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

https://cos316.princeton.systems/
Course Staff: Intros

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

• Research goals
  • ML for systems, systems for ML
  • Improving distributed applications in terms of performance, debuggability, and deployability
  • Edge Computing

Prof. Ravi Netravali
Instructor
Course Staff: Intros

• Joined Princeton faculty in 2018
• Often teaches COS 316
• Research in distributed and 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

Leon Schuermann
TA

• 2nd year PhD student working with Amit
• Works on secure embedded OSes and hardware-software co-design
• Likes systems because of the challenges in finding practical solutions to cope with constraints in environments where apps run
• First time TAing a course
Yue Tan
TA

• 5th year PhD student working with Amit
• Works on building (more) secure systems, including for new compute 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
Course Staff: Intros

Rui Pan
TA

• 2nd year PhD student working with Ravi
• Works on designing better systems and networks for machine learning
• Motivation: systems are the backbones for many apps that we all use, and small systems wins can make a big difference
  • Inspired by undergrad OS class!
• Has tutored for intro CS classes
Course Staff: Intros

- 2nd year PhD student working with Amit
- Works on consistency models for distributed databases
- Systems work is tangible: you can create what you design
  - Especially rewarding to tease out mathematical explanations for why things behave the way they do
- Has TAed multiple networking, security, and math courses
Course Staff: Intros

Jingyuan (Leo) Chen
TA

• 2nd year PhD student working with Amit
• Interested in applying PL techniques to improve security, performance, and debuggability of complex systems
• Passionate about designing powerful but simple interfaces to help developers build and debug systems
• First time TAing
Yinwei Dai
TA

• 2\textsuperscript{nd} year PhD student working with Ravi
• Works on the intersection of networked systems and data-intensive computing (ML, computer vision)
• Likes systems because of the challenges in creating robust/efficient solutions to complex problems
• Has TAed computer networks
Course Staff: Intros

Mike Wong
TA

• 3\textsuperscript{rd} year PhD student working with Ravi
• Works on improving the performance and resource-efficiency of machine learning systems
• Enjoys the satisfaction of making the apps that people use daily more performant, dependable, and deployable
• Has TAed COS 561
Learning Objectives & Course Components

• System Design **Principles**
  • Lectures
  • Problem Sets
  • Design assignment
  • 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

• 5 Major Themes
  • Naming
  • Layering
  • Caching
  • Concurrency
  • Access Control
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 systems/research
Problem Sets

• Focus on reinforcing and generalizing lecture content

• Done individually
Design Assignment

• Released today

• Builds on Lecture 1 (Netflix-like service), but at larger scale

• Writeup (600 word limit) + at least 1 *design* figure

• Will revisit this later in the course
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
  • Bring your laptop!

• 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 $\rightarrow$ 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 *must* collaborate (in groups of 2)
  • 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](https://cos316.princeton.edu/assignments)

<table>
<thead>
<tr>
<th>activity</th>
<th>your group*</th>
<th>course staff</th>
<th>COS 316 grads</th>
<th>classmates</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>discuss concepts with ...</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>acknowledge collaboration with ...</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>expose solutions to ...</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>view solutions from ...</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>plagiarize code from ...</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
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 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 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 14 will help you refine topic

• You design and build something you’re interested in!

• Small written component (< 2 pages)
What is Due When?

• 5 programming assignments; 5 problem sets
  • Each is due on Wednesday at 11pm Princeton Time
  • Due on different weeks

• Design writeups
  • Initial one due next Wednesday (9/13)
  • Final one due towards the end of the semester (date TBD)

• Final project is due on Dean’s Date at 5pm Princeton Time
Grading

• 50% - Programming Assignments (5 total, 10% each)
• 20% - Problem Sets (5 total, 4% each)
• 10% for design writeups (skewed mostly towards 2\textsuperscript{nd} one)
• 20% - Final Project

• No curve anticipated
  • Will \textbf{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 – one per module
  • Design writeup – one at beginning, one at end of semester
  • Final Project – You build something new

• Skills
  • Precepts – Attend Synchronously
  • Programming Assignments – 5 total
  • Final Project – Due on Dean’s Date