A practical course on

Advanced systems programming in C/Rust

Dimitrios Stavrakakis

Today’s topic!

Introduction

About us

Chair of Decentralized Systems Engineering

Our previous & current topics

- Trusted computing (SGX, ARM Trustzones)
- OS virtualization (containers) & Full virtualization and emulation (i.e. with qemu)
- Compiler-based systems designs
- Key value stores and distributed transactions
- High-Performance IO (DPDK, SPDK, RDMA)
- FPGAs
- Persistent memory
- ….

- Looking for a Master/Bachelor thesis topic? https://www.in.tum.de/dse/home/

Goals of the course

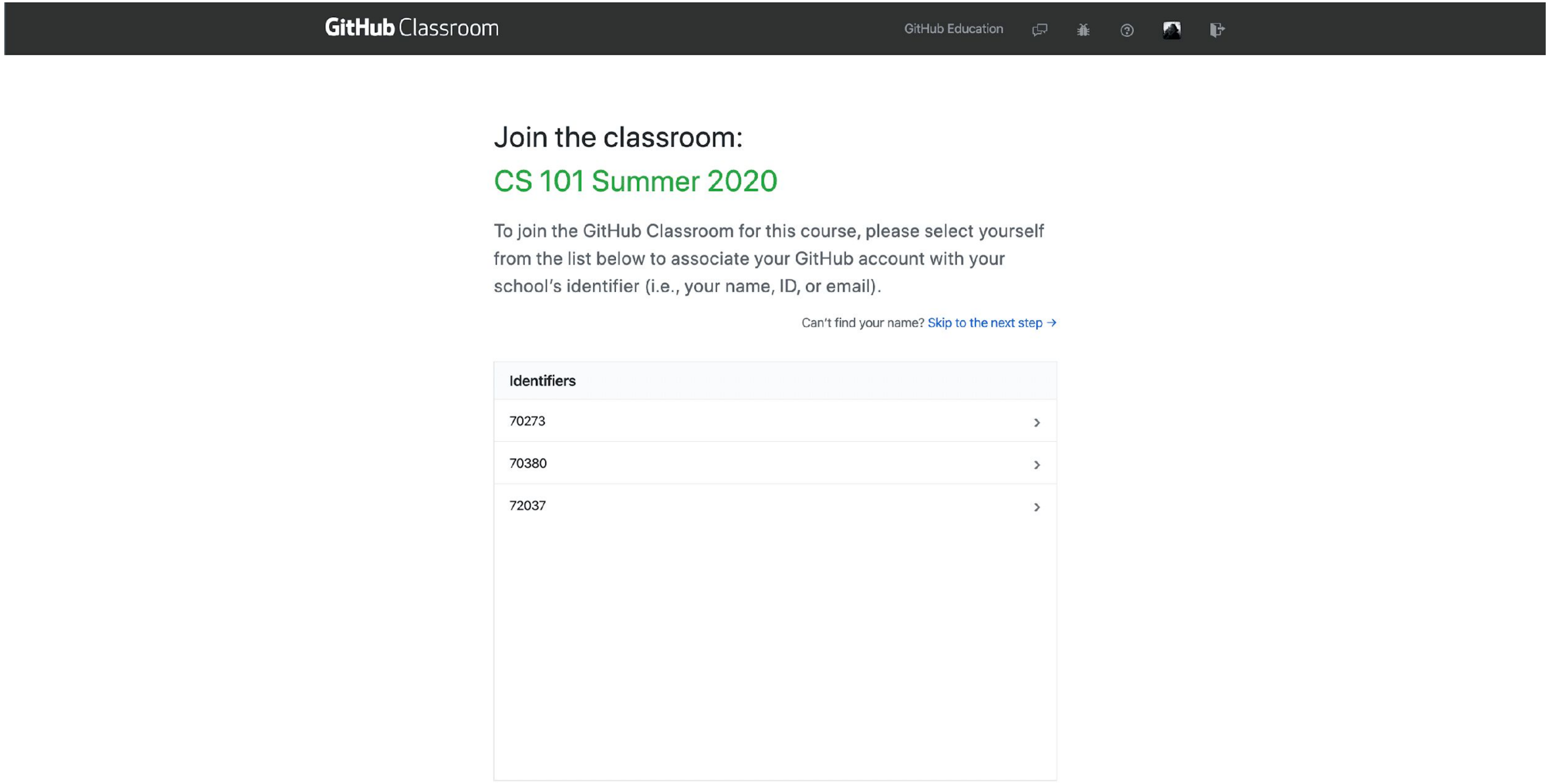
- Gain practical experience in assignments based on short theoretical lectures
- Learn how to use system APIs and memory/resource management
- Learn techniques for debugging and optimization of low-level code

Importantly, have fun!

About the course

- Platform: GitHub classroom (https://classroom.github.com)
  - GitHub organization: https://github.com/ls1-sys-prog-course
- 8 graded assignments -> 8 lecturers
  - No exams, presentations etc.
- Two weeks deadline for each topic
  - One topic per week, except for the last two topics (two weeks each)
- Each topic comes with tests to test your assignment
  - Tests run on our self-hosted ci runner (Submit only build scripts/code to solve the task!)
  - Solved test -> points -> grade
  - Hidden checks to avoid gaming the test suite
- Bug tracker: https://github.com/ls1-sys-prog-course/docs/issues

About the GitHub classroom



Join the classroom:

CS 101 Summer 2020

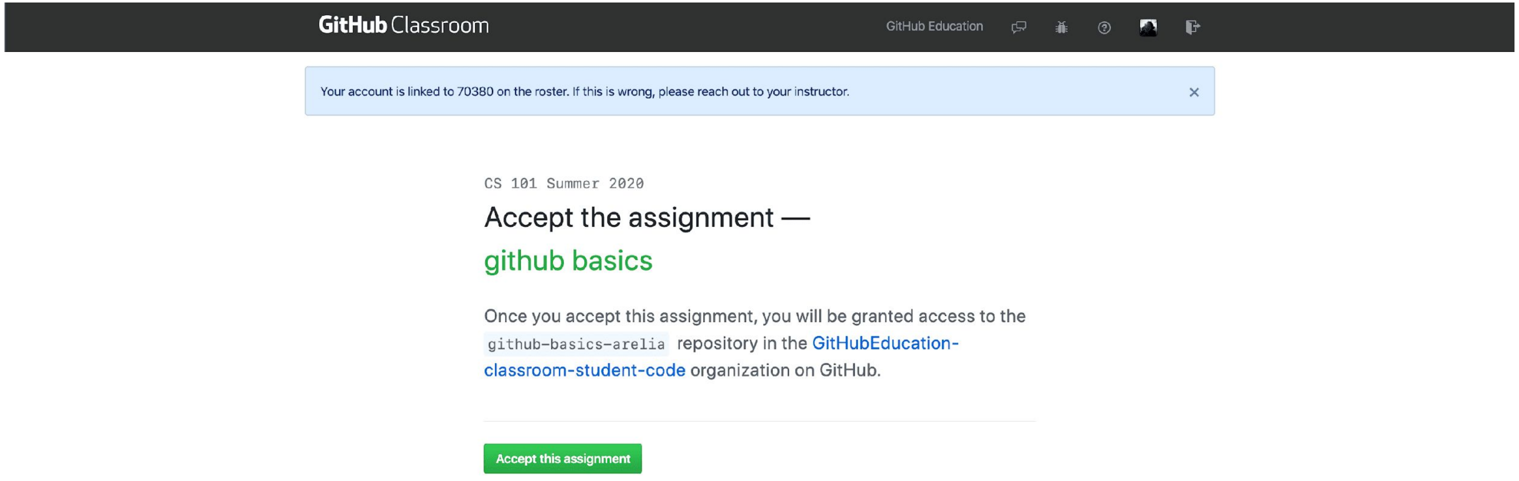
To join the GitHub Classroom for this course, please select yourself from the list below to associate your GitHub account with your school's identifier (i.e., your name, ID, or email).

Can't find your name? Skip to the next step →

Identifiers
70273
70380
72037

1. Pick your identifier and join the classroom

About the GitHub classroom



Your account is linked to 70380 on the roster. If this is wrong, please reach out to your instructor.

CS 101 Summer 2020

Accept the assignment — github basics

Once you accept this assignment, you will be granted access to the `github-basics-arelia` repository in the GitHubEducation-classroom-student-code organization on GitHub.

2. Accept assignment -> Creates repository

Organization

- Recorded lectures
  - Released every Monday
- Q&A session:
  - Every Thursday at 16:00-18:00
  - Explain the weekly assignment and answer questions regarding it
- Room for Q/A:
  - We will be using Zoom for the Q&A session. Join Zoom Meeting
  - Link: https://tum-conf.zoom.us/j/65566969019
  - Meeting ID: 655 6696 9019 P
  - Passcode: 762790
- Communication via Slack: https://ls1-courses-tum.slack.com/
  - if not joined yet please contact us

About the assignment setup

- Target: Linux on x86_64 CPUs
  - Setup a VM if you run a different operating system or install dual boot
- Choice between C, C++ and Rust
  - Can be switched for each task
  - Limited choice of allowed libraries (different per language / per task; see description)
- Each assignment comes with a template
  - Generic Makefile, needs to be adapted based on language
  - Tests will run **make all** to build project
- Python based test suite
  - One file per test (contains subtest)
  - Points per test
  - Started automatically when uploading, can be run locally

About the topics

Topic	Presenter
0. Introduction*	Dimitrios Stavrakakis
1. Kernel and system calls	Shady Issa
2. File I/O	Redha Gouicem
3. Processes	Sebastian Reimers
4. Concurrency and synchronization (ipc)	Myoung Jin Nam
5. Memory management	Moritz Lumme
6. Networking	Simon Ellmann
7. Performance	Atsushi Koshiba
8. Container virtualization	Peter Okelmann

*non graded assignment

How to find documentation

- References/documentation of your language:
    - C: https://en.cppreference.com/w/c
    - C++: https://en.cppreference.com/w/
    - Rust standard library: https://doc.rust-lang.org/std/index.html
- System call / Operating system documentation:
    - Each system call has a different page in manpage chapter **2**
    - On command line:
        - `$ man 2 read`
    - Online:
        - https://man7.org/linux/man-pages/man2/read.2.html

How to find examples

- Documentation is often a lie
  - Implementation sometimes easier to gasp than description
- Learn to Read the Source, Luke!
  - https://github.com/systemd/systemd (implements every syscall; modern C codebase)
  - Read-able libc: https://musl.libc.org/
  - Linux kernel: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/
    - Or search online: https://elixir.bootlin.com/linux/latest/source
- Get familiar with a code search tool:
  - Example: ripgrep: https://github.com/BurntSushi/ripgrep
  - Online search is often not great (I.e. github one)

When the code does not work…

- Use a debugger: gdb, rust-gdb
  - Learn most common commands (break; next; continue; print)
  - Enable debug symbols: `$ cc –Og –g main.c -o main`
  - Nicer graphical Interface: https://www.gdbgui.com/
  - Advanced (text) interface for low-level debugging: https://github.com/hugsy/gef
- Printf-Debugging:
  - Useful when debugging parallel issues/distributed code
  - C: `fprintf(stderr, "%s() at %s:%d: some var: %d\n", __func__, __FILE__, __LINE__, some_var);`
  - dbg! Macro in rust: https://doc.rust-lang.org/edition-guide/rust-next/dbg-macro.html

Meeting information

- Join Q&A session in Zoom:
  - Link: https://tum-conf.zoom.us/j/65566969019
  - Meeting ID: 655 6696 9019
  - Passcode: 762790
- Cheers!

How to find examples

- Documentation is often a lie
  - Implementation sometimes easier to gasp than description
- Learn to Read the Source, Luke!
  - https://github.com/systemd/systemd (implements every syscall; modern C codebase)
  - Read-able libc: https://musl.libc.org/
  - Linux kernel: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/
    - Or search online: https://elixir.bootlin.com/linux/latest/source
- Get familiar with a code search tool:
  - Example: ripgrep: https://github.com/BurntSushi/ripgrep
  - Online search is often not great (I.e. github one)