Course Overview

COS 316: Principles of Computer System Design

Lecture 2

Amit Levy & Ravi Netravali

Agenda

- Course staff introductions
  - Why we like systems?
- Course structure and goals
- Schedule and grading

Course Staff: Intros

Prof. Ravi Netravali
Instructor

- Joined Princeton faculty in 2021
- Teach COS 316 and COS 561
- Research in networked systems

- Research goals
  - Improving distributed applications in terms of performance and ease of use
  - Edge Computing
  - Systems for ML, ML for systems

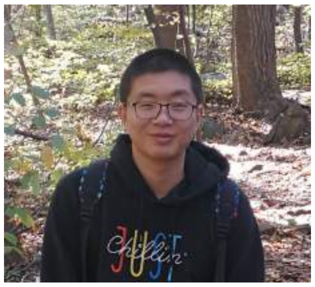
Course Staff: Intros

Prof. Amit Levy
Instructor

- Joined Princeton faculty in 2018
- Often teaches COS 316
- Research in operating systems

- Systems building blocks for building an endless number of applications
- Systems that allow developers to have the most flexibility and creativity
- … while being secure and performant

Course Staff: Intros



Zhenyu Song
TA

- 6th year PhD student working with Profs. Lloyd and Li
- Interested in using ML to improve system design
- Likes systems because it touches basic principles and has real world impacts.
- Has TAed for COS 418/518 (Distributed Systems)

Course Staff: Intros

Shai Caspin
TA

- 2nd year PhD student working with Amit
- Goal: making it easier to build safe/private/secure systems
- Motivation: lack of trust in technology, and wanting to ensure things will work as expected
- Has TAed “Intro to Systems”

Course Staff: Intros

Nick Kaashoek
TA

- 2nd year PhD student working with Prof. Lloyd
- Interested in distributed systems for the:
  - Complex problems and creative solutions they require
  - Practical, real-world applications
- First time TAing at Princeton

Course Staff: Intros

Wei Luo
TA

- 1st year Masters student
- Interested in applying ML to image/video processing systems, e.g., to improve image compression
- First time TAing at Princeton

Course Staff: Intros

Neil Agarwal
TA

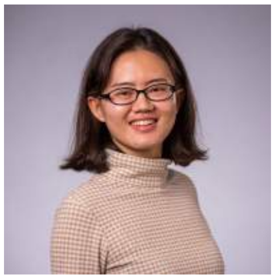
- 4th year PhD student working with Ravi
- Interested in the intersection of ML and Systems
- Goal: as ML becomes more popular and mature, build systems to enable its (practical) use at scale
- Has TAed COS 561

Course Staff: Intros

Ryan Torok
TA

- 2nd year Masters student working with Amit
- Works on improving the security and privacy of computer systems
- Goal: make the web safer for everyone
- Has TAed for COS 318 and COS 226

Course Staff: Intros



Yue Tan
TA

- 5th year PhD student working with Amit
- Works on improving security model for emerging computation paradigms, e.g., function-as-a-service
- Enjoy systems because it is challenging to develop principled large systems, but also rewarding since they better support current apps and enable new ones
- Has TAed for COS 316 and COS 418

Learning Objectives & Course Components

- System Design **Principles**
  - Lectures
  - Problem Sets
  - Final Project
- Skills (**Practice**)
  - Precepts
  - Programming Assignments
  - Final Project

Learning Objectives: System Design Principles

- What is the field of systems?
  - Learn to appreciate trade-offs in designing and building the systems you use.
  - Get better at understanding how systems work.
  - Learn to *use* systems better---write more efficient/secure/robust/etc. applications.

Lectures

- 6 Major Themes
  - Naming
  - Caching
  - Layering
  - Concurrency
  - Access Control
  - Scheduling

Lectures

- 6 Major Themes:
  - Naming
  - Caching
  - Layering
  - Concurrency
  - Access Control
  - Scheduling

Lectures

- Try your best to attend (in person)
  - Active thinking through concepts (you)
  - Active calibration of teaching (us)
- Explore fundamental concepts, ways of thinking, cutting-edge research

Problem Sets

- Focus on reinforcing and generalizing lecture content
- Done individually

Learning Objectives: Skills

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

Precepts

- Attend synchronously
- Hands on, active learning in small groups
- Coupled primarily with the programming assignments

Programming Assignments

- You're Building a Web Framework!
- Set of libraries and tools for building sophisticated 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

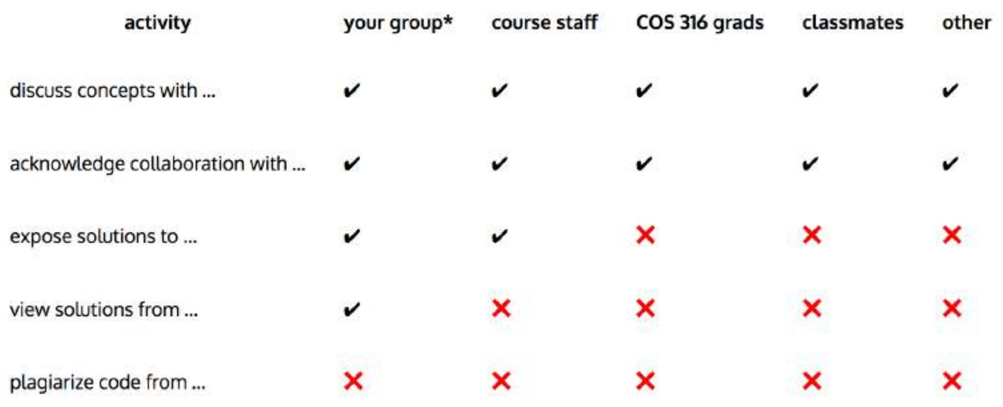
- COS126 & 217 told you how to design & structure your programs.
  - This class doesn’t.
- Poor (early) system design → much harder to get things right!
- Conversely, assignments won't require algorithms or data structures you're not already familiar with.
- Team-based assignments
  - Discuss potential solutions *before* implementing
  - Test-driven development

Assignments: Collaboration & Resources

- You can, and *should* any resources available on the Internet to complete assignments:
  - Go documentation, Stackoverflow, open source projects
  - Mailing lists, chat rooms, etc...
  - Cite sources in your comments or README!
- You *can* collaborate (in groups of up to 3)
  - Okay to share ideas/concepts (but *not* code) with other groups
- Take-a-walk rule:
  - If you discuss the assignment with other teams, do something else for an hour before returning to your code
- You may *not* ask instructors for help debugging your code.

Assignments: Collaboration & Resources

https://cos316.princeton.edu/assignments



activity	your group*	course staff	COS 316 grads	classmates	other
discuss concepts with ...	✔	✔	✔	✔	✔
acknowledge collaboration with ...	✔	✔	✔	✔	✔
expose solutions to ...	✔	✔	✗	✗	✗
view solutions from ...	✔	✗	✗	✗	✗
plagiarize code from ...	✗	✗	✗	✗	✗

Assignments: Submitting and Grading

- Submitting happens whenever you "push" to your "master" branch on GitHub
  - Push as many times as you like (we encourage you to do so *early and often*)
- Grading is automatic and immediate
  - No penalty for multiple submissions → we'll use your higest 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 Days

- 7 late days total for the semester
  - Granularity of 1 day
    - 11:02pm on Wednesday is 1 day late
    - 10:50pm on Thursday is 1 day late
- Assigned retroactively to give you the best possible overall grade
  - We do this for you!

Late Days Example

1. Jordan submits assignment #1 on time, but can't figure out how to pass the last test case. Their grade so far for the assignment is 95%.
2. 7 days after the deadline, Jordan figures out how to pass the last test and submits late, getting 100%.
3. Months later... Jordan underestimates their workload and isn't able to submit assignment 4 until 7 days after the deadline, but passes all tests to get 100%.
4. We assign the late days to assignment 4, so that Jordan's grade is 95% + 100%, as opposed to 100% + 0%.

Final Project

- Open ended systems building project; groups of 2 or 3
- Later precepts and Lecture 13 will help you refine topic
- You design and build something you're interested in!
- Small written component (< 2 pages)

What is Due When?

- Alternating Problem Sets and Assignments each week
  - Each is due on Wednesday at 11pm Princeton Time
- Final project is due on Dean's Date at 5pm Princeton Time

Grading

- 60% - Programming Assignments
  - 6 Assignment, each worth 10%
- 20% - Problem Sets
- 20% - Final Project
- No curve anticipated
  - Will **not** curve down (i.e., a 93% is an A no matter what)

Learning Objectives & Course Components

- System Design Principles
  - Lectures – Attend Synchronously
  - Problem Sets – Due every other week
  - Final Project – You build something new
- Skills
  - Precepts – Attend Synchronously
  - Programming Assignments – Due every other week
  - Final Project – Due on Dean's Date