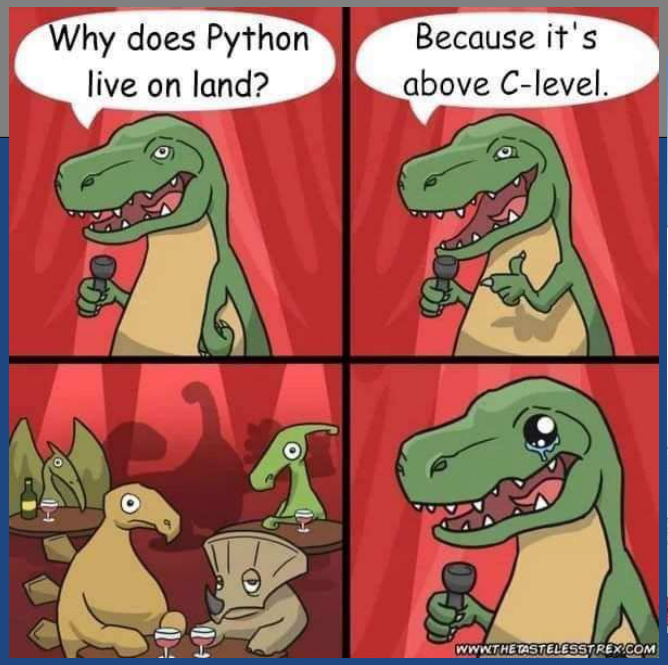
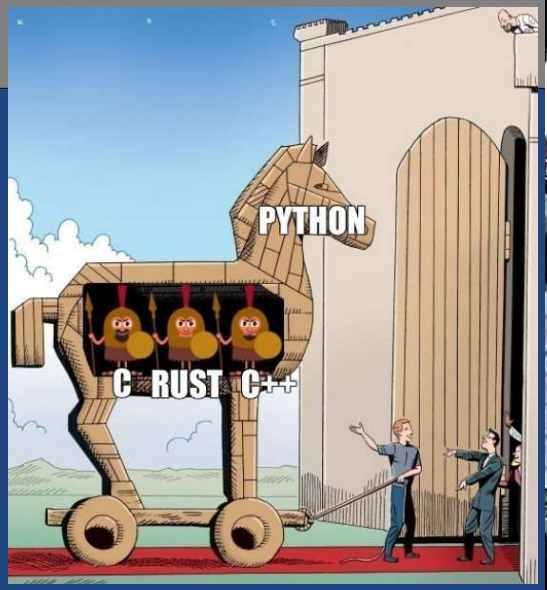
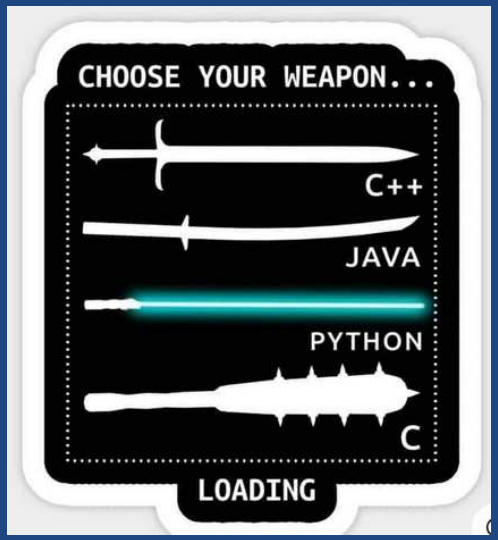
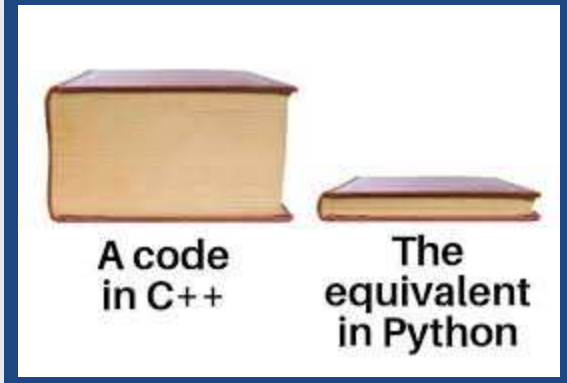






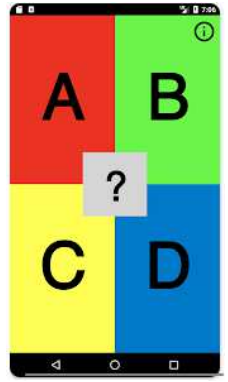


Welcome to CMPT 201 System Programming

Interactive Lectures

- Lectures will have
  - Lots of coding!
  - Lots of questions!
  - Lots of interactions!
- Install the ABCD app now!

Android

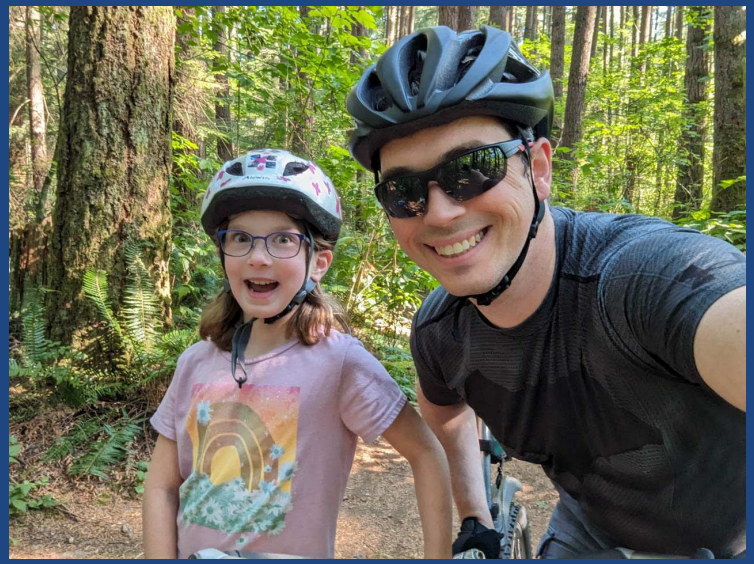


iPhone

Topics

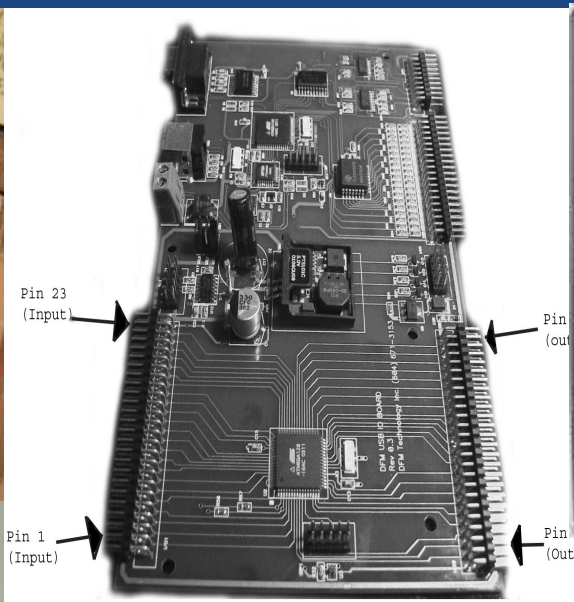
1) Introductions
2) What is System Programming?
3) Course overview
4) Demo of coding environment

Who’s Dr. Brian?



Dr. Brian (Fraser) (he/him)

- I love questions and feedback!



About Me

- Love Teaching: I can help share my excitement for programming, and for making the world a better place.
- Degrees: BSc & PhD from SFU (AI)
- Favourite Video Game: StarCraft 2, WoW, Elite Dangerous, Mario Kart
- Family: Married with 2 girls (9y & 11y)
- I recognize that I am privileged to be in my position with many advantages afforded to me throughout my life.
  - I work to build a positive inclusive experience for everyone.

Course Expectation

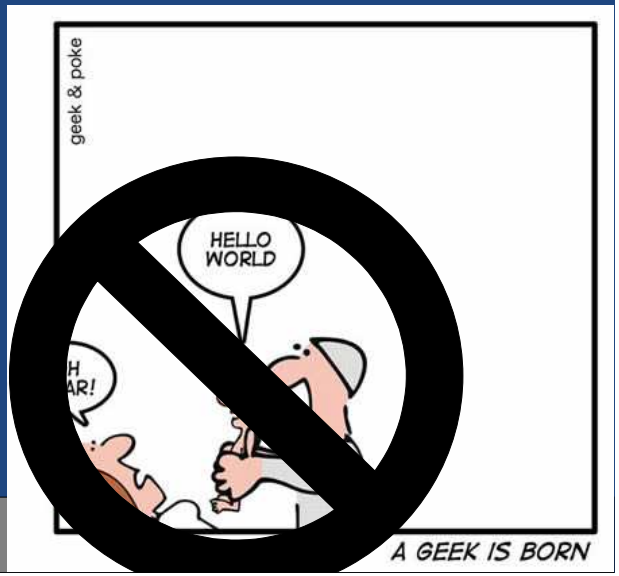
- Only one thing
  - Use a positive tone for all communication (asking questions, on Piazza forums, with TAs)
  - Anon trolling hurts and won't be tolerated
  - Students have wide range of backgrounds; respect it
- If sending a message
  - Give a little context (class, your name, topic, ...)
  - Email: If you are sending more than 2 per week on average, over multiple weeks, it may be too many.

Guide to Slides

- Slide Colour Guide (often...):
  - Green: headings.
  - Yellow: Highlighted text.
    - This course has a midterm and final
  - Blue: Term being defined.
    - Hour: 60 minutes.
  - Sweep-in Text: Blanked out text.
- Joke:
  - There are 10 types of people in the world...

Growth Mindset

- Programming is skill a person develops;
  not one they were born with.
  - Nobody was *born* being good at C.
  - Nobody was *born* being bad at C.
  - Everyone good at C has worked hard and learned it.
- Computer Scientists learn helpful dispositions such as:
  - Collaborative
  - Inventive
  - Persistent
  - Meticulous
  - ...



AI and Coding

I asked for 800x800 pixels;
it gave me 1024x1024.

Image generated by Copilot AI

Context on AI

- AI is dramatically changing software development
  - It can write (some?) code fast.
  - It can provide a (good?) analysis of code.
- Does this make software developers obsolete?

- I feel that to be an efficient developer with AI you must:
  1) Be an skilled software developer.
  2) Know how to efficiently use AI tools.

Why you need development skills?

- ..

- AI tools generate quantity over quality.
  - ***You*** must review the quality of generated code.
  - ***You*** must direct the AI how to meet your goals.

- ***You*** need the skills to know good code, and know what is needed to improve it.
  - This course is mainly about building your skills as a software developer.

Learning Software Development Skills

- Lecture teaches us ideas, but most of us learn software development best by..
  - Assignments are how we code in this course.
- Our Coding Process:
  a) Code a little (but of course it's wrong!)
  b) Run it and see where output is incorrect, or
     Read code and see it's a mess (design/details/...)
  c) Find where in the code is the problem.
  d) Decide how to correct the code.
  e) Goto 1.
- ABCD - With a partner
  Where in this does your learning happens?

Limit of AI

- AI Limits Learning
  - Using AI tools can..
  - This limits the skills you learn and stunts your growth.
- AI mostly "understands", most of the time
  - e.g.: asked to refactor code that used a Java interface, AI gave me 3 different ideas on how to instantiate an interface, but identical code for each!
- AI often favours Quantity over Quality
  - Some open source projects are banning AI because it creates a lot of low-quality code: experienced developers sift through to find good changes.
  - Benefits of AI use is an area of active research!

Use of AI in this course

- Use AI to help you learn by having it:
  - Explain code
  - Quiz you on course content
  - Give debug tips when truly stuck
  - Get help like stack-overflow
    e.g., "good C++ random number code")
- Don't use AI to write your code.
  Your value to a company is the skills you bring: you must be a strong developer to effectively use AI.
  - Don't let it make design decisions.
  - Don't let it write your code.
  - Don't let it refactor code to clean it up.

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.

Discussion

In groups of 3 to 4 people:

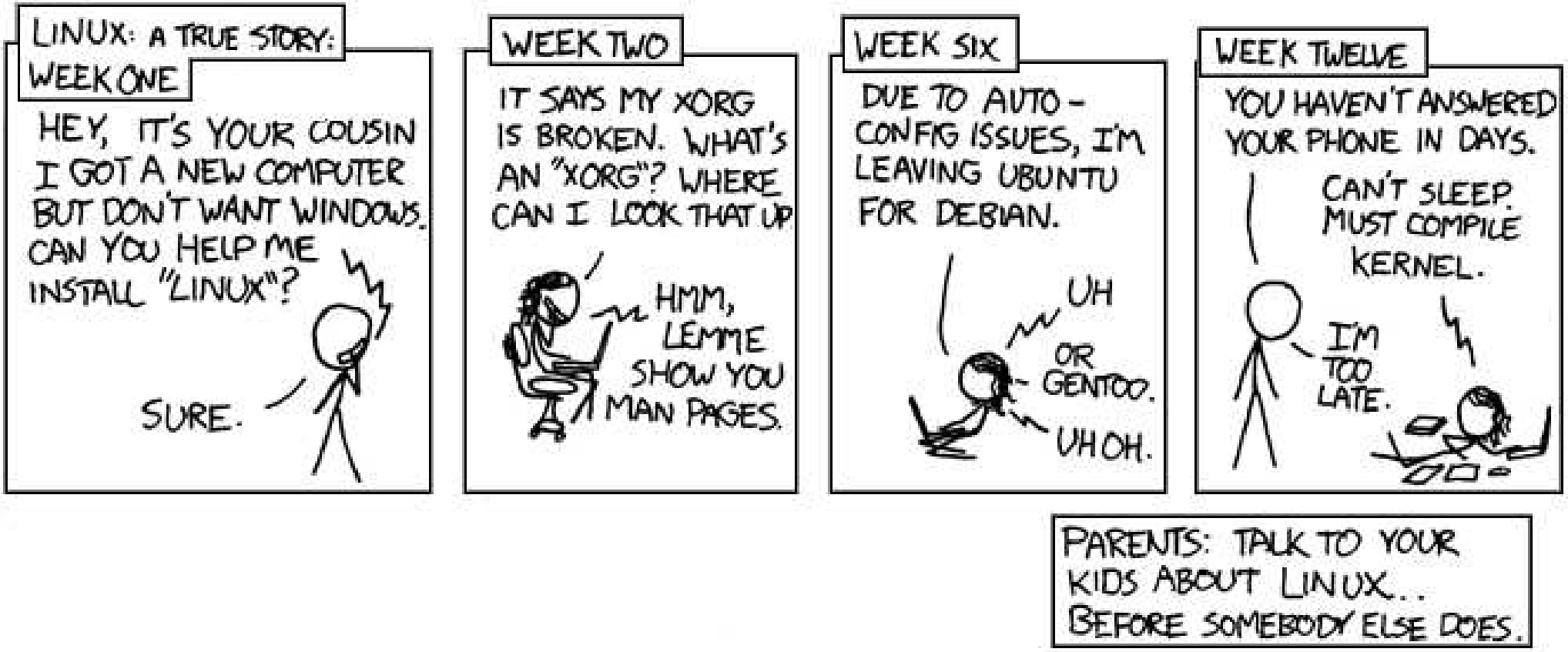
- Exchange contact info (email / Discord / ...)
- Answer the following:

1. What are 3 things that would be systems programming? Which is most interesting?

2. What is one important attribute of doing systems program?

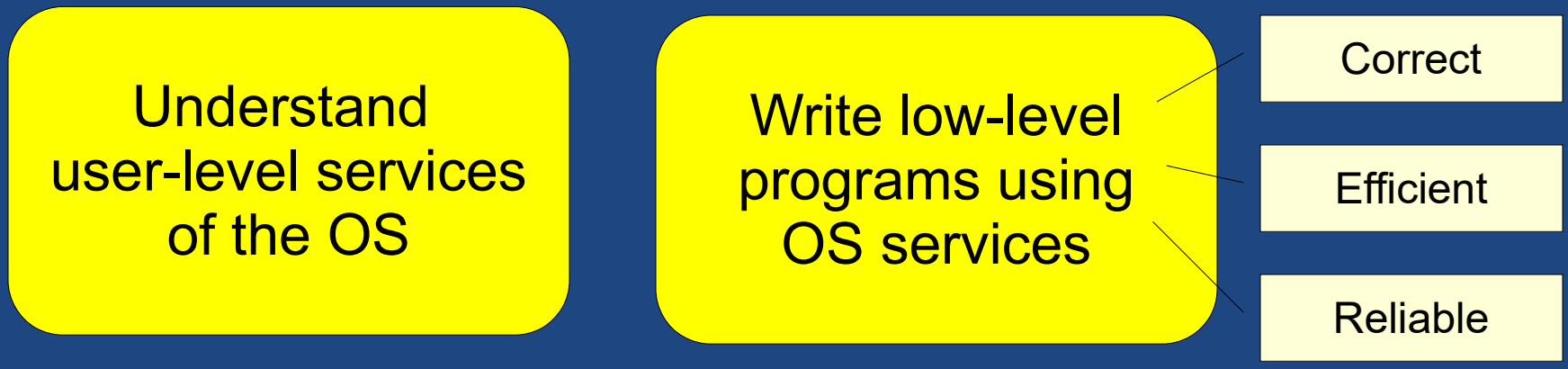
3. What do you think would be hardest about systems programming?

Course Overview



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



ABCD: Perspective

- How would you rate your programming ability?

a) Excellent!
b) Good
c) OK
d) Poor

- How good would you like to be at programming by the end of the course?

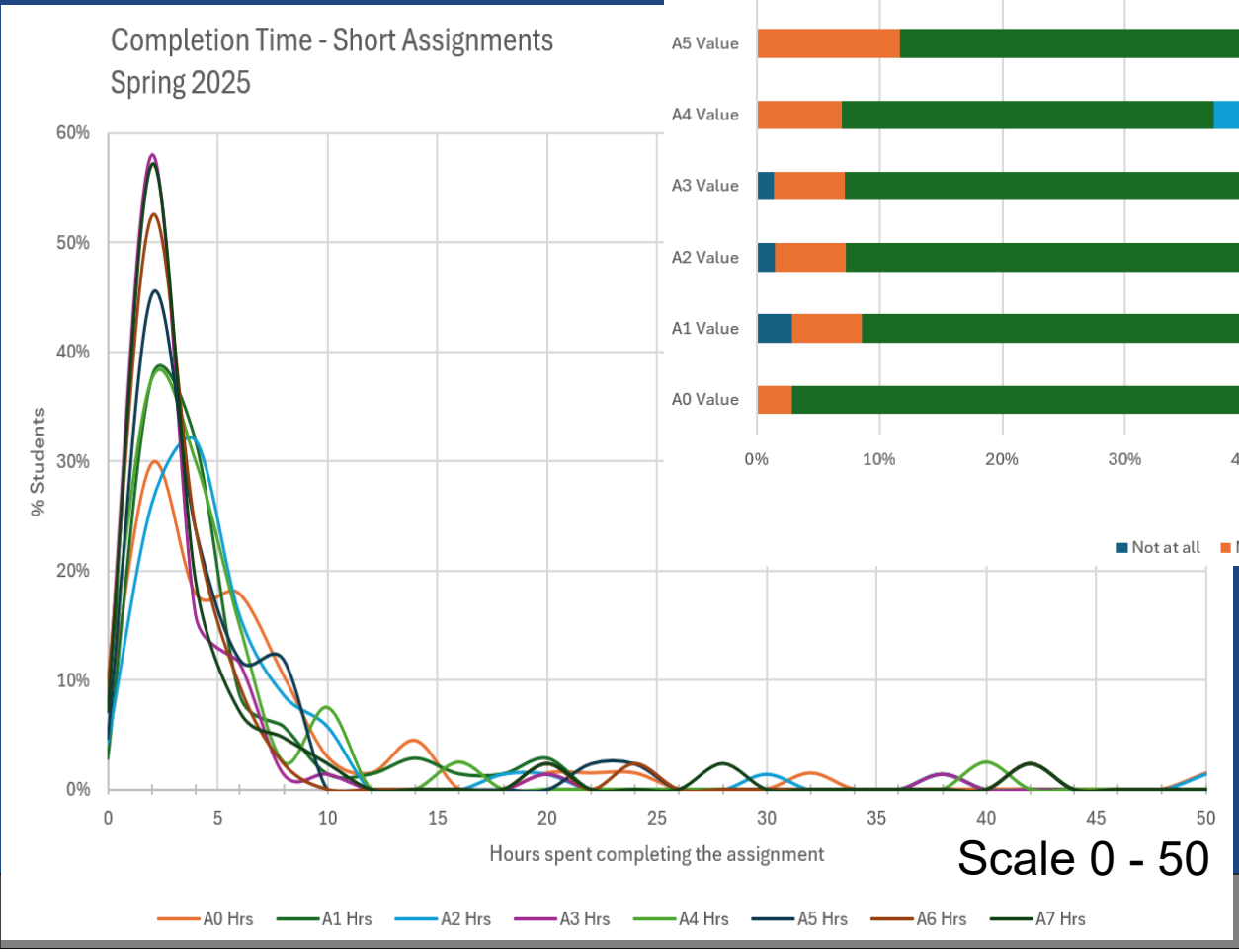
- How much work are you willing to put in?
  - Do you have the time?
  - Do you have the commitment?
  - CMPT 201 is a 4-credit course.

a) A whole lot
b) Average
c) Minimal
d) AI FTW!

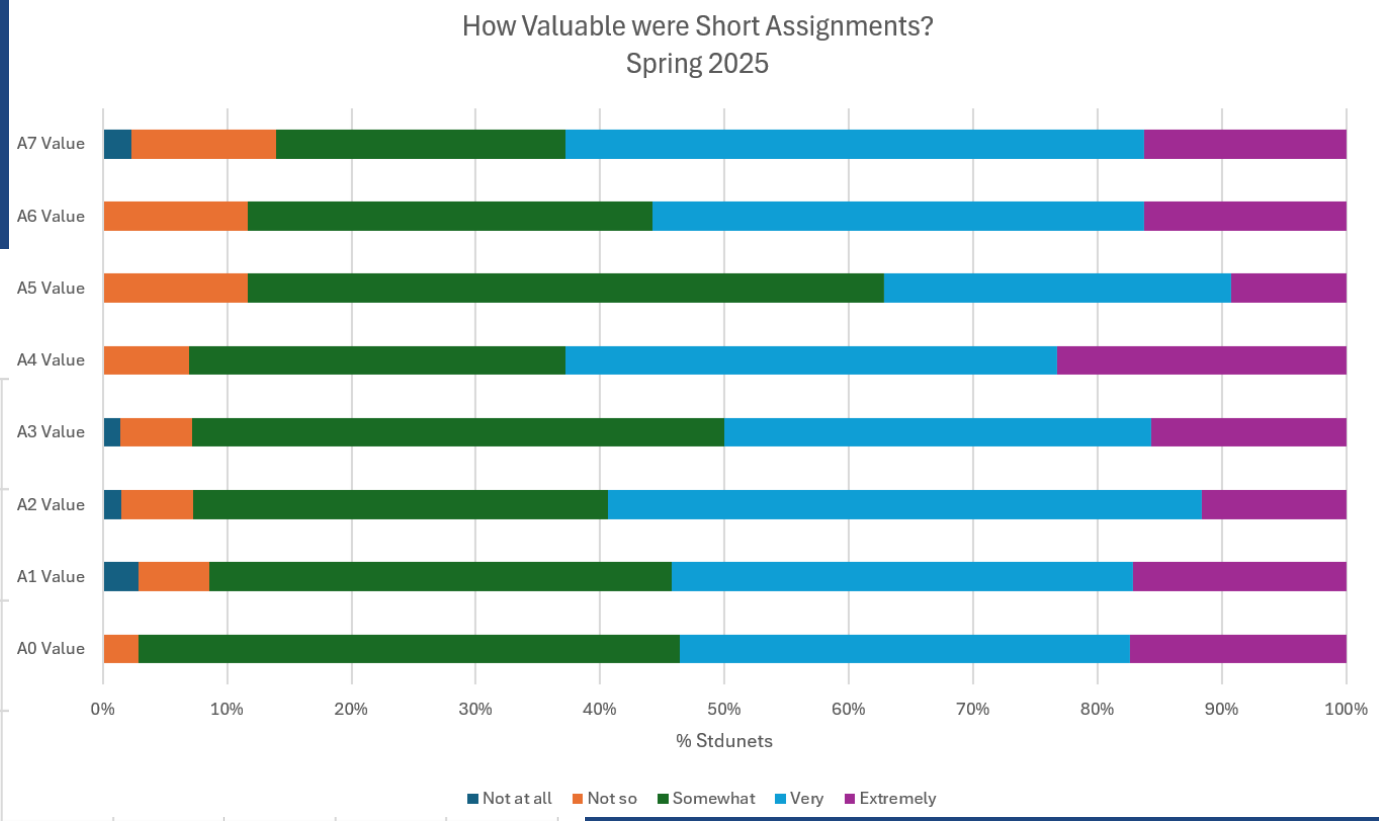
Weekly Schedule

- 11 Labs
  - Due Sundays; marked for completion.
- 8-9 Short Assignments (1st half of course)
  - Due Thursday and Sunday each week!
- 5 Long Assignments (2nd half of course)
  - Due every other Sunday

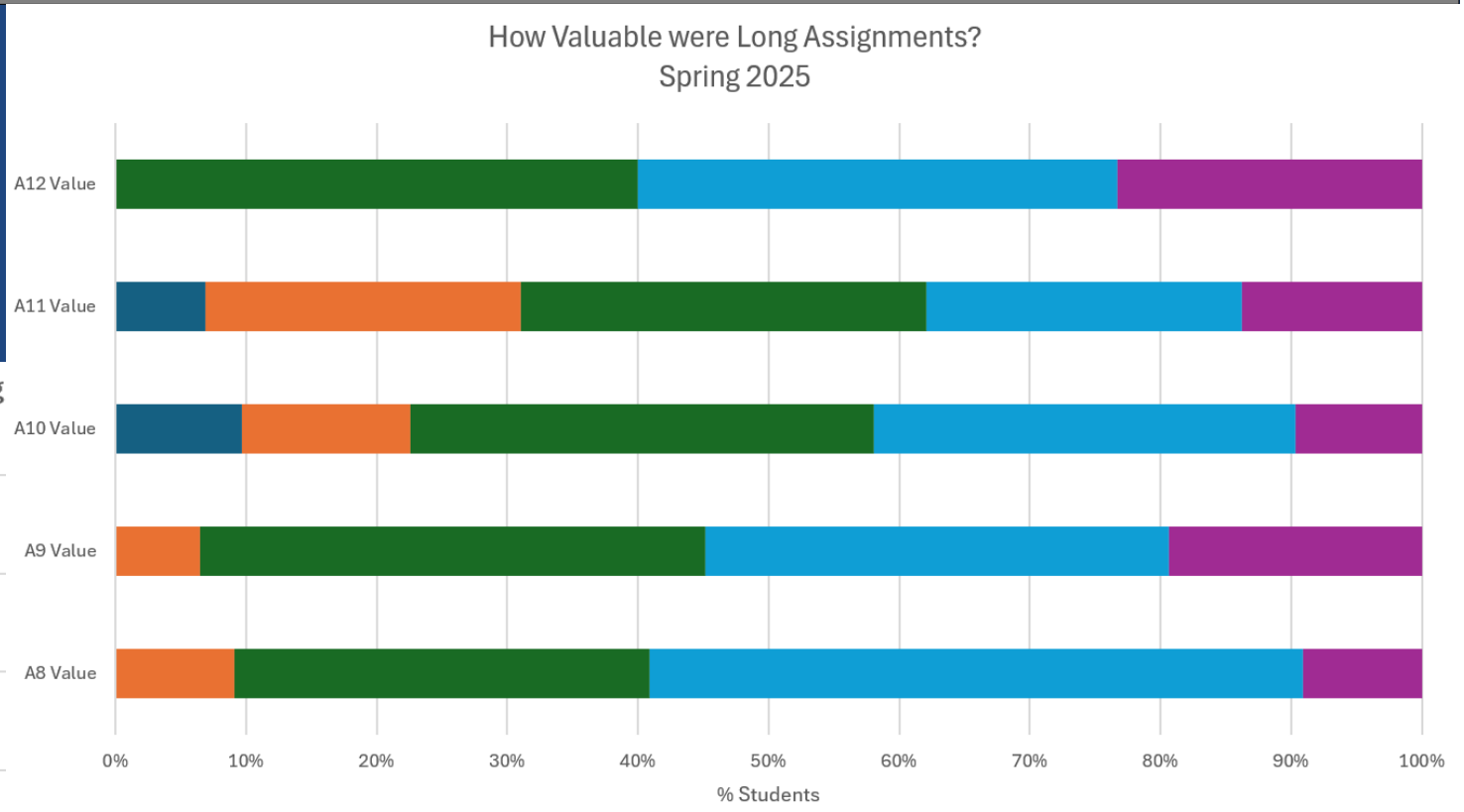
What to expect: **Short** Assignments



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

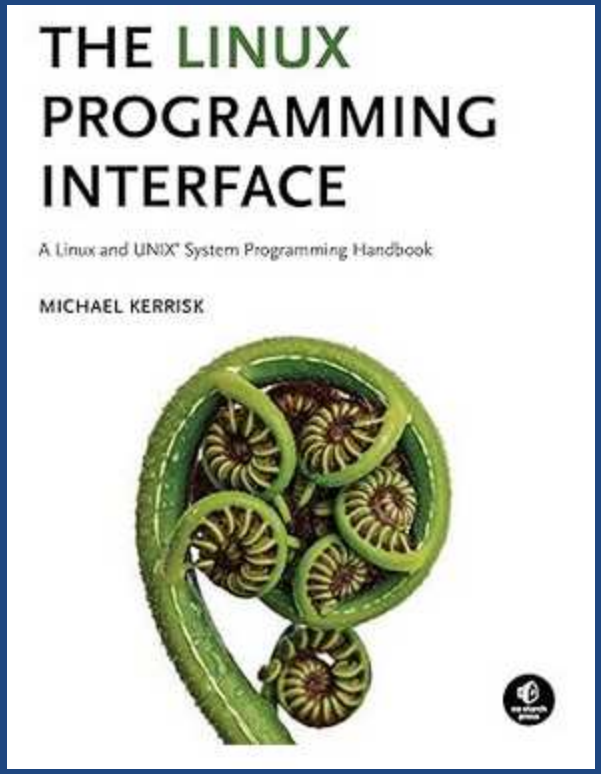
- From prereqs you know
  - Solid programming skills to write functions, loops, if, arrays, pointer, input/output.
  - Solid debugging skills to recognize defects and methodically track down errors.
  - Solid understanding of C or C++ programming.
- Passing prereq does not make this an easy course
  - 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

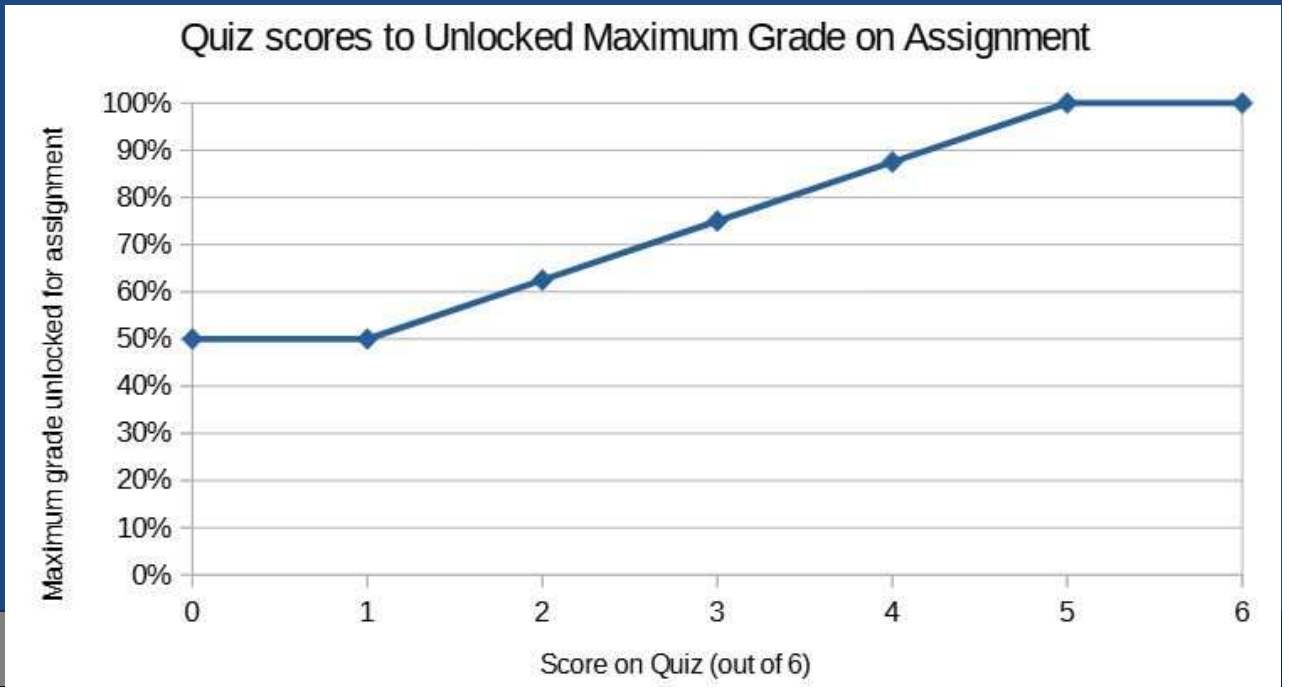
> • **Students must attain an overall passing grade on the weighted average of exams (quizzes/ midterms/final) in the course in order to obtain a clear pass (C- or better).**

- 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.

Unlocking Assignment Scores

- Assignments are a critical part of course learning
  - ~5 quizzes, all focused on assignment content.
  - Quizzes are worth 0% overall, but unlock half of the marks on assignments.

- Default maximum score on an assignment **is 50%.**
- Your quiz **unlocks** a maximum assignment score > 50%!



Policies

Late Policy

- **Assignment Late Policy**
  - **For labs**, there is a 60 minute grace period after a deadline to account for technical issues such as network failures or git issues.
  - **For assignments**, over the semester you have a total of **7 late-days** to use so late submissions can be marked.
    - Late-days are counted in fractional parts of a day: late by 1h uses 1/24th of a late-day.
    - You may use **at most 3 late-days on any given assignment**.
    - Late submissions (code pushed to your assignment repo after the due date), if it improves your grades from before the deadline, will automatically use late-days and you will received the latest grades. No need to tell us or request it, and we very much don't want to handle requests to "un-spend" a late-day.
  - Submissions outside of this late policy will not be marked.
  - Contact your instructor if there are extenuating circumstances; however, generally these flexible late-days should account for most illnesses, conflicting course deadlines, work commitments, and short-term family emergencies.
- **Extensions and Deferrals**
  In cases where some concession is requested for assignments or exams, you may need to email your instructor a completed SFU Academic Concession Self-Declaration Form. Doctor's notes are usually not required.

Remark Assignment Process

You must contact a TA for your section of the course within 7 days of an assignment's grade being released on CourSys to request a remark. It is best to see TAs during their office hours.

To remark an assignment, your TA will ask you to open the CMPT201 container and clone into a fresh directory a copy of the assignment in question. First you will run `git log` to show the history of commits to ensure nothing was submitted past the due date / late submission date. Then you will run the checker in that folder, and the TA may ask you to run certain additional steps to demonstrate the solution works.

The TA will then adjust your marks according, adding a comment in CourSys explaining what was changed on the marking.

Talk to your instructor if you have concerns with the remark procedure.

Copy-and-Paste Policy

On assignments (short and long), each copy-and-paste operation for assignments leads to at least 5.01% penalty. The fractional score triggers us to inspect whether cheating (e.g., pasting multiple lines of codes from others or AI tools) exists and if so, zero marks will be given.

Policies

- AI Tools

- You must have written the code yourself, and be able to re-create the application on an exam (or job interview)!
- If use use an AI tool, you must mention it in your code, such as putting the following comment at the top of your file:

```
// Used help from ChatGPT to find null pointer in tokenization code.
```

  - If you are finding that you need a lot of specific mentions on what AI is doing for you, then you are likely relying on it too much and not building your own software development and systems programming skills.
- Incorrect use of AI tools is considered a violation of the course's academic honesty policy and will earn a grade of 0 and an academic dishonesty report being filed with the university.

Allowable uses of AI

- Experiment with code shown in lectures.
- Help understand concepts.
- Help understand code and its behaviour.
- Answer questions such as:
  1. "Explain the return values of `fork()`?"
  2. "In this example, what are the arguments to `fork()`?"
  3. "Why does this code crashes when I enter an empty string?"
     (Note: you should first use the debugger and build your skills, but the AI can help)

Forbidden uses of AI

- Don't use it to write your lab or assignment code for you.
- Don't ask questions like:
  1. "What is an implementation of the following assignment?"
  2. "Write a C function which reads in user input and tokenizes it into an array."
  3. "Write a C program which passes the following tests."

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 figuring things out
  - Decide now if you are in for learning.
- Assignments
  - 2 short per week for first 4 weeks
  - 1 long per ~2-weeks
- Today
  - Start Lab 0 (not for marks, but get setup!)
  - Start A0 (“due” Sunday)
  - A0-A2 all actually due with A3 (handle late enrols)