Universität Bamberg

Operating Systems Engineering

Summer 2024

Lecture 1: Intro and Overview

Michael Engel (michael.engel@uni-bamberg.de)
Lehrstuhl für Praktische Informatik, insb. Systemnahe Programmierung
https://www.uni-bamberg.de/sysnap

Operating Systems Engineering

- This module concentrates on the central part ("kernel") of an operating system
  - the part of the system running in a privileged processor mode that interacts directly with hardware
- You will **investigate** the typical architecture of a **monolithic operating system kernel** and **implement** components of a typical OS
- The implementation and hardware-dependent parts are based on the modern and open **RISC-V** processor architecture.
- A central part of this module will consist of **code reading** and the **development of pieces of code** for a small operating system.
  - Different aspects of operating system functionality will be demonstrated through existing code

Syllabus

- From Source to Boot – compilation and system startup
- From Library to OS – what makes up an OS?
- Operating system interfaces and protection
- Multiple processes
- Multitasking
- Device drivers
- Synchronization
- Deadlocks and devices
- Virtual I/O and file systems
- Virtual memory

- In addition, we will discuss:
  - C crash course (on demand)
  - The RISC-V architecture

What will you learn?

- **Structure and implementation** details of a typical OS kernel
- **Programming on system level** ("bare metal") in C and/or Rust
  - …and **debugging**!
- **Algorithms** for typical OS tasks (scheduling, synchronization etc.)
- **Interaction of software and hardware** (CPU, peripheral) components
- Processor **architecture details** relevant for an OS (based on RISC V)
- Typical **development & emulation environments** for system software
- (optional) Bringup on real RISC V hardware

What do you need to know?

- Basic knowledge about operating systems and computer architecture
  - e.g. from bachelor level courses
- Programming experience
  - We will program in **C** [1,2]
  - Our **C crash course** is an overview of C for students with experience in (e.g.) Java programming
- Some Unix/Linux command line experience
  - You can learn this along the way
  - We will use Linux as development environment
- Experience with version control
  - We will use git (Uni Bamberg's gitlab instance)

- Useful: MIT's "The Missing Semester of Your CS Education" https://missing.csail.mit.edu

- **Exercises**
  - Theoretical exercises preparing for implementation tasks
  - Code reading and understanding, e.g. from MIT's xv6 OS [4]
  - Code design and writing

- **Examination**
  - Oral exam: present your solutions + answer questions

- **"Hackerspace" culture**
  - Discussions and questions in lectures are highly encouraged!

The RISC-V architecture

- RISC-V is an **open standard instruction set architecture** (ISA) based on established reduced instruction set computer (RISC) principles supporting 32 and 64 bit cores (and 128 bit…) [5]
- Developed at UC Berkeley since 2012
  - First as teaching tool (to replace the ubiquitous MIPS)
  - Then as open ISA for commercial applications
  - Simple base architecture extensible by application-specific extensions (e.g. fp, vectors, bit manipulation, compressed…)
- **RISC-V is a standard, not an implementation**
- Hundreds of (open/close source) implementations exist
  - ASICs – C906 core (D1 chip), SiFive (Unmatched board), …
  - FPGA – picorv32, FemtoRV, SERV, …
  - Emulators – qemu, tinyemu, …

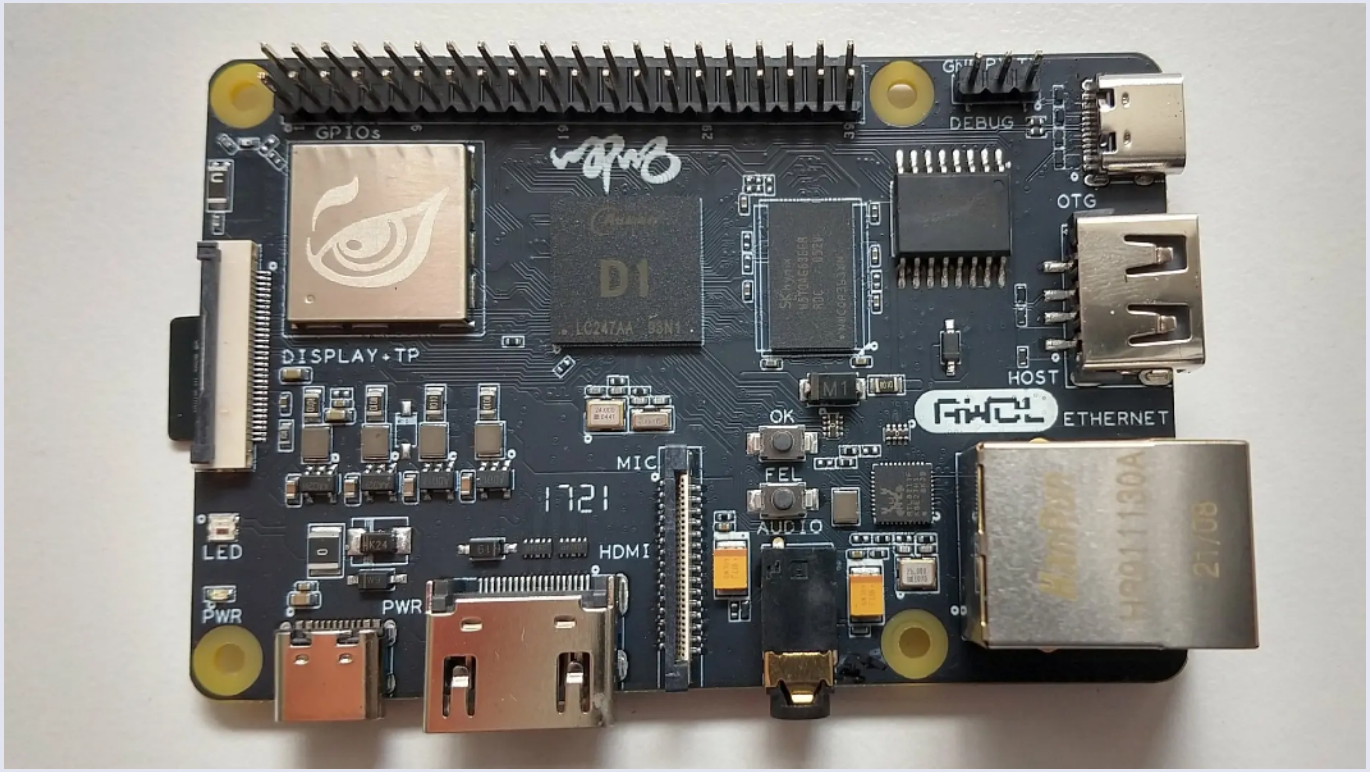
Development environment

- **Compilers and IDE**
  - for C: GNU gcc
  - Debugger: GNU gdb
  - IDE: your choice (Makefile+vi/emacs, Visual Studio, …)

- **Emulation**
  - Very useful – real hardware is more difficult to handle
  - We use Fabrice Bellard's qemu (https://qemu.org)
  - However: differences between hardware and emulation!

Real hardware

- D1 Nezha RISC-V boards: https://linux-sunxi.org/D1
- Single-core 1 GHz 64-bit RISC-V core, 1 GB RAM



System Programming Master Project

Universität Bamberg

- **Also in this summer semester! SYSNAP-Proj-M**
- **Topic: System-oriented Programming of Concurrency Platforms**
- Lecturer: Florian Schmoll
- https://vc.uni-bamberg.de/course/view.php?id=61230

References

1. Brian W. Kernighan, Dennis M. Ritchie, *The C Programming Language*, 2nd Edition, Pearson 1988, ISBN 978-0131103627
2. Jens Gustedt, *Modern C*, Manning 2019, ISBN 978-1617295812
3. Russ Cox, Frans Kaashoek and Robert Morris, *xv6: a simple, Unix-like teaching operating system*, MIT, RISC-V rev. 1, 2020 https://pdos.csail.mit.edu/6.S081/2020/xv6/book-riscv-rev1.pdf
4. David Patterson and Andrew Waterman, *The RISC-V Reader: An Open Architecture Atlas*, Strawberry Canyon 2017, ISBN 978-0999249116 https://github.com/Lingrui98/RISC-V-book/blob/master/rvbook.pdf

Links

- Course website:
  - https://www.uni-bamberg.de/sysnap/studium/operating-systems-engineering/
- C programming books:
  - https://stackoverflow.com/questions/562303/the-definitive-c-book-guide-and-list
- MIT's "The Missing Semester of Your CS Education"
  - https://missing.csail.mit.edu
- xv6 port to the Nezha D1 board
  - https://github.com/michaelengel/xv6-d1
- qemu Emulator
  - https://www.qemu.org/