

Practical course

Advanced Systems Programming in C/C++/Rust

(SoSe 21)

Jörg Thalheim
TU Munich

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/l1s1-sys-prog-course>
- 9 graded assignments -> 9 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/l1s1-sys-prog-course/docs/issues>

About the GitHub classroom

GitHub Classroom

GitHub Education



Join the classroom:

CS 101 Summer 2020

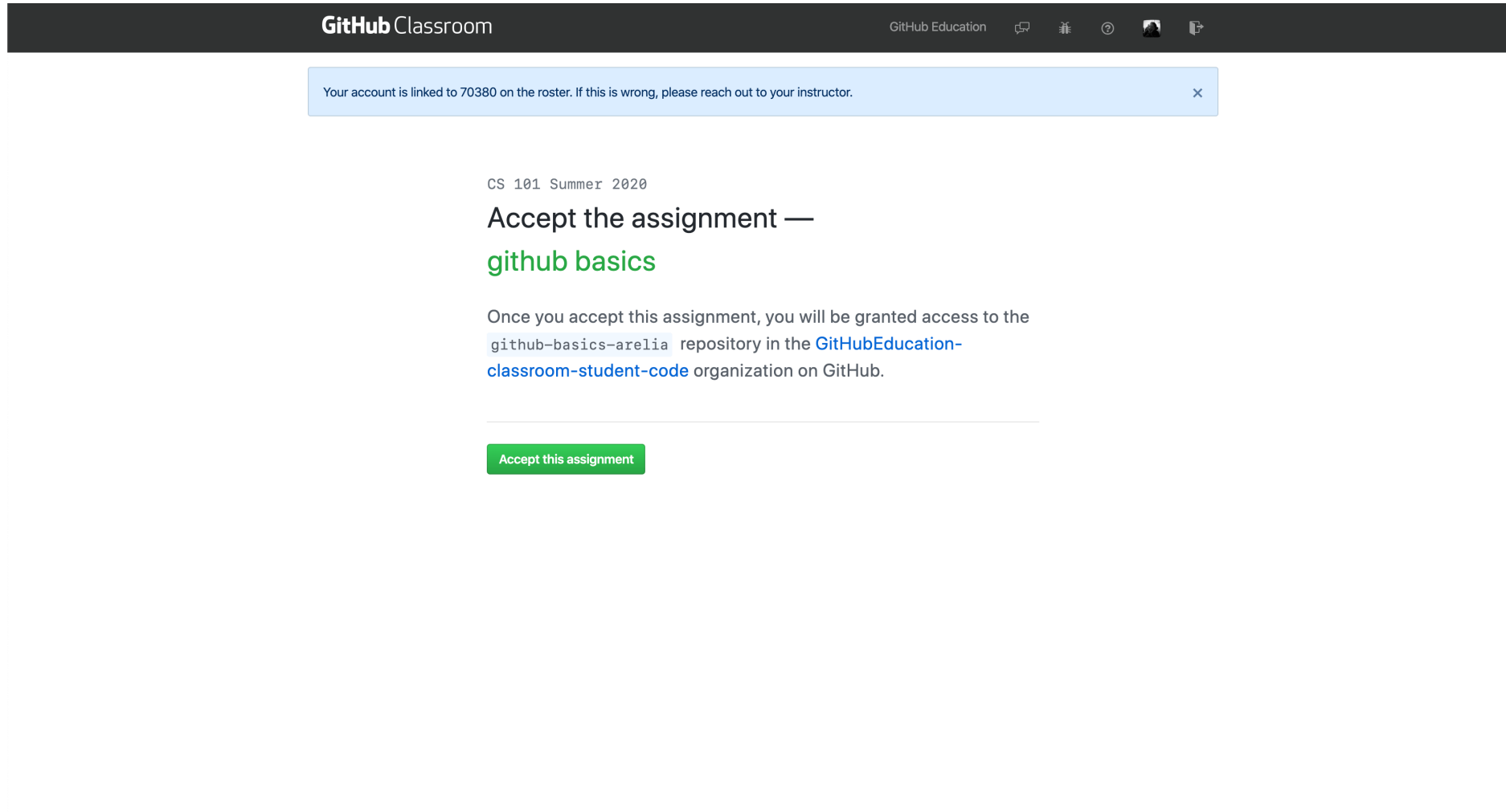
To join the GitHub Classroom for this course, please select yourself from the list below to associate your GitHub account with your school's identifier (i.e., your name, ID, or email).

Can't find your name? [Skip to the next step →](#)

Identifiers	
70273	>
70380	>
72037	>

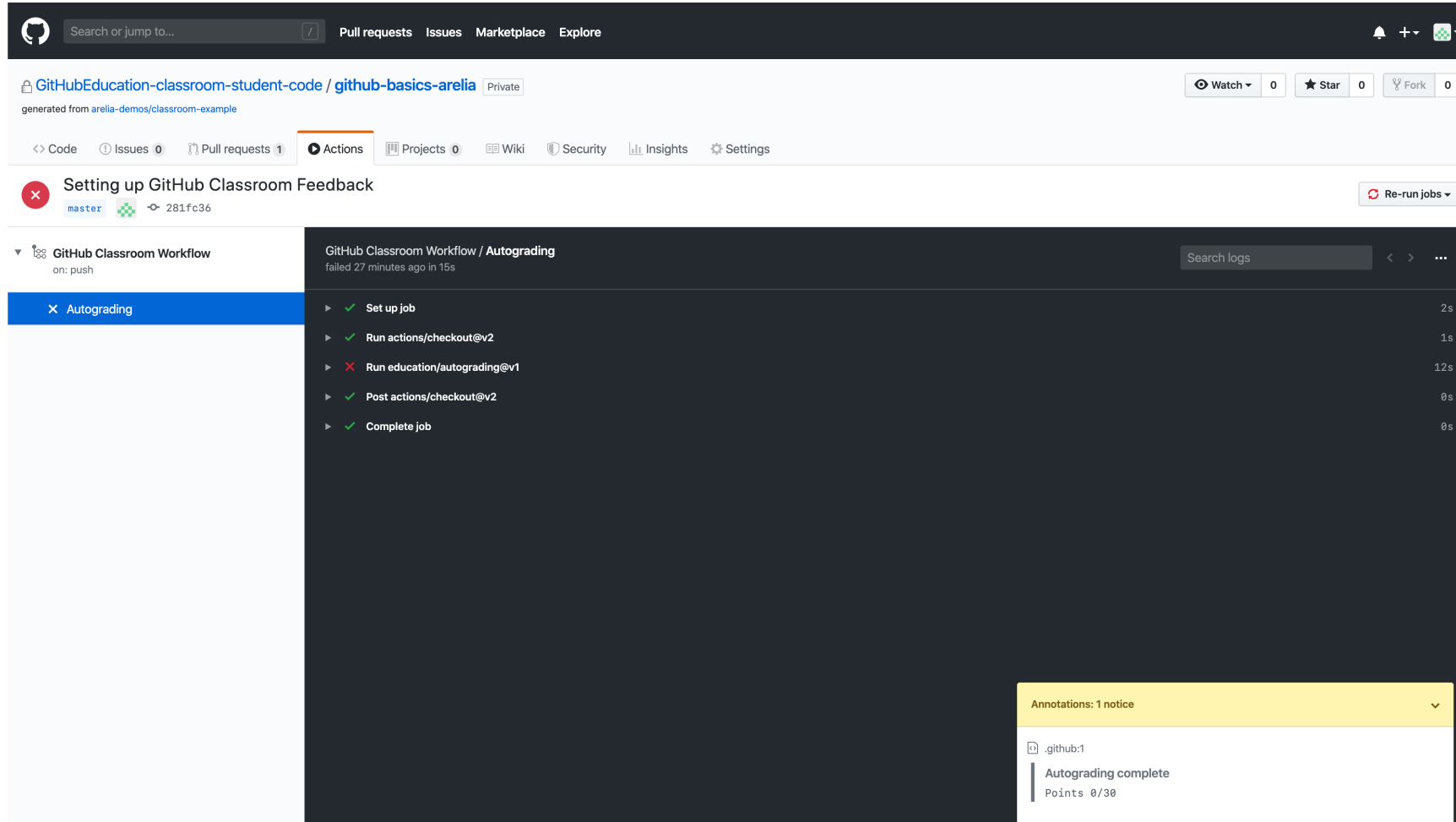
1. Pick your identifier and join the classroom

About the GitHub classroom



2. Accept assignment -> Creates repository

About the GitHub classroom



The screenshot shows a GitHub repository named 'github-basics-arelia' under the organization 'GitHubEducation-classroom-student-code'. The repository is private and has 0 stars and 0 forks. The 'Actions' tab is selected, showing a workflow named 'Setting up GitHub Classroom Feedback' on the 'master' branch. The workflow run 'Autograding' is highlighted, showing a list of steps: 'Set up job' (2s), 'Run actions/checkout@v2' (1s), 'Run education/autograding@v1' (12s), 'Post actions/checkout@v2' (0s), and 'Complete job' (0s). The workflow run is marked as 'failed 27 minutes ago in 15s'. A search bar is visible above the workflow run details. At the bottom right, there is a yellow box with the text 'Annotations: 1 notice' and a dropdown arrow. Below this, a notice from '.github:1' states 'Autograding complete' and 'Points 0/30'.

3. Submit code to get the tests passed.

Organization

- Recorded lectures
 - Released every Monday
- Q&A session:
 - Every Thursday at 16:00-17:00
 - 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 invited yet please send email to D.Giantsidi@sms.ed.ac.uk

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	Github
0. Introduction*	Jörg Thalheim	Mic92
1. Kernel and system calls	Redha Gouicem	rgouicem
2. File I/O	Harshavardhan Unnibhavi	harshanavkis
3. Processes	Matthias Hille	mahille
4. Concurrency and synchronization (ipc)	Shady Issa	shadyalaa
5. Memory management	Dimitrios Stavrakakis	dimstav23
6. Sockets	Dimitra Giantsidi	dgiantsidi
7. Performance	Maurice Bailleu	mbailleu
8. Container virtualization	Jörg Thalheim	Mic92
9. Virtualization (KVM)	Atsushi Koshiba	AtsushiKoshiba

*no 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>

Looking forward

- First test assignment will be released on Thursday
- Join Q&A session in Zoom:
 - Link: <https://tum-conf.zoom.us/j/65566969019>
 - Meeting ID: 655 6696 9019 P
 - Passcode: 762790
- Cheers!