Practical course

Advanced Systems Programming in C/Rust

(SoSe 2023)

Preliminary meeting

Chair of Distributed Systems and Operating Systems
https://dse.in.tum.de/

IMPORTANT

This course is not offered to *IN2397* (CSE course)

About us

Chair of Distributed Systems and Operating Systems

Prof. Pramod Bhatotia https://dse.in.tum.de/

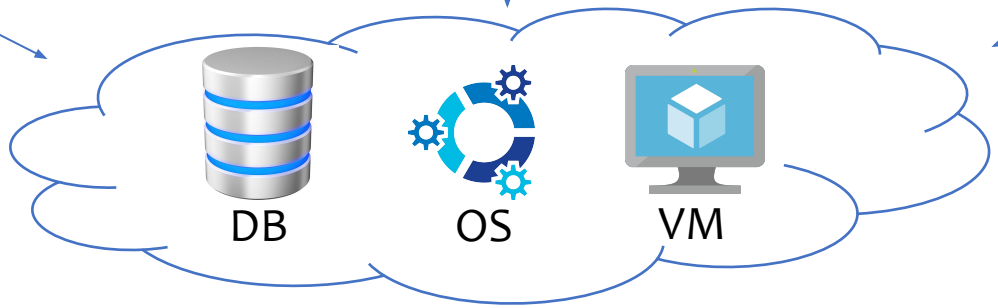
Masanori Misono		masanori.misono@in.tum.de
Jiyang Chen		jiyang.chen@tum.de
Charalampos (Babis) Mainas		charalampos.mainas@tum.de
Francisco Romão		francisco.romao@tum.de
Sebastian Reimers		sebastian.reimers@tum.de
Tianchi Yu		yu-tianchi@outlook.com

Systems programming applications

Internet of things

Transportation

Healthcare



Low level systems programming is an essential building block for high level applications

Software core properties

Performance

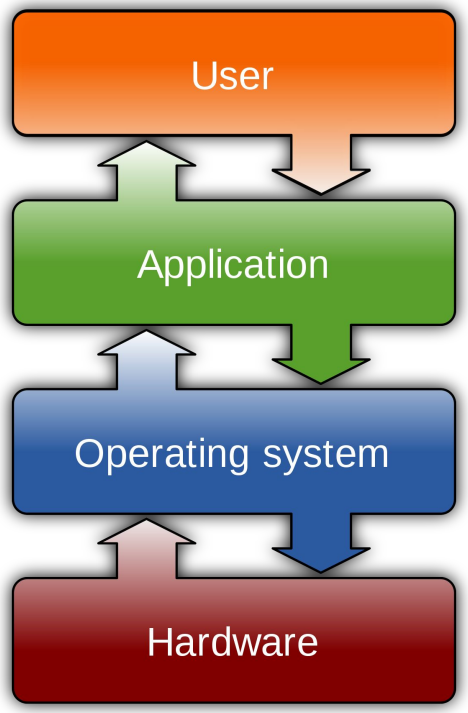
Reliability

Security

Efficient low level systems programming is critical to ensure these properties

System stack

Systems programming spans in multiple system levels and application domains

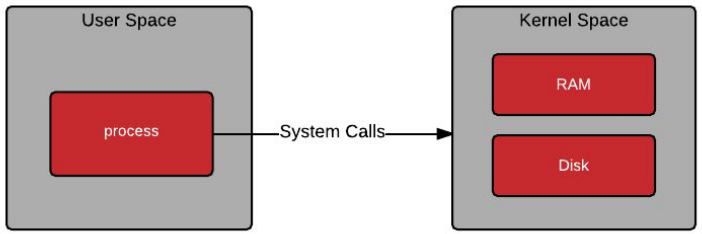


Time to get hands-on experience!

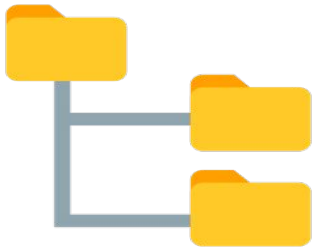
Course topics

- This course covers some of the most important aspects of systems programming:

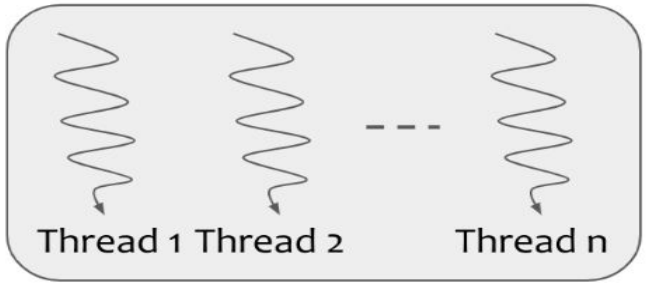
Kernel and system calls



File I/O



Concurrency and synchronization

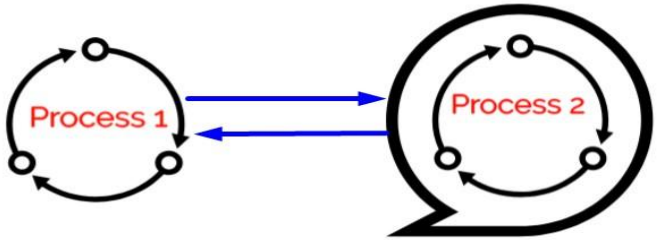


Memory management

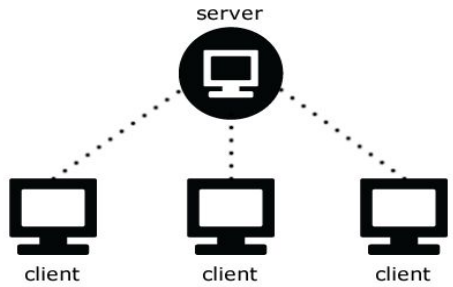
Course topics

- This course covers some of the most important aspects of systems programming:

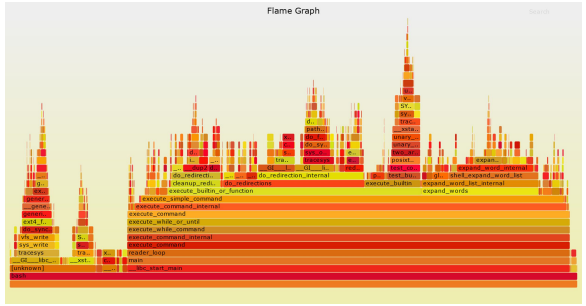
Processes



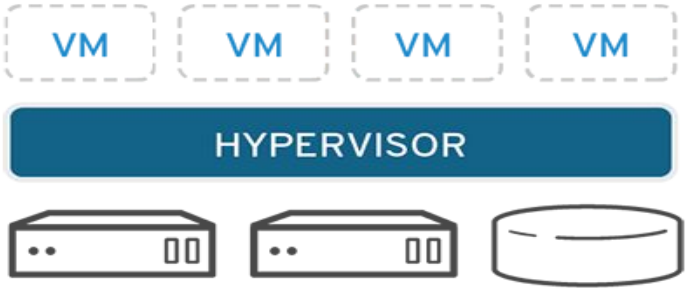
Networking



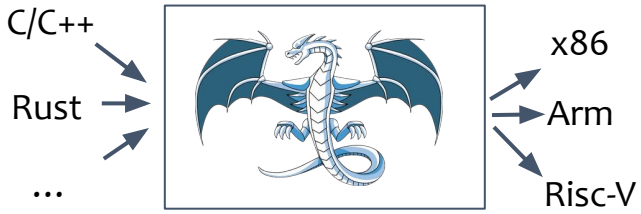
Performance profiling



Virtualization (containers/KVM)



Compilers / LLVM



Lab format

- Lab assignments
  - **10 practical programming exercises**
  - Deadline of 2 - 3 weeks depending on the difficulty/workload
  - Online submission
- Weekly meeting
  - Video with theoretical background coverage
  - Question and answer session to explain and discuss each assignment
  - Slack channel for questions
- Assessment:
  - **10 programming assignments (100%)** with public & private unit tests
  - No further exam / quiz / projects

Assignment examples (Actual tasks may change)

- Implement some functionalities of a filesystem
- Conversion of a single-thread program to a multithreaded version using locks
- Write your own memory allocator for better performance
- Implement your own client/server applications

Grading system

- Github classroom (https://classroom.github.com/)
  - Template repository for each task with detailed instructions & test cases
- Automated tests
  - Points are distributed among the exercises based on the estimated workload (30-60 points)
  - Specially designed test cases with gradually increasing difficulty
  - Hidden tests to detect & prevent gaming the grading system
- Grading scheme :

From	To	Grade	From	To	Grade
0	119	5.0	210	224	2.7
120	134	4.7	225	239	2.3
135	149	4.3	240	254	2.0
150	164	4.0	255	269	1.7
165	179	3.7	270	284	1.3
180	194	3.3	285	300	1.0
195	209	3.0			

Languages / OS

- 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)
  - You can also use the remote desktop (lxhalle.in.tum.de)

Learning goals

- 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 tasks

- Importantly, have fun!

Prerequisites

- We don't have any compulsory pre-requisites, but we prefer
  - **Knowledge equivalent to the lectures**
    - Fundamentals of Programming (IN0002)
    - Introduction to Computer Architecture (IN0004)
    - Basic Principles: Operating Systems and System Software (IN0009)
  - **Programming knowledge**
    - Foundations of programming (C, C++ or Rust)
    - Work in a Linux environment
  - **Knowledge on Git and GitHub**

If the prerequisites are unclear/strict -- please check with the instructor!

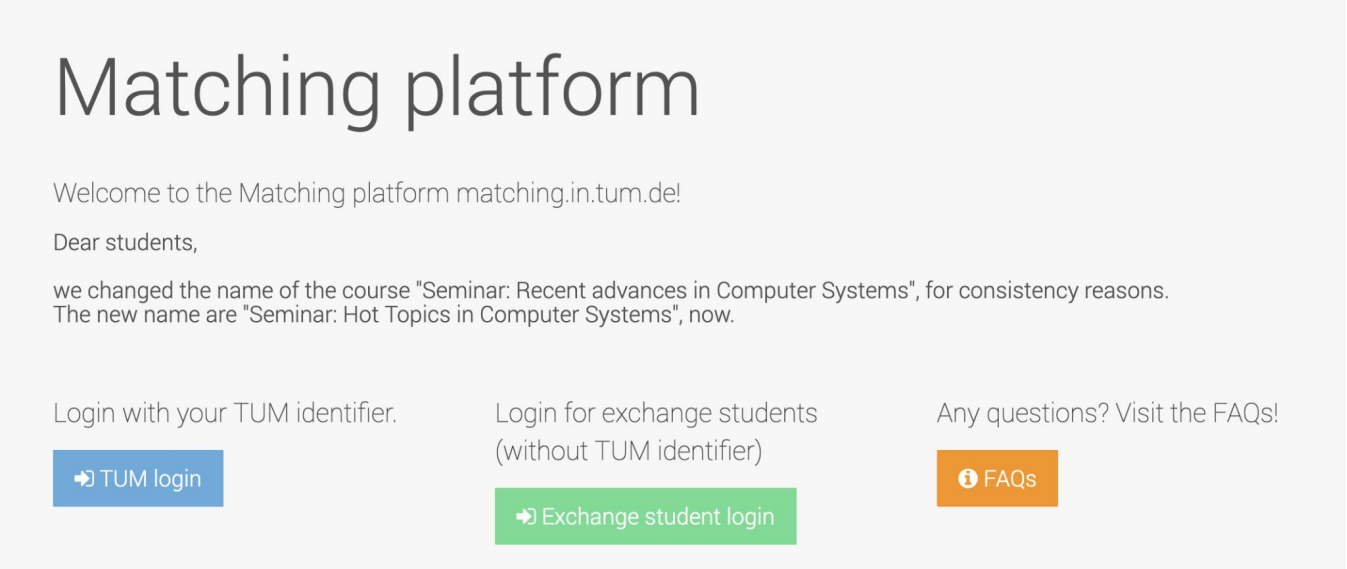
Code of conduct

- University plagiarism policy
  - https://www.in.tum.de/en/current-students/administrative-matters/student-code-of-conduct/

- Decorum
  - Promote freedom of thoughts and open exchange of ideas
  - Cultivate dignity, understanding and mutual respect, and embrace diversity
  - Racism and bullying will not be tolerated

- **Please write your own code!**

Interested?



Sign up on the TUM matching platform

Contacts

- Masanori Misono
  - masanori.misono@in.tum.de

We *strongly* prefer slack for all communications. Please join the slack channel.

Workspace: http://ls1-courses-tum.slack.com/

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

Channel: #ss-23-sys-prog

Join us with TUM email address (@tum.de)

IMPORTANT

This course it not offered to *IN2397* (CSE course)