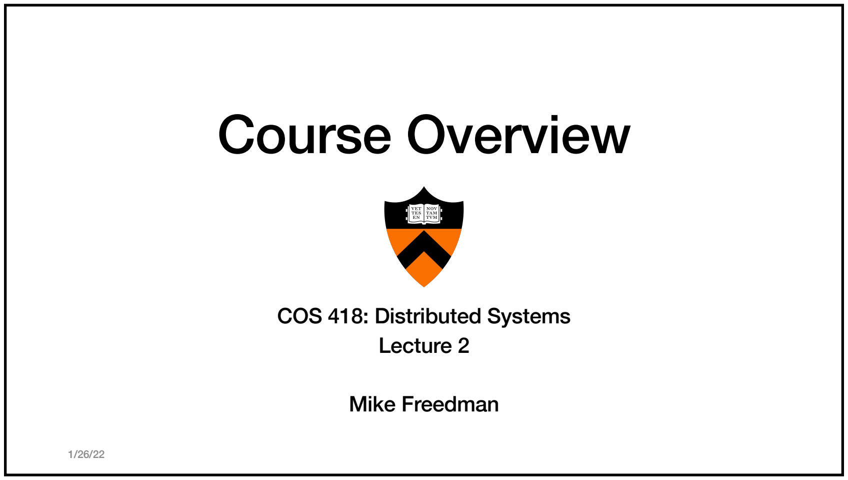


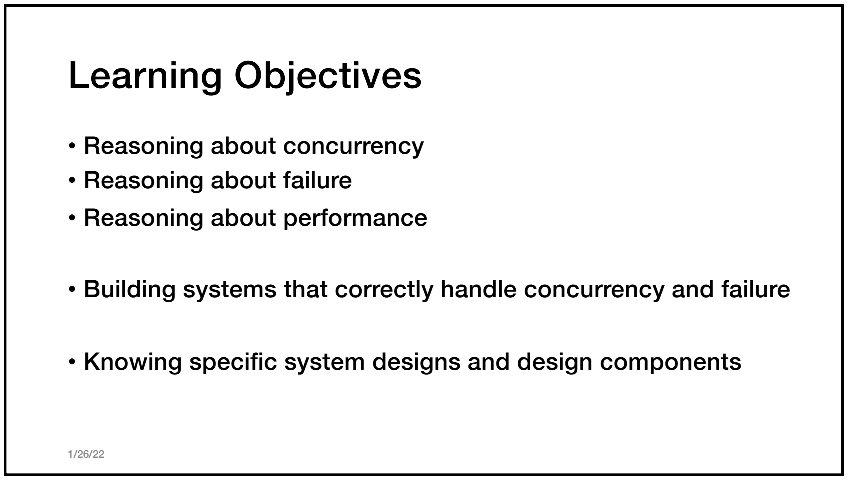
Course Overview

COS 418: Distributed Systems
Lecture 2

Mike Freedman

1/26/22

1



Learning Objectives

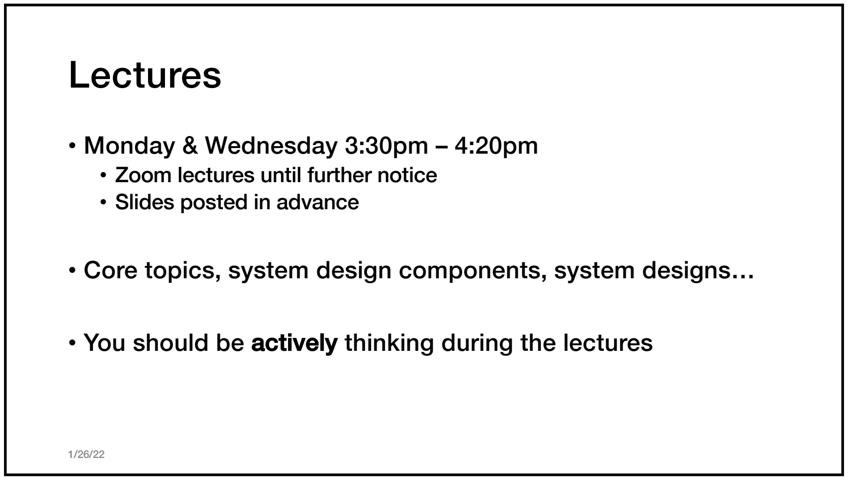
- Reasoning about concurrency
- Reasoning about failure
- Reasoning about performance

- Building systems that correctly handle concurrency and failure

- Knowing specific system designs and design components

1/26/22

2



Lectures

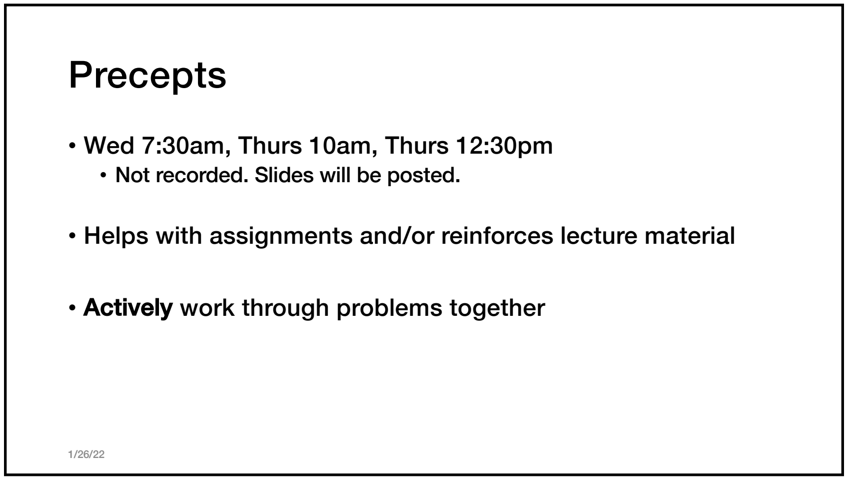
- Monday & Wednesday 3:30pm – 4:20pm
  - Zoom lectures until further notice
  - Slides posted in advance

- Core topics, system design components, system designs…

- You should be **actively** thinking during the lectures

1/26/22

3



Precepts

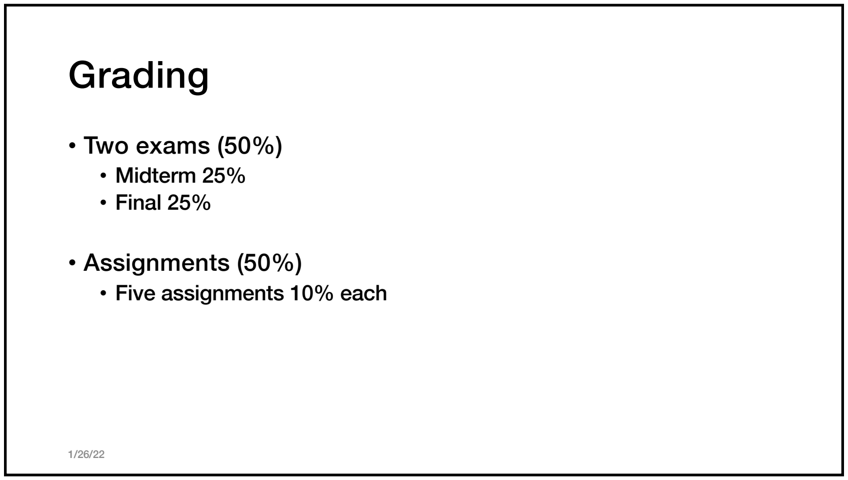
- Wed 7:30am, Thurs 10am, Thurs 12:30pm
  - Not recorded. Slides will be posted.

- Helps with assignments and/or reinforces lecture material

- **Actively** work through problems together

1/26/22

4

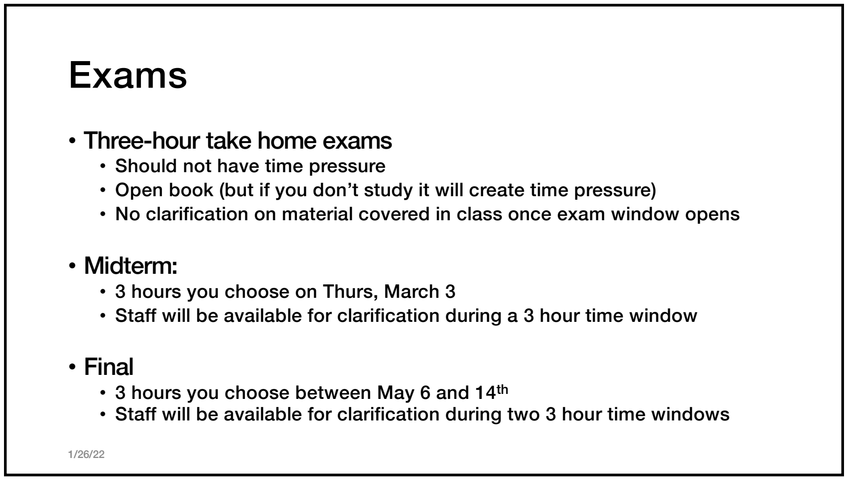


Grading

- Two exams (50%)
  - Midterm 25%
  - Final 25%
- Assignments (50%)
  - Five assignments 10% each

1/26/22

5

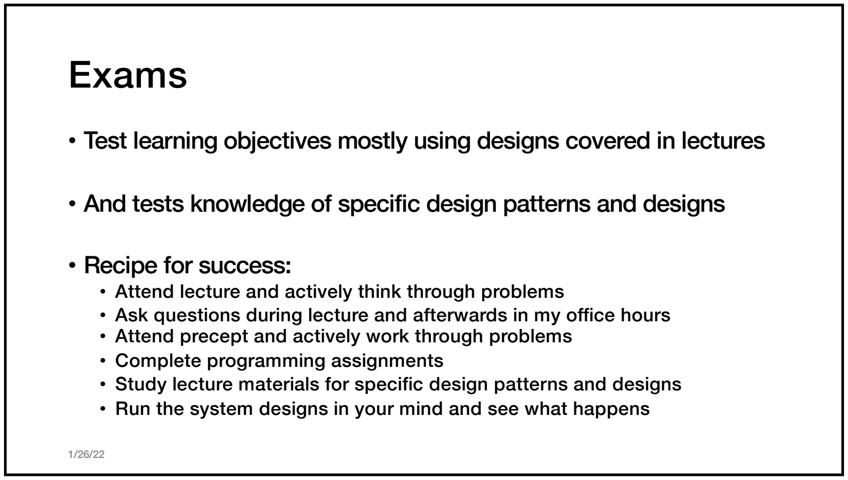


Exams

- Three-hour take home exams
  - Should not have time pressure
  - Open book (but if you don't study it will create time pressure)
  - No clarification on material covered in class once exam window opens
- Midterm:
  - 3 hours you choose on Thurs, March 3
  - Staff will be available for clarification during a 3 hour time window
- Final
  - 3 hours you choose between May 6 and 14th
  - Staff will be available for clarification during two 3 hour time windows

1/26/22

6

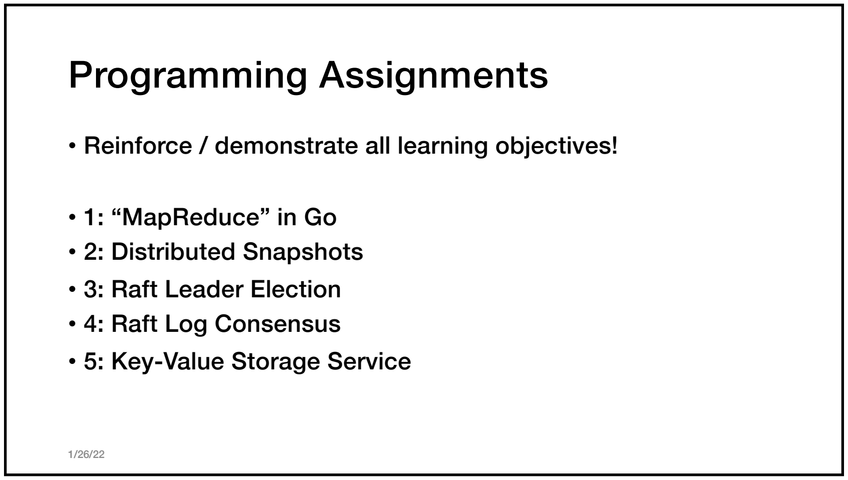


Exams

- Test learning objectives mostly using designs covered in lectures
- And tests knowledge of specific design patterns and designs
- Recipe for success:
  - Attend lecture and actively think through problems
  - Ask questions during lecture and afterwards in my office hours
  - Attend precept and actively work through problems
  - Complete programming assignments
  - Study lecture materials for specific design patterns and designs
  - Run the system designs in your mind and see what happens

1/26/22

7

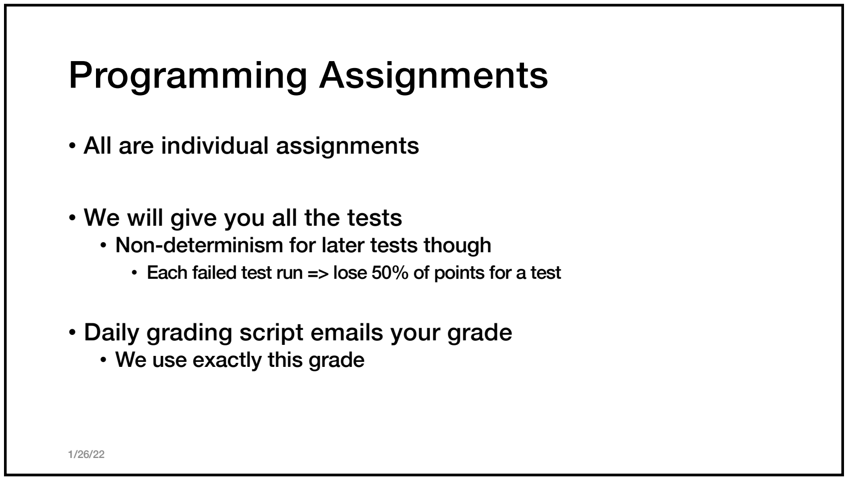


Programming Assignments

- Reinforce / demonstrate all learning objectives!
- 1: "MapReduce" in Go
- 2: Distributed Snapshots
- 3: Raft Leader Election
- 4: Raft Log Consensus
- 5: Key-Value Storage Service

1/26/22

8

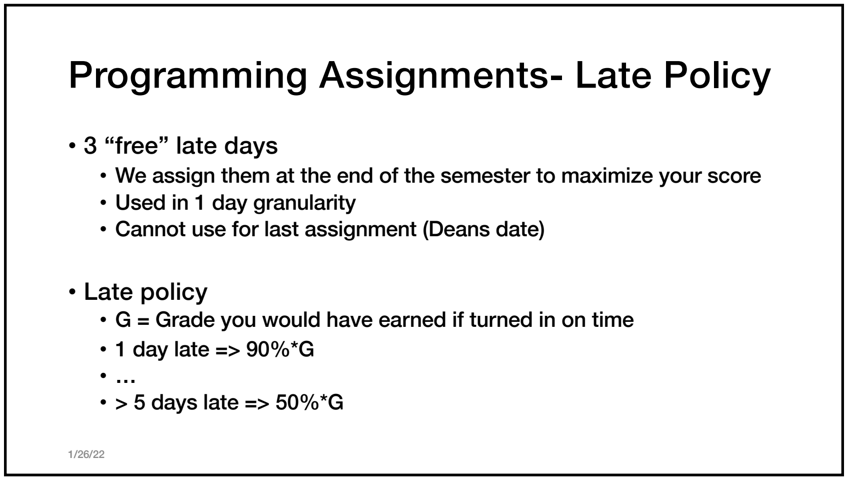


Programming Assignments

- All are individual assignments
- We will give you all the tests
  - Non-determinism for later tests though
    - Each failed test run => lose 50% of points for a test
- Daily grading script emails your grade
  - We use exactly this grade

1/26/22

9

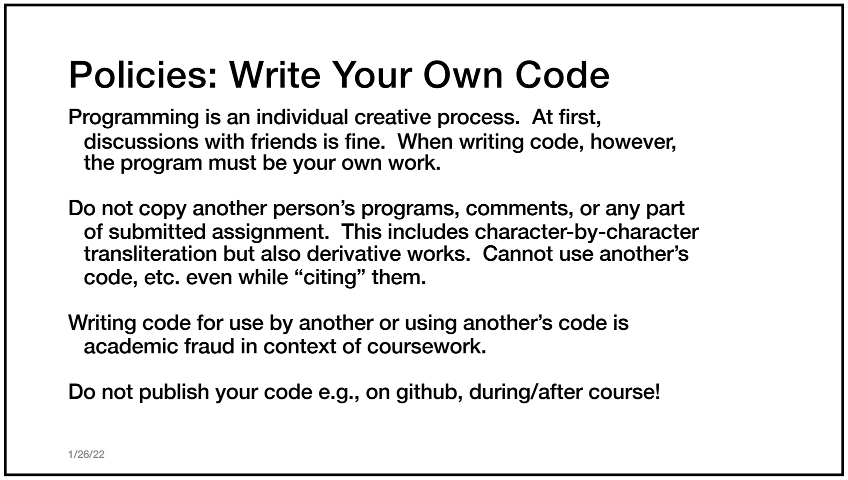


Programming Assignments- Late Policy

- 3 “free” late days
  - We assign them at the end of the semester to maximize your score
  - Used in 1 day granularity
  - Cannot use for last assignment (Deans date)
- Late policy
  - G = Grade you would have earned if turned in on time
  - 1 day late => 90%*G
  - …
  - > 5 days late => 50%*G

1/26/22

10



Policies: Write Your Own Code

Programming is an individual creative process. At first, discussions with friends is fine. When writing code, however, the program must be your own work.

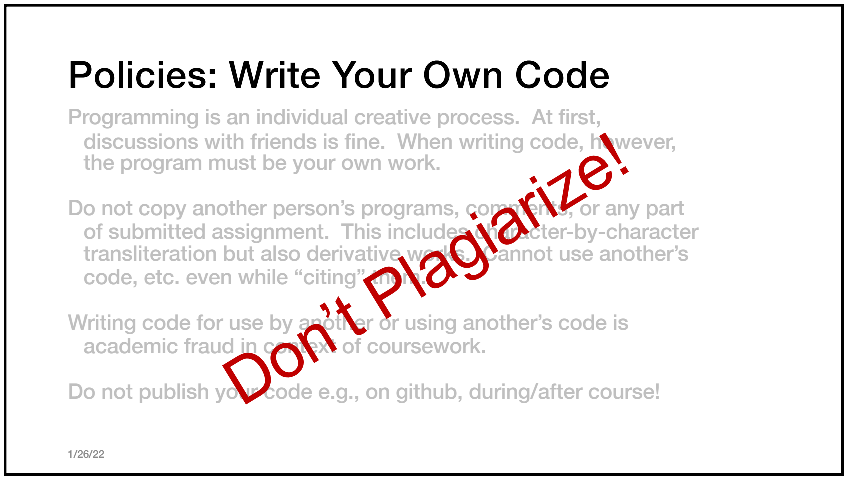
Do not copy another person’s programs, comments, or any part of submitted assignment. This includes character-by-character transliteration but also derivative works. Cannot use another’s code, etc. even while “citing” them.

Writing code for use by another or using another’s code is academic fraud in context of coursework.

Do not publish your code e.g., on github, during/after course!

1/26/22

11



Policies: Write Your Own Code

Programming is an individual creative process. At first, discussions with friends is fine. When writing code, however, the program must be your own work.

Do not copy another person’s programs, comments, or any part of submitted assignment. This includes character-by-character transliteration but also derivative works. Cannot use another’s code, etc. even while “citing” them.

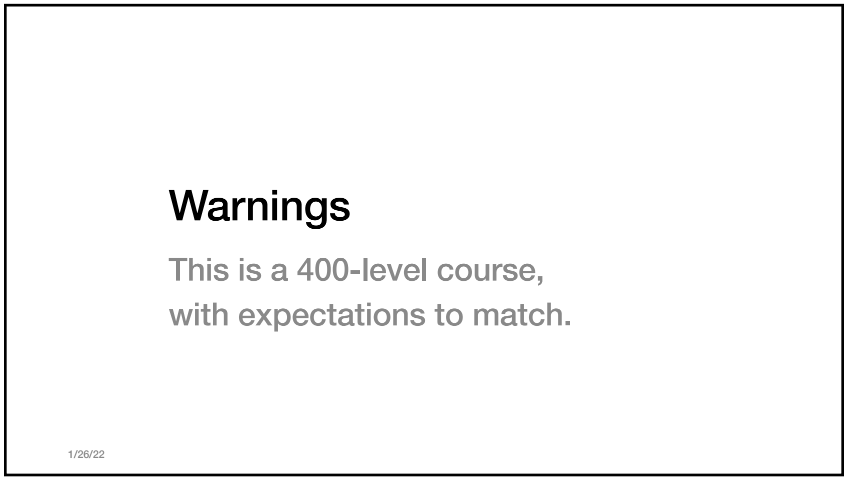
Writing code for use by another or using another’s code is academic fraud in context of coursework.

Do not publish your code e.g., on github, during/after course!

Don’t Plagiarize!

1/26/22

12

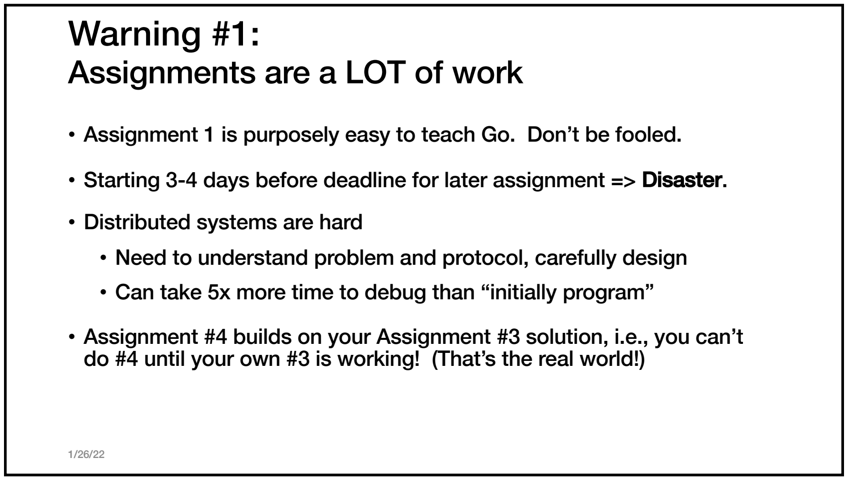


Warnings

This is a 400-level course,
with expectations to match.

1/26/22

13

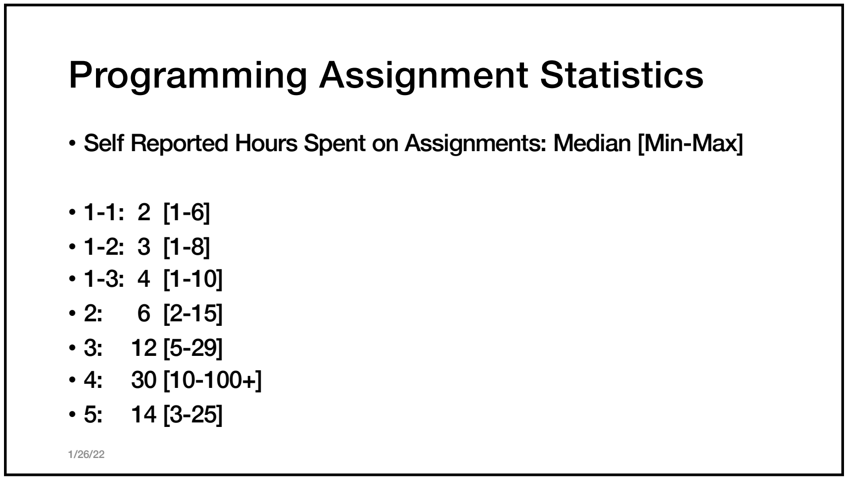


Warning #1:
Assignments are a LOT of work

- Assignment 1 is purposely easy to teach Go. Don't be fooled.
- Starting 3-4 days before deadline for later assignment => **Disaster**.
- Distributed systems are hard
  - Need to understand problem and protocol, carefully design
  - Can take 5x more time to debug than "initially program"
- Assignment #4 builds on your Assignment #3 solution, i.e., you can't do #4 until your own #3 is working! (That's the real world!)

1/26/22

14



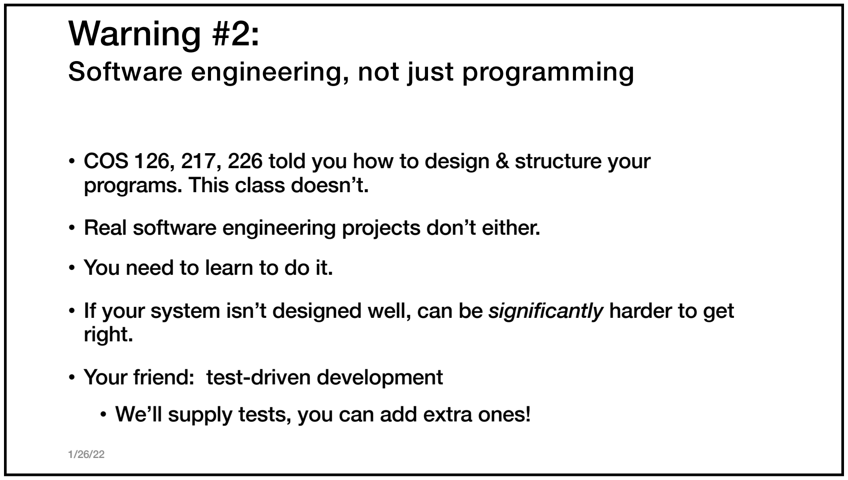
Programming Assignment Statistics

- Self Reported Hours Spent on Assignments: Median [Min-Max]

- 1-1: 2 [1-6]
- 1-2: 3 [1-8]
- 1-3: 4 [1-10]
- 2: 6 [2-15]
- 3: 12 [5-29]
- 4: 30 [10-100+]
- 5: 14 [3-25]

1/26/22

15



Warning #2:
Software engineering, not just programming

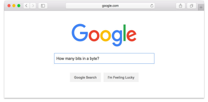
- COS 126, 217, 226 told you how to design & structure your programs. This class doesn't.
- Real software engineering projects don't either.
- You need to learn to do it.
- If your system isn't designed well, can be *significantly* harder to get right.
- Your friend: test-driven development
  - We'll supply tests, you can add extra ones!

1/26/22

16

Warning #3:

Don’t expect 24x7 answers



How many bits in a byte?

- Try to figure out yourself
- Ed discussion not designed for debugging
  - Utilize right venue: Go to office hours
  - Send detailed Q’s / bug reports, not “no idea what’s wrong”
- Staff are not on pager duty 24 x 7
  - Don’t expect response before next business day
  - Questions Friday night @ 11pm should not expect fast responses. Be happy with something before Monday.
- Implications
  - Students should answer each other (+ it’s worth extra credit)
  - Start your assignments early!

1/26/22

17

Programming Assignments

- Recipe for disaster
  - Start day assignment is due
  - Write code first, think later
  - Test doesn’t pass => randomly flip some bits
  - Assume you know what program is doing

1/26/22

18

Programming Assignments

- Recipe for success
  - Start early (weeks early)
  - Think through a complete design
  - Progressively build out your design (using tests to help)
  - Checkpoint progress in git (and to github) frequently
  - Debug, debug, debug
    - Verify program state is what you expect (print it out!)
    - Write your own smaller test cases
    - Reconsider your complete design
  - Attend office hours

1/26/22

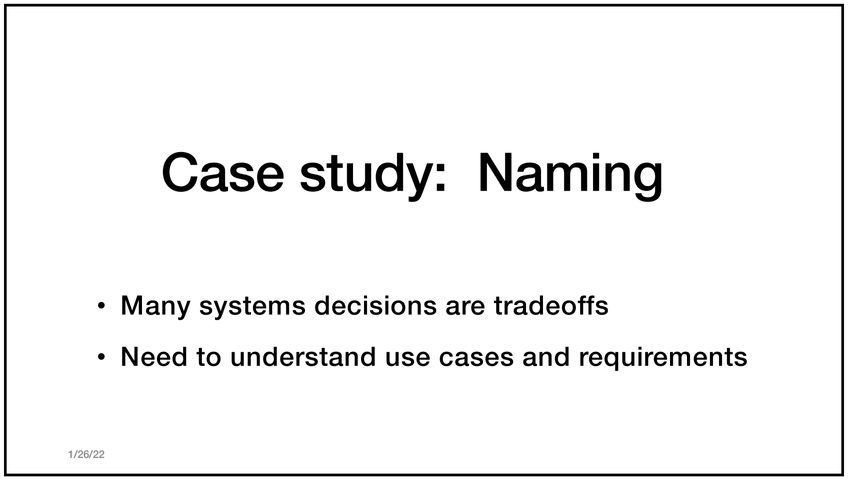
19

Programming Assignment Office Hours

- 10+ hours of office hours per week (TAs + LAs)
- Schedule posted on Ed discussion
- Expectations
  - No: “You have a bug on line 17.”
  - Yes: Helping you think like they would think about a problem

1/26/22

20

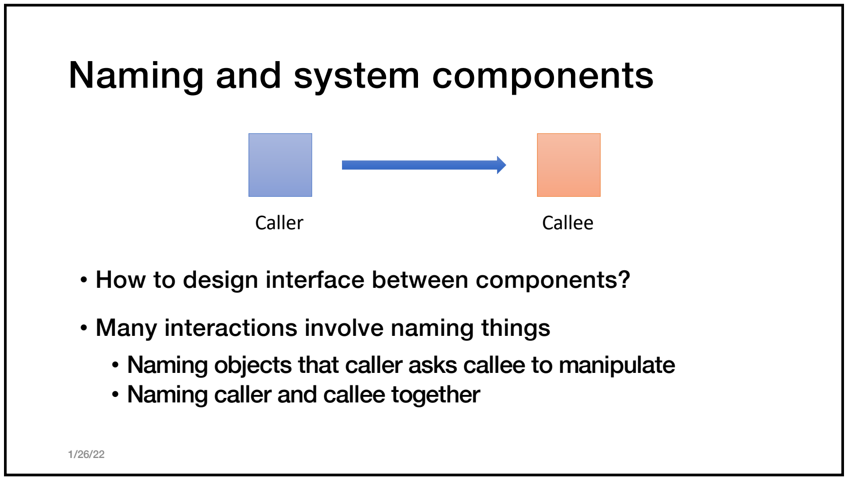


Case study: Naming

- Many systems decisions are tradeoffs
- Need to understand use cases and requirements

1/26/22

21



Naming and system components

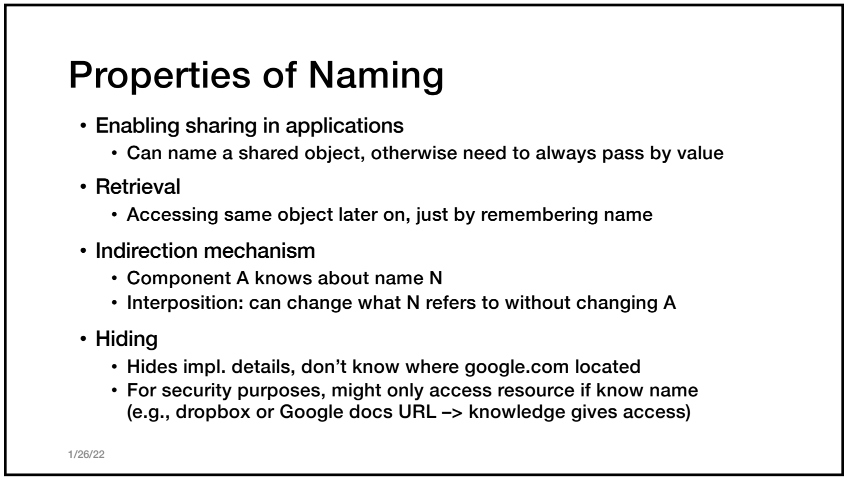
Caller

Callee

- How to design interface between components?
- Many interactions involve naming things
  - Naming objects that caller asks callee to manipulate
  - Naming caller and callee together

1/26/22

22

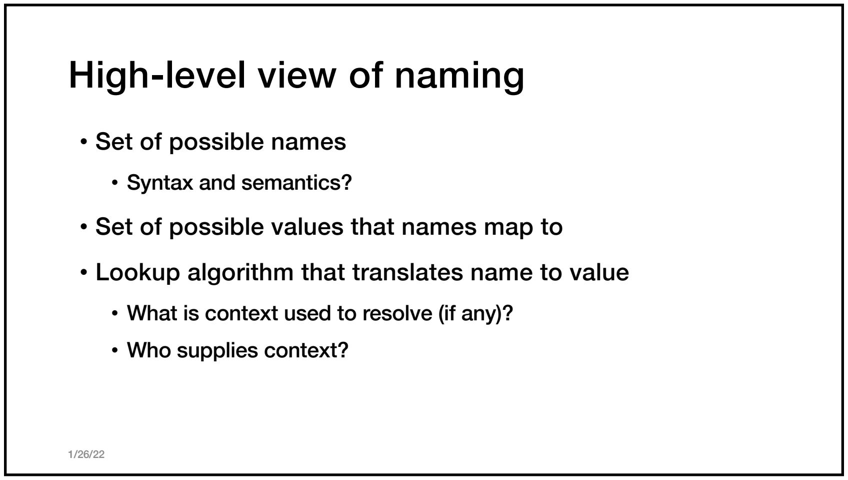


Properties of Naming

- Enabling sharing in applications
  - Can name a shared object, otherwise need to always pass by value
- Retrieval
  - Accessing same object later on, just by remembering name
- Indirection mechanism
  - Component A knows about name N
  - Interposition: can change what N refers to without changing A
- Hiding
  - Hides impl. details, don't know where google.com located
  - For security purposes, might only access resource if know name (e.g., dropbox or Google docs URL –> knowledge gives access)

1/26/22

23

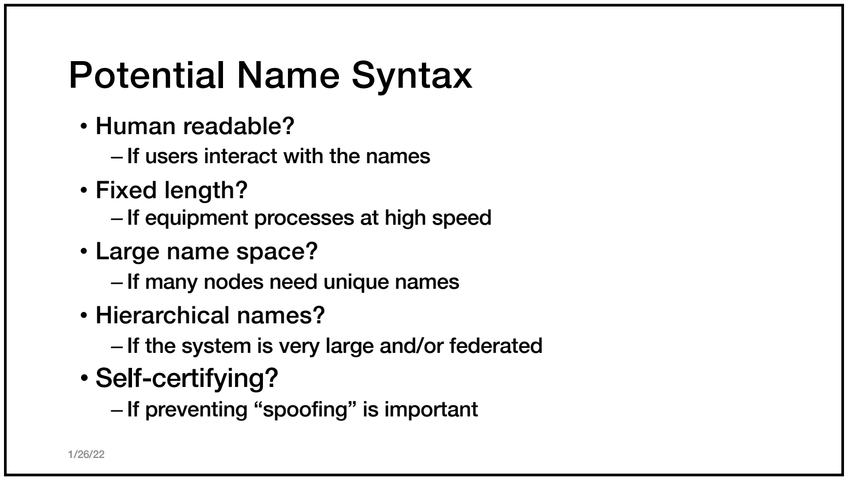


High-level view of naming

- Set of possible names
  - Syntax and semantics?
- Set of possible values that names map to
- Lookup algorithm that translates name to value
  - What is context used to resolve (if any)?
  - Who supplies context?

1/26/22

24

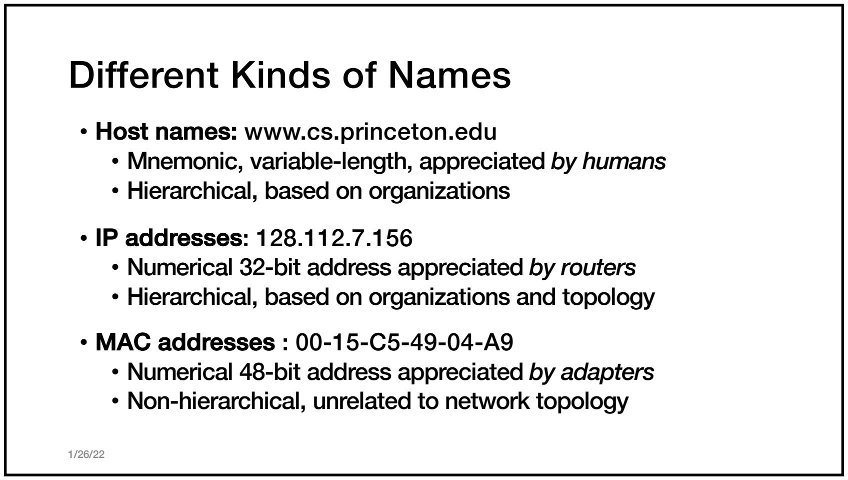


Potential Name Syntax

- Human readable?
  - If users interact with the names
- Fixed length?
  - If equipment processes at high speed
- Large name space?
  - If many nodes need unique names
- Hierarchical names?
  - If the system is very large and/or federated
- Self-certifying?
  - If preventing “spoofing” is important

1/26/22

25

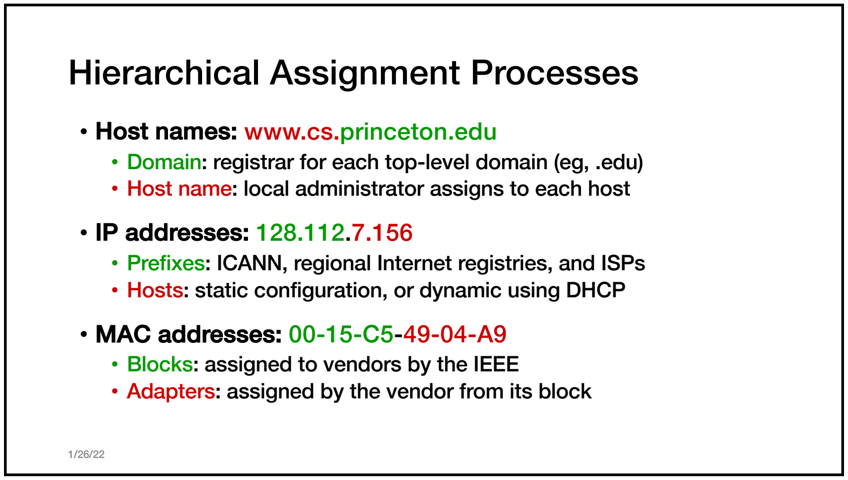


Different Kinds of Names

- **Host names:** www.cs.princeton.edu
  - Mnemonic, variable-length, appreciated *by humans*
  - Hierarchical, based on organizations
- **IP addresses:** 128.112.7.156
  - Numerical 32-bit address appreciated *by routers*
  - Hierarchical, based on organizations and topology
- **MAC addresses** : 00-15-C5-49-04-A9
  - Numerical 48-bit address appreciated *by adapters*
  - Non-hierarchical, unrelated to network topology

1/26/22

26

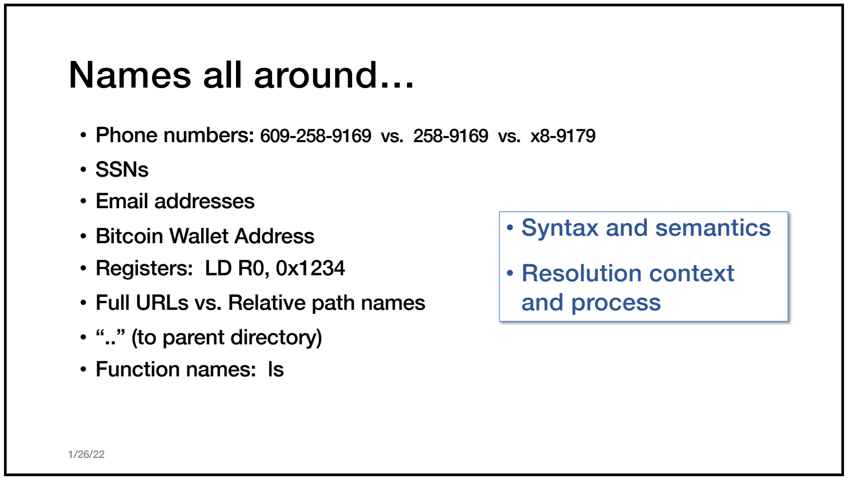


Hierarchical Assignment Processes

- **Host names:** www.cs.princeton.edu
  - Domain: registrar for each top-level domain (eg, .edu)
  - Host name: local administrator assigns to each host
- **IP addresses:** 128.112.7.156
  - Prefixes: ICANN, regional Internet registries, and ISPs
  - Hosts: static configuration, or dynamic using DHCP
- **MAC addresses:** 00-15-C5-49-04-A9
  - Blocks: assigned to vendors by the IEEE
  - Adapters: assigned by the vendor from its block

1/26/22

27



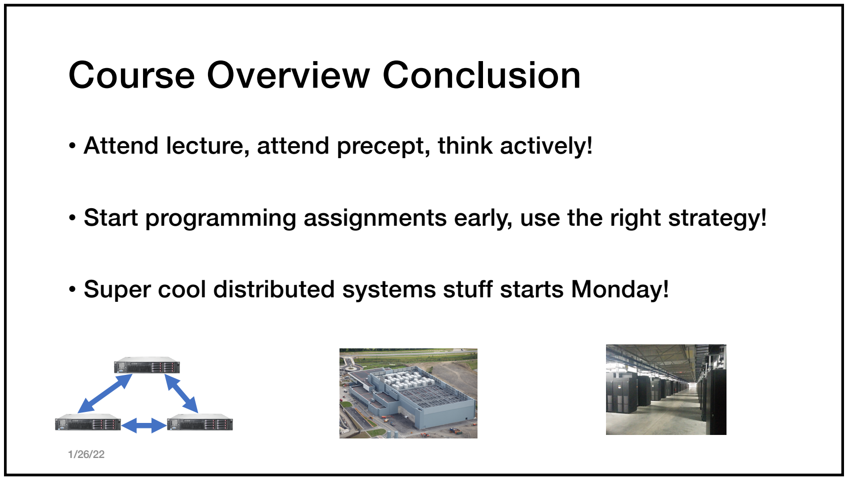
Names all around…

- Phone numbers: 609-258-9169 vs. 258-9169 vs. x8-9179
- SSNs
- Email addresses
- Bitcoin Wallet Address
- Registers: LD R0, 0x1234
- Full URLs vs. Relative path names
- “..” (to parent directory)
- Function names: ls

- Syntax and semantics
- Resolution context and process

1/26/22

28



Course Overview Conclusion

- Attend lecture, attend precept, think actively!
- Start programming assignments early, use the right strategy!
- Super cool distributed systems stuff starts Monday!

1/26/22

29