Introduction

Class 0

Overview

1. Staff introductions
2. Class overview
3. Tips for success

Staff Introductions

Instructor: Brandon Nguyen

- PhD Candidate under Prof. Scott Mahlke
- Undergrad at UT Austin in ECE
  - Primary focus: Computer architecture and embedded systems
  - Secondary focus: Digital signal processing
  - (spent my first two years as a BME doing premed...)
- Interests include computer architecture, compilers, and systems software
  - Weirdo who enjoys classes like 427, [2345]70, [34]73, 482, 583

Random fun stuff about me

- Modern and historical fencing (HEMA)
- *Slight* addiction to Genshin Impact
- Finally gotten around to indulge in photography and videography

Course Overview

What is this class

- This class is for *anyone* wanting to become more effective at using their computer for development work
- This isn't necessarily a "tools" class
  - Tools come and go: does anyone remember COBOL and CVS?
- Each workplace will have its own tools and workflows
- The ultimate goal of this class is to help you learn to pick up, learn, and use new tools to solve problems
- The tools you learn along the way are the icing on the cake
- That being said, we will be focusing on Unix/Unix-like systems and shells in this class
  - Windows Command Prompt is not suitable for this class
  - Windows Subsystem for Linux (WSL) is suitable, however

Expectations

- Have a basic understanding of program control flow
    - e.g. if statements, loops, functions
- Have experience expressing your solutions in program statements
- Have some experience with a C or C++ or similar language
    - Let me know if you need help with the language itself

Expectations

- Work is intended to be done alone
  - It can help to point each other to useful resources you find
  - Your code should be your own
- No ban on AI, but...
  - Are you doing yourself a favor by robbing yourself the experience of researching and figuring out a solution?
  - *Your* development is the goal here, not the final product you produce

Expectations

Technology

- Have a computer that runs Windows, macOS, or Linux that you can install software on
- Chromebooks are welcome if they have Linux Beta (Crostini)
- Ubuntu 22.04 is going to be the reference environment for class
    - Most Linux distributions will have similar behavior for things in this class, so don't fret if you're on 20.04 or Debian or whatever $^{\text{i use arch btw}}$
- While most things can be done on macOS, some tools have different behavior between Linux and macOS

Course communication

- Canvas: Announcements and a fancy gradebook
- Piazza: Course content and logistics related questions
- Discord: Casual and informal chat and questions
- Email: For personalized correspondance and more personal matters
    - Please start the subject line with "EECS 201" so I can find it

Course structure

Flipped classroom

- "Lecture at home, homework at lecture"
- Weekly content videos
- Quick quiz over videos with time window (keep up with class!)
- In-person class, attendance optional
    - Mini-review
    - Q/A
    - Activities
    - **Time to work on assignments with instructor around for questions!**
    - Survey for extra credit

Course structure

Weekly "basic" assignment

- Guided light assignments to familiarize you with tools and what you can do with them
- Directly related to material covered in the videos

Advanced component

- Less guidance than basic assignments
- Often touches on some related things not directly covered in class
- Provides practical experience in perusing documentation and applying what you know
- Can also be fulfilled by doing a project
    - Checked out at an office hour
    - More to come about this...

Grading

- Point accumulation
- Two major grade categories: **Basic** and **Advanced**
- **Class** is an extra category
- Basic has soft cap of 60 total points
- Advanced has a soft cap of 40 total points
- Class has no soft cap
- Final score is the *adjusted* sum of these categories
    - You can make more than 60 Basic or 40 Advanced points
    - Points after these soft caps are worth half (more on this)
    - There is no averaging: you just add numbers
    - You can see how letter grades get assigned in the syllabus

Grading

Basic

- There will be at least 10 basic assignments, worth 6 points each
- That means you only need to do 10 to get all 60 points
- The remaining assignments serve as a buffer for you to miss/skip
- There is a rounding scheme
- **Points past 60 are worth 50%: an 11th assignment would only be worth 3 points**
- If you do 12 basic assignments:
  - 12 * 6 = 72 -> 60 + 12/2 = 66
- Assignments submitted via GitLab have a soft deadline: submitting on time will get you extra points
  - [6,∞) unrounded points nets you 1 point
  - [4, 6) unrdounded points nets you 0.5 points
- Assignments submitted via Gradescope have a hard deadline: no late submissions

Grading

Advanced

- Each advanced assignment is worth *at least* 10 points
- That means you only need to do 4 to get all 40 points
- There is no rounding
- You can also do *one* project for a total of 40 points
    - You can submit *one* partially completed project for partial credit
- **Similarly, points past 40 are worth 50%: an 11th assignment would only be worth 5 points**
- If you do 12 advanced assignments and the project...
    - 12 * 10 + 40 = 160 -> 40 + 120/2 = 100: no need to do basic assignments 😃

Grading

Advanced

- Assignments submitted via GitLab have a soft deadline: submitting on time will get you extra points
  - $[10,\infty)$ points nets you 1 point
  - $[7, 10)$ points nets you 0.5 points
- Assignments submitted via Gradescope have a hard deadline: no late submissions

Grading

Class

- Each content quiz and class survey are worth 1 point
- Participation in class activities can net you points
- Surveys are time sensitive and available only for a week after class
- Quizzes are due at the beginning of class
- Surveys cannot be done late
- Late quizzes are worth half credit
- This category has no soft cap

Grading

tl;dr you get points for each assignment and your letter grade is based on the total points

Grading

Repeat after me: a point is a point

- I'm free to give you free points here or there
- There's no trick if you get extra points on something
- There are no proportional grades: if a Basic assignment is worth 9 points instead of 6, you can get 9 points
- The only time a division sign ever appears in calculating grades is when you exceed a category total

Grading
Summary

- **11 surveys = 11 Class points**
- **11 quizzes = 11 Class points (if submitted on time)**
- 11 Basic assignments = 66 Basic points (8 have early-submission credit)
- 10 Advanced assignments = 100 Advanced points (6 have early-submission credit)
- 1 project = 40 Advanced points

Strategies for success

- Grading scheme is very flexible
- It's on you to keep up
    - **Schedule some time to watch the videos and stick to it as if it were a lecture**
    - **Take notes!** It engages you more! Don't fall into the "I can watch it later" trap!
    - Class time will give you time to complete homework
    - Doing the video quizzes will add up in the end, enough to save you from multiple assignments!
- Sometimes I'll toss some easy Basic or Advanced assignments your way for easy points

Strategies for success
Conventional approach

- Do each quiz on time and fill out each survey: **22 Class points**
  - 11 quizzes, 11 surveys
- Fully complete 10 Basic assignments: **60 Basic points**
- So far, 82 points: already at a B-
- Fully complete two Advanced assignments: **20 Advanced points**
- Now we're at 102 points: A+
- This isn't taking into account the points you get for doing GitLab assignments on time, or assignments worth more than normal
- If you miss quizzes or surveys, you can make for the shortfall by doing assignments and with their early submission credit
  - The grading was originally balanced around 10 Basic and 4 Advanced assignments
  - Class points are there to ease the load

Strategies for success
Conventional approach workload

- Basic assignments: roughly 1 hour each
- Advanced assignments: roughly 4 hours each
- Lecture videos + quiz: roughly 1.5 hours
- Each week:
  - Watch/take notes on lecture videos and do the quiz: **1.5 hours**
  - Come to class/do Basic assignment: **1.5 hours**
- **~3 hours a week**
- **Sprinkle in two Advanced assignments when you can**
- Earn an A+ 😊

Any questions?

Demo

Why are you here? How'd you hear about this class?

What's your relationship with computers up until this point?

What's your goal with EECS?

Addenda

Environment

- Terminal emulator: **Alacritty**
  - Former rxvt-unicode user until I learned how bad its font handling was
- Shell: **Zsh**
- Window manager: **i3**
- Compositor: **picom**
  - Does window transparency effects
- Notification server: **dunst**
  - Displays notifications

Software

- Editor: **Vim**
- System monitoring: **htop**
- Notetaking: **Xournal++**