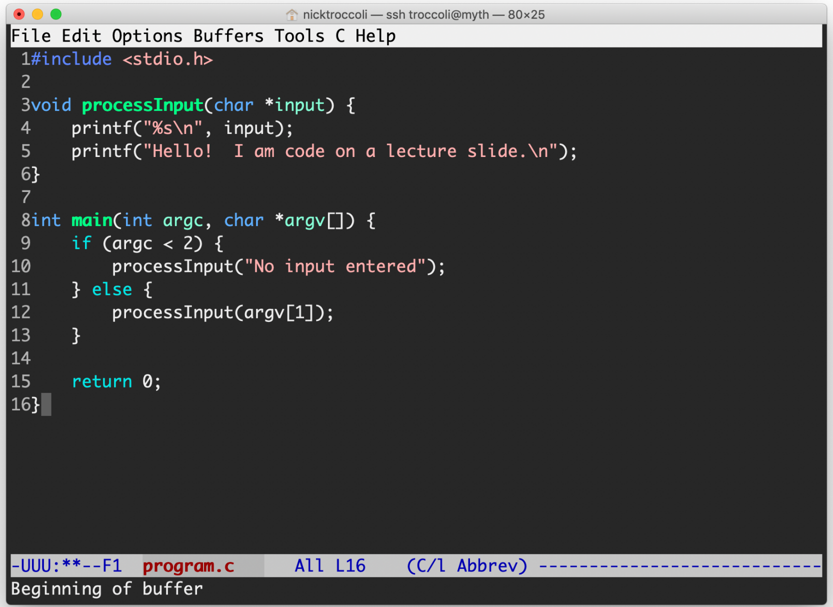
Grading

*	10%	Lecture Check-in Quizzes
*****	45%	Assignments
**	20%	Final Project
*	10%	Mid-quarter Assessment
**	15%	Lab Participation

Read our full course policies document:
https://cs107.stanford.edu/syllabus.html

Assignments

- 6 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 – no TA review
  - Style graded via *automated tests* and TA code review, given as bucket score
  - Grades returned via course website

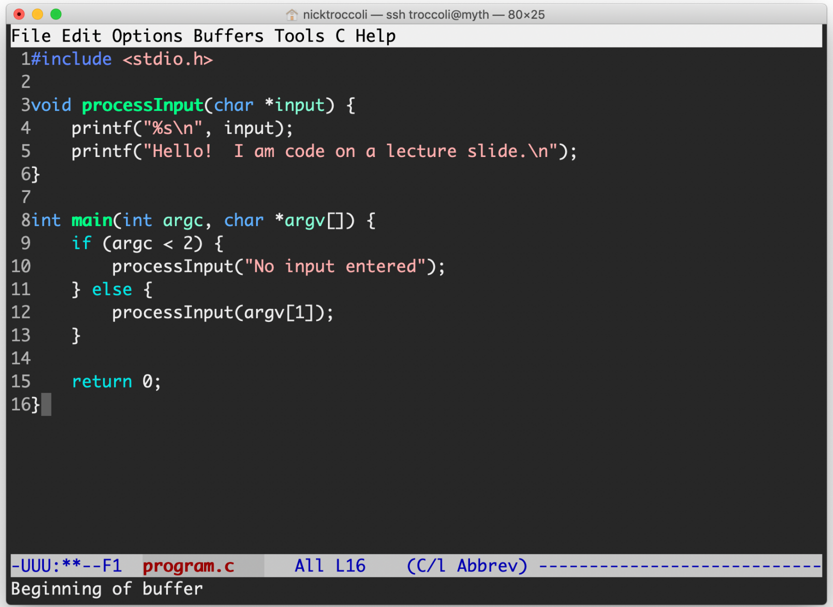

nicktroccoli — ssh troccoli@myth — 80×25
File Edit Options Buffers Tools C Help

```
#include <stdio.h>

void processInput(char *input) {
    printf("%s\n", input);
    printf("Hello!  I am code on a lecture slide.\n");
}

int main(int argc, char *argv[]) {
    if (argc < 2) {
        processInput("No input entered");
    } else {
        processInput(argv[1]);
    }

    return 0;
}
```

```
-UUU:**--F1  program.c      All L16    (C/l Abbrev) ---------------------------
Beginning of buffer
```

Assignment Handout Helper Hours

- Getting started on an assignment is the biggest barrier to success
- The day after an assignment is released, the CAs will hold special AHH Hours
- You must read the assignment handout before coming, but you can ask questions in a group about the assignment
- No debugging questions allowed, only questions about this particular assignment
- We want to encourage you to read the assignment as early as possible ☺
- Starting with assign1 next week


nicktroccoli — ssh troccoli@myth — 80×25
File Edit Options Buffers Tools C Help

```
#include <stdio.h>

void processInput(char *input) {
    printf("%s\n", input);
    printf("Hello!  I am code on a lecture slide.\n");
}

int main(int argc, char *argv[]) {
    if (argc < 2) {
        processInput("No input entered");
    } else {
        processInput(argv[1]);
    }

    return 0;
}
```

```
-UUU:**--F1  program.c      All L16    (C/l Abbrev) ----------------------------
Beginning of buffer
```

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.
- -	Shows many significant issues and does not represent passing work.
0	No work submitted, or barely any changes from the starter assignment.

Late Policy

- **Start out with 5 “free late days”**: each late day allows you to submit an assignment up to 24 additional hours late without penalty. (No late days permitted for assign0)
- **Hard deadline 48 hours** after original due date
- Penalty per day after late days are exhausted (1 day: 80% cap; 2 days: 60% cap)
- Late days are “pre-granted extensions” – additional extensions for exceptional circumstances must be approved by the **instructor**. Please communicate with us! We are here to accommodate you as much as possible.

Grading

*	10%	Lecture Check-in Quizzes
*****	45%	Assignments
**	20%	Final Project
*	10%	Mid-quarter Assessment
***	15%	Lab Participation

Read our full course policies document:
https://cs107.stanford.edu/syllabus.html

Final Project

- The final project has you implement your own “heap allocator” – a fundamental component of many of the programs we will write this quarter.
- We first use it as clients, and then we implement it!
- Capstone individual project that sums up topics from throughout the quarter.
- **You must do the final project in order to pass the class.**

Question Break!

Read our full course policies document:
https://cs107.stanford.edu/syllabus.html

Grading

*	10%	Lecture Check-in Quizzes
*****	45%	Assignments
**	20%	Final Project
*	10%	Mid-quarter Assessment
**	15%	Lab Participation

Read our full course policies document:
https://cs107.stanford.edu/syllabus.html

Mid-Quarter Assessment

- Available from Wed. 8/4 – Fri 8/6
- Open-book, timed assessment; you can choose when to start
- We aim for a lower-stakes assessment instead of a traditional midterm:
  - Weighted 10%
  - Flexible timing
  - Assessment will be written to minimize time pressure
  - Primary goal: reinforce concepts learned to that point
- Covers material from assign1-4

Grading

*	10%	Lecture Check-in Quizzes
*****	45%	Assignments
**	20%	Final Project
*	10%	Mid-quarter Assessment
**	15%	Lab Participation

Read our full course policies document:
https://cs107.stanford.edu/syllabus.html

Lab Sections

- Weekly 1 hour labs led by a CA, starting *next* week, offered on either Tuesdays, Wednesdays, or Thursdays.
- Hands-on practice in small groups with lecture material and course concepts.
- Graded on attendance + participation
- Lab preference submissions open no earlier than **Tuesday 6/22 at 5PM PST** and **are not first-come first-serve**. You may submit your preferences anytime until **Saturday 6/26 at 5PM PST**. Sign up on the course website.

Question Break!

Read our full course policies document:
https://cs107.stanford.edu/syllabus.html

Grading

*	10%	Lecture Check-in Quizzes
*****	45%	Assignments
**	20%	Final Project
*	10%	Mid-quarter Assessment
**	15%	Lab Participation

Read our full course policies document:
https://cs107.stanford.edu/syllabus.html

Getting Help

- Post on the **Discussion Forum**
  - 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!)
- Visit **Helper Hours**
  - 24/7 open community where you can chat about course topics or just hang out
  - Using Nooks
  - Work anytime with other students or come at scheduled times for TA help; schedule will be posted on course website tomorrow.
  - Best for **group work**, **coding/debugging questions (with TAs only!)** or **longer course material discussions**
- **Email** the Course Staff
  - cs107@cs.stanford.edu – please do not email CAs individually
  - Best for **private matters** (e.g. grading questions, OAE accommodations).

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 Honor Code handout 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 or answers
  - Do not give your solutions to others or post them on the web or our Ed forum.
  - Report any inappropriate activity you see performed by others.
- Assignments are checked regularly for similarity with help of software tools.
- 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.
  - If you realize that you have made a mistake, you may retract your submission to any assignment at any time, no questions asked.

https://cs107.stanford.edu/collaboration

OAE Accommodations

- Please email the course staff (cs107@cs.stanford.edu) as soon as possible with any accommodations you may need for the course.
- We are eager to do everything we can to support you and make you successful in CS107!

Incompletes from previous quarters

- Please email the course staff (cs107@cs.stanford.edu) as soon as possible with your situation.
- Do not enroll in the course.

Waitlist

- CS 107 does not usually have an enrollment cap and/or waitlist – this summer quarter is unusual
- Course enrollment fluctuates significantly during shopping period – you may still get into the course
- I do not control the waitlist
- Waitlisted students should have normal access in the meantime
- Email cs107@cs.stanford.edu if you run into issues

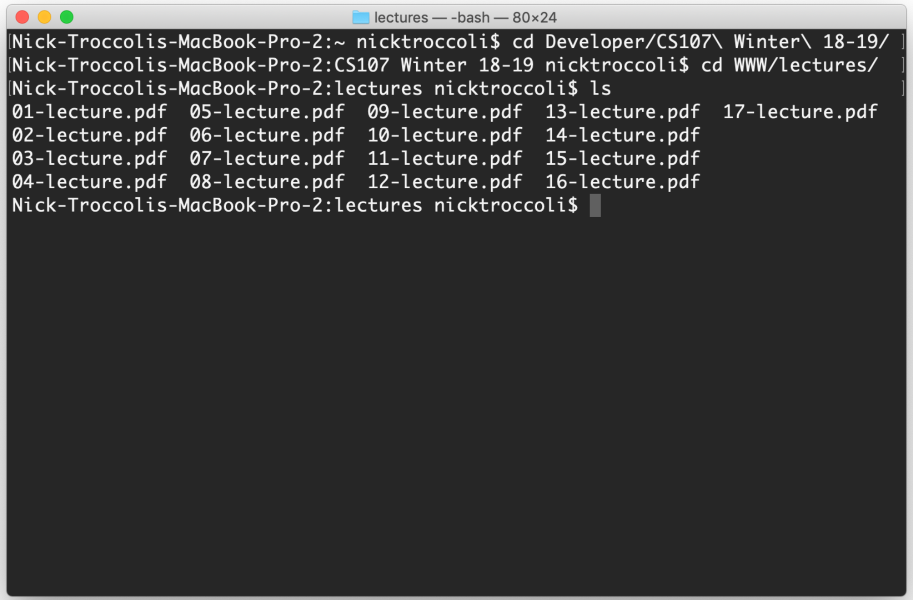
Question Break!

Plan For Today

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

What is Unix?

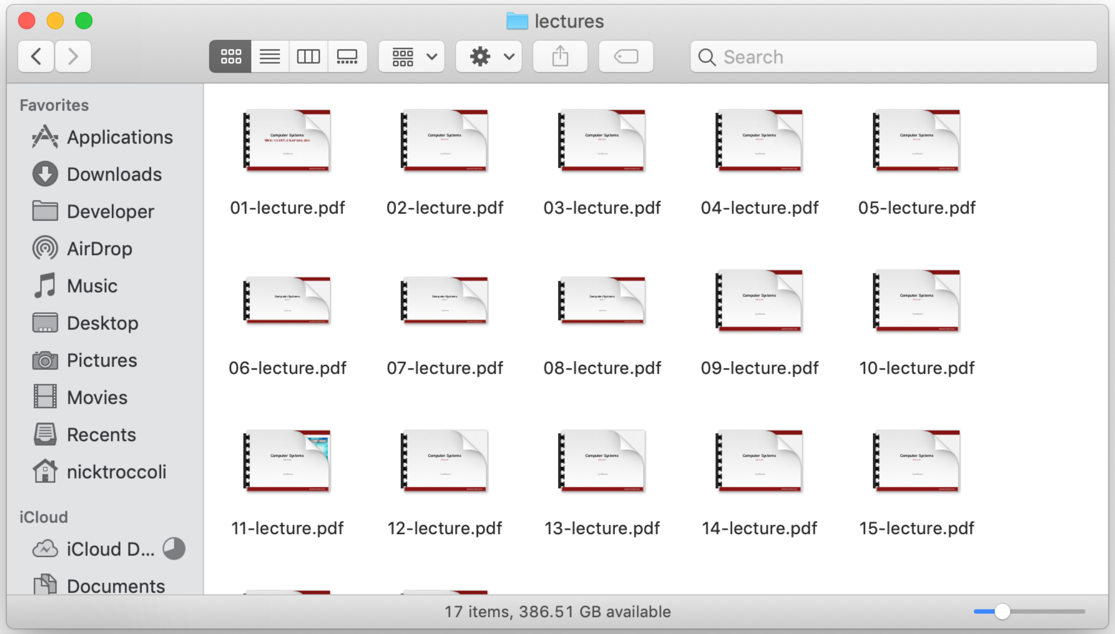
- **Unix:** a set of standards and tools commonly used in software development.
  - **macOS** and **Linux** are operating systems built on top of Unix
- You can navigate a Unix system using the **command line** ("terminal")
- Every Unix system works with the same tools and commands



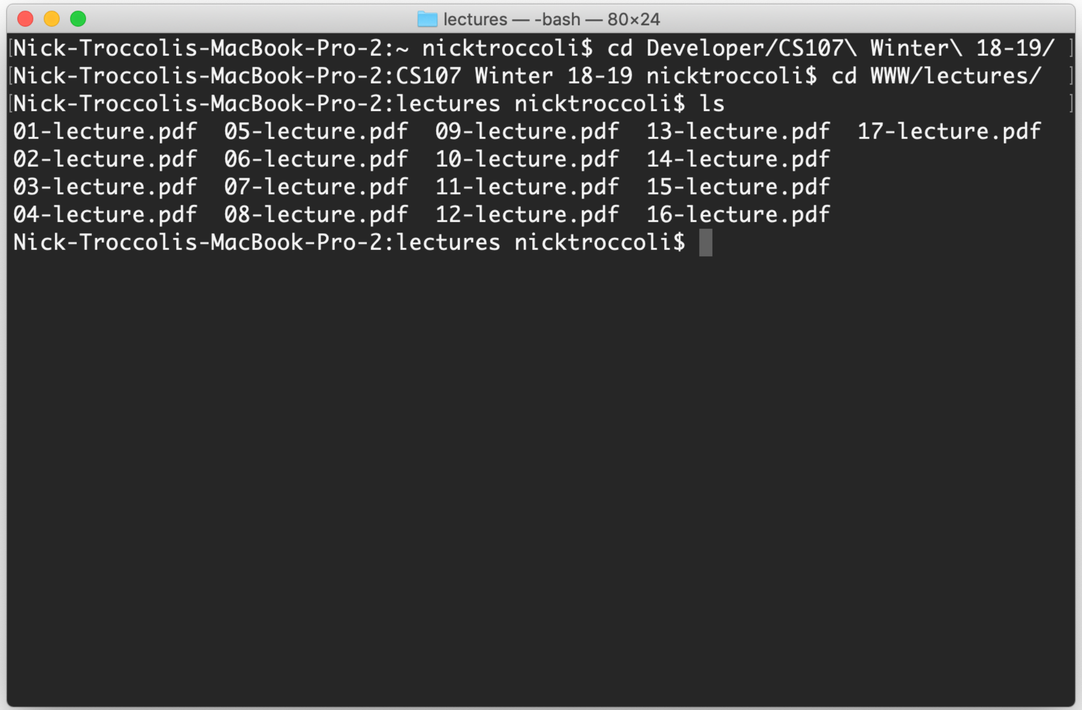
```
Nick-Troccolis-MacBook-Pro-2:~ nicktroccoli$ cd Developer/CS107\ Winter\ 18-19/
Nick-Troccolis-MacBook-Pro-2:CS107 Winter 18-19 nicktroccoli$ cd WWW/lectures/
Nick-Troccolis-MacBook-Pro-2:lectures nicktroccoli$ ls
01-lecture.pdf  05-lecture.pdf  09-lecture.pdf  13-lecture.pdf  17-lecture.pdf
02-lecture.pdf  06-lecture.pdf  10-lecture.pdf  14-lecture.pdf
03-lecture.pdf  07-lecture.pdf  11-lecture.pdf  15-lecture.pdf
04-lecture.pdf  08-lecture.pdf  12-lecture.pdf  16-lecture.pdf
Nick-Troccolis-MacBook-Pro-2:lectures nicktroccoli$ 
```

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

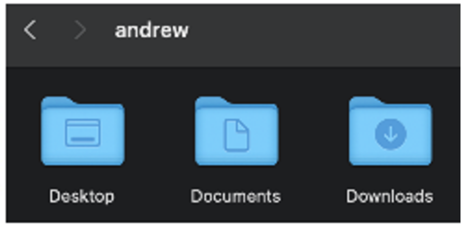


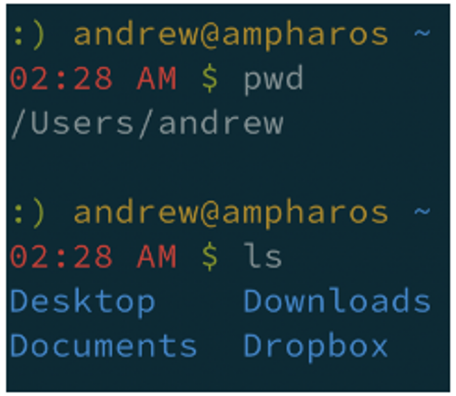
```
Nick-Troccolis-MacBook-Pro-2:~ nicktroccoli$ cd Developer/CS107\ Winter\ 18-19/
Nick-Troccolis-MacBook-Pro-2:CS107 Winter 18-19 nicktroccoli$ cd WWW/lectures/
Nick-Troccolis-MacBook-Pro-2:lectures nicktroccoli$ ls
01-lecture.pdf  05-lecture.pdf  09-lecture.pdf  13-lecture.pdf  17-lecture.pdf
02-lecture.pdf  06-lecture.pdf  10-lecture.pdf  14-lecture.pdf
03-lecture.pdf  07-lecture.pdf  11-lecture.pdf  15-lecture.pdf
04-lecture.pdf  08-lecture.pdf  12-lecture.pdf  16-lecture.pdf
Nick-Troccolis-MacBook-Pro-2:lectures nicktroccoli$ 
```

Text-based interface

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).



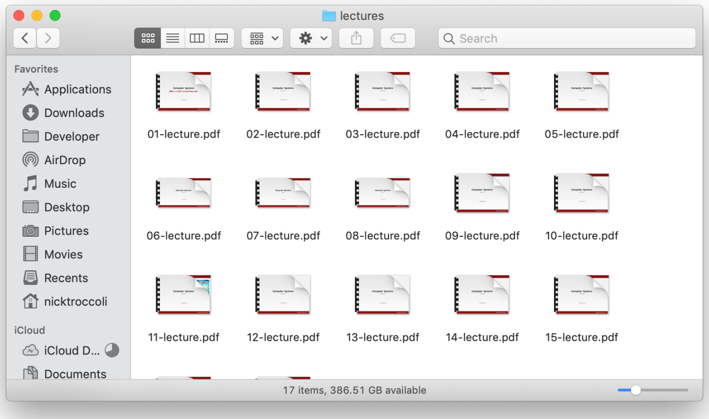


```
:) andrew@ampharos ~
02:28 AM $ pwd
/Users/andrew

:) andrew@ampharos ~
02:28 AM $ ls
Desktop      Downloads
Documents    Dropbox
```

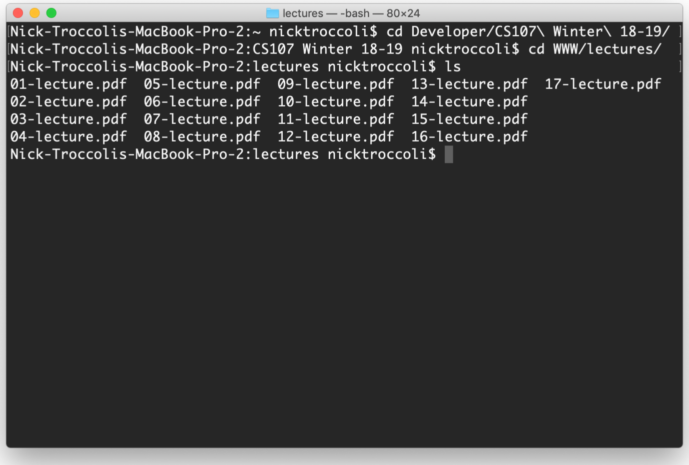
Command Line Vs. GUI

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 **execute programs**.



Graphical User Interface



```
Nick-Troccolis-MacBook-Pro-2:~ nicktroccoli$ cd Developer/CS107\ Winter\ 18-19/
Nick-Troccolis-MacBook-Pro-2:CS107 Winter 18-19 nicktroccoli$ cd WWW/lectures/
Nick-Troccolis-MacBook-Pro-2:lectures nicktroccoli$ ls
01-lecture.pdf  05-lecture.pdf  09-lecture.pdf  13-lecture.pdf  17-lecture.pdf
02-lecture.pdf  06-lecture.pdf  10-lecture.pdf  14-lecture.pdf
03-lecture.pdf  07-lecture.pdf  11-lecture.pdf  15-lecture.pdf
04-lecture.pdf  08-lecture.pdf  12-lecture.pdf  16-lecture.pdf
Nick-Troccolis-MacBook-Pro-2:lectures nicktroccoli$ 
```

Command-line 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'll use Unix and the command line to implement and execute our programs.

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

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 course website for more commands and a complete reference.

Learning Unix and the Command Line

- Using Unix and the command line can be intimidating at first:
  - It looks retro!
  - How do I know what to type?
- It’s like learning a new language:
  - At first, you may have to constantly look things up (**resources** on course website!)
  - It’s important to 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/resources/getting-started.html

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 limitations:

- No advanced features like operator overloading, default arguments, pass by reference, classes and objects, ADTs, etc.
- No extensive libraries (no graphics, networking, etc.) – Small language footprint means not much to learn ☺
- Weak compiler and almost no 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 define new variable types with classes, and call methods on objects.

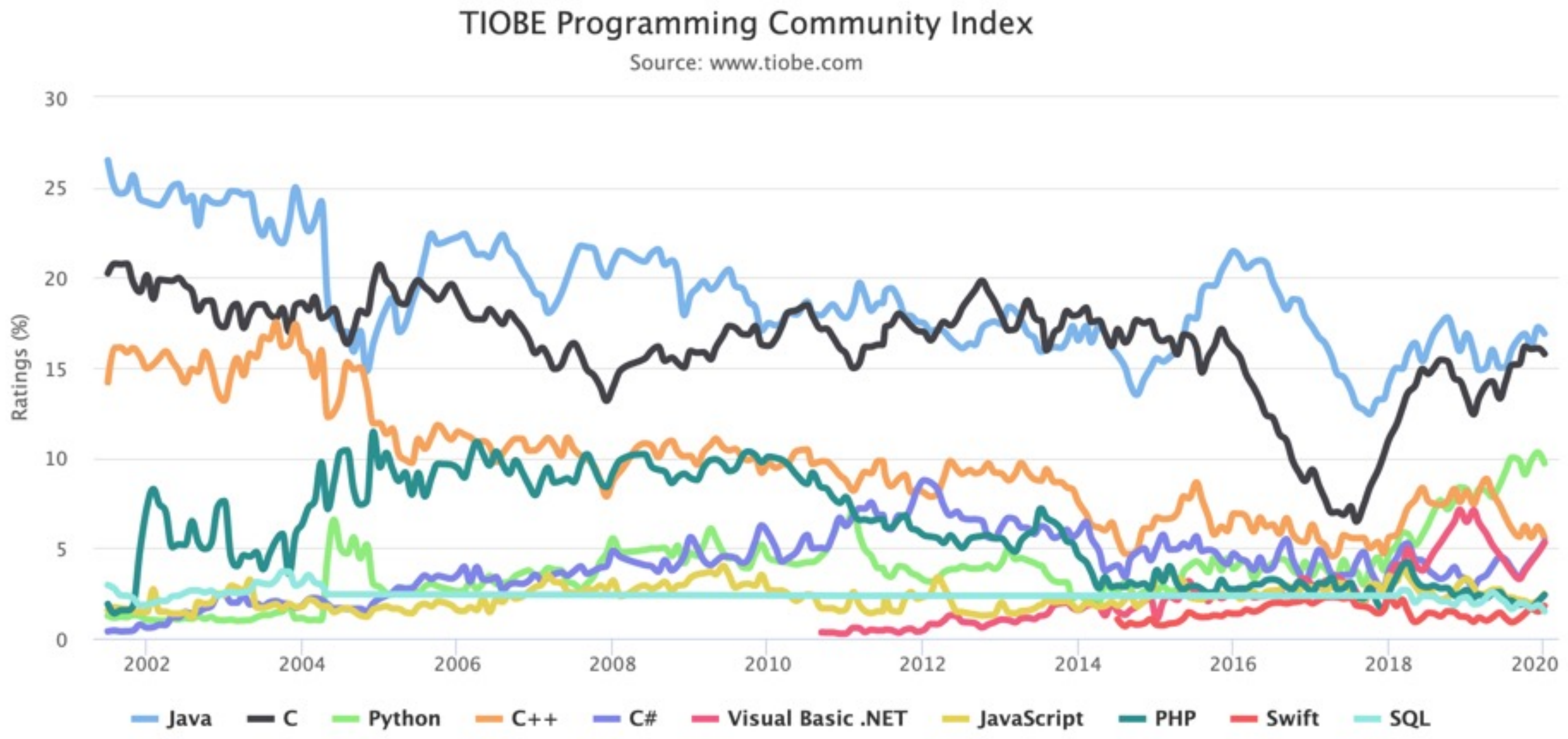
Python is also procedural, but dynamically typed: you still write functions and call methods on objects, but the development process is very different.

Java is object-oriented: virtually everything is an object, and everything you write needs to conform to the object-oriented design pattern.

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

```
/*
 * 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

```
/*
 * 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;
}
```

Question Break!

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

$\left.\begin{array}{l} \text{emacs} \\ \text{make} \end{array}\right\}$ Now

$\left.\begin{array}{l} \text{gdb} \\ \text{valgrind} \end{array}\right\}$ 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

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/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.

Question Break!

Get up and running with our guide:

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

Assign0

Assignment 0 (Intro to Unix and C) is already released on the course website and is due in one week on **Mon. 6/28 at 11:59PM PT**.

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

Lectures 2 and 3

Videos/quizzes on Canvas, slides on course website.

- Complete Lecture 2 quizzes by **Friday 6/25 1:00pm PDT**
- Lecture 2 review in-class on Friday 6/25

- Complete Lecture 3 quiz by **Monday 6/28 1:00pm PDT**
- Lecture 3 review in-class on Monday 6/28

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 page, information about the Honor Code in CS107, and more about CS107 course policies and logistics.

We're looking forward to an awesome quarter!

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?*