Course Overview

COS 316: Principles of Computer System Design

Lecture 2

Wyatt Lloyd & Rob Fish

Course Staff: Intros

- Yang Duan
- Sofiia Druchyna
- Jianan Lu
- Minhao Jin
- Christopher Branner-Augmon
- Nan Li
- Alan Zhang
- Robert Fish
- Wyatt Lloyd
- TBD LabTAs

Learning Objectives & Course Components

- System Design **Principles**
  - Lectures
  - Exams

- Skills (**Practice**)
  - Precepts
  - Programming Assignments

Learning Objectives: System Design Principles

- Appreciate trade-offs in designing and building systems
  - Generality vs. performance, performance vs. security, …
- Understand how systems work
  - Common high-level techniques in systems
  - Reasoning about concurrency
- Understand the general systems stack, how it fits together, and some of how it works
  - OS, Network, Distributed Systems, Security

Lectures

- Attend!
  - Active thinking through concepts (you)
  - Active calibration of teaching (us)
  - Review each lecture afterwards, ask questions in Office Hours

- Explore fundamental concepts,
  ways of thinking,
  important parts of systems stack,
  cutting-edge research

Lectures

- 5 Major Themes:
  - Naming
  - Layering
  - Caching
  - Concurrency
  - Access Control

Learning Objectives: Skills

- Go programming language, and "Systems" programming
- Version control with git
- Working in groups (sometimes!)
- "Systems programming": sockets programming, concurrency, modular design, unit testing, performance measurement, ...

Precepts

- Attend!
- Hands on, active learning in small groups
  - Bring your laptop!
- Coupled primarily with the programming assignments

Programming Assignments

- You're Building a Web Framework!

- Set of libraries and tools for building complex web applications
  - Abstracts connection and protocol handling
  - Routes requests to controllers/handlers
  - Caching for common queries and computations
  - Multiplexes concurrent access to databases
  - Translates database objects into programming language constructs
  - User authentication and authorization

- Examples: Rails, Django, Express, Apache Struts, Laravel

WARNING
Systems Building is *not just* Programming

- COS 126 & 217 told you how to design & structure your programs.
  - This class doesn’t.
- If your system is designed poorly, it can be much harder to get right!
- Conversely, assignments won't require algorithms or data structures you're not already familiar with.
- Your friends:
  - Discussing potential solutions *before* implementing (with TAs or classmates)
  - Test-driven development

Assignments: Collaboration & Resources

This slide is really important

- You can, and *should* use *general* resources available on the Internet to complete assignments:
  - Go documentation, Stackoverflow, open source projects
  - Mailing lists, chat rooms, etc...
  - Cite sources in your README!
- You cannot use AI tools like Copilot or ChatGPT
- For partner assignments, you can work with a partner. For solo assignments you can discuss high-level challenges and ideas for addressing them. You cannot talk about or view code with any other students.
  - Take-a-walk rule: If you discuss the assignment with other teams, do something else for an hour before returning to your code

Assignments: Collaboration & Resources

https://www.cs.princeton.edu/courses/archive/fall24/cos316/policies.html

activity	your group for the 2 assignments*	course staff	COS 316 grads	classmates	other	AI tools (e.g., ChatGPT, Copilot)
discuss concepts with ...	✔	✔	✔	✔	✔	✘
acknowledge collaboration with ...	✔	✔	✔	✔	✔	✔
expose solutions to ...	✔	✔	✘	✘	✘	✘
view solutions from ...	✔	✘	✘	✘	✘	✘
plagiarize code from ...	✘	✘	✘	✘	✘	✘

Assignment Partners

- 2 will be with partners
- 3 will be solo

Assignments: Submitting and Grading

- Submitting happens whenever you "push" to your "master" branch on GitHub
  - You can push as many times as you'd like (we encourage you to do so *often*)

- Grading is automatic and immediate
  - There is no penalty for multiple submissions. We will use your highest graded submission (push)
  - Each automatic grading is posted as a comment to the last commit of each push. It includes a break down of tests cases, including which failed.

Programming Assignment Late Penalties

- 90% for work submitted up to 24 hours late
- 80% for work submitted up to 2 days late
- 70% for work submitted up to 3 days late
- 60% for work submitted up to 5 days late
- 50% for work submitted after 5 days late

Programming Assignment Late Days

- 3 late days total for the semester
  - Granularity of 1 day
    - 1 minute late is 1 day late
    - 23 hours and 59 minutes late is 1 day late

- Assigned retroactively to give you the best possible overall grade
  - We do this for you!

- Additional late days? No!
  - Only w/ involvement of your dean or with a doctor's note

EdStem

- Great place for questions and answers
- Use EdStem in lieu of email in almost all cases
  - Can send course staff a message using private posts
- Post conceptual questions about lecture material or assignments
- **Not** for debugging your code
  - Go to office hours to learn how to do that!
- Expectations: We will check EdStem and answer questions **once a day**, on weekdays

Programming Assignment Support

- This is a 3xx class: assignments are more challenging and with less **direct** support
- Office hours, office hours, office hours!
- TAs and LabTAs will help you learn how to **think** about and **approach** solving a problem
- They **will not** debug your code for you
  - They are not allowed to touch your laptop!

Exams

- Midterm in TBD evening of midterm week
  - 3 hour “no time pressure” exam
  - Midterm review session during week after fall break
    - Review OF the midterm not for it

- Final on December 15, 2024 from 4-7pm
  - 3 hour “no time pressure” exam
  - Weakly cumulative (i.e., most questions from 2nd half of course, but some from 1st half)

Grading

- 50% - Programming Assignments
  - 5 Assignment, each worth 10%
- 50% - Exams
  - Midterm and final each 25%

Changes from prior semesters

- Based on learning goals

Learning Objectives & Course Components

- System Design **Principles**
  - Lectures
  - Exams

- Skills (**Practice**)
  - Precepts
  - Programming Assignments