Universität Bamberg

Operating Systems Engineering

Lecture 1: Intro and Syllabus

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
- Based on seminal publications, students will investigate different architectures of kernels, such as monolithic, micro- and exokernels, hypervisors and also unikernels. Mechanisms and policies of operating systems will be analyzed with respect to their functional as well as non-functional properties. The analysis of mechanisms dependent on a specific processor architecture will be explained using 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

- C crash course
- Development environment for operating systems
- Overview of the xv6 OS
- The RISC-V architecture
- Operating system interfaces
- Operating system organization
- Virtual memory and page tables
- System calls and traps
- Device drivers and interrupts
- Locking
- Scheduling
- File systems
- Concurrency revisited

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
  - 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

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

The xv6 operating system

- xv6 is a **modern reimplementation of Sixth Edition Unix** in ANSI C for multiprocessor x86 and RISC-V systems [3]
- Created for pedagogical purposes in MIT's Operating System Engineering course
- Simple and small enough to cover in a semester, yet still contains the important concepts and organization of Unix
  - **Kernel code size**:
    330 lines RISC-V assembler code
    5372 lines C code
    1410 lines C headers
- 21 of the most important **Unix system calls**
  - fork, exit, wait, pipe, read, write, close, kill, exec, open, mknod, unlink, fstat, link, mkdir, chdir, dup, getpid, sbrk, sleep, uptime
- Simple **Unix-like user land utilities**
  - cat, echo, grep, kill, ln, ls, mkdir, printf, rm, sh, wc

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…) [4]
- Developed at UC Berkeley since ~10 years
  - 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, …

Operating Systems Engineering

- **Paper reading:** Papers about OS topics from the last decades
- **Code reading:** Check the xv6 implementation
- **Core writing:** Implement feature from paper in xv6

- **Exercises**
  - Paper reading and discussion rounds
  - Practical code reading
  - Code design and writing

- **Examination**
  - Report
  - Oral project presentation + question round

- **Hackerspace culture**
  - Lectures = discussions, material to prepare for each week's lecture

Operating Systems seminar

- *Also in this summer semester!*

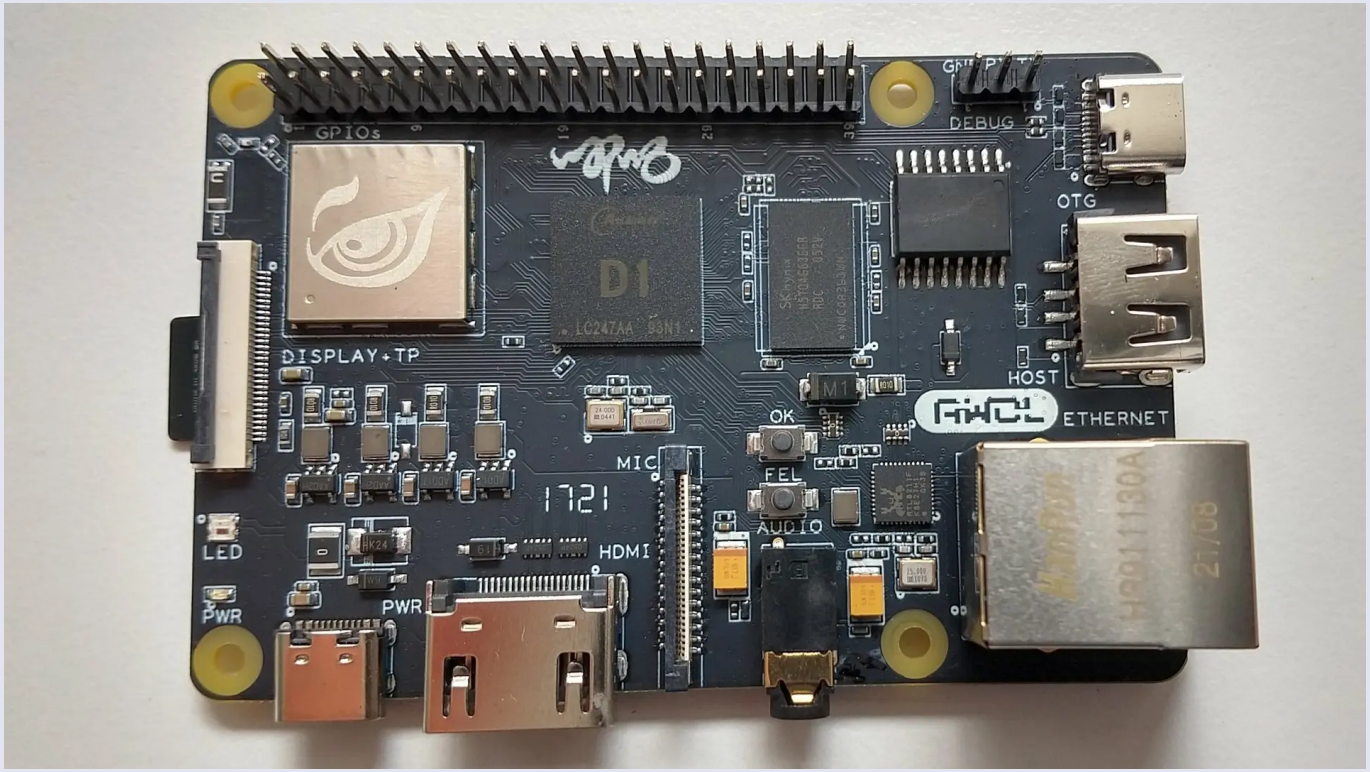
- Topic: the **Synthesis operating system** [5,6]
- Developed 30 years ago in the context of Alexia Massalin's PhD at Columbia University, Synthesis was groundbreaking in many aspects which have not been realized in subsequent operating system research and development projects, e.g.:
  - Code generation at run time,
  - Adaptive scheduling, with auto-tuning,
  - Optimistic Locking,
  - Extensible kernel, and
  - Superoptimization
- Idea: re-evaluate the ideas of Synthesis with a view on current hardware system and application requirements
- https://www.uni-bamberg.de/sysnap/studium/seminar-operating-systems/

OSE Project

- Operating Systems Engineering project in the winter semester
- Idea: investigate a given OS design approach
  - e.g. microkernels, hypervisors, single-address-space OS, …
- **Design and implement your own OS** based on chosen approach
  - Bare metal = running directly on real hardware
  - …as well as in emulation (e.g. qemu)
- Hardware platforms:
  - D1 RISC-V (64 bit)
  - ESP32-C3 RISC-V (32 bit)
  - Raspberry Pi (ARM V8 32/64 bit)

Real hardware

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



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
5. Calton Pu, Henry Massalin, and John Ioannidis. *The synthesis kernel*, Computing Systems 1.1 (1988): 11-32
6. John Regehr, *It's Time for a Modern Synthesis Kernel*, blog article https://blog.regehr.org/archives/1676

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