Welcome to CMPT 201
Systems Programming

Topics

1) Introductions
2) What is Systems Programming?
3) Course overview
4) Demo of coding environment

Who’s Joe Lunderville?

- Joe or Joseph, he/they
- I’m a sessional instructor!
- Grad student, working in quantum computing
- But I have *lots* of professional experience
  - 20ish years of C++, C#, Python, BASH, JS, assembler…
  - Eclectic career: web, info systems, mobile, build engineering, media, embedded, now research…?
- I don’t like doing things in order

We Are Still Sorting Out Logistics

- The course website is still showing information from Fall!
- This is mostly accurate, because I'm teaching the same material…
- But the dates are all wrong. They are wrong in CourSys as well.
- They should be fixed before the next lecture.

What is Systems Programming?

Systems Programming

- Systems Programming
  - Low-level programming that directly interacts with hardware or the OS.
- Languages Used
  - Need ability to manage raw memory access and other low level tasks.
  - Ex: C, C++, Rust
  - Python and Java don't allow you to do that.

Systems Programming Perspectives...

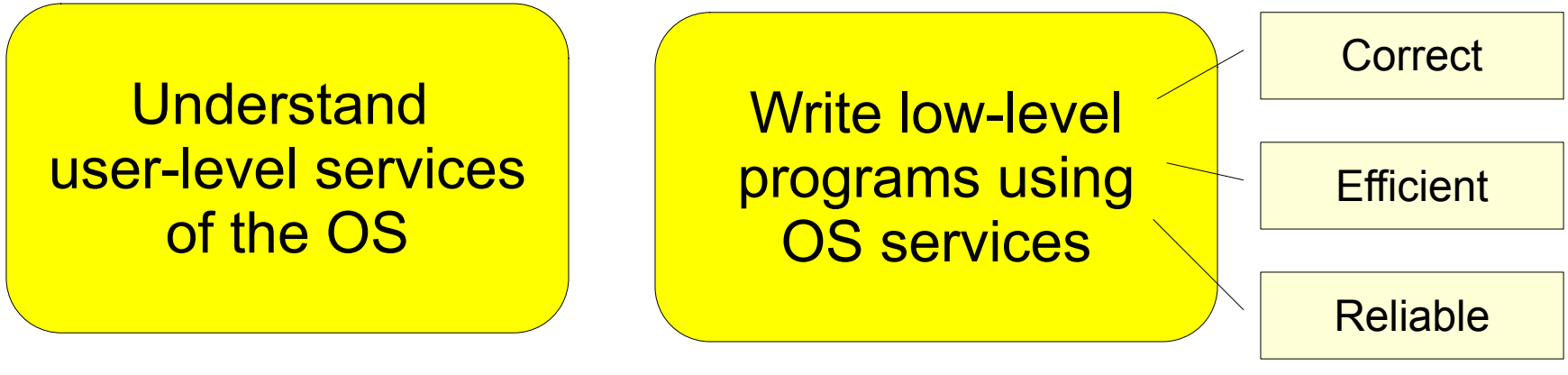
- The preceding explanation is pretty... fine
  - But I think we kind of beg the question: what is "low level"?
- I have slightly different perspective:
  - The parts of software that are *most common*
- And another:
  - What you need to explain program behaviours that *aren't part of the high-level abstraction*

Systems Programming

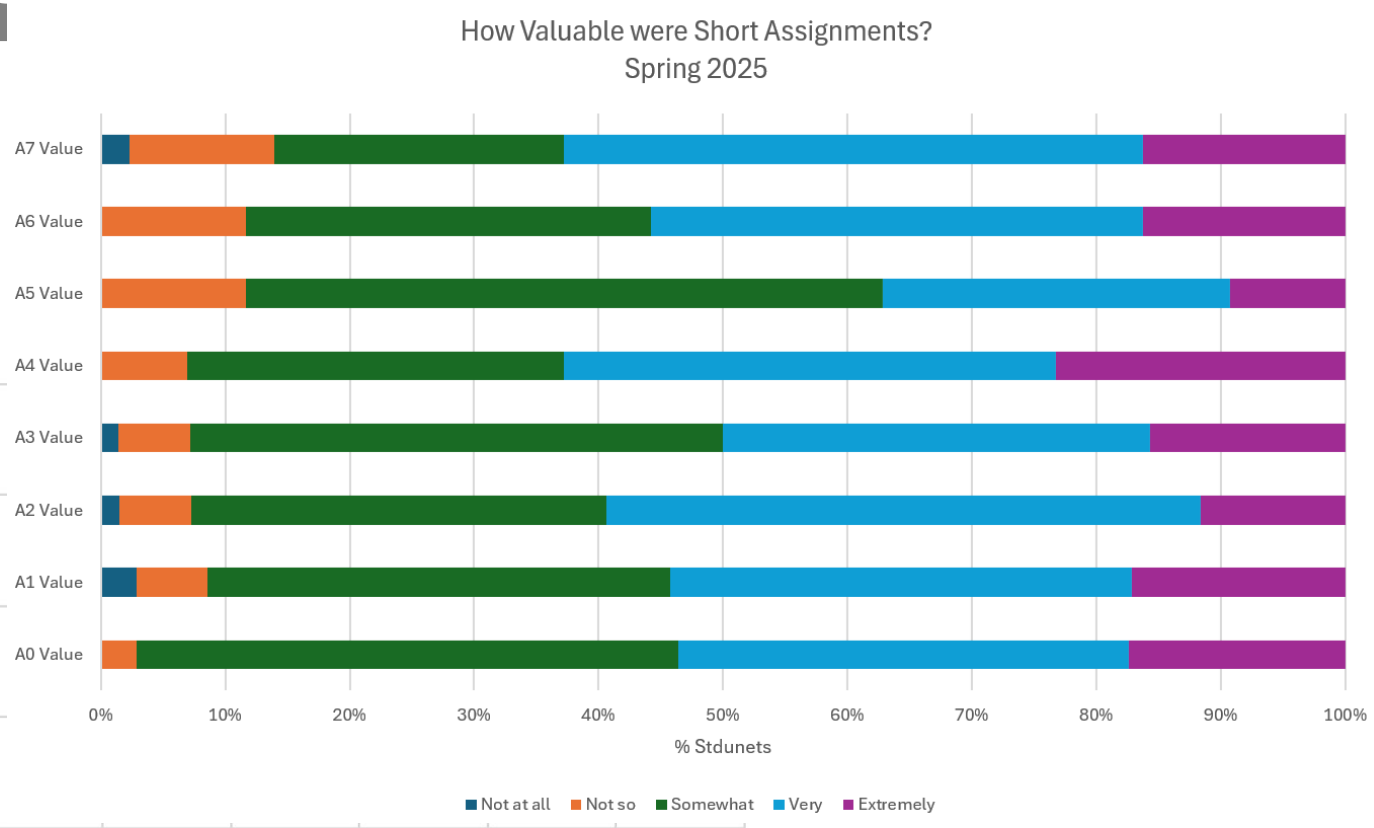
- We will be working with a lot of simple data structures and simple algorithms, and focusing on the interaction with the OS
  - We will also see what it looks like when they fail
  - Just under the surface of any significant application or language runtime
- Useful even if you don’t write this code yourself
  - It will help you understand what’s possible when designing at a higher level
  - It will help you stay oriented when things fail

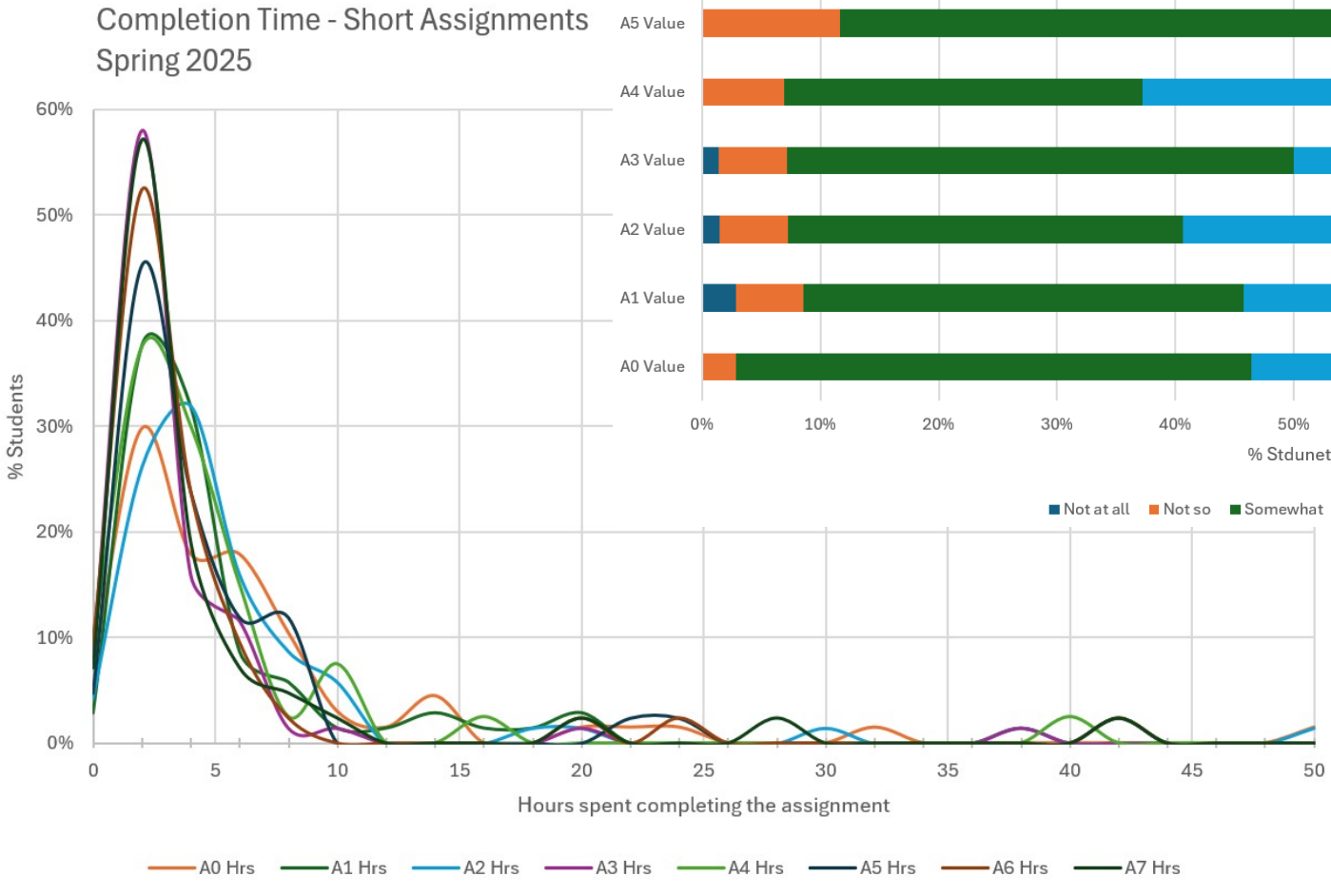
Course Overview

- Goal
  - Be a confident developer with low-level OS services.
  - Course is very applied
    - May spend hours solving build issues, and debugging complex behaviour.
- Course Components



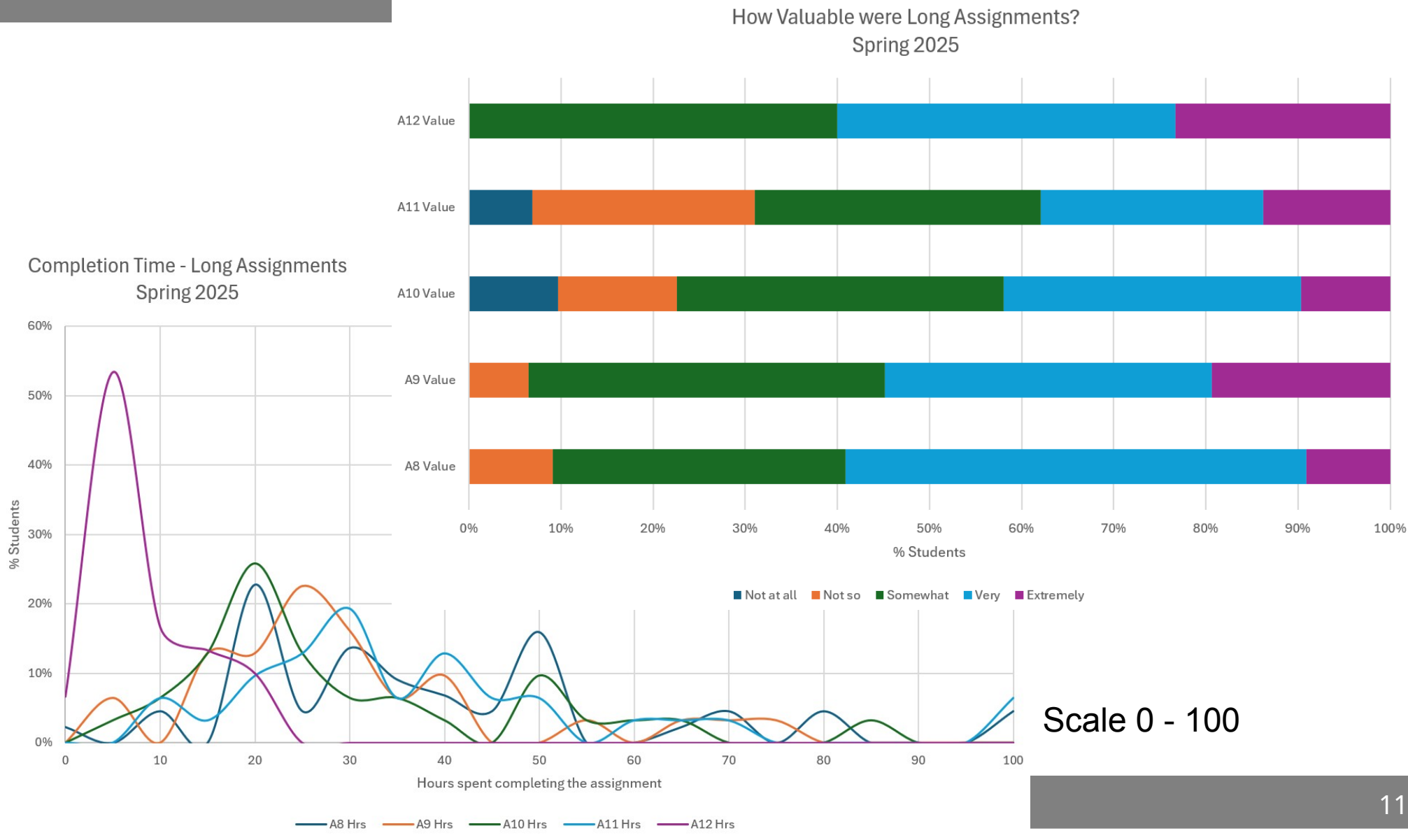
What to expect: **Short** Assignments





Scale 0 - 50

What to expect: **Long** Assignments



Scale 0 - 100

What to expect

- Course is polarizing

Really love it!

The assignments really helped solidify my understanding of concepts

..teaching is very engaging and informative

(In long assignments,) you're given a spec, some general pointers and guidelines, and then basically told figure it out. (These) were, hard, yes, but super gratifying to do.

Really hate it!

This course feels like 6 credits worth.

The homework is completely unreasonable.

...assignments are just soo much tougher going beyond the lecture material

[Assignments] are a full time job that you cannot complete. They can take up to 40 hours easily then if you do not figure out one seg fault you just spent 40 hours in your week to get a 0 on a 10% assignment.

Topics

- Linux command-line interface (CLI), shell scripting, and basic development tools
- Processes and threads
- Memory management
- Virtual memory
- Scheduling
- Synchronization
- Storage and file system abstractions
- Communication abstractions such as IPC, sockets, and RPC
- Basics of OS security and cryptographic functions.

What you know

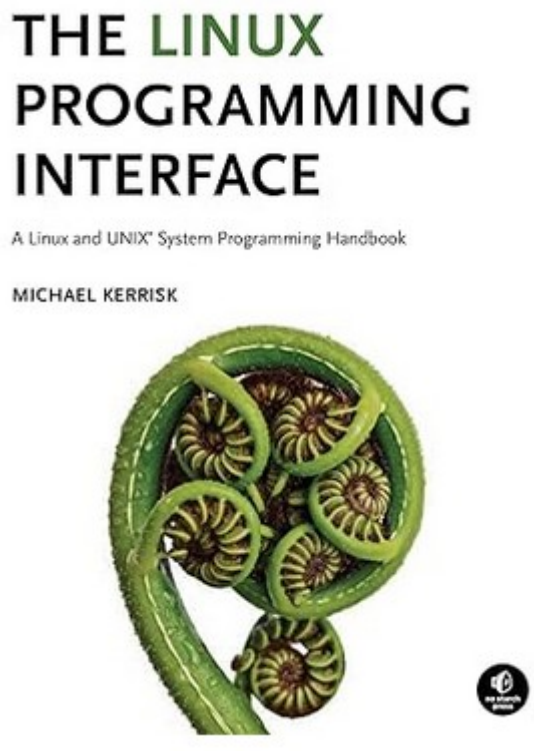
- From prereqs you know
  - Solid programming skills to write functions, loops, if, arrays, pointer, input/output.
  - Solid debugging skills to recognize defects and methodically track down errors.
  - Solid understanding of C or C++ programming.
- Passing prereq does not make this an easy course
  - Lower grades in prereq?
    or less familiar with above skills?
    Then expect to spend a lot of time becoming excellent this semester.
  - CMPT201 does not teach C and uses Linux terminal

Opinionated

- Course is opinionated about good coding.
  - Focuses on Linux command-line interface (CLI)
  - Uses C for low-level programming
    (we don’t teach C).
  - These are frequently missing skill in new grads,
    so we’ll learn it well here!
- You may not love this approach to coding,
  but it will expand your skills!

Book

- Highly recommended book
  The Linux Programming Interface:
  A Linux and UNIX System Programming Handbook,
  Michael Kerrisk, 2010
- Why Recommended?
  - Arguably the best book on systems programming using Linux.
  - Almost required: course draws on it
  - If struggling, then very highly recommended to read along during semester!



Admin Review

- Assessment
  - Midterm: 15%
  - Final: 15%
  - Programming assignments: 66%
    - 8 short (2% each), 5 long (10% each)
  - Labs: 4%
  - Grade breakpoints (“% for B+?”) may be non-standard
    - **Students must attain an overall passing grade on the weighted average of exams (quizzes/ midterms/final) in the course in order to obtain a clear pass (C- or better).**
- Academic Honesty
  - I am passionate about proving who did their own work.
  - Corollaries:
    - I'll give you credit for the work you do.
    - I'll catch those who don't do their own work.

Policies

- Late Policy
- Regrading Policy

Policies

- AI Tools Policy
  - When in doubt, don’t use them

Roles

- We play many roles
  - I’m also a student – I could be in your class
  - Here and now I’m your instructor
- My personal take: I recommend seeing the role you play as something that will change
  - I’ve been in many professional roles: junior, team lead, contractor – and then I came back here as an undergrad; it’s not humbling, it’s just a role I chose
  - Don’t make it your identity
  - But take it seriously: play that role effectively
  - If it’s the wrong role, find a different one

Roles: Teaching Staff

- I am here to facilitate your learning, as are the TAs
- We should be reasonably available and accommodating
  - The size of the class puts limits on this
  - So does the need for consistency
- Part of our role is to evaluate you
  - SFU has an external role: it promises, through your degree, that you have certain skills
  - I promise to SFU that I have evaluated you when I submit grades

Roles: Students

- As students, you have a specific role as well
- When you help each other learn and are respectful, everybody learns better
- When everyone participates honestly, the degree you get is more credible and more valuable
- It’s not just your right, it is your *responsibility* to ask for help

Roles: Students

- Participating honestly means not copying answers
  - For the labs we will be explicitly encouraging you to work together
  - On assignments, talk concepts but don't reuse code snippets
- This also means not using AI tools
  - In principle I don't have a problem with them
  - But that's not the skill we're teaching here: to learn that skill, *you have to make the prompts*
  - If you feed *my* assignment description and *my* test suite into the model, you're just copying *my* work

Roles: Students

- When communicating with me and with TAs, include context
  - Class number in the subject
  - GitHub ID if it’s about a programming assignment
  - Your name or student number if you’re using e.g. a Discord or Piazza account

Demo

- VM for doing Assignments
  (Like take-home tests)
  - Launch the VM
  - Explain ./start_here.sh
  - Explain record
  - Neovim snapshots

Summary

- Course is *hands-on*
  - Expect to learn systems programming skills
  - Expect to spend quite a bit of time fumbling around
    - Learning to problem-solve and debug effectively in a challenging environment is part of this
  - Decide now if you are in for learning
- Assignments
  - 2 short every week for first 4 weeks
  - 1 long every 2(ish) weeks after that