

# COMP201

## Computer Systems & Programming

Lecture #01 – Introduction



**KOÇ**  
**UNIVERSITY**

Aykut Erdem // Koç University // Spring 2021

# A little about me...

Koç University  
Associate Professor  
2020-now



Hacettepe University  
Associate Professor  
2010-2020



Università Ca' Foscari di Venezia  
Post-doctoral Researcher  
2008-2010



- I explore better ways to understand, interpret and manipulate visual data.

Middle East Technical University  
1997-2008  
Ph.D., 2008  
M.Sc., 2003  
B.Sc., 2001



MIT  
Fall 2007  
Visiting Student



VirginiaTech  
Visiting Research Scholar  
Summer 2006



- My research interests span a diverse set of topics, ranging from image editing to visual saliency estimation, and to multimodal learning for integrated vision and language.



<https://aykuterdem.github.io>

# Plan For Today

- Course Introduction
- COMP201 Course Policies
- Unix and the Command Line
- Getting Started With C

**Disclaimer:** Slides for this lecture were borrowed from  
—Nick Troccoli's Stanford CS107 class

# COMP201 on Zoom

- You are encouraged to turn on your cameras in order to keep human touch with others and not to feel alone, but this is clearly not a must!
- I will upload the slides prior to the class so that you can take a look/study the lecture material before the class. We'll take regular question breaks to address your questions.
- In Zoom, everyone is muted by default. However, during the question breaks, you can unmute yourself or alternatively use Zoom chat to ask your questions.



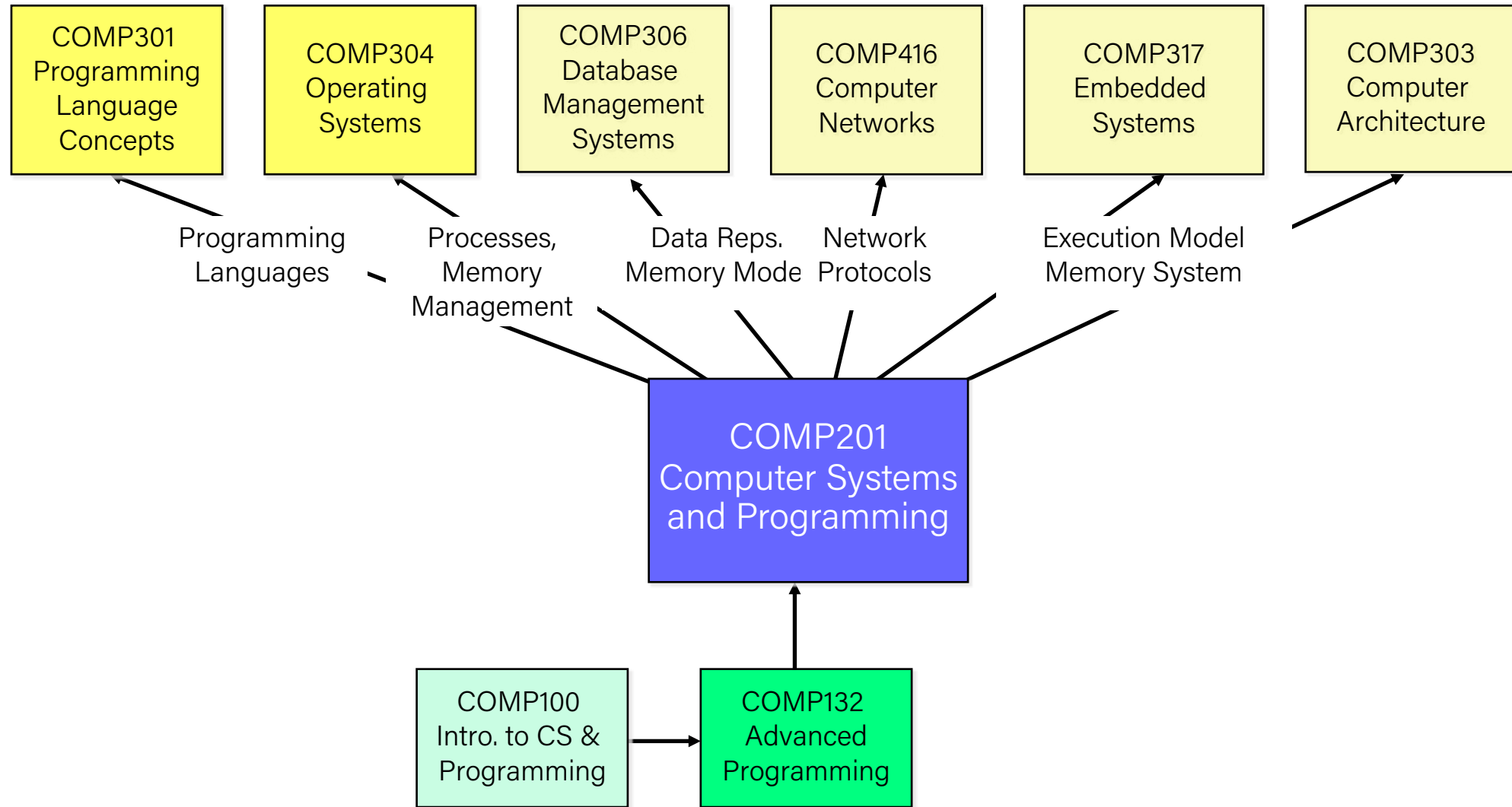
# Lecture Plan

- Course Introduction
- COMP201 Course Policies
- Unix and the Command Line
- Getting Started With C

# What is COMP201?

- A new addition to the COMP curriculum, designed from ground-up.
- The third course in the line of COMP's introductory programming courses (COMP100, COMP132, and COMP201)
  - COMP100 teaches you the notion of computational thinking and how to solve problems as a programmer (using Python)
  - COMP132 introduces you object-oriented programming paradigm (using Java)
- COMP201 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.

# Role within COMP Curriculum



# What is COMP201?



# Computer Systems and Programming

- 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
- How **cache memories work** and **how to exploit them** to improve the performance of your programs
- Uses the **C** programming language
- Programming **style** and software development practices



# COMP201 Learning Goals

The goals for COMP201 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
- finding bottlenecks and improving runtime performance
- working effectively in a Unix development environment

and have **exposure** to

- a working understanding of the basics of cache memories

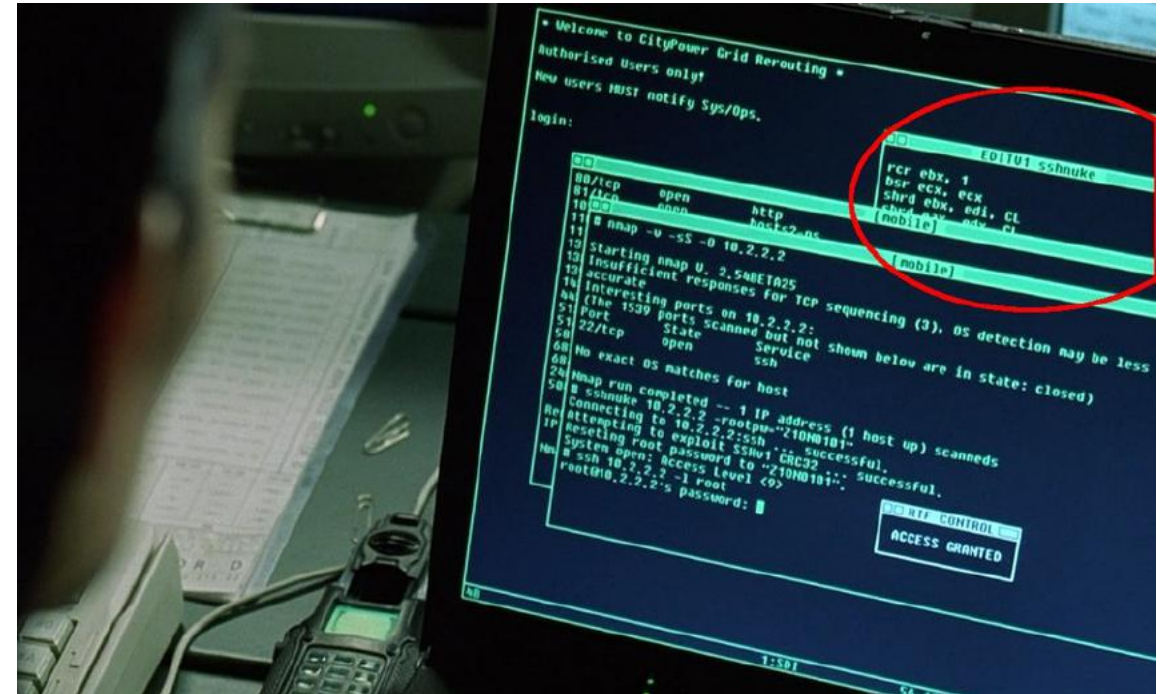


# COMP201 Learning Goals

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



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



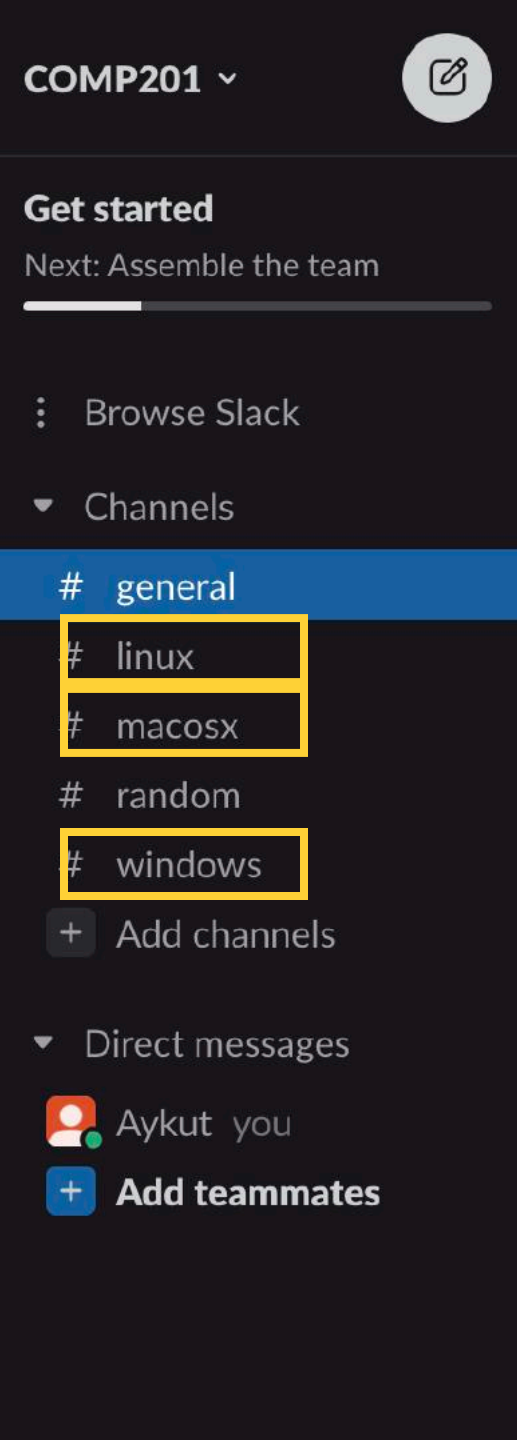
Trinity saving the world by  
hacking into the power grid  
using Nmap Network Scanning  
*The Matrix Reloaded, 2003*

# 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. **The Memory Hierarchy** - *How does the memory system is organized as a hierarchy of different storage devices with unique capacities*
7. **The Heap Allocators** - *How do core memory-allocation operations like malloc and free work?*

# COMP201 - The Online Edition

- This quarter, we will try our best to make your COMP201 experience as close as the face-to-face teaching version.
- **We are working to emphasize community and connection.**
- We understand that the pandemic poses unique challenges for each and every one of us involved.
- We will constantly evaluate and listen to ensure the class is going as smoothly as possible for everyone.
- Please don't hesitate to reach out if you want our support. We are always here for you.

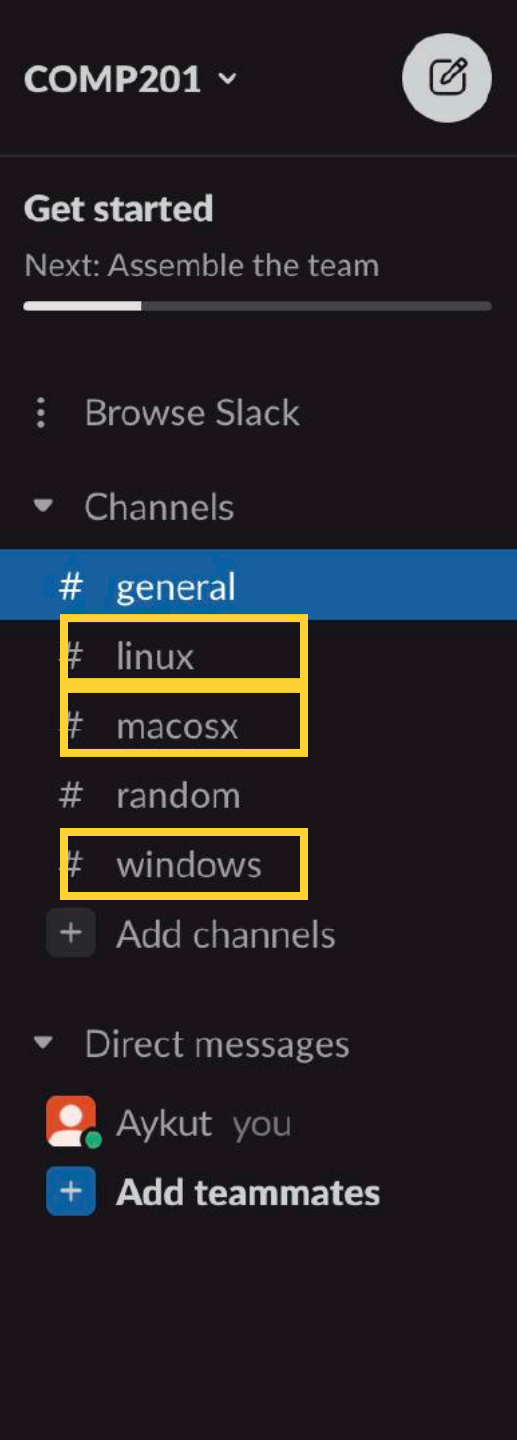


# COMP201 Slack Workspace



<https://join.slack.com/t/comp201/signup>





COMP2

## Slack Developer Community Code of Conduct

This code of conduct governs Slack Platform's Community events and discussions.

### Introduction

- Diversity and inclusion make our community strong. We encourage participation from the most varied and diverse backgrounds possible and want to be very clear about where we stand.
- Our goal is to maintain a safe, helpful and friendly community for everyone, regardless of experience, gender identity and expression, sexual orientation, disability, personal appearance, body size, race, ethnicity, age, religion, nationality, or other defining characteristic.
- This code and related procedures apply to unacceptable behavior occurring in all community venues, including behavior outside the scope of community activities — online and in-person— as well as in all one-on-one communications, and anywhere such behavior has the potential to adversely affect the safety and well-being of community members.

### Expected Behavior

- Be welcoming.
- Be kind.
- Look out for each other.

### Unacceptable Behavior

- Conduct or speech which might be considered sexist, racist, homophobic, transphobic, ableist or otherwise discriminatory or offensive in nature.
  - Do not use unwelcome, suggestive, derogatory or inappropriate nicknames or terms.
  - Do not show disrespect towards others. (Jokes, innuendo, dismissive attitudes.)
- Intimidation or harassment (online or in-person). Please read the [Citizen Code of Conduct](#) for how we interpret harassment.
- Disrespect towards differences of opinion.
- Inappropriate attention or contact. Be aware of how your actions affect others. If it

[https](https://slack.com/docs/developer-community/code-of-conduct)

[nup](https://slack.com/docs/developer-community/code-of-conduct)

# Teaching Team



Aykut Erdem



Ahmed Imam Shah



Samet Demir



Farzin Negahbani

# Course Website

<https://aykuterdem.github.io/classes/comp201/>

\*lecture videos on Panopto – can be accessed through Blackboard or from the course webpage

Question Break!

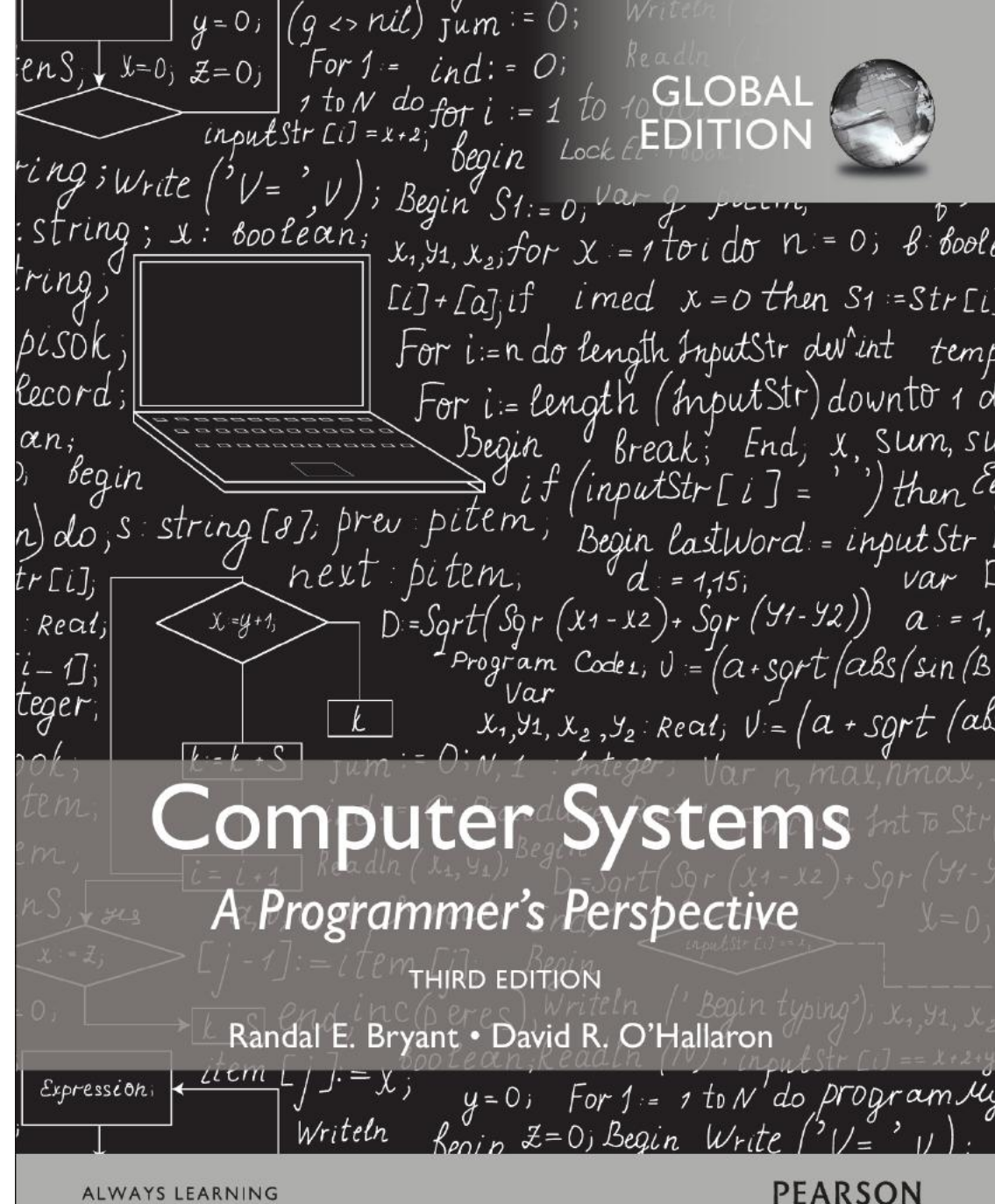
# Lecture Plan

- Introduction
- **COMP201 Course Policies**
- Unix and the Command Line
- Getting Started With C



# Textbooks

- *Computer Systems: A Programmer's Perspective* by Bryant & O'Hallaron, 3<sup>rd</sup> Edition
  - 3<sup>rd</sup> edition matters – important updates to course materials
- A C programming reference of your choice
  - *The C Programming Language* by Kernighan and Ritchie
  - Other C programming books, websites, or reference sheets



# Course Structure

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

Monday	Wednesday	Friday
Lecture	Lecture	Lab-A
		Lab-B
		Lab-C

- **assg0:** out later on Wednesday, due Feb 24
- **C bootcamp:** this week (during lab hours)

# Grading

38%	Programming assignments
21%	Labs
20%	Midterm exam
21%	Final exam

# Grading

38%      Programming assignments

21%      Labs

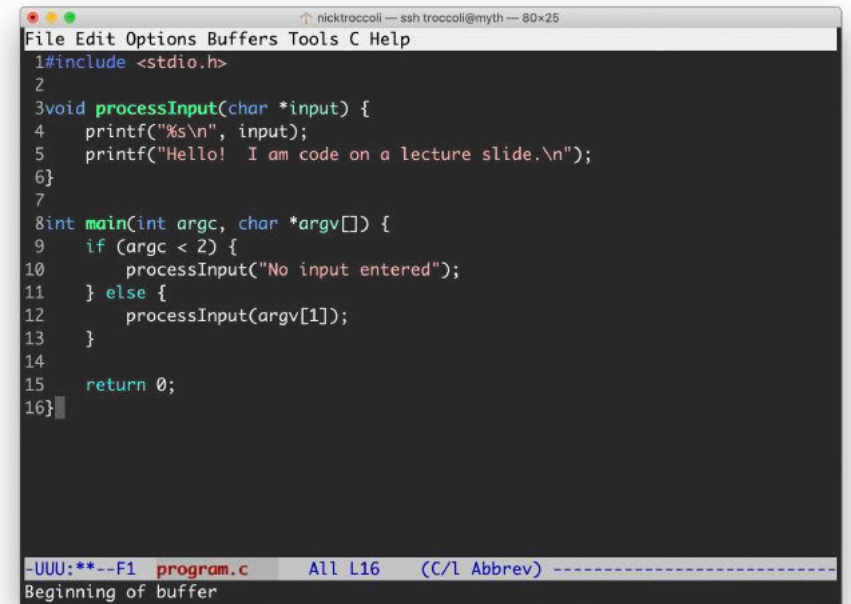
20%      Midterm exam

21%      Final exam

# Assignments

- 7 programming assignments completed individually using **Unix command line tools**
  - Free software, pre-installed on linuxpool cluster dedicated to COMP students
  - GitHub Classroom + Repl.it
  - 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,
  - Grades returned via Blackboard

GitHub Classroom

A screenshot of a terminal window showing a C program. The window title is "nicktroccoli — ssh troccoli@myth — 80x25". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "C", and "Help". The code is as follows:

```
1#include <stdio.h>
2
3void processInput(char *input) {
4    printf("%s\n", input);
5    printf("Hello! I am code on a lecture slide.\n");
6}
7
8int main(int argc, char *argv[]) {
9    if (argc < 2) {
10        processInput("No input entered");
11    } else {
12        processInput(argv[1]);
13    }
14
15    return 0;
16}
```

The status bar at the bottom shows "-UUU:\*\*--F1 program.c All L16 (C/l Abbrev) -----" and "Beginning of buffer".



# Late Policy

- **Start out with 5 *grace days*:** each late day allows you to submit an assignment without penalty if you have free grace days left.
- **Hard deadline:** No submissions will be accepted **48 hours** after the original due date
- Penalty per day after grace days are exhausted
  - 1 day: 20% off
  - 2 days: 40% off

Question Break!

# Grading

38%	Programming assignments
21%	Labs
20%	Midterm exam
21%	Final exam

# Lab Sections

- Weekly 110-minute labs led by a TA, starting *next* week, offered on Fridays.
- Hands-on practice with lecture material and course topics.



- Graded on attendance + participation (*verified by submitting work at the end*)
  - Your lowest 3 scores will be dropped, hence there will be no make-up
- Lab preference submissions will be open **Tuesday 2/16 at 5PM** and **are not first-come first-serve**. You may submit your preferences anytime until **Friday 2/19 at 5PM**. Sign up info will be posted on Blackboard

Question Break!



# Grading

38%	Programming assignments
21%	Labs
20%	Midterm exam
21%	Final exam

# Midterm and Final Exams

- Online exams through Blackboard
  - Midterm: At week 9, date and time will be announced later
  - Final: Date and time will be announced later
- The exams will include multiple-choice questions as well as open-ended fill-in-the-blanks or short answer type questions.
- The exams will be released in multiple sessions, in which each student is required to complete each session in a limited time.

Question Break!

# Grading

38%	Programming assignments
21%	Labs
20%	Midterm exam
21%	Final exam

**NOTE:** Once the final letter grades are determined, a student can choose to be graded with "S" grade if the final grade is "D" or above.

Question Break!

# 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 **Online Office Hours**
  - More info to come soon!
- **Email the Course Staff**
  - Best for **private matters** (e.g. grading questions).

# Koç University Honor Code

- For assignments/quizzes/exams students should be required to digitally add and approve a version of the agreement below.

*I hereby declare that I have completed this examination individually, without support from anyone else.*

*I hereby accept that only the below listed sources are approved to be used during this open-source examination:*

*(i) Coursebook,*

*(ii) All material that is made available to students via Blackboard for this course, and*

*(iii) Notes taken by me during lectures.*

*I have not used, accessed or taken any unpermitted information from any other source. Hence, all effort belongs to me.*



# Honor Code and COMP201

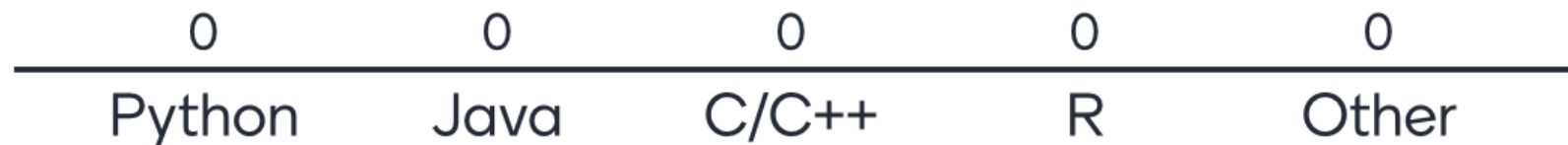
- Please help us ensure academic integrity:
  - Indicate any assistance received on HW (books, friends, etc.).
  - Do not look at other people's solution code or answers
  - Do not give your solutions to others or post them on the web or to the forum.
  - Report any inappropriate activity you see performed by others.
- Assignments are checked regularly for similarity with help of automated 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. Come to use before we come for you.
- 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.

Question Break!

# Poll Time

Go to [www.menti.com](https://www.menti.com) and use the code 92 17 52 7

# What is your favorite programming language?

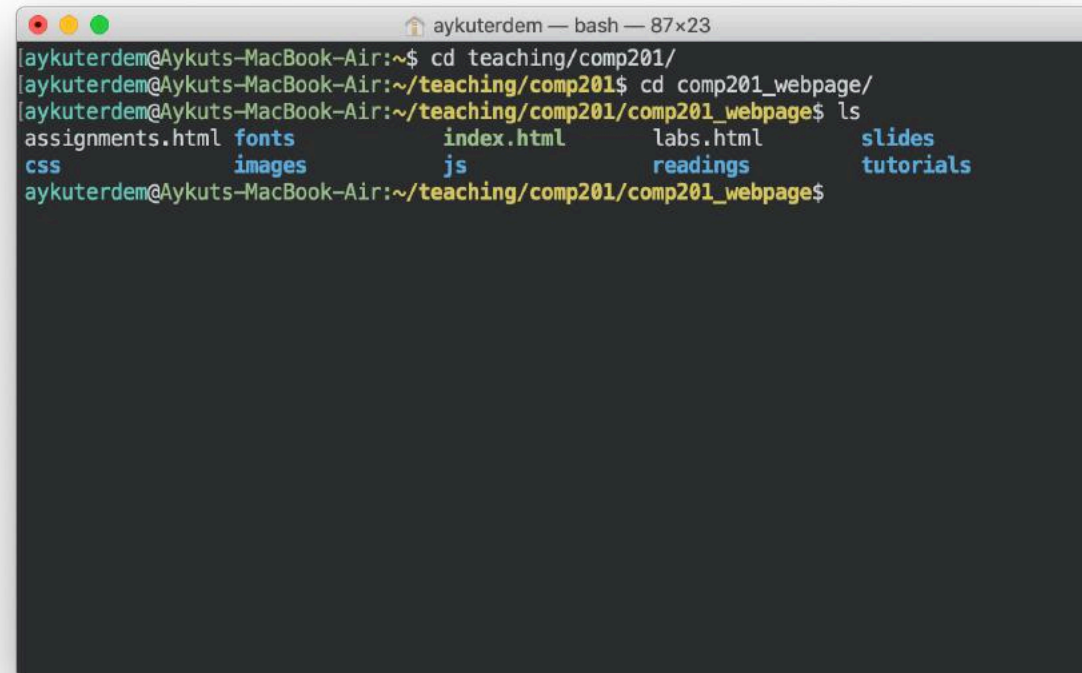


# Lecture Plan

- Introduction
- COMP201 Course Policies
- Unix and the Command Line
- Getting Started With C

# What is Unix?

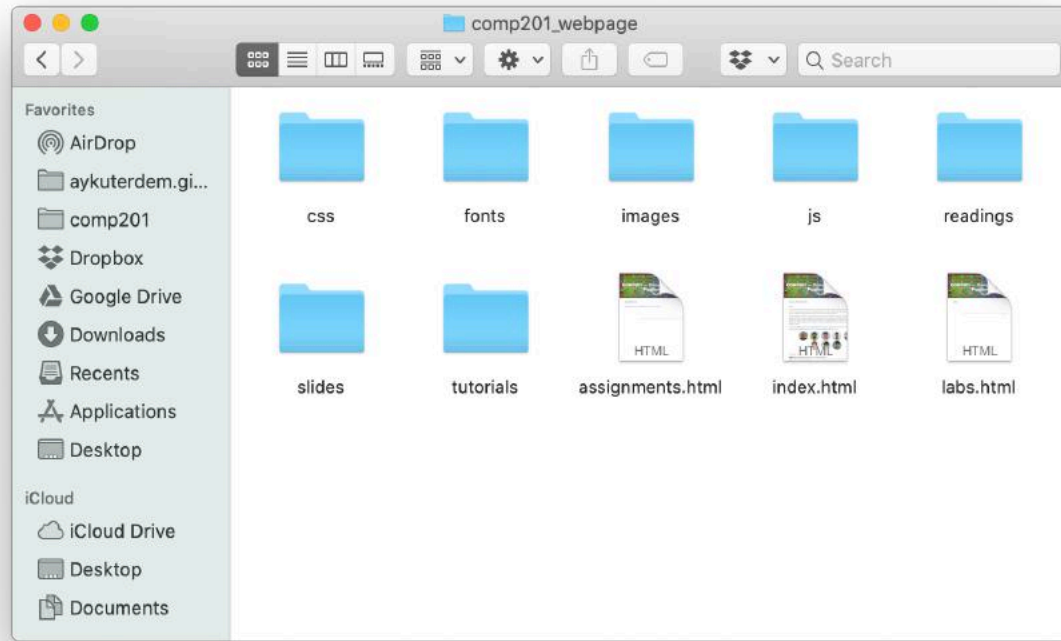
- **Unix**: a set of standards and tools commonly used in software development.
  - **macOS** and **Linux** are operating systems built on top of Unix
- You can navigate a Unix system using the **command line** ("terminal")
- Every Unix system works with the same tools and commands



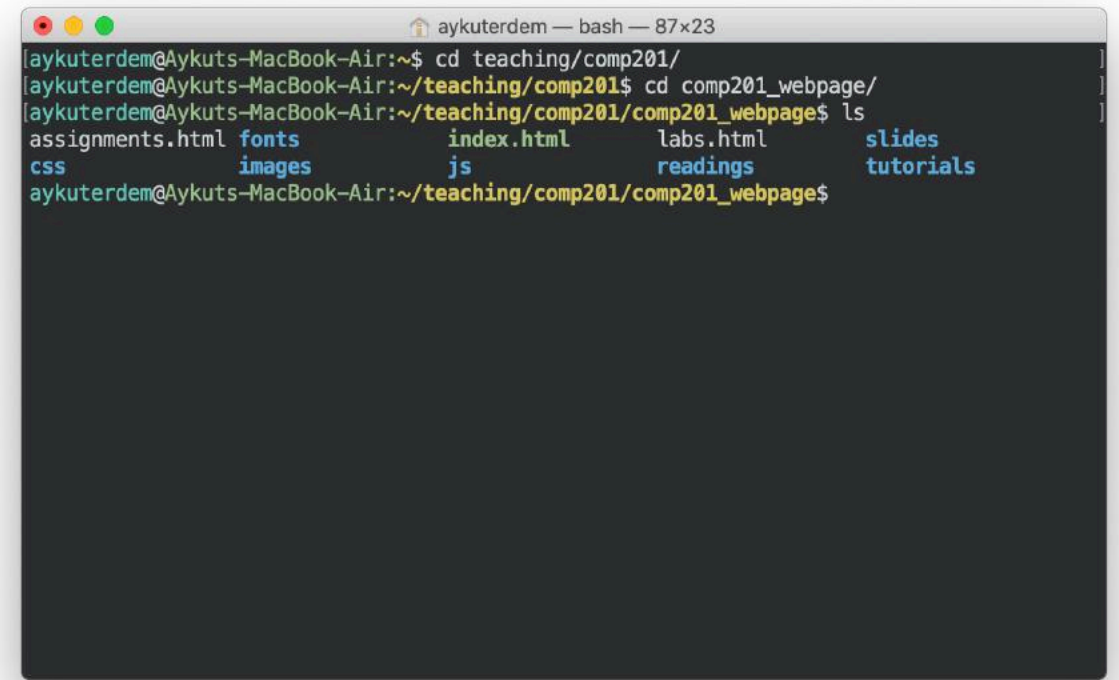
```
aykuterdem — bash — 87x23
aykuterdem@Aykuts-MacBook-Air:~$ cd teaching/comp201/
aykuterdem@Aykuts-MacBook-Air:~/teaching/comp201$ cd comp201_webpage/
aykuterdem@Aykuts-MacBook-Air:~/teaching/comp201/comp201_webpage$ ls
assignments.html  fonts          index.html     labs.html      slides
css               images         js             readings       tutorials
aykuterdem@Aykuts-MacBook-Air:~/teaching/comp201/comp201_webpage$
```

# 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



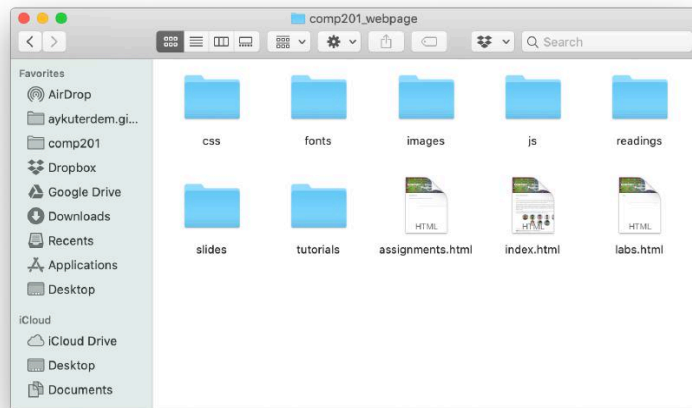
Text-based interface



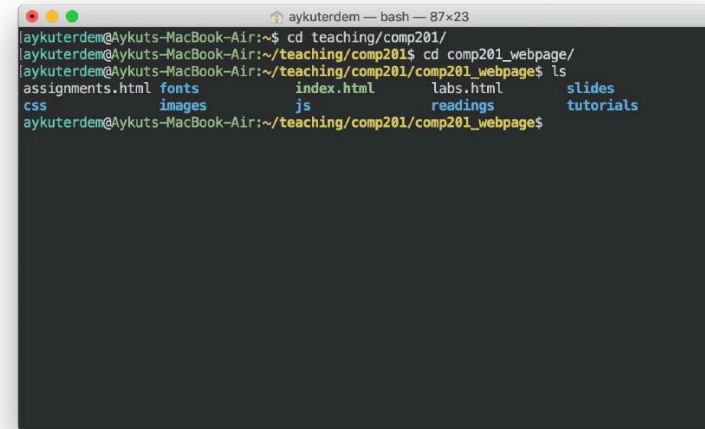
# Command Line vs. GUI

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



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.

# Demo: Using Unix and the Command Line



# Unix Commands Recap

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



**Lab 1:**  
The Linux Shell  
(*next week*)

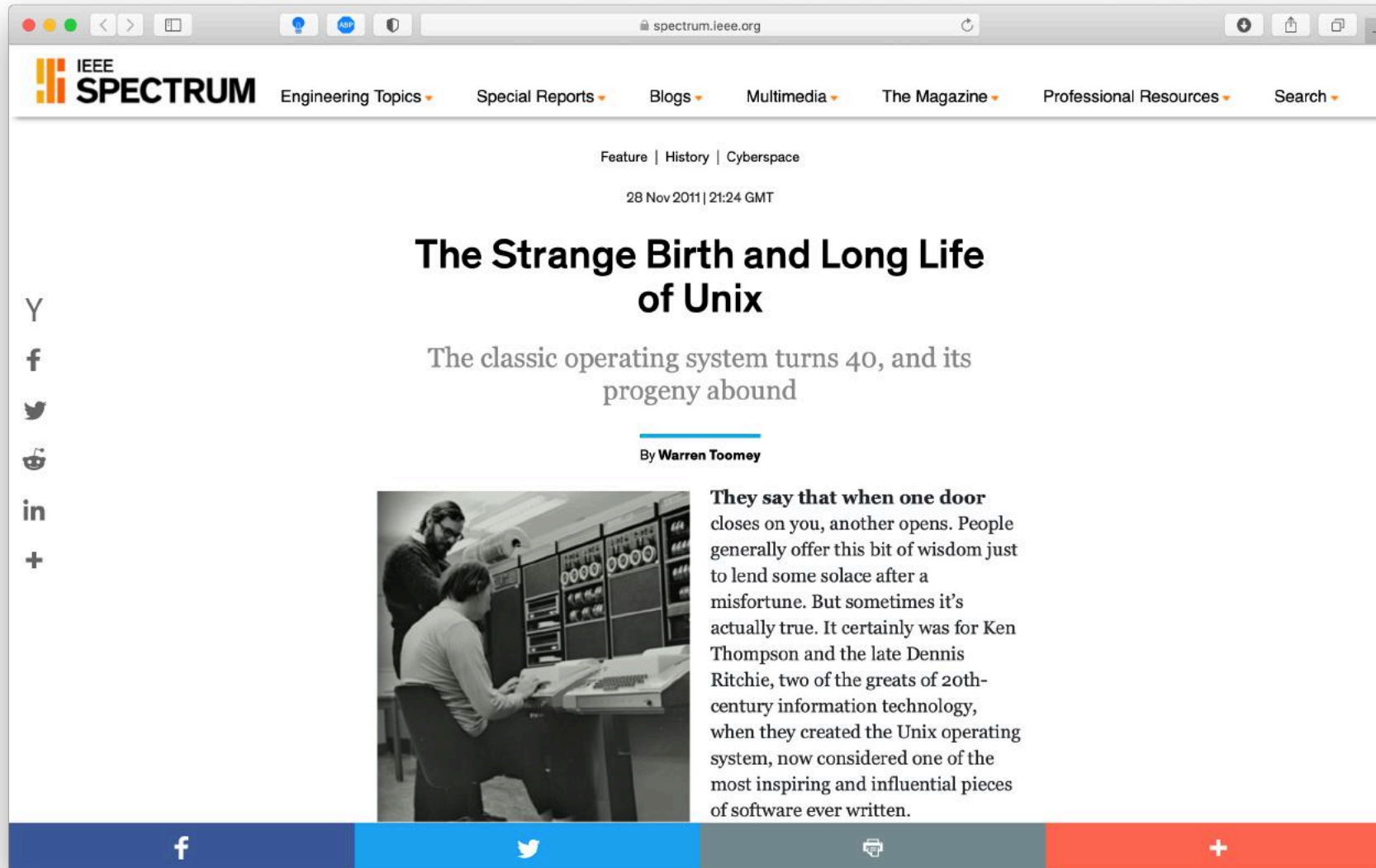
See the Resources page of the course website for more commands, and a complete reference.

# 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** page 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!

# Additional Reading



<https://spectrum.ieee.org/tech-history/cyberspace/the-strange-birth-and-long-life-of-unix>



# Lecture Plan

- Introduction
- COMP201 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 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.

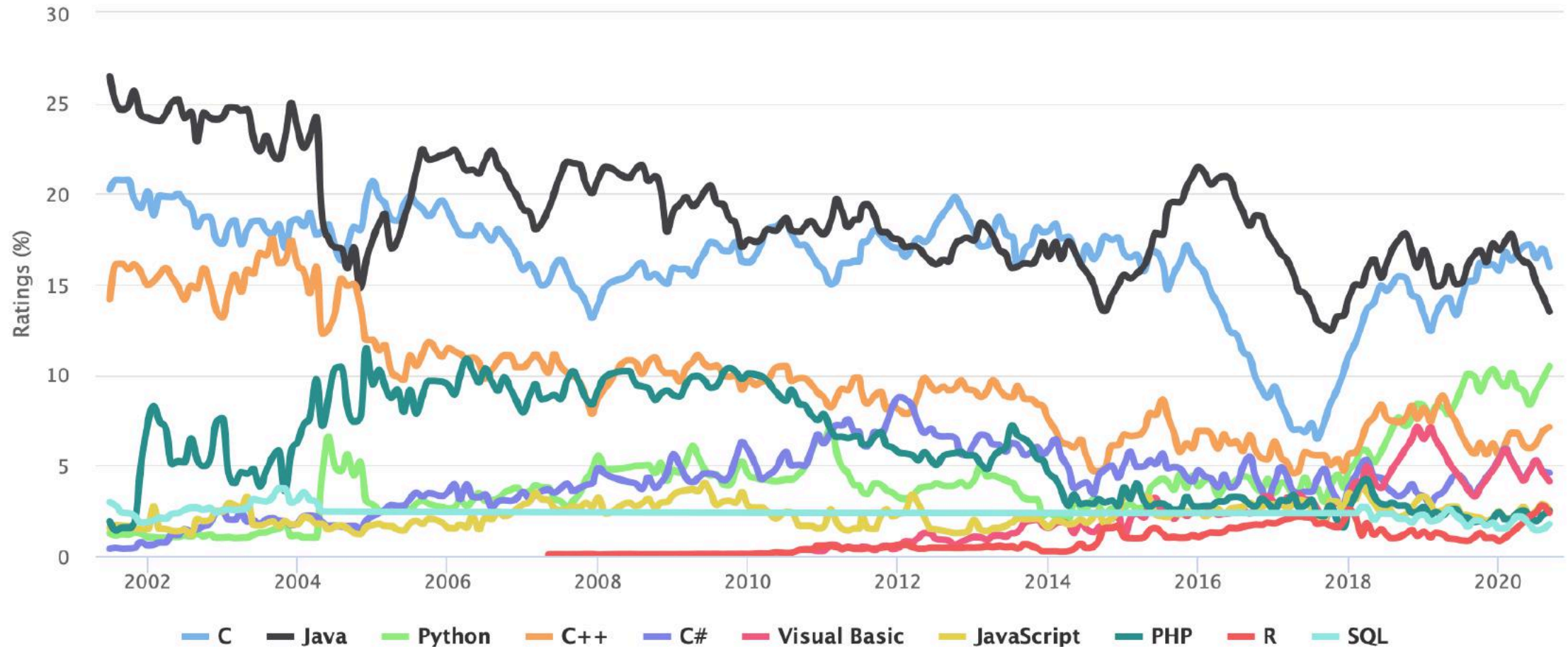
# Why C?

- 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

## TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



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

# 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;  
}
```



# 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;  
}
```

## Program comments

You can write block or inline comments.

# 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;  
}
```

## Import statements

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

# 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 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

# 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;
}
```



# Console Output: printf

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

```
// Example
```

```
char *classPrefix = "COMP";
```

```
int classNumber = 201;
```

```
printf("You are in %s%d", classPrefix, classNumber);    // You are in COMP201
```

`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)

Question Break!

# Writing, Debugging and Compiling

We will use:

- the **vi/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

# Demo: Compiling And Running A C Program



# Working On C Programs Recap

- **ssh** – remotely log in to `linuxpool` computers (*later*)
- **Vi/Emacs** – text editor to write and edit C programs
  - Use the mouse to position cursor, scroll, and highlight text
  - `:w` / `Ctl-x Ctl-s` to save, `:q` / `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 codes are accessible at Blackboard

# Recap

- COMP201 is a programming class, which uses C to teach 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 regularly visit the course website, <https://aykuterdem.github.io/classes/comp201> and follow the announcements on Blackboard.
- **We're looking forward to an exciting quarter!**
- **Next time:** *How a computer represents integer numbers? What are the limitations?*

