

System and Runtime Software Interaction with Modern Hardware

Fordypningsemne / theory module TDT09

Høst 2021

Michael Engel

Motivation

- “If I have seen further, it is by standing on the shoulders of giants”
 - Isaac Newton in a 1675 letter to Robert Hooke
- Research does not exist in a vacuum
 - Research is (also) social interaction
 - “Collaboration for the Greater Good”
- However, science today is often *competition*
 - Gain & maintain a competitive advantage
 - Attract research funds, great researchers, personal benefits, ...
- How does this fit with publishing results?
 - Hiding or leaving out details, denying access to research results ("artifacts"), ...



[Image from wikipedia,
public domain]





So what is the problem?

- Reading a paper doesn't imply you understand it
 - Details might be missing
 - Things might be unclear
 - Details of the paper might be
 - ...unintentionally or intentionally incorrect
 - ...described but never implemented 🤔
- ***All these things can slip through the usual peer review process for conferences and journals!***
- Reproducibility of research results is important
 - Gives confidence that work exists and is useful
- Recent trend: require reproducibility
 - Delivery of paper + "artifacts" = code, data, ...
 - Different levels of artifact evaluation



[Images by ACM]

Levels of usefulness

Functional	Reusable	Available	Replicated	Reproduced
No Badge				
Artifacts documented, consistent, complete, exercisable, and include appropriate evidence of verification and validation	Functional + very carefully documented and well-structured to the extent that reuse and repurposing is facilitated. In particular, norms and standards of the research community for artifacts of this type are strictly adhered to.	Functional + placed on a publicly accessible archival repository. A DOI or link to this repository along with a unique identifier for the object is provided.	Available + main results of the paper have been obtained in a subsequent study by a person or team other than the authors, using, in part, artifacts provided by the author.	Available + the main results of the paper have been independently obtained in a subsequent study by a person or team other than the authors, without the use of author-supplied artifacts.



[Images by ACM]

Idea

- A scientific paper in CS is more than just a PDF
 - Especially in operating systems and related topics
- Software and hardware described in the paper...
 - was developed, benchmarked and evaluated → *code* and *data*
- **Reading and understanding** a paper is often not enough
- Often, theory modules are in classic "seminar style"
 - Read one or two papers, discuss, write summary paper & present
 - ...this is a bit boring – for you as well as for me
 - ...what does this really teach you?
- **Idea:** Give an intro to research from the perspective of a PhD student
 - You have to find an interesting research topic that keeps you interested and motivated for (at least) three years
- You have to find out what work already exists in this topic area
 - ...either to base your own ideas upon existing work
 - ...and/or to build something much better than previous researchers! 😊

Our approach for TDT09

- Many students of this year's OS course wanted an advanced course
 - This takes long to implement and get approved
 - But the theory module is something similar
 - Should prepare you for your master degree!
- Reading and *understanding* papers
 - Understanding works best if you try to build a system
- Our approach: **reproduce** a given research result
 - Extract the central **idea** from the main paper
 - Find a way how to fit the idea into a new environment
 - Implement (+evaluate) the idea
 - Write and present about the topic + your experience

xv6 – the "system and runtime software" part

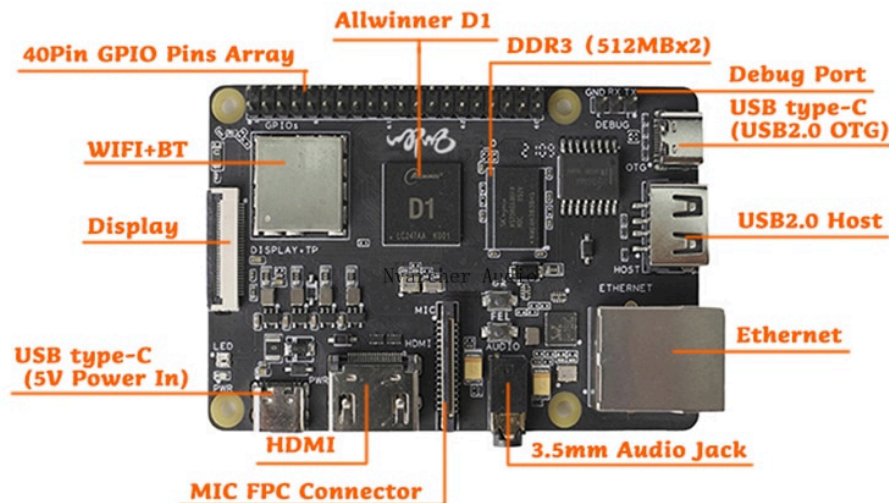
- Small educational operating system developed at MIT
 - x86 (outdated) and RISC-V (current) versions
 - Written in C (+ a tiny bit of unavoidable assembler code)
 - Similar in functionality to 6th Edition Unix (ca. 1977)
 - Some reduced functionality (e.g. no users/protections)
 - Some extended functionality (basic virtual memory)
 - ...ideal basis to explore and extend
 - (Quite) well documented and small
- Lots of code examples and a large community around xv6
 - Don't hesitate to interact with students around the world
 - Possible collaboration with students in John Regehr's advanced OS course at the University of Utah

RISC-V – the "modern hardware" part

- Modern open ISA RISC architecture
 - 32, 64, 128 bit versions: base integer instruction set + extensions (FP, vector instructions, hypervisor, ...)
 - Scalable: from microcontroller to high-performance manycore
 - Architecture (somewhat) similar to MIPS
- Developed at UC Berkeley, originally as teaching tool
- Large number of implementations available
 - Emulators: qemu, tinyemu, Spike, gem5, ...
<https://riscv.org/exchange/software/>
 - FPGA processor cores
<https://riscv.org/exchange/cores-socs/>
 - Boards and systems on chip
<https://riscv.org/exchange/>

Platform

- Use an emulator
 - qemu-system-riscv64
Alternative: tinyemu
 - Virtual hardware + I/O
- Real hardware! (*optional*)
 - Boards donated by **RISC-V International**
 - **Thanks a lot!**
 - "Nezha" Allwinner D1 board
 - 1 GHz single core RISC-V 64 SoC
 - 1 GB DDR3 DRAM
 - UART, Ethernet, USB, HDMI, SD-card, Wifi/Bluetooth, audio...
- I'm currently finishing the xv6 port to the D1...



Possible Topics

- Extend the xv6 OS based on research ideas
 - Typically, based on one main + some background papers
 - Some topics are more recent, some are a bit older
 - Still, all relevant OS research, tiny but interesting ideas and concepts
- Complete list of topics after the weekend
 - Read through the list, prioritize topics
- Possible topics:
 - Tickless scheduling, priority-based scheduling, ...
 - Virtual memory copy-on-write + process checkpointing and migration (2 students)
 - Access control lists + capabilities (1-2 students)
 - Redundant multithreading
 - Paravirtualization on a hypervisor
 - Implement container technology (1-2 students)
 - Lazy process switching
 - Shared library support
 - System and process tracing and profiling
 - ***your own idea here***

Deliverables: "3P"

- **Paper** – 10–15 pages + references: **December 5th**
 - Describe the background of the respective publication and *your experience (+results) reproducing it*
- **Presentation** – 20 minutes + Q&A: **week of December 6th**
 - Prepare a presentation based on the contents of your paper and your experience
- **Prototype** – this can range from... (**December 5th**)
 - A concept how to adapt the paper's idea to xv6
 - A prototype implementation in xv6
 - ...or, if all else fails, da*n good reasons why this could never work :)

Learning Outcomes

- Set up a development environment for OS research
 - Cross-compiler toolchain, debugger, emulator
 - RISC-V is a relatively new platform, problems might show up
- Read and understand a (moderately) large code base
 - xv6 kernel: ca. 6500 lines of code
- Read & understand a scientific publication on an OS topic
 - Extract the relevant information to *reproduce the idea*
 - Create a concept how to *adapt the idea* to xv6
 - (Try to) *Implement the idea* in xv6
- Improve your scientific writing and presentation skills
 - Write a paper that describes the background of the respective publication's topic *and your experience (+results) reproducing it*
 - Prepare and present a 20 minute (+ Q&A) presentation

Organization

- **Development platform**
 - Linux, MacOS, Windows+WSL should all work
 - But please give feedback if something fails
- **Work as a single student or in groups of two**
 - A group of two will get a more complex task, of course
- **Meetings** ~every two weeks
 - Do you prefer group or individual meetings?
 - I'm in Germany from Sept. 23–Oct. 8, but we can have zoom meetings on demand if problems show up during this time
- **Lab space**
 - Trying to find some, it's difficult...
- **Online collaboration tool**
 - Slack? Discord? IRC? What do you prefer?

References

1. Writing papers:

Roy Levin and David D. Redell: "How (and How Not) to Write a Good Systems Paper"
https://www.usenix.org/legacy/publications/library/proceedings/dsl97/good_paper.html

2. Preparing presentations:

Andreas Zeller: "How to give a good research talk" (see also his Twitter account @AndreasZeller)
<https://www.st.cs.uni-saarland.de/edu/specmine11/slides-good-talk-howto.pdf>

3. The state of OS research:

Rob Pike's talk "System software research is irrelevant":
<http://herpolhode.com/rob/utah2000.pdf>

Timothy Roscoe's Usenix ATC/OSDI2021 keynote "It's Time for Operating Systems to Rediscover Hardware":
<https://www.youtube.com/watch?v=36myc8wQhLo>

4. xv6 for RISC-V (+ qemu/compiler setup):

Web: <https://pdos.csail.mit.edu/6.828/2021/xv6.html>

Book: <https://github.com/mit-pdos/xv6-riscv-book>

Code: <https://github.com/mit-pdos/xv6-riscv>

5. RISC-V documentation – the RISC-V reader: <https://github.com/Lingrui98/RISC-V-book>

6. Nezha D1 technical information: https://linux-sunxi.org/Allwinner_Nezha