CS 521: Systems Programming

Lecture 1

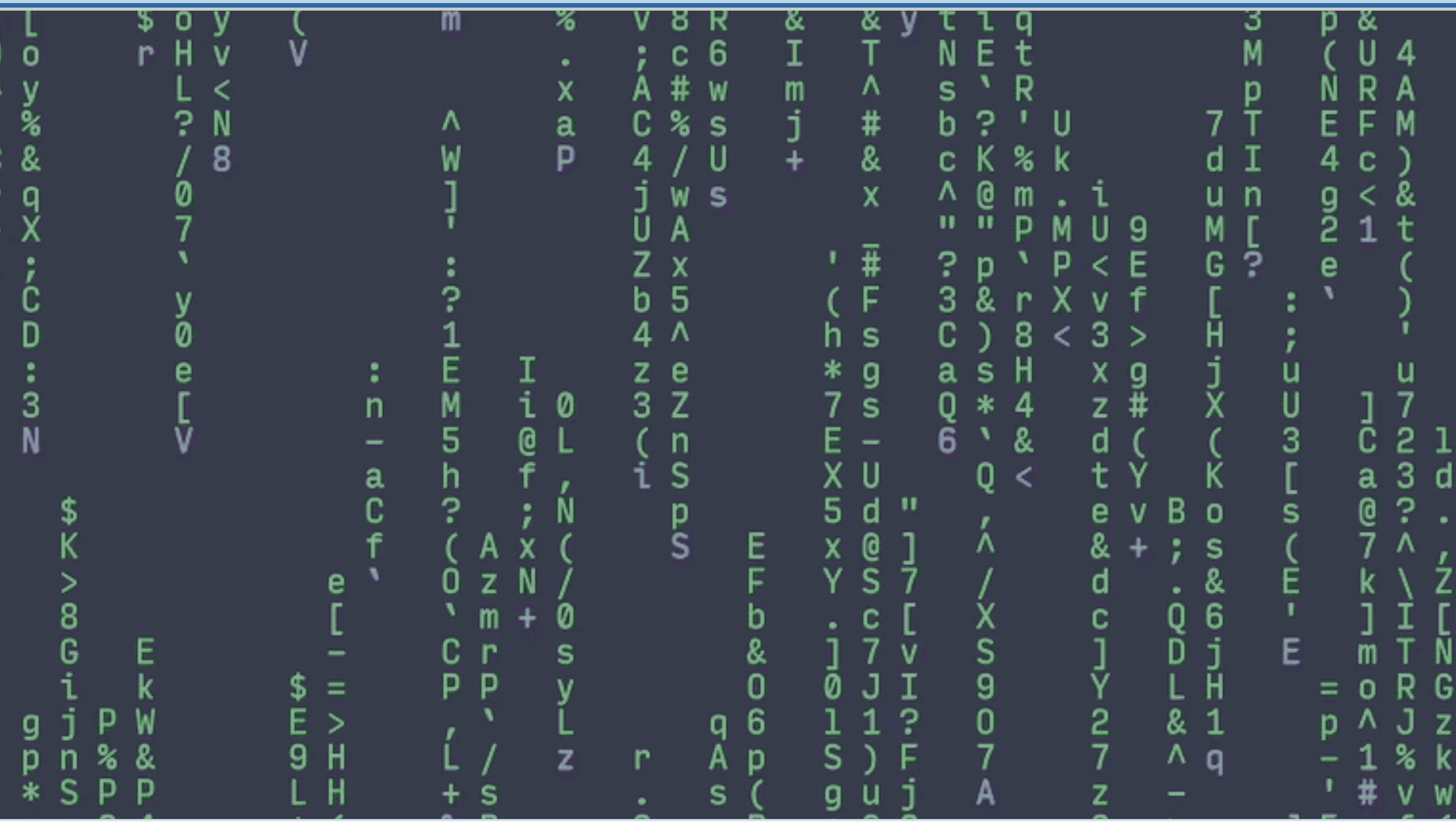
Welcome to CS 521

- Looking forward to a great semester!
- This class will be taught in a *hybrid* modality:
  - *On-Campus Lecture:* **M, W** · 10:30 – 11:35am · **ED** 103
  - *Remote Lecture:* **F** · 10:30 – 11:35am · Zoom Live Stream
- Staff:
  - **Instructor**: Matthew Malensek
    - mmalensek@usfca.edu
    - Office Hours: **M, W** 11:45am – 12:45pm · **HR** 407B
  - **TA**: Athene Marston
    - Office Hours: **Th** 5:00 – 6:30pm · **HR** CS Labs

What's this class about?

- How to develop and debug **systems** software
  - Software that is a *platform* for other software
  - Operating systems, command line utilities, server-side software
- Setting up and maintaining a UNIX development environment
  - How to work effectively from the UNIX command line
- Low-level information representation
- Designing and writing concurrent (multi-threaded) software
- Performance evaluation of systems software

So, basically:



Today's Agenda

- Syllabus
- Course Overview

Today's Agenda

- **Syllabus**
- Course Overview

Course Website

- https://www.cs.usfca.edu/~mmalensek/cs521
- Contains assignments, office hours, and lots of other helpful information
- I will regularly update the **schedule** page with:
  - Upcoming topics, readings, due dates
  - Lecture slides, videos
- Grades will be posted on Canvas
- Project submissions: GitHub

Communication

- If you need help, our main mode of communication is CampusWire
  - Q&A link on the website takes you there
  - Post questions! If you have a question, then everyone else probably has the same question. 
  - You can post anonymously if you wish
- You are also welcome to email the instructor or communicate with course staff via USFCS Slack.
  - But Campuswire will usually be the fastest way to get an answer

Quick Note: Prerequisites

- This is a bridge class; you should be in the first year of the bridge program.
- CS 514 with a grade of B or better
- Math 501 with a grade of B or better

Course Overview

1. Setting up a Linux development environment
2. Writing *idiomatic* code in C and Rust
   - Pointers, dynamic memory management, ownership and borrowing, memory representation and safety, data structures
3. Operating Systems: processes, system calls, the shell
4. Concurrency, parallel programming, networking
   - Threads, CUDA, GPU programming, sockets

Books

NOTE: These are **optional**.

- *The C Programming Language, 2nd edition*. Brian W. Kernighan and Dennis M. Ritchie. Prentice-Hall, 1988.
- *The Rust Programming Language, 2nd Edition*. Steve Klabnik and Carol Nichols. No Starch Press, 2023.

Course Structure

- Assignments:
  - Labs: assigned roughly once per week (~16)
  - Projects: larger, tie labs and other concepts together (~4)
- Assessment: quizzes every ~3 weeks
- In class: we'll do a lot of live coding and experimentation
  - I generally do not provide starter code for your projects, but we will develop it in class.

Grading

Score	Grade
100 – 93	A
92 – 90	A-
89 – 87	B+
86 – 83	B
82 – 80	B-
79 – 77	C+
76 – 73	C
72 – 70	C-
69 – 67	D+
66 – 63	D
62 – 60	D-
59 – 0	F

Grade Distribution

- Labs: 25%
- Projects: 25%
- Quizzes: 50%
  - Team: 5%
  - Individual: 45%

Lab Assignments

- Labs will be assigned each Friday during the Zoom lecture
  - Due in 10 days
  - Late policy: you can turn in any lab or project up to **7** days late for full credit, but no credit thereafter
- A grading specification will be included with each lab
  - Grading is straightforward: complete the required functionality for the grade you want.
- You must have your labs checked **in person**, interactively

Projects

- Projects work the same as labs, but they are larger in scope
  - Deadlines will be adjusted based on the scope
- You'll need more creativity for the projects; some parts will be left up to you to design.
- **Plan** out your design, work methodically, and debug as you go
  - You'll learn strategic debugging and defensive programming
  - Especially when working with C, you will find that many things take much longer to complete than you expect!

Quizzes

- Given roughly every three weeks
  - Makeup quizzes will not be administered **unless** arranged at least **one week** in advance.
- Helps us go back and review what we've covered
- The strategy to do well is:
  1. Take note of concepts that are demoed in class. These are most important.
  2. Review the slides.
  3. If something isn't clear, review the lecture videos and book chapters
- I will drop your lowest quiz score (including the *final*)

Group Quiz

- After the regular quiz, you'll take it again
  - this time as a group (probably 4-5 students)
- Good way to review and discuss the problems

Academic Honesty

- You are responsible for the code you turn in.
  - If you get help from an external source, you must provide attribution and fully understand what you are submitting.
- **Do not cheat**. Review the Honor Code, and if in doubt about whether or not something is cheating, ask the professor.
  - The course staff will run cheat detection software that includes past assignments.
  - "Collaboration" that involves sharing code/solutions is considered cheating.
  - If you cheat, **you will get a 0 on the assignment or an F in the class**.

Today's Agenda

- Syllabus
- **Course Overview**

Oh Say Can You C?

- Many of the assignments in this course will be in C
- The C programming language was invented around 1970
  - It's old.
  - Legend has it that Dennis Ritchie invented it while he was riding around in his horse-drawn carriage
- C can be tough, and it's not the friendliest language out there.

Why are we learning C?

- Most operating systems (the software that runs **your** software ) are written in C
  - Including Linux, macOS, Windows, *BSD, etc.
  - C-based APIs and calling conventions
- High performance but still reasonably high level (it's not *assembly code*, at least)
- Easy(ish) to manipulate bits, registers, and other low-level constructs, manage memory, and interface with hardware

What about Rust?

- In recent years, we've had a bit of a systems programming renaissance!
- Rust has emerged as one of the most important languages in this space
  - It provides many of the advantages of C while also being ***memory safe*** and eliminating bugs that plague C programs
    - (especially ***security bugs***)
  - It's mature, well-supported, and slowly making its way into the Linux and Windows operating systems
- There are other options, like **Zig**. But they don't have quite the same traction… yet?

But does that apply to me?

- You might not see yourself writing systems software as your ideal career, so why even take this class?!
- I can sell you two big concrete reasons:
  - Maybe you love Python (or some other language) but occasionally it's just too slow. Rewrite the slow part in Rust or C, keep using your favorite language for the rest
  - Knowing how things work at a low level gives you a **huge** advantage when designing software, even if you're working at a higher level
- And… you may just find out that you love systems software!
  - It could happen! 

TIOBE Language Rankings

Indisputable proof that C is the best:

Jan 2021	Jan 2020	Change	Programming Language	Ratings	Change
1	2	^	C	17.38%	+1.61%
2	1	v	Java	11.96%	-4.93%
3	3		Python	11.72%	+2.01%
4	4		C++	7.56%	+1.99%
5	5		C#	3.95%	-1.40%

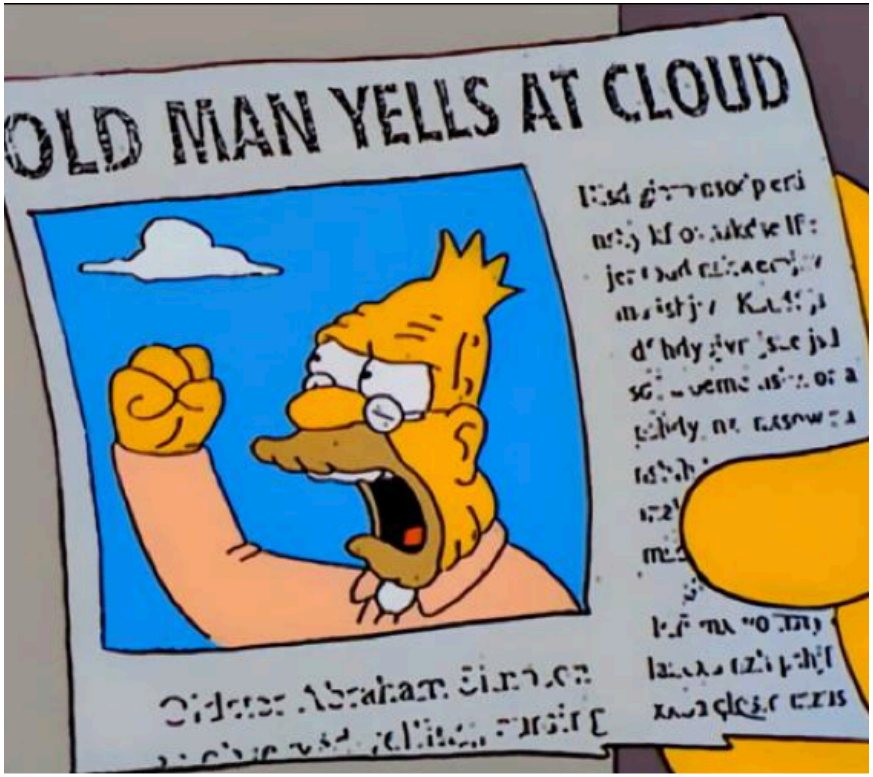
(see https://www.tiobe.com/tiobe-index/)

Virtual Machines

- We'll be using Linux virtual machines in CS 521
  - Everybody gets a shiny new VM of their own!
- VMs will be hosted on `gojira` in the CS network and run **Arch Linux**
- If you're not familiar with Linux or the command line, this will be a great way to learn.

Why VMs? [1/2]

- We'll all have the same configuration and can help each other out
- Regardless of whether you use a Mac, Windows, or Linux machine
- C code that works on one machine is **NOT** guaranteed to work elsewhere (more on that later…)



Why VMs? [2/2]

- It's how business is done today
  - When you start working at Google, Amazon, Hooli, etc. you're going to spend a lot of time in a Linux VM
- Knowing your way around Linux and the command line will pay off
- You'll be able to access your VM (and work on your projects) from anywhere in the world
  - You can't escape!

One Last Thing

- We need to make sure you can log into the CS department Linux machines
- If you are in a CS classroom or the CS labs, you can connect to `CSLabs` wifi and use `ssh` to connect directly to our ***jump host***, `stargate`.
- If you're anywhere else, you'll need to install the USF VPN and connect to it first
- Then: `ssh your-user-name@stargate.cs.usfca.edu`
- Let's see how many people we can get logged in…
  - and if that works, we can do a small activity!