Advanced Operating Systems

Summer Semester 2023/2024

Martin Děcký

1

Introduction

About the Course

- **Lecture**
  - Thursdays at 2:00 p.m. in lecture room S10
    - From February 22nd to May 23rd 2024
  - Follow up to the *Operating Systems* (NSWI200) course from winter semester
    - We assume prior understanding of basic concepts
- **https://d3s.mff.cuni.cz/teaching/nswi161/**
  - Up-to-date information and current affairs
  - Slide decks of past lectures and other study 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 implementation mechanisms of operating systems**
  - Relevant not only for system-level development
    - All abstractions are leaky to a certain degree (black boxes are rarely truly black)
    - Functional and extra-functional properties of a piece of software (reliability, performance, etc.) hard to fully assess and/or guarantee without the understanding of the properties of the underlying tiers
- **Insight into design principles of operating systems**
  - Based on specific context, requirements and constraints
    - One size does not fit all
    - Not everything has been already optimally solved

Course Structure

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

Coarse-grained Course Topics

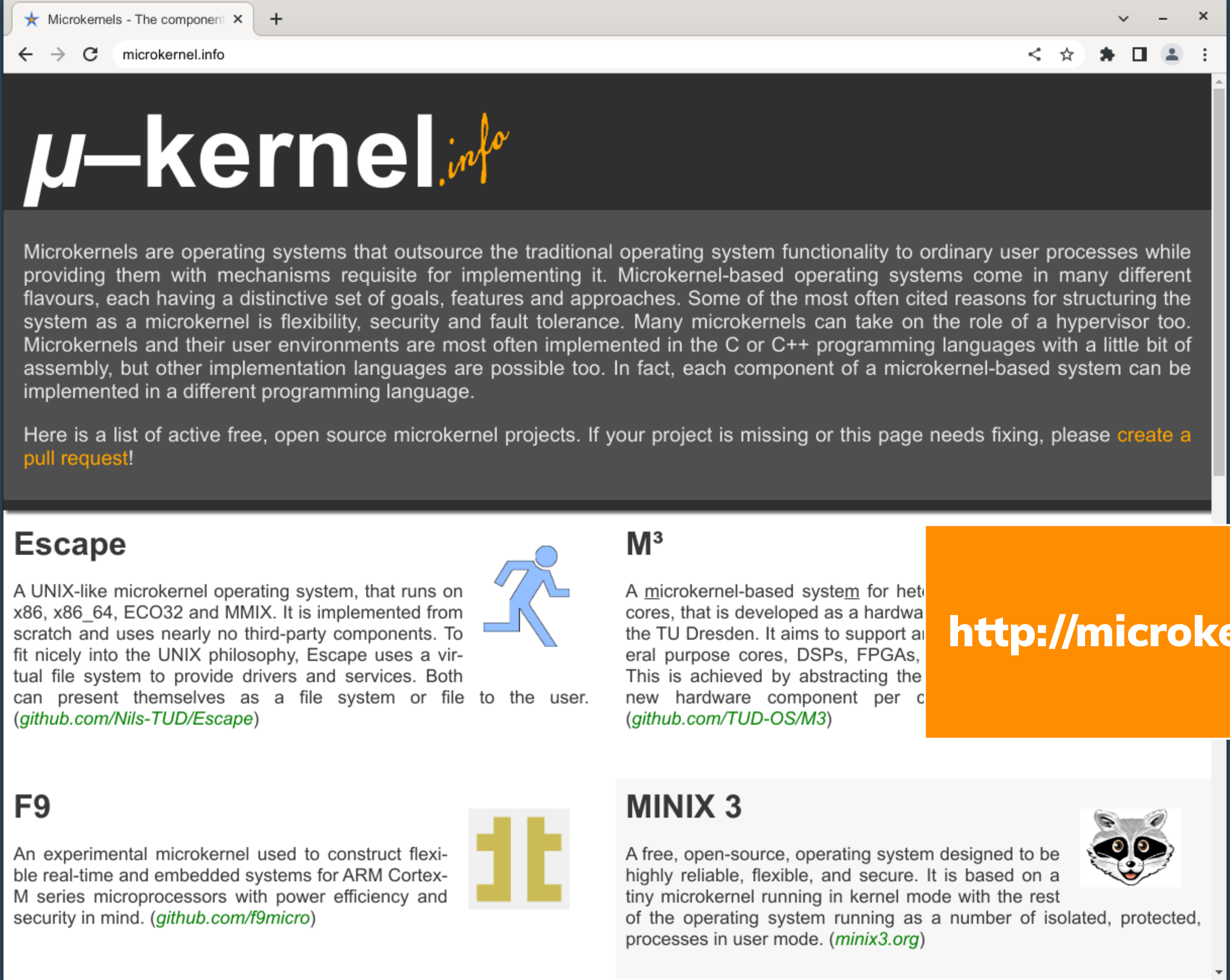
- **Subject to change of order and shift of focus**
  - Languages languages and techniques
  - Run-time environments and interfaces
  - Abstractions and interactions
  - Compatibility and portability
  - Architectures and requirements
  - Design patterns and configurations
  - Safety, security and reliability
  - Memory and resource management

Coarse-grained Course Topics

- File systems and data storage
- Observability and performance
- Debugging, tracing and instrumentation
- Communication
- Concurrency, parallelism and synchronization
- Service management
- Virtualization
- Verification, validation, certification
- 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
    - Web search and LLM queries for suggestions
    - Similar courses at other universities
    - Textbooks and academic papers from reasonable venues (e.g. OSDI, SOSP, ATC, FAST, HotOS, EuroSys, SIGOPS, etc.)
    - Dedicated on-line resources (e.g. LWN.net, OSDev.org)
    - **Open source operating systems**
    - **Hands-on experience**



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*)

M^3

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*)

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*)

http://microkernel.info

How Can a LLM Help?

- **In its own words of ChatGPT 4:**
  - *As a computer science student enrolled in an Advanced Operating Systems course, you can leverage ChatGPT in various ways to enhance your learning and understanding of the subject matter. Here's how I can be particularly useful:*
    - *If you're struggling with any of the topics listed in your course, you can ask me to provide explanations or analogies to help you grasp these concepts better.*
    - *You can pose specific questions about course material. I can provide detailed answers or guide you to resources for further reading.*
    - *I can help you understand code snippets, debug issues, or demonstrate best practices in coding, relevant to operating systems.*
    - *We can explore different software architectures and design patterns used in operating systems. I can provide insights on how these patterns are implemented in real-world systems.*

How Can a LLM Help?

  - *I can suggest academic papers, textbooks, or online resources that delve deeper into these topics.*
  - *Use me as a tool for revising topics or practicing for exams. I can help you with summaries, key points, or by conducting mock quizzes.*
  - *I can provide a historical perspective on the evolution of operating systems, highlighting the reasons behind certain design choices and how they have shaped modern operating systems.*
  - *You might explore how different operating systems handle similar tasks. I can help you compare and contrast these approaches, which can deepen your understanding of the pros and cons of various architectural decisions.*
  - *I can help you conceptualize and simulate hypothetical operating system scenarios. For example, we can discuss what might happen under certain conditions in memory management or task scheduling, enhancing your problem-solving skills in these areas.*
- *Remember, while I can be a valuable resource for information and guidance, it's important to cross-reference any critical information with your course materials or other authoritative sources, especially for academic work.*

Source: DALL·E 3 via ChatGPT 4

Credits

- **Traditional approach**
  - Written exam based on the actual topics taught
    - Half of the maximal number of points required for passing
    - List of questions/tasks will be published at the end of the semester
- **Hands-on approach**
  - Individual or small team implementation project
    - Goals and criteria need to be agreed upon between the candidates and the lecturer & guarantor
    - Do not hesitate to approach us (the sooner the better)
- **Contributor approach**
  - Preparing and presenting a standalone lecture or an extended demonstration
    - Topic not necessarily limited to the list shown before
    - The same two points as just above

Implementation Project

- **Random topic suggestions**
  - Your own serious / fun / pet / study project
    - Some non-trivial connection to operating systems required
    - Can be an extension of a previous work (e.g. your winter semester assignment) or something you plan to extend in the future (e.g. your master thesis)
      - But obviously not something you have already finished
  - Targeted contribution to an (open source) operating system project
    - **Tip:** Many open source projects have a list of Google Summer of Code projects
    - **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)
  - Co-author of the HelenOS microkernel multiserver operating system (since 2004)
- **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)
  - Contributing to the HarmonyOS NEXT microkernel-based operating system
- **Kernkonzept GmbH**
  - Senior Software Engineer (since 2021)
  - Contributing to the L4Re microkernel-based operating system framework

About Kernkonzept

Owner-managed

Founded 2012

Spin-off from TU Dresden

International team of 35

Wide experience since 1996

Continuously growing

Close to research and innovative

Operating system specialists

Located in Dresden, Germany

Kernkonzept Markets

AUTO-MOTIVE

HIGH ASSURANCE

CYBER SECURITY

SECURE ENDPOINT

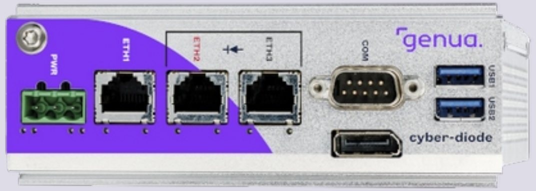
SMART HOME

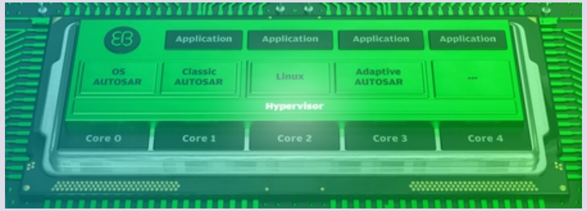
SECURE CLOUD

INDUSTRIAL IOT

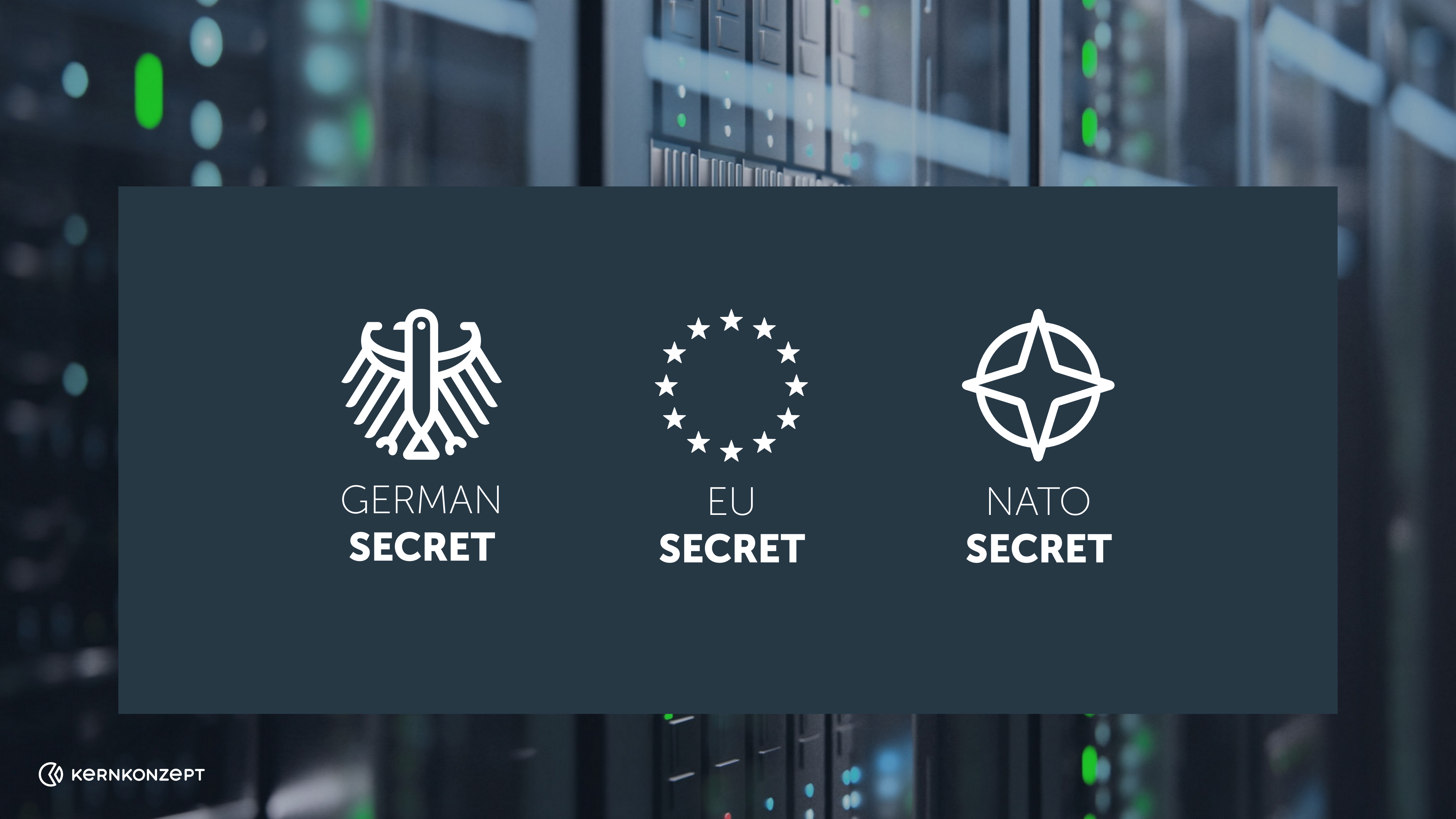
AVIONICS

Kernkonzept Customers





- **infodas**
  - *SDoT Security Gateway* and other products
    - German SECRET classification
- **genua**
  - *Secure laptop, Cyber data diode*
    - BSI approval for NATO SECRET, EU SECRET
- **Elektrobit**
  - Wholly-owned subsidiary of Continental
  - *EB Corbos Hypervisor*
    - Bare-metal mixed-criticality hypervisor for automotive systems (targeting Adaptive AUTOSAR)
    - Actually running in *Volkswagen ID.3* and other cars
- **Electrolux, Airbus, IABG, etc.**



GERMAN
SECRET

EU
SECRET

NATO
SECRET

1.1

Recommendations

Please interact

Make Sure This Course Is Useful

- **Not possible to cover every topic in the finest detail**
  - Let us focus on the topics that **you** care about
  - Let us skip parts that you already understand
- **Ask questions**
  - There is no point in listening to something you do not understand
  - There are really no stupid questions
- **Discuss**
  - Despite best effort, everyone is biased
  - This course is not about dogmas, but about nuances
  - Think about why and how would you do things differently

Please take notes

Taking Notes Helps

- **Passive listening does not lead to understanding**
  - Everything seems reasonable and logical while listening (obviously)
  - Nobody remembers all the details from a 90-minute lecture
    - Especially in a few days
  - Reformulating what you hear into concise notes helps detect that you might be missing something
- **Slides do not contain all the information**
- **Pro tips**
  - Explain what you have learned to somebody else (based on your notes)
    - Talk to a friend, roommate, etc.
  - Actively force yourself to ask a question
    - Even if everything seems clear

Please try things out

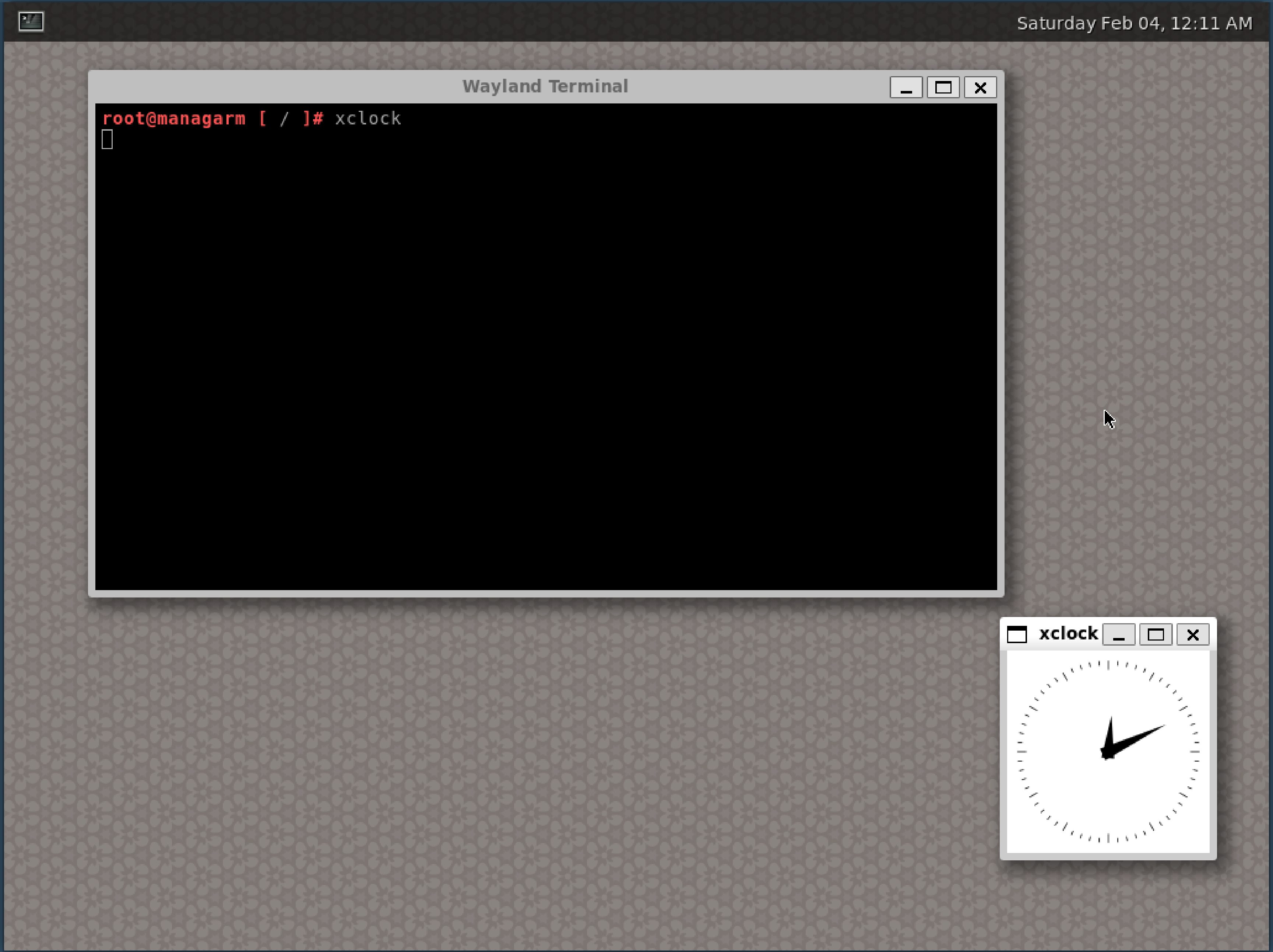
Exploring Is Always Better than Watching

- **People are generally optimistic**
  - *Have I really understood everything?* – Yes, of course!
  - *Is my current understanding sufficient for a practical application?* – Yes, of course!
  - *Am I still missing some details?* – Why should I?
- **You can be sure about the points above only if you put them to test**
  - Experiment with the ideas we talk about
    - Create a prototype, look up an actual implementation
  - Run the code
    - Configure it, tweak it, debug it, extend it
  - Try to break it
    - Finding a weak spot tests your understanding to the limit

Exercise

Explore a New Operating System

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



```
root@managarm [ / ]# xclock
```

Exploration Tips

- **Read the available documentation**
  - At first, do not drown yourself in details
    - Just skim it and focus on the key aspects
      - Structure of the components of the system
      - Languages and tools used
      - How to make it work
- **Get the sources**
  - Examine the directory structure
  - Look briefly into the build system
    - Remember: 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

- **Go back to the documentation**
  - Map the structure from the documentation to the sources
  - Are there some easily distinguishable / recognizable 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

Exploration Tips

- **Explore the user space**
  - What are the components?
  - What are the libraries?
  - How does a syscall look “from the other side”?
  - How is the user experience?
  - How to create a simple *hello world* program?
- **Explore the documentation and sources in detail**
  - What do you like and what do you dislike?

D3S

Thank you!

Questions?