CS592: Introduction to Program Analysis

1. Introduction

Kihong Heo

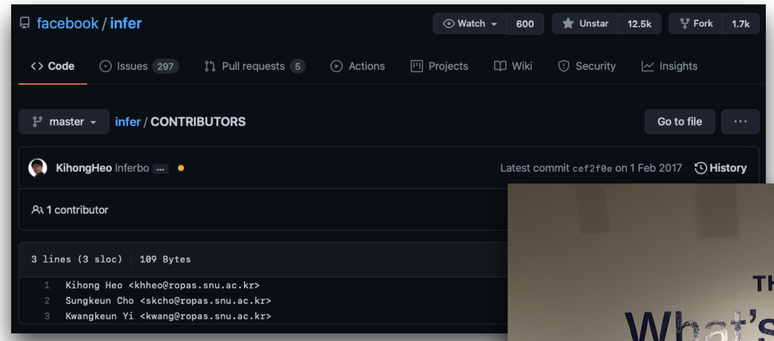
KAIST

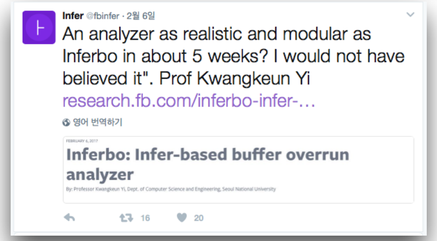
About Me

- Instructor: Kihong Heo (허기홍, kihong.heo@kaist.ac.kr)
- KAIST CS / GSIS / Programming Systems Lab.
- Homepage: https://kihongheo.kaist.ac.kr / https://prosys.kaist.ac.kr
- Office: N5 2321
- Office Hours: Mon 10:30 - 11:30 at Zoom (by appointment)

My Research

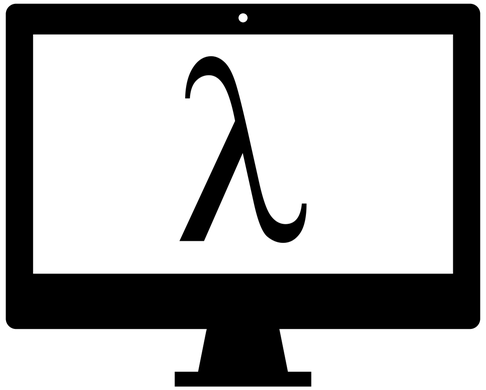
- Goal: solid PL theories ⟺ powerful programming systems
- Keywords: program analysis, programming language, SW security
- Good (also fierce) memories:





*https://research.fb.com/blog/2017/02/inferbo-infer-based-buffer-overrun-analyzer/

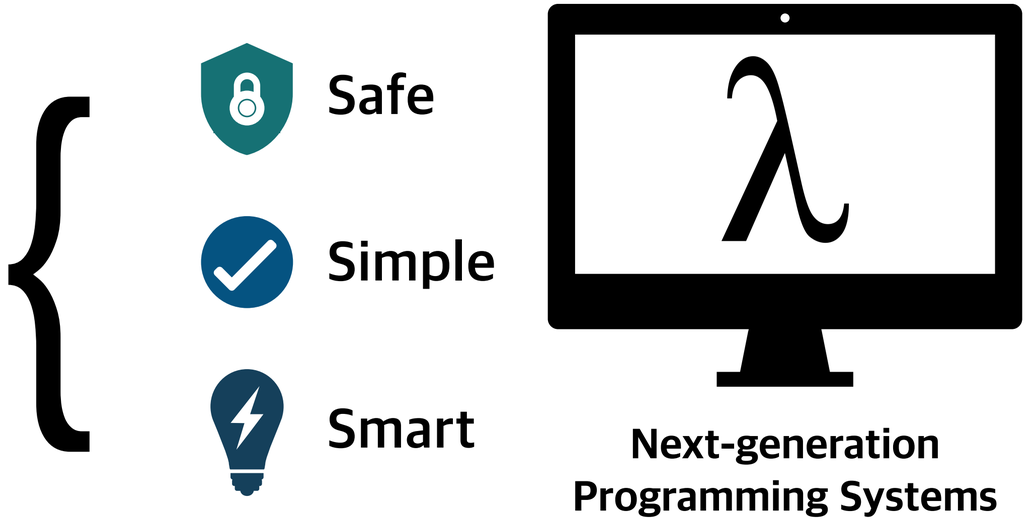
My Research



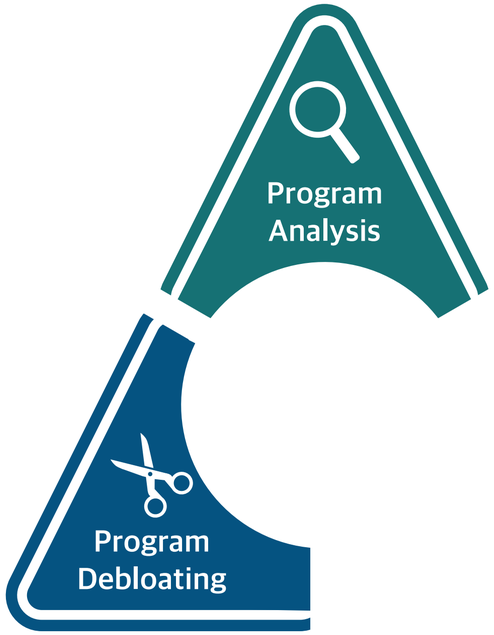
Next-generation
Programming Systems

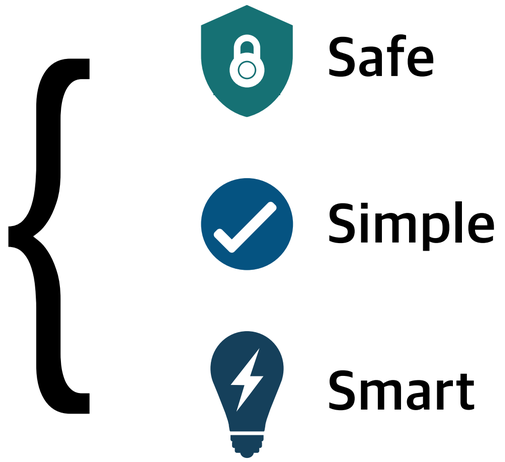
My Research

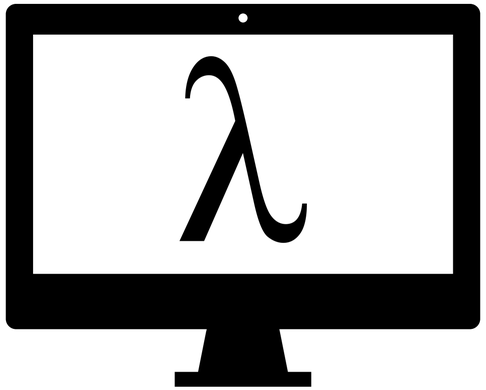

Program
Analysis


Safe
Simple
Smart
Next-generation
Programming Systems

My Research

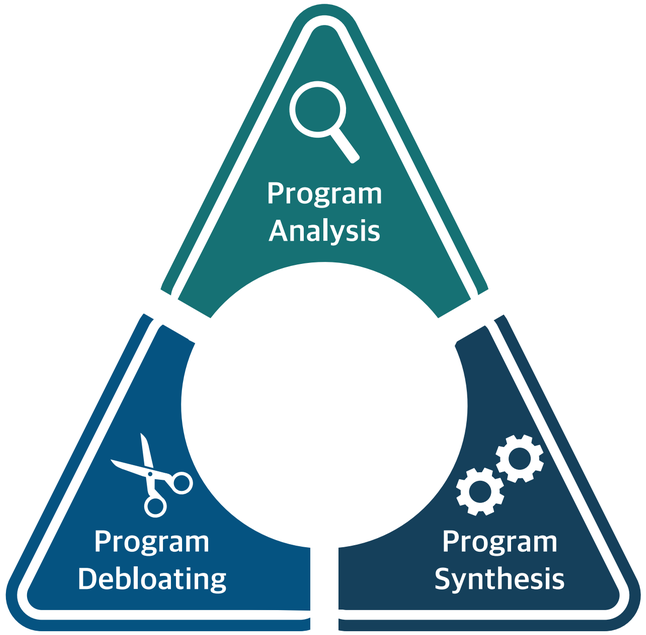






Next-generation Programming Systems

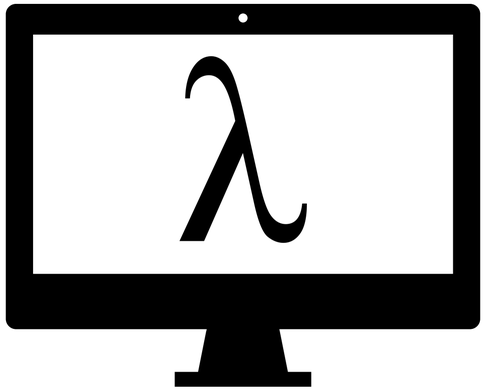
My Research



Safe

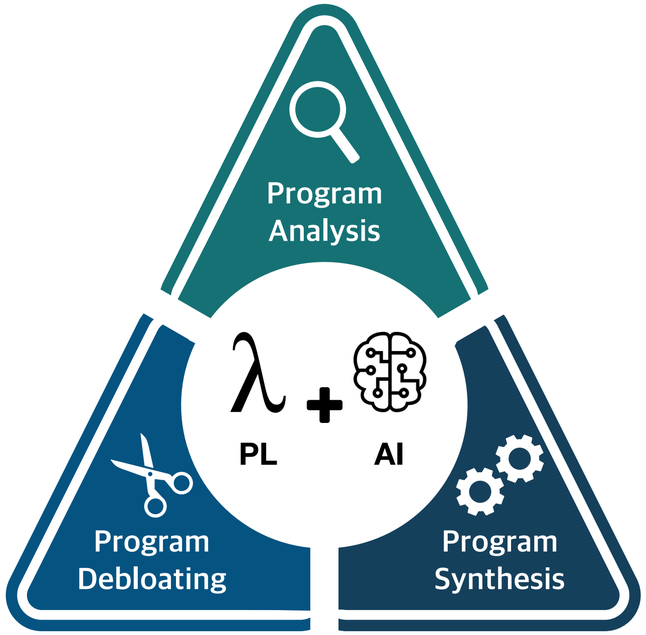
Simple

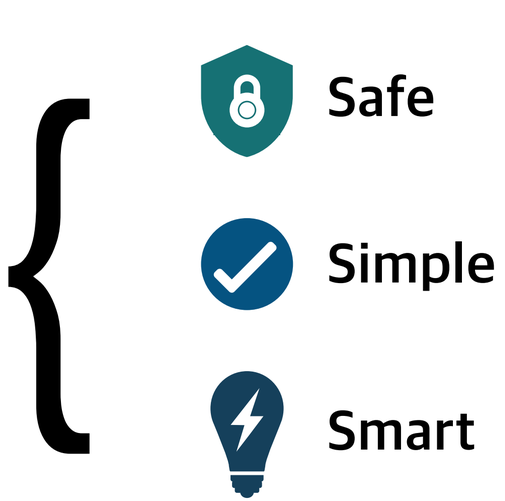
Smart

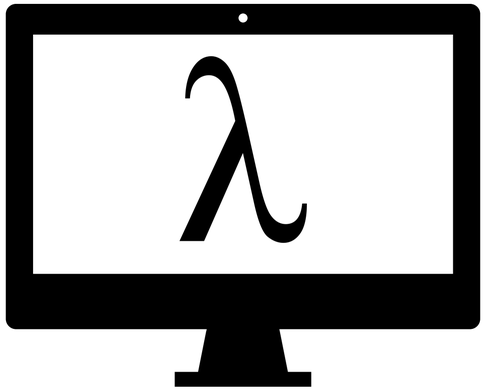


Next-generation Programming Systems

My Research







Next-generation Programming Systems

Course Information

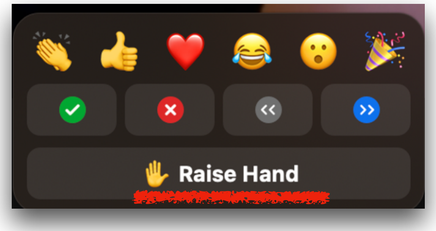
- Course Website: https://github.com/prosyslab-classroom/cs592-2021-fall
- Q&A Board: https://github.com/prosyslab-classroom/cs592-2021-fall/issues
- TAs (mailing list: ta.2021f@prosys.kr)
  - Hyunsu Kim (김현수, hyunsu.kim00@kaist.ac.kr)
  - Wooseok Kang (강우석, kangwooseokeq@kaist.ac.kr)
- Textbook:
  - Lecture slides will be provided
  - Xavier Rival and Kwangkeun Yi,
    Introduction to Static Analysis: an Abstract Interpretation Perspective, MIT Press, 2020

Important Notice (1): Academic Integrity

- DO NOT share the course contents (e.g., assignments or exams) with others
  - Esp., Github public repository, chegg.com, etc
- DO NOT discuss the details of solutions with others
- Any integrity violation: at LEAST F
- If you have questions: QnA board > TAs > instructor

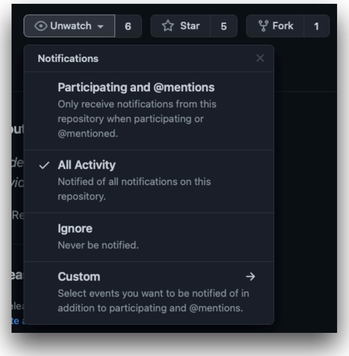
Important Notice (2): In-class Operations

- Prerequisites:
  - Prepare your camera (on by default) and mic (off by default)
  - Use your name in your Zoom profile
- Questions (either in Korean or in English):
  - Raise hand and speak up, or
  - Write your question in a chat
- Reaction:
  - E.g., Thumb up (virtually or physically)



Important Notice (3): Out-of-class

- All Q&A and public notices: Github issue board
  - “Watch” all notifications
- Private notices (grading, homework links, etc): KLMS
- Questions are always welcome except for
  - Too detailed ones (TAs are not debuggers!)
  - Directly related to the solutions



A long time ago
in a galaxy far, far away….

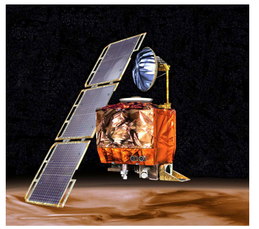
SOFTWARE BUGS

Software Bugs: A Persistent Problem

- A long time ago, far far away

The Patriot Missile (1991)
Floating-point roundoff
28 soldiers died

The Ariane-5 Rocket (1996)
Integer Overflow
$100M



NASA's Mars Climate Orbiter (1999)
Meters-Inches Miscalculation
$125M

- Unfortunately, it becomes your own problem now

Software Bugs: A Persistent Problem

- A long time ago, far far away
- U

COST OF A SOFTWARE BUG

$100

If found in Gathering Requirements phase

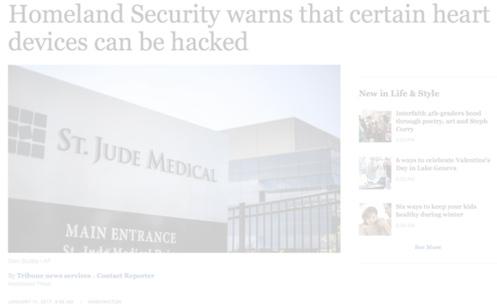
$1,500

If found in QA testing phase

$10,000

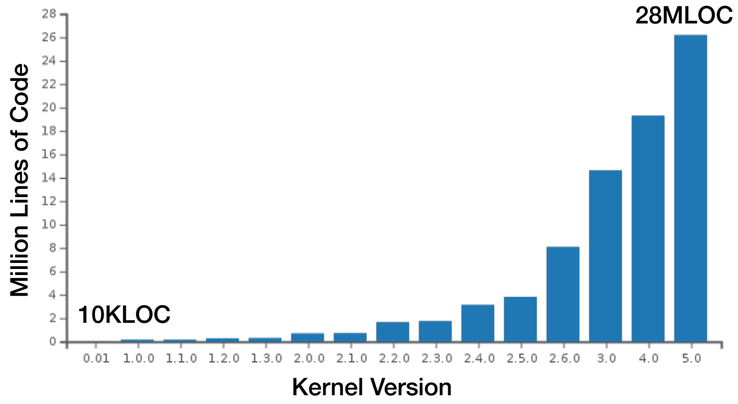
If found in Production

- IBM Systems Sciences Institute, 2015

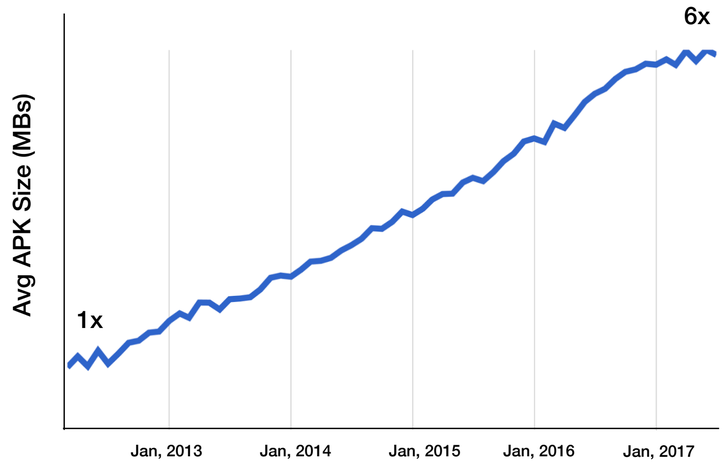


Why Software Still Fails?

Size of Linux Kernel



Avg. Size of Android Apps



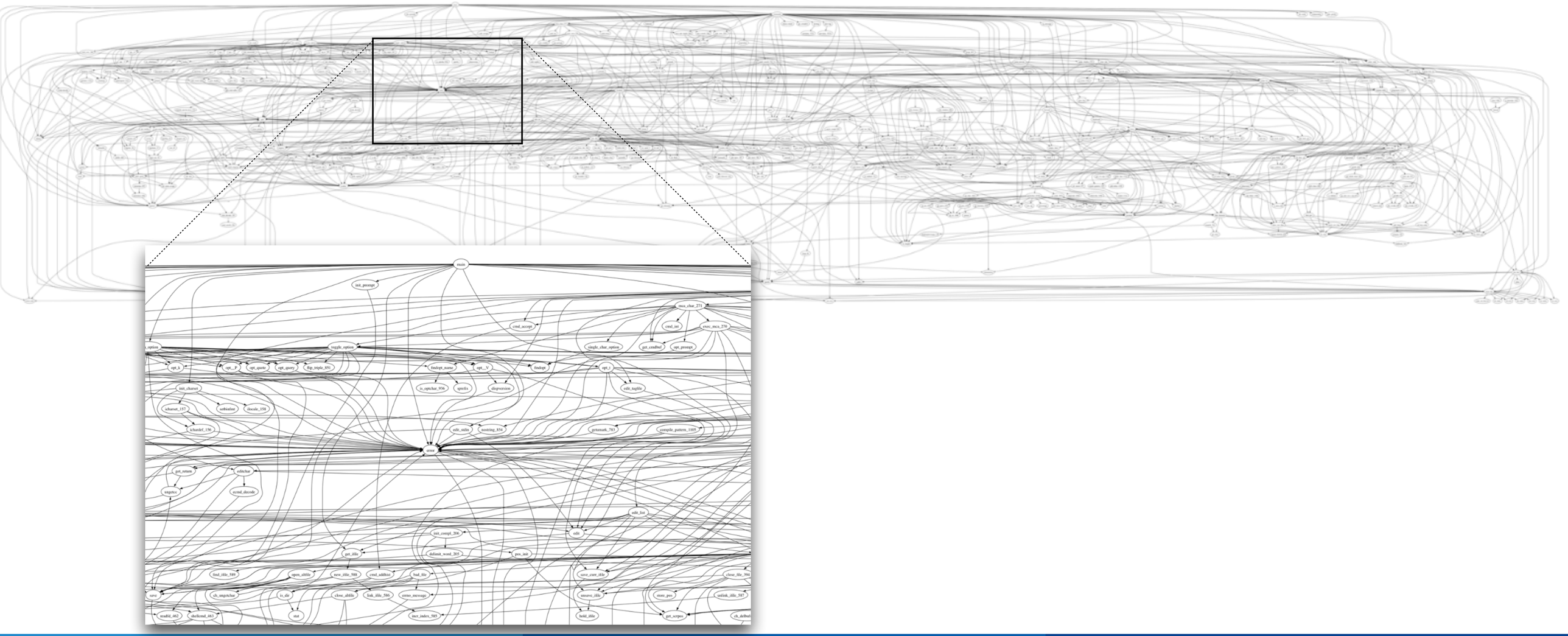
X

GitHub

10M+ New Developers
44M+ New Repositories
87M+ New Pull Requests
in 2019

Software Complexity

less-382 (23,822 LOC)



Course Objectives: Principles

Q: How to **formally estimate** software behavior automatically before its execution?

Course Objectives: Principles

Artifact

Subject

Principle

$$\vec{F} = m\vec{a}$$

$$\nabla \cdot E = 0 \quad \nabla \times E = -\frac{1}{c}\frac{\partial H}{\partial t}$$

$$\nabla \cdot H = 0 \quad \nabla \times H = \frac{1}{c}\frac{\partial E}{\partial t}$$

Static Program Analysis

- General methodology to predict software behavior
  - **static**: before execution
  - **automatic**: software is analyzed by software (program analyzer)
  - **systematic**: foundational theory (Abstract Interpretation)
- Applications:
  - bug-finding, verification, code optimization, etc

“SW MRI”

Success Stories

Domain-specific Verification

General-purpose Bug-finding

Windows Device Driver

Microsoft

Astrée

Airbus Controller

ENS / AbsInt

coverity®

Stanford / Synopsys

Infer

Facebook

Sparrow

SNU / Fasoo.com

PolySpace TECHNOLOGIES

Mathworks

CODESONAR

GrammaTech

lgtm.com by Semmle™

Semmle / Github

JULIA

JuliaSoft

GCC

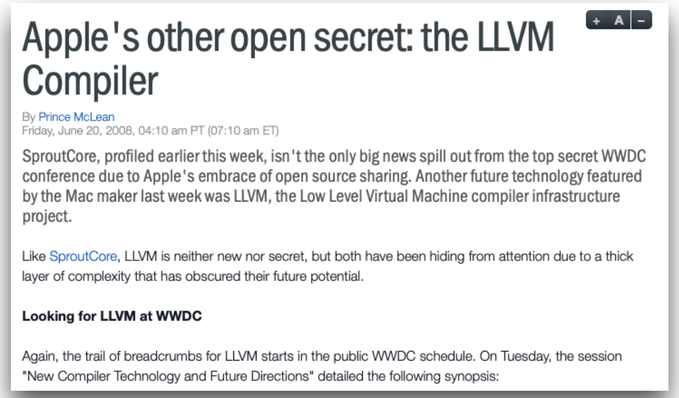
LLVM/Clang

Course Objectives: Practice & Challenge

- Homework: **design** & **implement** program analyzers
  - 6 (main) + 1 (dummy) assignments
- Programming assignments in OCaml using LLVM
  - You will write your analyzers in OCaml
  - Your analyzer will analyze LLVM IR code
- Why LLVM? (https://llvm.org)
- Why OCaml? (https://ocaml.org)

The LLVM Compiler Infrastructure

- The de-facto standard & well-structured compiler toolchain
  - parser, code optimizer, linker, loader, debugger, etc
- A wide variety of frontends: C/C++, Obj-C, Swift, Fortran, etc
  - translated to the LLVM IR (intermediate representation)



Apple's other open secret: the LLVM Compiler

By Prince McLean
Friday, June 20, 2008, 04:10 am PT (07:10 am ET)

SproutCore, profiled earlier this week, isn't the only big news spill out from the top secret WWDC conference due to Apple's embrace of open source sharing. Another future technology featured by the Mac maker last week was LLVM, the Low Level Virtual Machine compiler infrastructure project.

Like SproutCore, LLVM is neither new nor secret, but both have been hiding from attention due to a thick layer of complexity that has obscured their future potential.

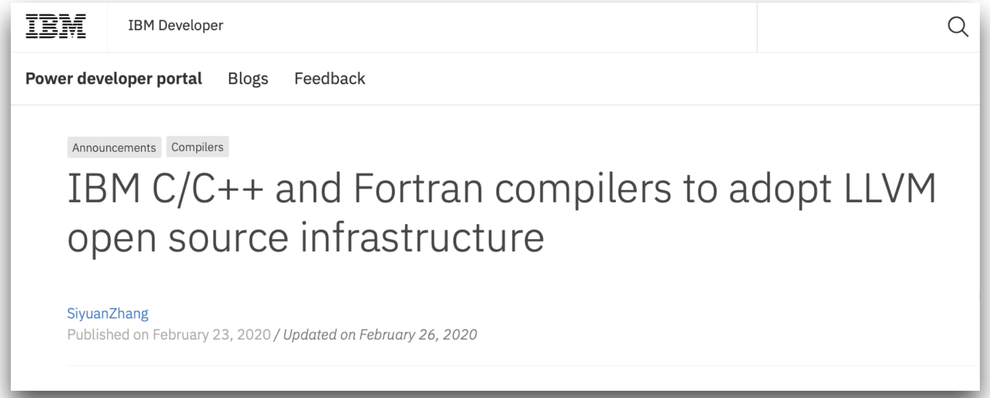
Looking for LLVM at WWDC

Again, the trail of breadcrumbs for LLVM starts in the public WWDC schedule. On Tuesday, the session "New Compiler Technology and Future Directions" detailed the following synopsis:

Google Chrome is replacing Microsoft's C++ compiler with Clang

By Muhammad Jarir Kanji · Mar 6, 2018 14:06 EST · HOT!

Alongside bringing better touch support and automatic ad-blocking for 'intrusive' ads to the desktop version of Chrome, Google is also making some changes to its browser under the hood. The company is now starting to build Chrome for Windows using the Clang compiler which it already uses for other platforms like macOS and Linux.



IBM Developer

Power developer portal Blogs Feedback

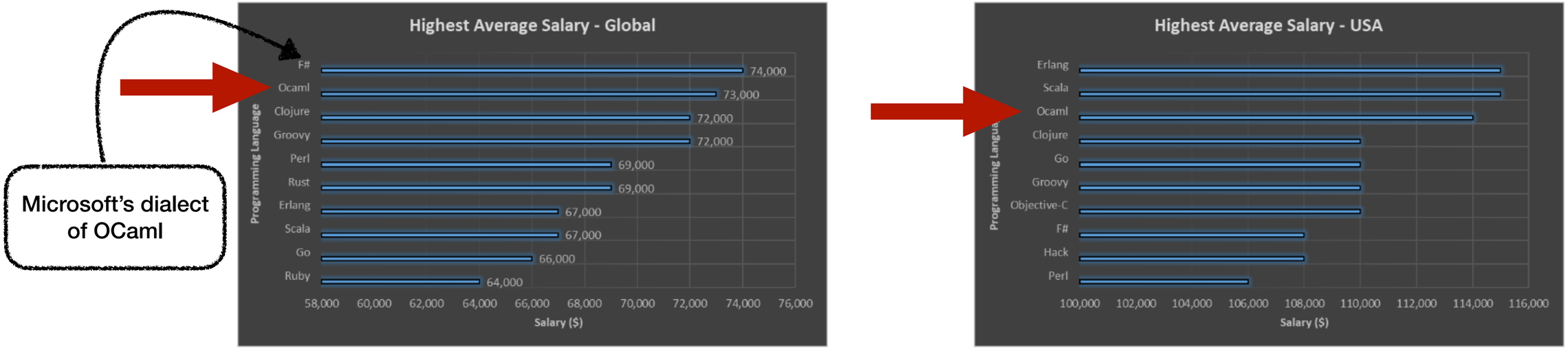
Announcements Compilers

IBM C/C++ and Fortran compilers to adopt LLVM open source infrastructure

SiyuanZhang
Published on February 23, 2020 / *Updated on February 26, 2020*

The OCaml Language

- Simple, safe, realistic and high-level programming language
- Official OCaml bindings to LLVM API supported
- A lot of growing demands from academia and industry



StackOverflow, 2018

Homework

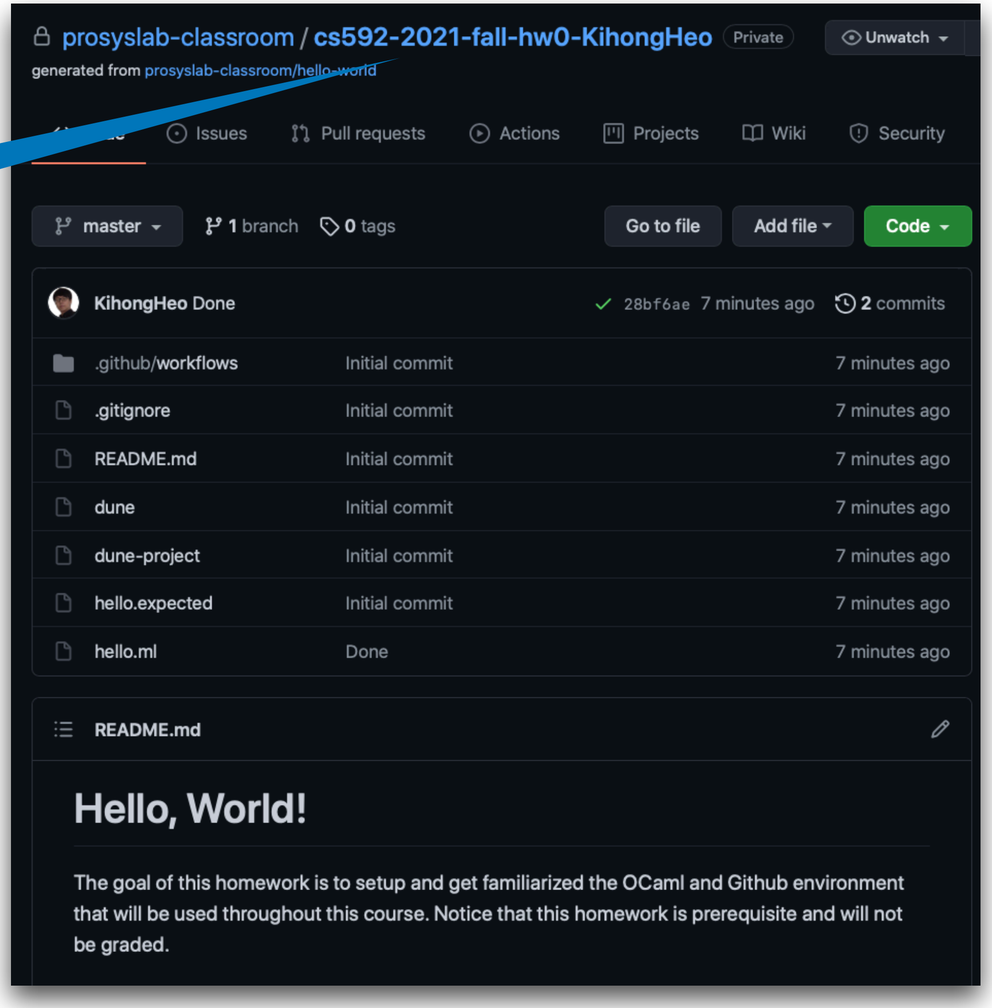
- All submissions will be managed using Github / Github Classroom
  1. For each HW, a unique invitation URL will be posted at KLMS
  2. Once you accept, a private repo for your assignment will be created
  3. You can push as many commits as you want before the deadline
  4. The final commit of your master branch will be graded
- 80% credit for 1-day late, 50% credit for 2-days late, NO credit otherwise

Homework 0: Hello, World!

- Goal: setting up and getting familiarized with OCaml and Git environments
  - Implement your “hello-world” program in OCaml
  - Test on your machine
  - Push to your Github repository
  - See the result in Github Action
- The invitation URL will be posted at KLMS

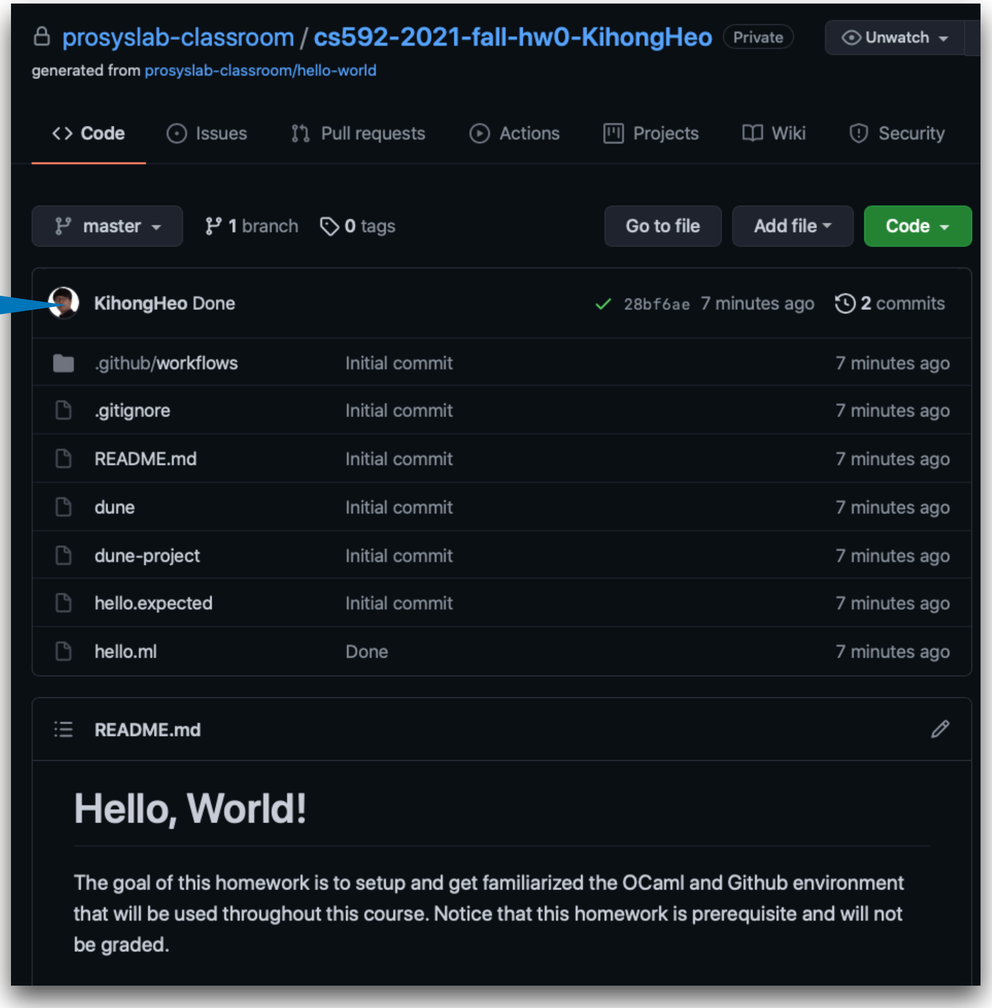
Homework 0: Hello, World!

1. Accept the invitation and have your repository



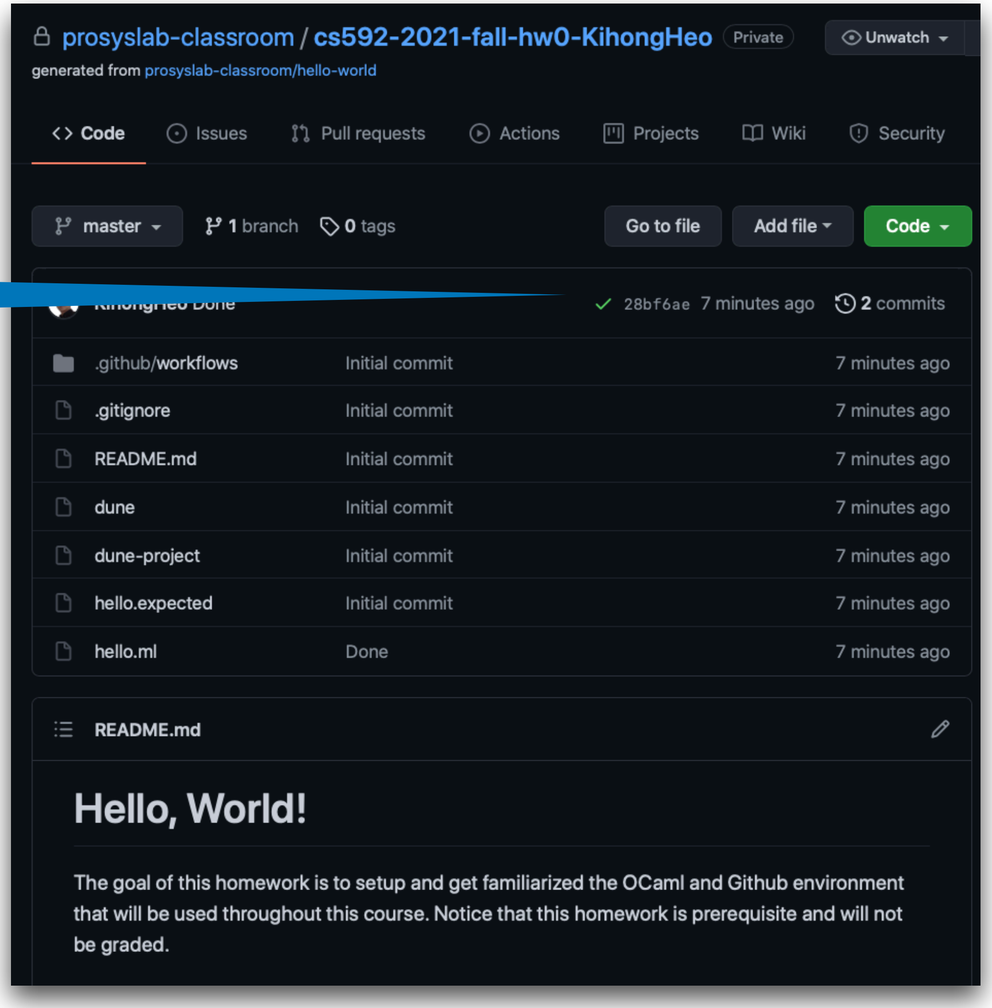
Homework 0: Hello, World!

2. Commit your code



Homework 0: Hello, World!

3. See your result



Misc

- Submit your Github account via the google form (see the Github issue board)
- Rules for programming assignments
  - Preserve the structures (directories, files, types, etc)
  - Don’t install further Github App