Practical course

Advanced Systems Programming in C/Rust

https://github.com/ls1-sys-prog-course/docs

Introduction

Chair of Distributed Systems and Operating Systems

https://dse.in.tum.de/

About us

Chair of Distributed Systems and Operating Systems

Our research topics include

- Operating Systems and Virtualization
- Distributed Systems and Cloud Computing
- Hardware Security and HW/OS Co-Design
- Binary Translation and Memory Models
- Quantum Software Systems
- ….

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

Goals of the course

- Acquire fundamental knowledge to build robust systems
- Familiarize yourself with end-to-end system design
- Learn techniques for profiling, debugging and optimization of low-level code
- Get a good understanding of memory- and resource management
- Improve hands-on experience through a variety of programming task

Importantly, have fun!

Course format

- Programming assignments using GitHub Classroom
  - Programming exercises released almost weekly
  - Deadline of 2 - 3 weeks depending on the difficulty/workload
  - Online submission & automatic grading
- Weekly Q&A meeting (attendance is optional)
  - Question and answer session to explain and discuss each assignment
- Slack channel for questions and discussion
  - https://ls1-courses-tum.slack.com/
- Grading
  - **Programming assignments (100%)** with public & private unit tests
  - No further exam / quiz / projects
- Latest Information: https://github.com/ls1-sys-prog-course/docs

About the assignment setup

- Languages
  - Choice between **C, C++ and Rust**
  - Can be switched for each task
  - Limited choice of allowed libraries (different per language)

- OS Environment information
  - All executables must run on **Linux, x86_64**
  - Use virtual machines if you run a different OS (i.e. Hyper-V on Windows)

Outline

- ~~Course introduction~~
- How to use Github Classroom
- Assignment (Task 0)

Github Classroom

- Create GitHub account: https://github.com/

- NOTE: Students can get many benefits from GitHub
  - https://education.github.com/students

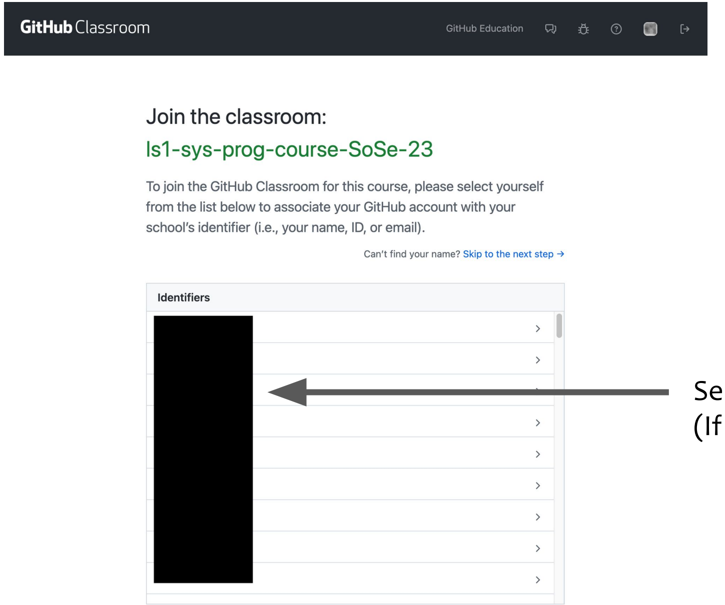
Join assignments

- For each assignments, we send **an invitation link** to the assignment via email
  - Example link: https://classroom.github.com/a/XXXXXXX

- Join an assignment by clicking the link

Connect student identifier (only first time)

- When joining the GitHub Classroom first time, it ask your identifier
- Select **your matriculation number** as an identifier



Join the classroom:

ls1-sys-prog-course-SoSe-23

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).

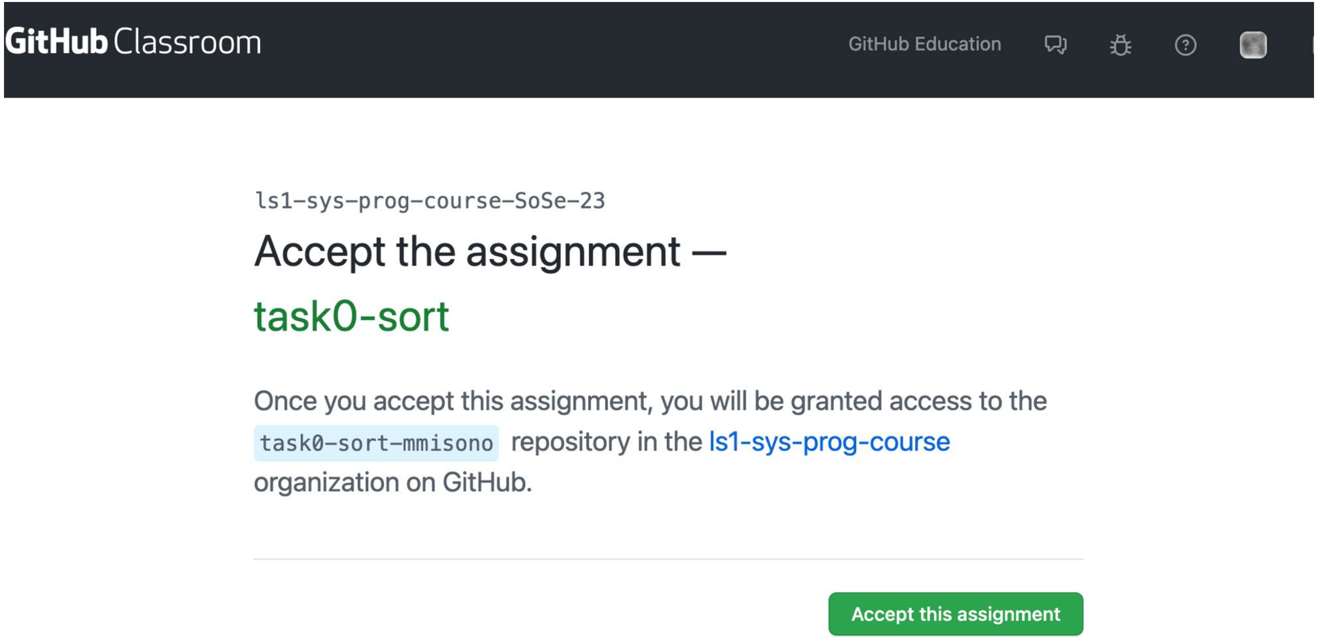
Can't find your name? Skip to the next step →

Identifiers

[redacted]

Select your matriculation number
(If you cannot find your number, please let us know)

Accept the assignment



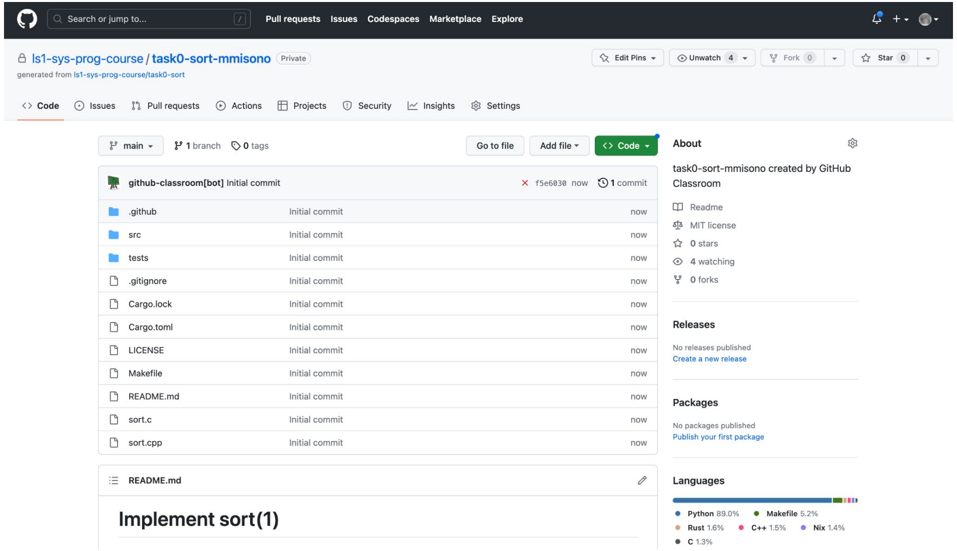
ls1-sys-prog-course-SoSe-23

Accept the assignment — task0-sort

Once you accept this assignment, you will be granted access to the `task0-sort-mmisono` repository in the ls1-sys-prog-course organization on GitHub.

Let’s do the task!

- Your repository for the task is automatically created in ls1-sys-prog-course organization
- Push your code (solution) to this repository
- **Do not edit tests and .github directory** (We will check their integrity later)



github-classroom[bot] Initial commit — f5e6030 now — 1 commit

.github	Initial commit	now
src	Initial commit	now
tests	Initial commit	now
.gitignore	Initial commit	now
Cargo.lock	Initial commit	now
Cargo.toml	Initial commit	now
LICENSE	Initial commit	now
Makefile	Initial commit	now
README.md	Initial commit	now
sort.c	Initial commit	now
sort.cpp	Initial commit	now

README.md

Implement sort(1)

About

task0-sort-mmisono created by GitHub Classroom

Readme
MIT license
0 stars
4 watching
0 forks

Releases

No releases published
Create a new release

Packages

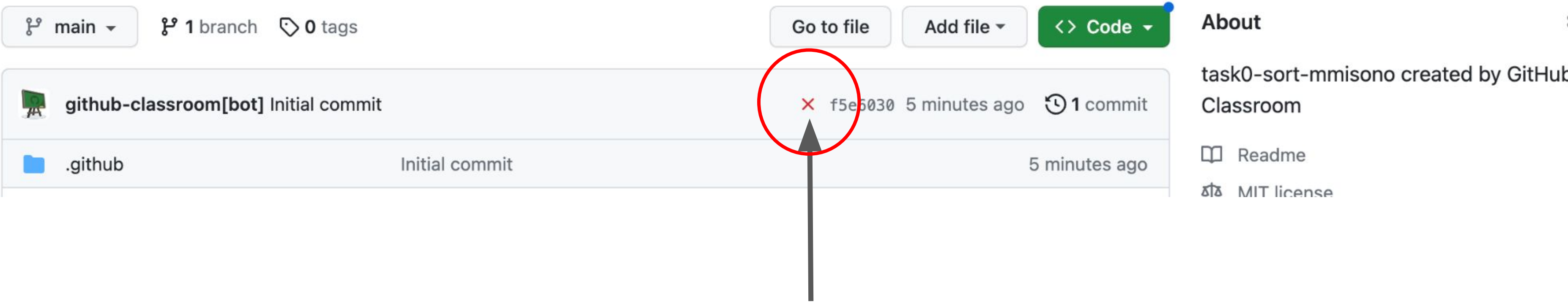
No packages published
Publish your first package

Languages

● Python 89.0% ● Makefile 5.2% ● Rust 1.6% ● C++ 1.5% ● Nix 1.4% ● C 1.3%

Check your score and test results

- **GitHub CI automatically tests your code** when you push
- You can check your score and test detail by checking GIthub CI’s report



github-classroom[bot] Initial commit ✕ f5e5030 5 minutes ago 1 commit

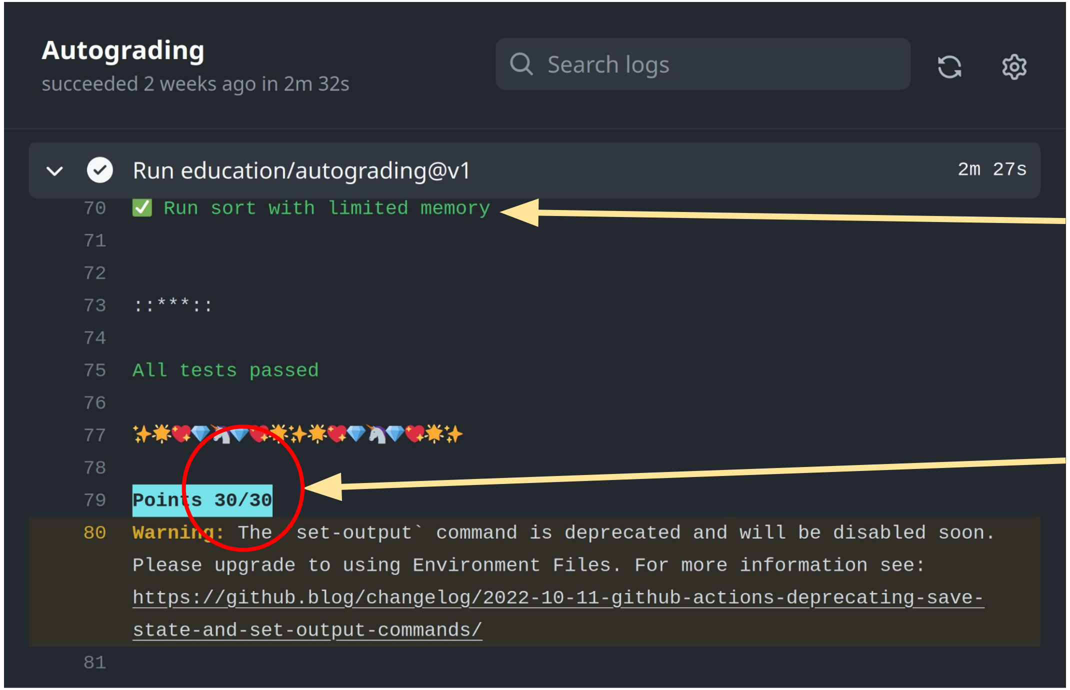
.github Initial commit 5 minutes ago

task0-sort-mmisono created by GitHub Classroom

Click this to check the detail
If you passed all the tests, this will be a green check mark

Check your score and test results (contd.)

- You can check your score and test detail by checking GIthub CI’s report



Autograding

succeeded 2 weeks ago in 2m 32s

Run education/autograding@v1 2m 27s

```
✅ Run sort with limited memory


::***::

All tests passed

✨🌟💖💎🦄💎💖🌟✨🌟💖💎🦄💎💖🌟✨

Points 30/30
Warning: The `set-output` command is deprecated and will be disabled soon.
Please upgrade to using Environment Files. For more information see:
https://github.blog/changelog/2022-10-11-github-actions-deprecating-save-
state-and-set-output-commands/

```

Find the output and result of test runs here

You can find your score in the detail of autograding

NOTE

- Your score will be the result of **the most latest commit before the deadline**
- Only commits on the main branch count

Outline

- ~~Course introduction~~
- ~~How to use Github Classroom~~
- Assignment (Task 0)

Assignment (task 0)

- Implement sort (score: 0 points)
- Goal: get used to GitHub and GitHub classroom
- Refer to the task repository for the detail

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 (github code search, elixir, etc.)

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

Latest information

https://github.com/ls1-sys-prog-course/docs