Advanced Operating Systems
Summer Semester 2022/2023

Martin Děcký

1

Introduction

About the Course

- **Lecture**
  - Thursdays at 3:40 p.m. in lecture room S1
    - From February 16th to May 18th 2023
    - **Exception:** April 13th 2023 → lecture room S5
  - Follow up on the *Operating Systems* (NSWI004) course from winter semester
    - We might assume the understanding of some basic concepts
- **https://d3s.mff.cuni.cz/cz/teaching/nswi161/**
  - Up-to-date information and current affairs
  - Slide decks of past lectures and other materials
  - Urgent updates will be sent out using e-mails via the Student Information System

Further Information

- **https://gitlab.mff.cuni.cz/teaching/nswi161/forum**
  - GitLab forum for both technical and organizational inquires
  - Just create a new issue and/or subscribe to the notifications
- **Lecturer**
  - Martin Děcký
    - Employed by Kernkonzept GmbH, no permanent office at Charles University at the moment
    - Consultations on demand after a prior agreement (ideally before or after the lecture)
    - decky@d3s.mff.cuni.cz
- **Guarantor**
  - Petr Tůma
    - Office S 205 (Malá Strana)
    - petr.tuma@d3s.mff.cuni.cz

Course Goals

- **Insight into operating system design and implementation mechanisms**
  - Not only for system-level development
    - All abstractions are leaky to a certain degree, black boxes are rarely truly black
    - Many extra-functional properties of a piece of software (performance, reliability, etc.) cannot be properly assessed and/or guaranteed without the understanding of the underlying properties
- **Insight into the context, requirements and constraints in which operating systems exist**
  - One size does not fit all
  - Not everything has been already solved

Course Structure

- **Two main interleaving “tracks”**
  - Operating systems implementation
    - Overview of technical aspects
  - Operating systems architecture
    - High-level overarching concerns
- **Guest lectures**
  - Individual lectures by industrial experts
  - No fixed schedule at this moment (follow the course web site)

(Planned) Course Topics

- Languages, run times, bootstrap
- Interfaces and interactions, compatibility, portability, abstractions
- Debugging, performance, observability
- Memory hierarchy, memory management
- File systems, storage
- Communication, networking, off-loading, distributed computing

(Planned) Course Topics

- Architecture, design and configuration
- Requirements, validation, certification, verification
- Concurrency, parallelism, synchronization
- Safety, security, reliability
- Virtualization
- Resource and service management
- Real time

Literature and Resources

- **This course is not based on a specific textbook**
  - Individual references will be presented as necessary
  - The usual sources of useful information
    - English Wikipedia for the general overview
    - Similar courses at other universities
    - Academic papers from good venues (e.g. OSDI, SOSP, ATC, FAST, HotOS, EuroSys, SIGOPS, etc.)
    - On-line resources (e.g. LWN.net)
    - **Open source operating systems**

Microkernels - The componen

microkernel.info

µ–kernel.info

Microkernels are operating systems that outsource the traditional operating system functionality to ordinary user processes while providing them with mechanisms requisite for implementing it. Microkernel-based operating systems come in many different flavours, each having a distinctive set of goals, features and approaches. Some of the most often cited reasons for structuring the system as a microkernel is flexibility, security and fault tolerance. Many microkernels can take on the role of a hypervisor too. Microkernels and their user environments are most often implemented in the C or C++ programming languages with a little bit of assembly, but other implementation languages are possible too. In fact, each component of a microkernel-based system can be implemented in a different programming language.

Here is a list of active free, open source microkernel projects. If your project is missing or this page needs fixing, please create a pull request!

Escape

A UNIX-like microkernel operating system, that runs on x86, x86_64, ECO32 and MMIX. It is implemented from scratch and uses nearly no third-party components. To fit nicely into the UNIX philosophy, Escape uses a virtual file system to provide drivers and services. Both can present themselves as a file system or file to the user. (*github.com/Nils-TUD/Escape*)

M³

A microkernel-based system for het
cores, that is developed as a hardwa
the TU Dresden. It aims to support a
eral purpose cores, DSPs, FPGAs,
This is achieved by abstracting the
new hardware component per c
(*github.com/TUD-OS/M3*)

http://microkernel.info

F9

An experimental microkernel used to construct flexible real-time and embedded systems for ARM Cortex-M series microprocessors with power efficiency and security in mind. (*github.com/f9micro*)

MINIX 3

A free, open-source, operating system designed to be highly reliable, flexible, and secure. It is based on a tiny microkernel running in kernel mode with the rest of the operating system running as a number of isolated, protected, processes in user mode. (*minix3.org*)

D3S

Credits

- **Standard approach**
  - Written test during the exam period based on the course topics
    - Half of the total amount of points required for passing the exam
    - Further details will be clarified during the semester
- **Hands-on approach**
  - Individual or small team implementation project
    - Assignment, goals and criteria need to be agreed upon between the candidates and the lecturer & guarantor
    - If interested, do not hesitate to approach us (but soon enough)
- **Contributor approach**
  - Picking one of the course topics for a standalone lecture or an extended demonstration
    - Again, needs to be agreed upon soon enough

Implementation Project

- **Random topic suggestions**
  - Your own pet project
    - Some non-trivial connection to operating systems is required
    - Can be an extension of a previous work (e.g. the winter semester assignment) or something you plan to extend in the future (e.g. into your master thesis)
      - But obviously not something you have already finished
  - Targeted contribution to an open source operating system project
    - Tip: Have a look at the list of ideas for the Google Summer of Code
    - **Pro tip:** http://www.helenos.org :)
    - **Pro tip from my employer:** https://www.l4re.org :)
  - Original implementation of an idea from a research paper
    - Could be both rewarding and treacherous

About the Lecturer

- **Charles University, Faculty of Mathematics and Physics**
  - MSc. (2005), Ph.D. (2015)
  - Researcher at the Department of Distributed and Dependable Systems (2008 – 2017)
  - HelenOS project co-author (http://www.helenos.org)
- **Huawei Technologies**
  - Senior Research Engineer at the Munich Research Center (2017 – 2019)
  - Principal Research Engineer and co-founder at the Dresden Research Center (2019 – 2021)
- **Kernkonzept**
  - Senior Software Engineer (since 2021)
  - https://www.kernkonzept.com

ABOUT KERNKONZEPT

Owner-managed	Founded 2012	Spin-off from TU Dresden
International team of 30	Wide experience since 1996	Continuously growing
Close to research and innovative	Operating system specialists	Located in Dresden, Germany

KERNKONZEPT MARKETS

Please Do Interact!

Interaction Is Always Welcomed

- **Influence the topics of this course**
  - There is no point in yapping about something you already know or do not care about
- **Ask questions**
  - There is no point in listening to something you do not understand
  - As usual: There are no stupid questions
- **Discuss**
  - Despite best effort, everyone is biased
  - This course is not about dogmas, but about nuances
  - Think about how and why would you do things differently

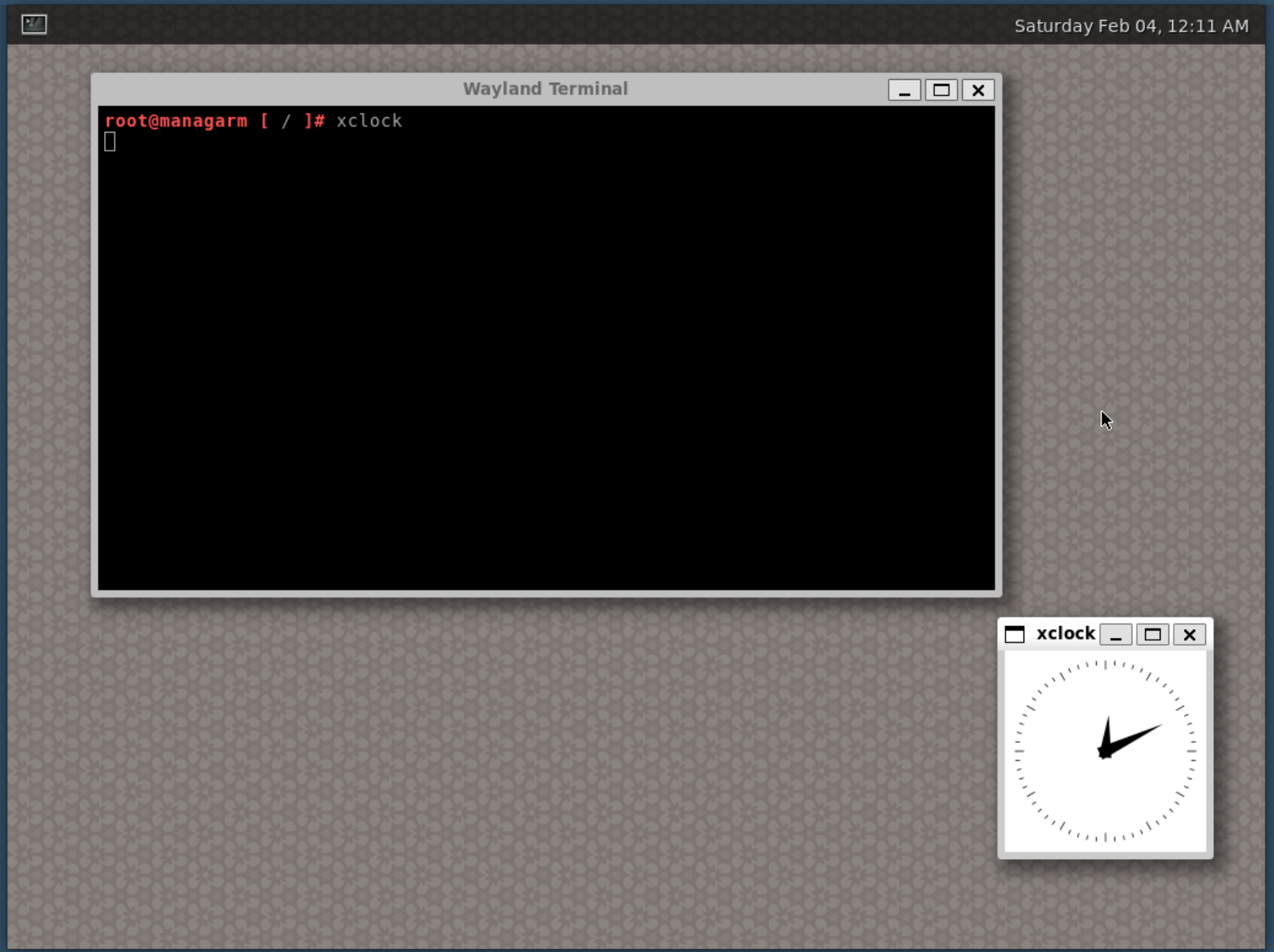
Please Try Things Out!

Exploring Is Always Better than Watching

- **Passive listening during a lecture**
  - Everything seems reasonable and logical (of course)
  - Potential issues are not obvious (people are generally optimistic)
- **Easy tips**
  - Run the code we talk about
    - Configure it, tweak it, modify it
  - Explain what you have learned in your own words
    - Talk to your roommates, friends
  - Force yourself to ask a question

Exercise: Explore a New Operating System

- **Managarm**
  - https://managarm.org
  - General-purpose
  - Desktop-oriented
  - Microkernel-based
  - Asynchronous kernel design
  - Some degree of Linux compatibility


Saturday Feb 04, 12:11 AM
Wayland Terminal
root@managarm [/]# xclock
xclock

Exploration Tips

- **Read the available documentation**
  - Don’t go into all the details, just skim it and focus on the key aspects
    - Structure of the components of the system
    - Languages and tools used
- **Get the sources**
  - Examine the directory structure
  - Look briefly into the build system
    - Not everything that is compiled is necessarily linked together

Exploration Tips

- **Start from the bottom**
  - What is the boot protocol?
    - What standard boot loader is used?
    - Is there a custom (2nd-stage) boot loader?
    - Where is the boot entry point?
  - Examine the linker script(s)
    - What is the memory layout of the kernel?
    - Where is the assembly entry point to the kernel?
    - Where is the high level language entry point to the kernel?
  - Explore the call graph of the kernel from the high level language entry point

Exploration Tips

- **Map the structure from the documentation to the sources**
  - Are there some easily distinguishable parts of the kernel?
    - Platform-specific vs. platform-neutral code?
    - Drivers?
    - Support for threads?
    - Page table management?
    - Syscall handlers?
- **Build the sources**
  - Prepare the build environment according to the documentation
  - Run the build
  - Run the built image
- **Explore the user space**

Thank you!

Questions?