DAT320
Operating Systems and Systems Programming

Course Introduction — Fall 2021

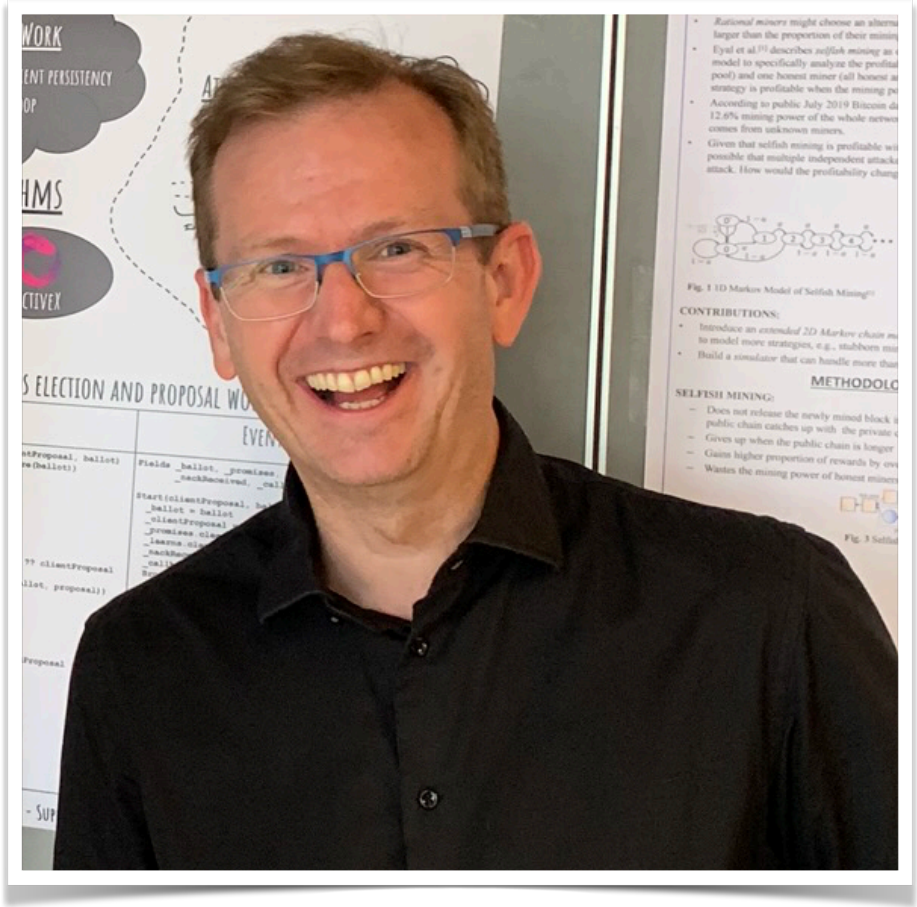
Lecturer: Hein Meling

In This Lecture

- Class overview and administration
- What is an operating system?
- What is a process?

Teaching Staff: Instructor

- Method of communication
  1. Discord
  2. Email: hein.meling@uis.no
  3. Meeting, office: E437



Teaching Assistants

Bachelor and Master Students

- Jan Markus Høie
- Stian Brekken Antonsen
- Jostein Hagen Lindhom
- Oskar Skjærvø Gjølga (Partly Remote)
- Alexander Bjørnum Brynildsen (Remote)
- Hans Erik Frøyland (NTNU, Remote)

PhD students

- **Rodrigo Saramago** (PhD student)
- Hanish Gogada (PhD student)

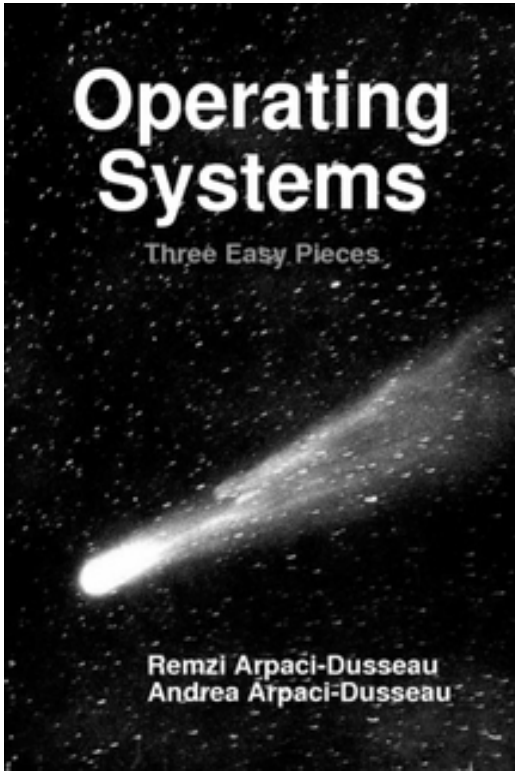
Course Web Site

- https://github.com/dat320-2021/
  - Syllabus
  - Lecture and Lab Plans
  - Sign-up instructions and guides
  - Reading Material and Code Examples from Lectures
  - Discord #announcement channel is the main place for updates

Syllabus

- Chapters from the textbook
- Additional papers and blog posts
- Lab projects
- Slides published throughout the semester

Textbook



- **Operating Systems: Three Easy Pieces**
  Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau
  Arpaci-Dusseau Books
  August, 2018 (Version 1.00)

- http://pages.cs.wisc.edu/~remzi/OSTEP/

- Printed copies available in the campus book store and online (see link)

Textbook Overview

Intro	Virtualization		Concurrency	Persistence	Appendices
Preface	3 *Dialogue*	12 *Dialogue*	25 *Dialogue*	35 *Dialogue*	*Dialogue*
TOC	4 Processes	13 Address Spaces code	26 Concurrency and Threads code	36 I/O Devices	Virtual Machines
1 *Dialogue*	5 Process API code	14 Memory API	27 Thread API code	37 Hard Disk Drives	*Dialogue*
2 Introduction code	6 Direct Execution	15 Address Translation	28 Locks code	38 Redundant Disk Arrays (RAID)	Monitors
	7 CPU Scheduling	16 Segmentation	29 Locked Data Structures	39 Files and Directories	*Dialogue*
	8 Multi-level Feedback	17 Free Space Management	30 Condition Variables code	40 File System Implementation	Lab Tutorial
	9 Lottery Scheduling code	18 Introduction to Paging	31 Semaphores code	41 Fast File System (FFS)	Systems Labs
	10 Multi-CPU Scheduling	19 Translation Lookaside Buffers	32 Concurrency Bugs	42 FSCK and Journaling	xv6 Labs
	11 *Summary*	20 Advanced Page Tables	33 Event-based Concurrency	43 Log-structured File System (LFS)	
		21 Swapping: Mechanisms	34 *Summary*	44 Flash-based SSDs	
		22 Swapping: Policies		45 Data Integrity and Protection	
		23 Complete VM Systems		46 *Summary*	
		24 *Summary*		47 *Dialogue*	
				48 Distributed Systems	
				49 Network File System (NFS)	
				50 Andrew File System (AFS)	
				51 *Summary*	

Lectures and Labs

- Combination of slides, live demo/coding, videos and whiteboard
- Bring your laptop to lectures
- There will be 7 mandatory lab assignments

Weekly Schedule

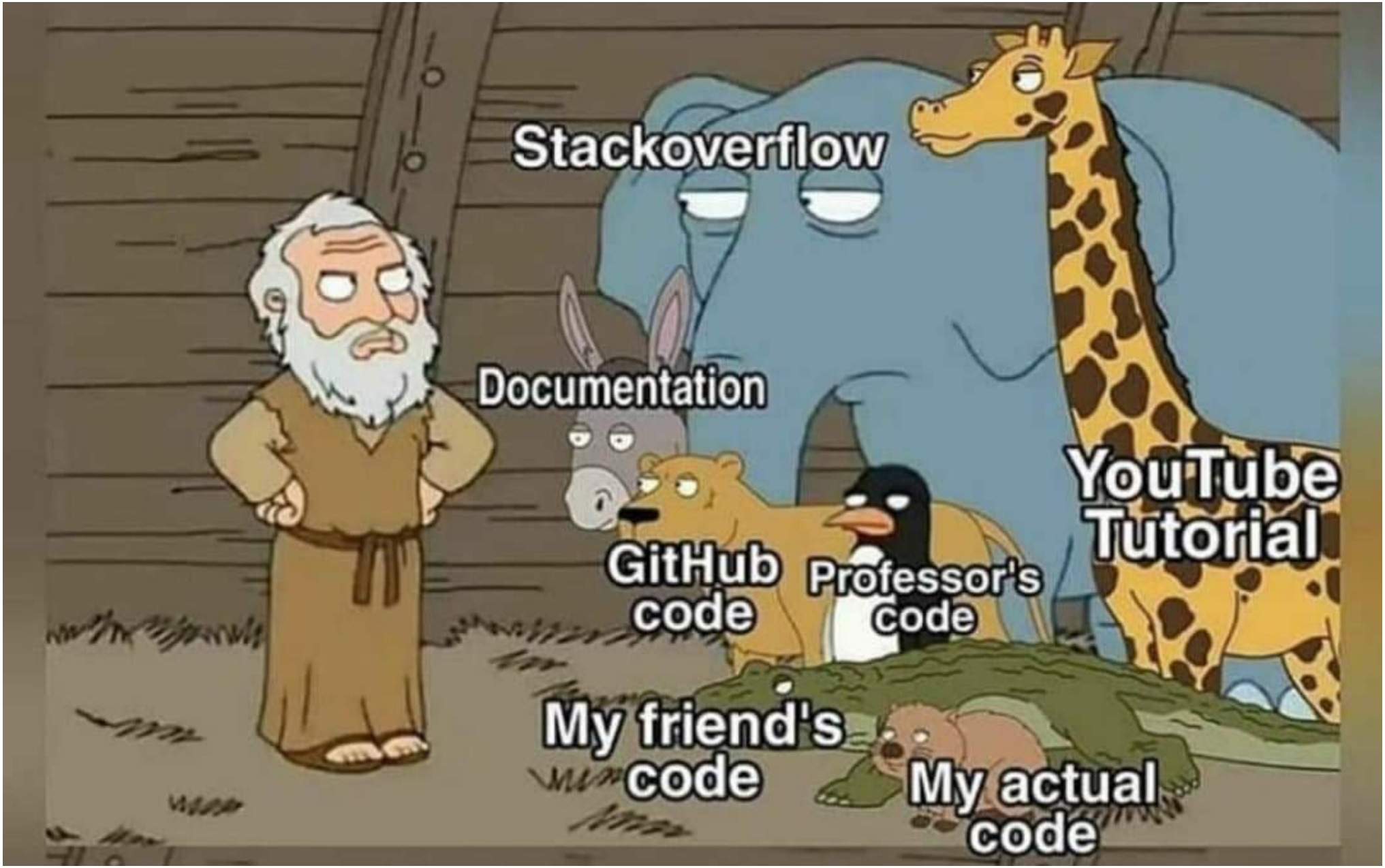
Activity	Day	Time	Room
Lab	Monday	14:15 - 18:00	Discord / E454 / E456
Lab	Tuesday	14:15 - 18:00	Discord / E454 / E456
Lecture	Wednesday	12:15 - 16:00	E102
Lecture	Thursday	08:15 - 10:00	E102
Lab	Friday	12:15 - 16:00	Discord / E454 / E456

Lecture Plan 2021

W	Date	Day	Ch	Topics
34	25.08	Wed		Course Intro and Tutorials
34		Wed	2	Introduction to Operating Systems
34		Wed	4	Abstraction: The Process
34	26.08	Thu	5	Process API
34		Thu	6	Mechanism: Limited Direct Execution
35	01.09	Wed		Introduction to Go programming
35		Wed	7	Scheduling: Introduction
35		Wed	8	Scheduling: Multi-Level Feedback Queue
35	02.09	Thu	9	Scheduling: Proportional Share
36	08.09	Wed		Organizing Go Code
36		Wed	13	Abstraction: Address Spaces
36		Wed	14	Memory API
36	09.09	Thu	15	Mechanism: Address Translation
36		Thu	16	Segmentation
37	15.09	Wed	17	Free-Space Management
37		Wed	18	Paging: Introduction
37	16.09	Thu	19	Paging: Faster Translation (TLBs)
38	22.09	Wed	20	Paging: Smaller Tables
38		Wed	21	Beyond Physical Memory: Mechanisms
38	23.09	Thu	22	Beyond Physical Memory: Policies
39	29.09	Wed	23	Complete Virtual Memory Systems
39		Wed	23	The Mystery of O(N^2) Matrix Traverse (21:58)
39		Wed	23	Buffer Overflow (17:30)
39		Wed	23	Spectre & Meltdown (13:44)
39		Wed	26	Concurrency: Introduction
39	30.09	Thu	27	Thread API
40	06.10	Wed		Rob Pike: Go Concurrency Patterns
40		Wed		Live Coding: Shared Integer w/Mutual Exclusion
40		Wed	28	Locks
40		Wed	29	Lock-based Concurrent Data Structures
40	07.10	Thu	30	Condition Variables
41	13.10	Wed		Fall break -- no lectures planned (may change if U.S. travel still not possible)
41	14.10	Thu		
42	20.10	Wed	31	Semaphores
42		Wed	32	Common Concurrency Problems
42	21.10	Thu		Network Programming with gRPC in Go
43	27.10	Wed	10	Multiprocessor Scheduling
43		Wed	39	Files and Directories
43	28.10	Thu	40	File System Implementation
44	03.11	Wed	48	Distributed Systems
44		Wed	48	Live Coding: TCP Echo Client/Server in Go
44		Wed	49	Sun's Network File System
44	04.11	Thu		
45	10.11	Wed		
45	11.11	Thu		
46	17.11	Wed		
46	18.11	Thu		

Lab Plan for 2021

Lab	Topic	Grading	Approval	Submission	Deadline
1	Introduction to Unix	Pass/fail	Automatic	Individually	September 3
2	Introduction to the C Programming Language	Pass/fail	Automatic	Individually	September 10
3	Introduction to Go Programming	Pass/fail	Automatic	Individually	September 17
4	Scheduling	Pass/fail	TA Approval	Individually	October 8
5	Memory Management	Pass/fail	TA Approval	Group	October 22
6	Concurrency and Parallelism	Pass/fail	TA Approval	Group	November 5
7	Introduction to Docker	Pass/fail	TA Approval	Individually	November 12


Stackoverflow
Documentation
YouTube Tutorial
GitHub code
Professor's code
My friend's code
My actual code


Meling
my working code
What the hell is this?

Lab Assignments: Overview

- First three labs are relatively easy
  - Should be able to do them now, by reading material and watching videos online
- Remaining labs do depend on the course material
  - Should be able to solve them by reading the lab descriptions and content in the text book and possibly other online resources
- **All labs are required** to sit for the exam

Lab Assignments: Requirements

- QuickFeed score of **at least 90%** is required for a lab to be approved
- Automatic Approval: Labs 1-3
- Manual Approval: Lab 4-7
  - May check for plagiarism
  - Expect you can explain your code and show that it runs

Lab Assignments: Approvals

- Approvals queue is managed via Discord (or Whiteboard)
- Approvals can take place both before or *after the deadline*
  - Code must be committed and pushed to GitHub before the deadline

Lab Assignments: Approvals II

- **Before requesting approval**
  - Check that QuickFeed score > 90 %
  - Prepare yourself and your machine (terminal and code editor)
    - To effectively demo your program
    - To effectively show and explain your code

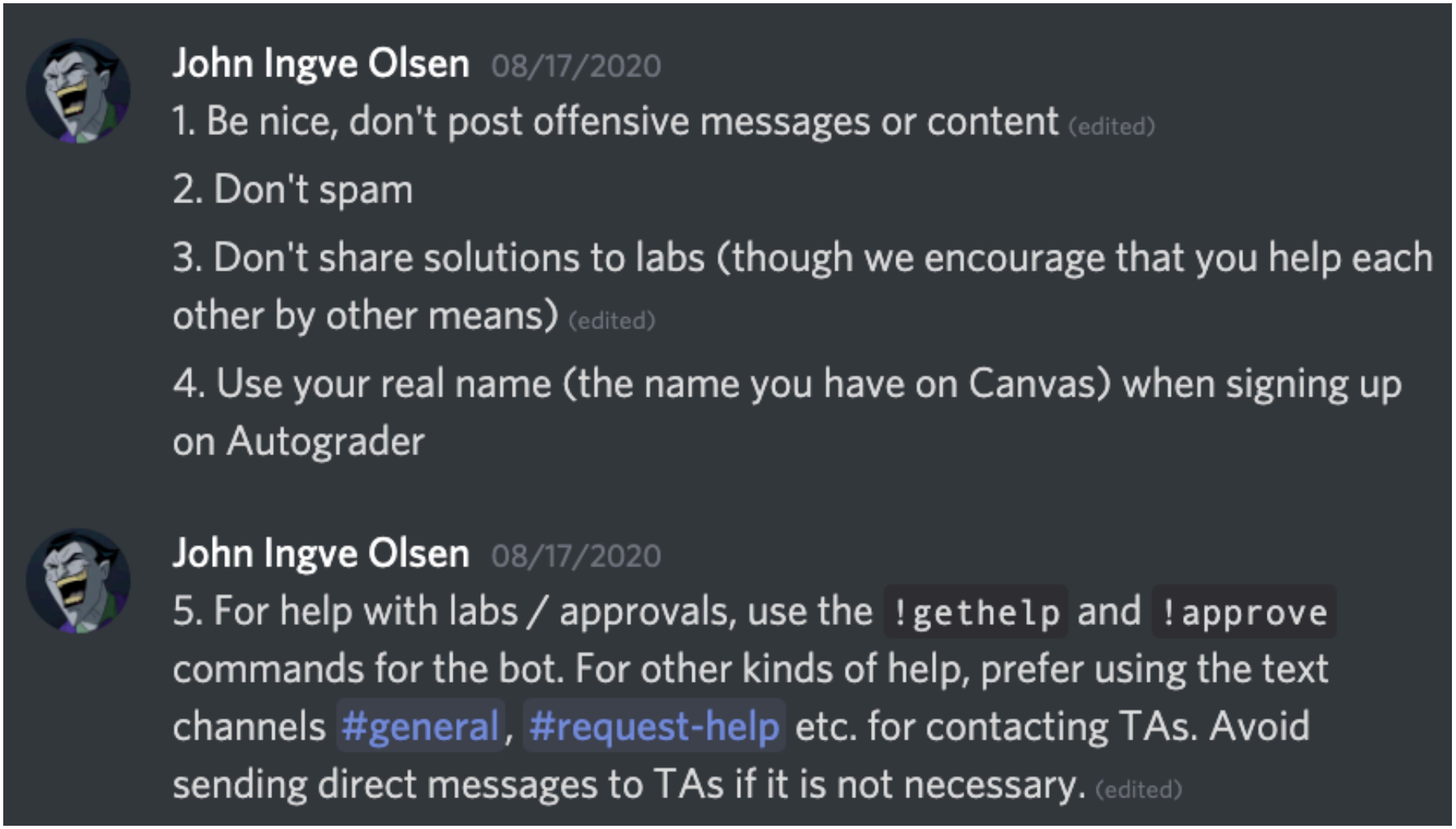
Lab Assignments: Not Approved

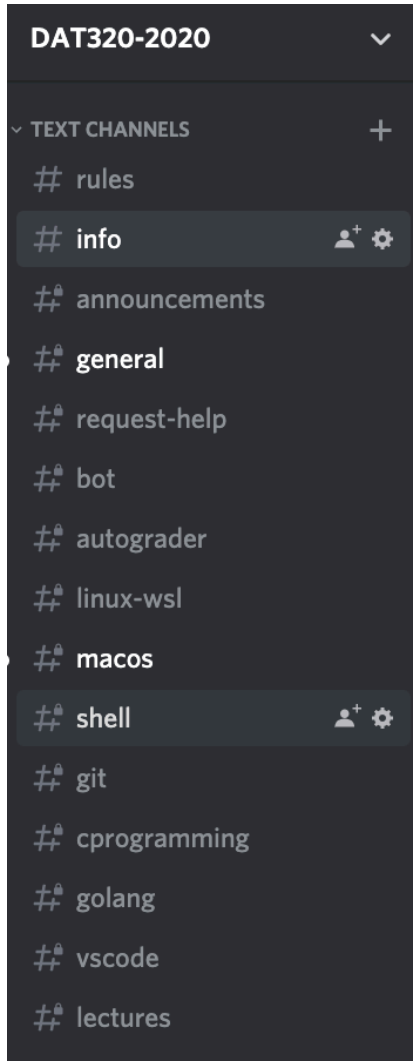
- If a lab is not approved
  - **One additional attempt to approve** for each lab
- But **max three additional attempts** overall

Lab Assignments: Slip Days

- There are **seven (7) slip days** in this course
- Deadlines are recorded as per the last commit for each lab
  - If you continue to work on a lab after the deadline
  - You spend from your slip days budget
- Note: Manual Approvals will typically take place after the last commit.
  - It is okay to continue to fix an issue after the deadline, if you have already received approval, e.g. with 95% and you want to get 100%

Rules of Engagement





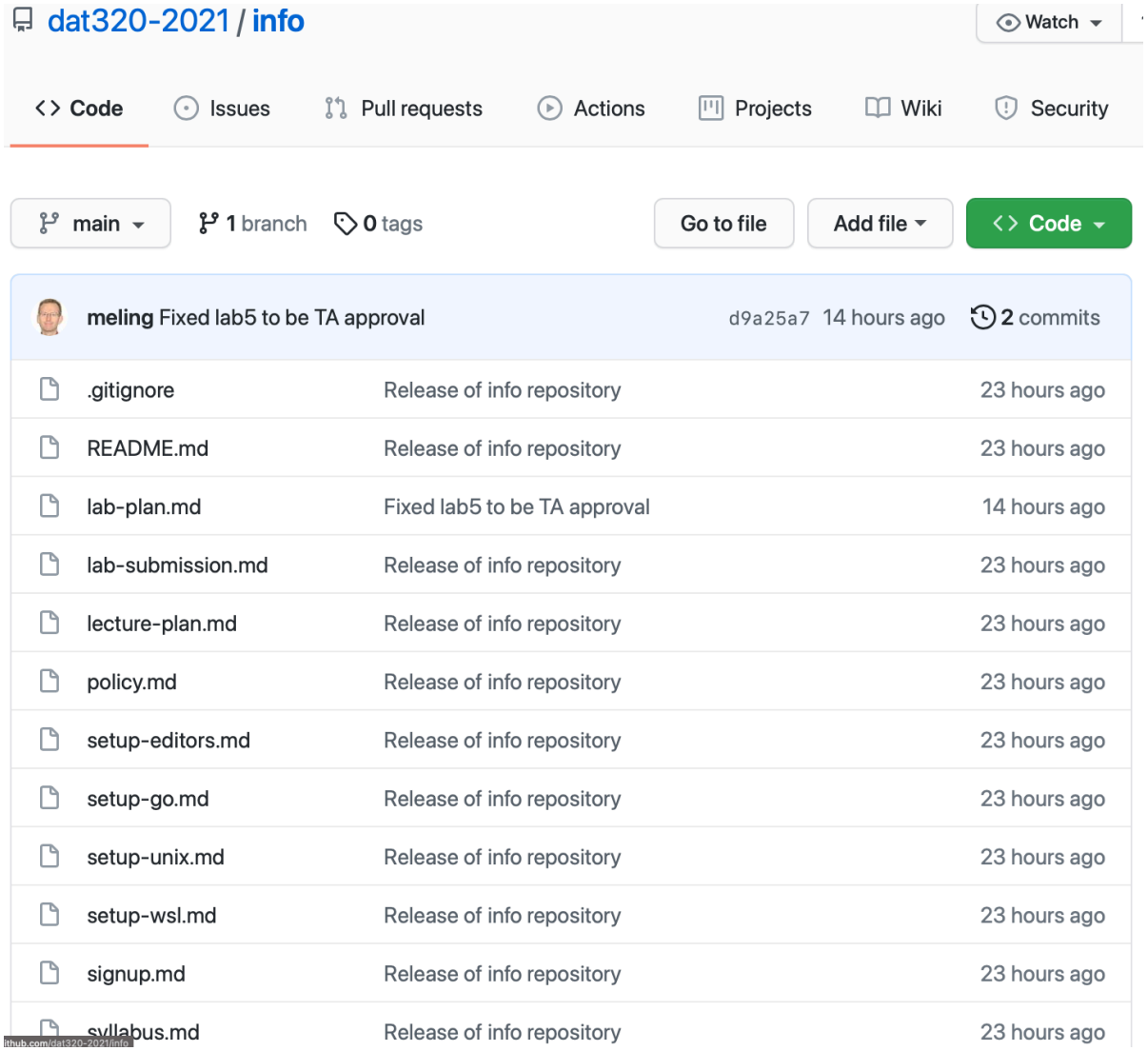
Unix Account Registration

- To complete the lab you'll need a Unix user account
- Go to http://user.ux.uis.no/ today!
- Register ASAP
- Physical access to the Linux lab (E353)
  - PIN code: xxxx

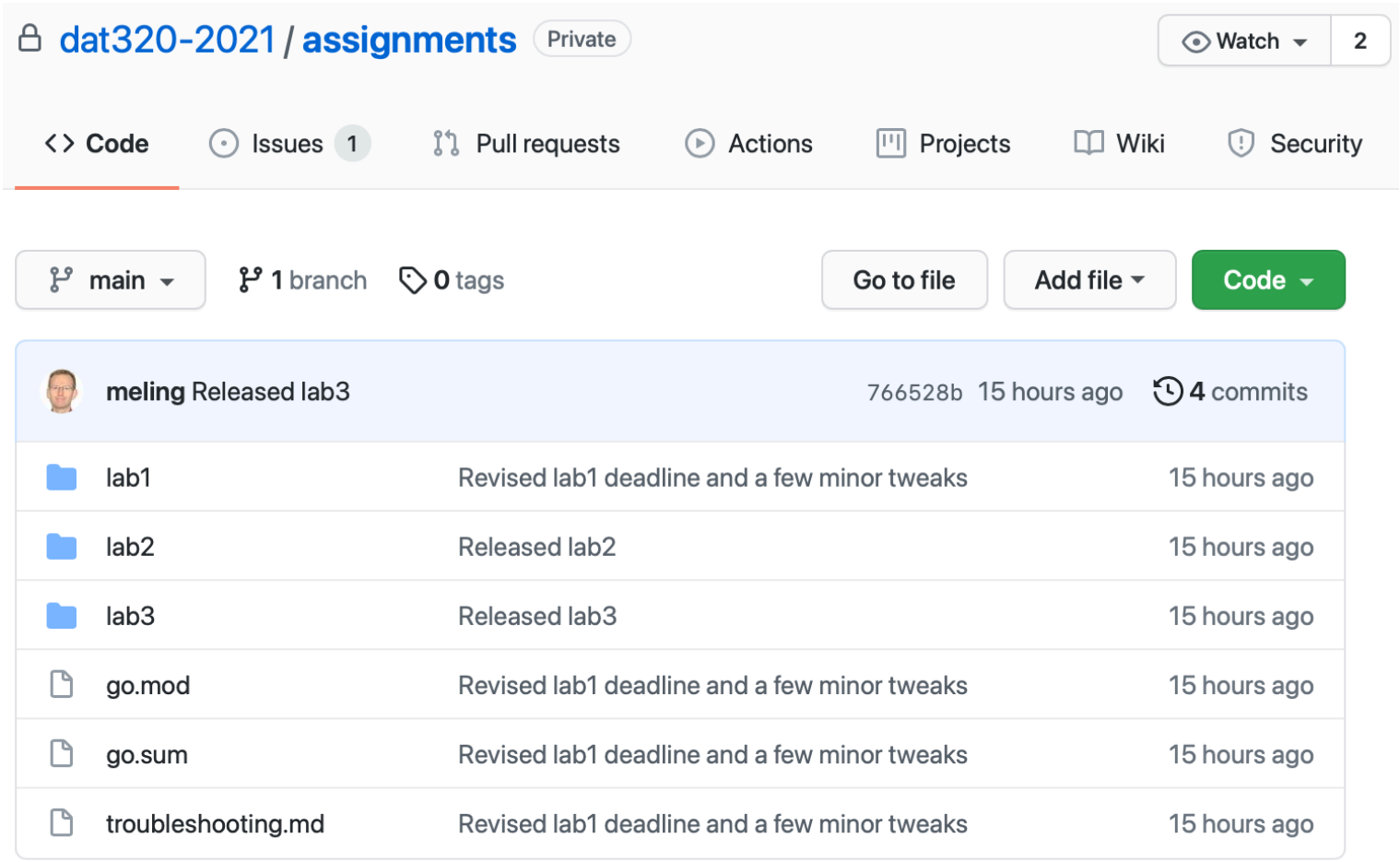
QuickFeed

- We will use QuickFeed to manage
  - Courses, Users and Groups on GitHub
  - Lab submissions, testing and scoring
- It's ready @ http://uis.itest.run/
  - Sign up using your GitHub account
    - Use real name and student number on QuickFeed
  - (you can create a new GitHub account for courses if you prefer)

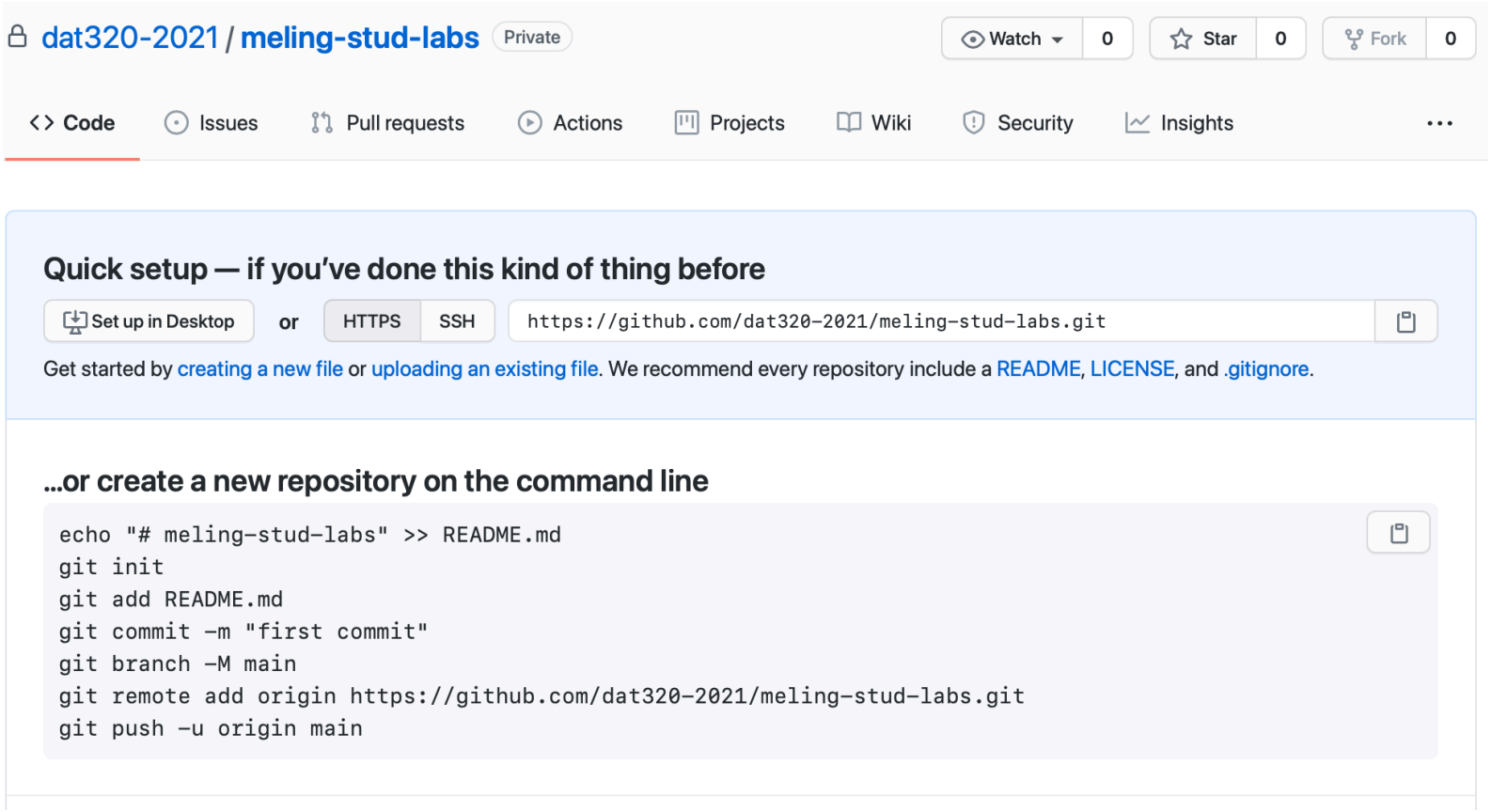
QuickFeed Repositories



QuickFeed Repositories



QuickFeed Repositories



Instructions for Submitting a Lab Assignment to QuickFeed

This section give step-by-step instructions on how to submit assignments. In the following, you are expected to run commands from a terminal environment.

Here are two videos describing these steps: Part 1 (~10 minutes) and Part 2 (~19 minutes).

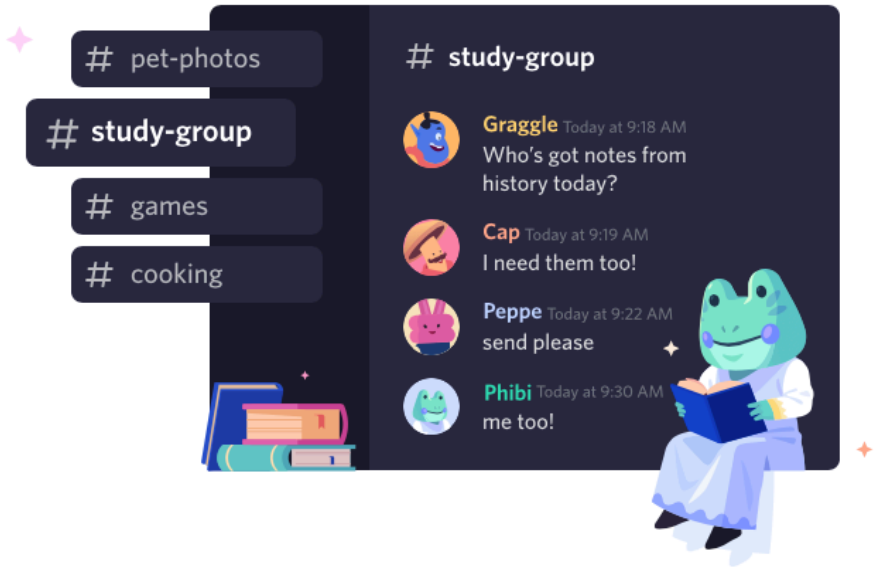
- On macOS, Terminal can be started via Spotlight, by typing the first few letters of `terminal`.
- On Ubuntu Linux, you can click on the Activities item at the top left of the screen, then type the first few letters of `terminal`.
- On Windows, follow these instructions to install the Windows Subsystem for Linux, if you haven't done so already.

Group Registration

- lab5 and lab6 will be carried out in groups
  - General rule: two students per group
  - Working alone or three member groups only accepted by application stating a reason: Send application to Rodrigo on Discord
  - Note each member of a group is expected to contribute equally (see also specific requirements in the policy.md document)
- Registration will be carried out on QuickFeed later

Join Discord

- Instructions for signing up for Discord account and joining the course chat is available on GitHub here



Frank & Ernest by Bob Thaves

Why Go?

Why Go?

- General-purpose programming language
  - Low-level, but garbage collected
- Fast, scalable and easy to learn
  - Designed for ease of reading code
- Builtin support for concurrency
- Easy to build command line tools, clients and servers (microservices)
  - Key enablers for cloud computing services

https://golang.org/

Learning a new Programming Language

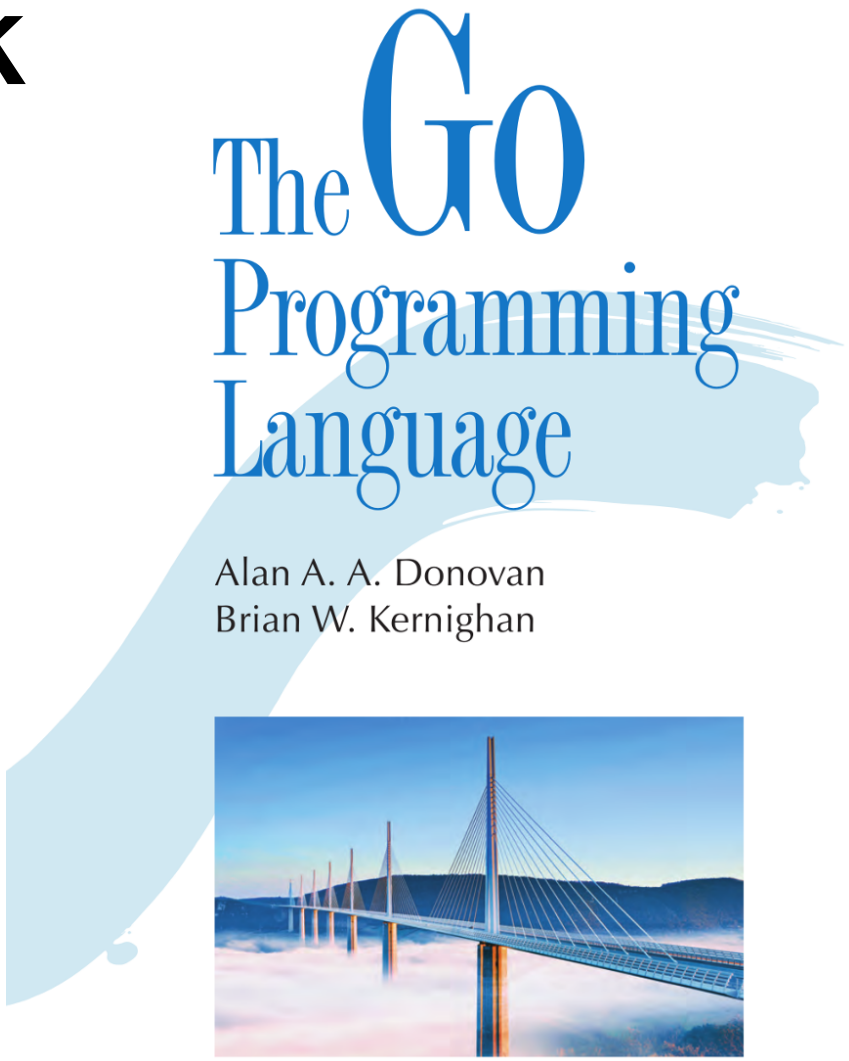
- Impossible to learn systems programming in theory
- You need to *get your hands dirty*!
- Start today!
- Best way to learn is to do real programming!
  - Compiler returns great error messages — great way to learn!

https://golang.org/

Supplement Book

The Go Programming Language
Alan A. A. Donovan and Brian W. Kernighan
Addison-Wesley; ISBN: 978-0134190440
Published Oct 26, 2015

http://gopl.io/



Why Study Operating Systems?