universität
innsbruck

seL4 Microkernel

Interactive Theorem Proving

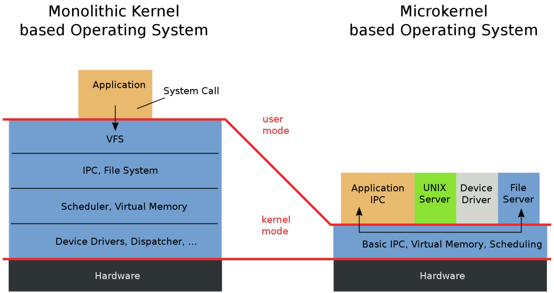
David Föger

Microkernel

> *A concept is tolerated inside the microkernel only if moving it outside the kernel [...] would prevent the implementation of the system's required functionality.* *(Jochen Liedtke)*

Microkernel

> *A concept is tolerated inside the microkernel only if moving it outside the kernel […] would prevent the implementation of the system's required functionality.* *(Jochen Liedtke)*


Monolithic Kernel based Operating System
Microkernel based Operating System
Application
System Call
VFS
IPC, File System
Scheduler, Virtual Memory
Device Drivers, Dispatcher, ...
Hardware
user mode
kernel mode
Application IPC
UNIX Server
Device Driver
File Server
Basic IPC, Virtual Memory, Scheduling
Hardware

(https://en.wikipedia.org/wiki/Microkernel)

Microkernel

Properties

- kernel complexity stable
- more overhead (mode and context switches)
    - IPC performance crucial
- small (typically 2000x smaller than monolithic kernel)
- easier to implement, port, optimize, debug

Microkernel

Properties

- kernel complexity stable
- more overhead (mode and context switches)
  - IPC performance crucial
- small (typically 2000x smaller than monolithic kernel)
- easier to implement, port, optimize, debug

Use of microkernels – OKL4

- predecessor to seL4
- used in Qualcomm modem chips
- Apple uses similar version of L4 microkernel in iOS secure enclave

What is seL4?

secure embedded Microkernel

- high-performance, i.e. *fast*
- high-assurance, i.e. *secure*
- open source, GPLv2, $\approx$10k LOC

What is seL4?

secure embedded Microkernel

- high-performance, i.e. *fast*
- high-assurance, i.e. *secure*
- open source, GPLv2, $\approx$10k LOC

Design

- developed from scratch with aim for *formal verification*
- radically minimal: memory management in user space, HW shines through
- main focus: real-world usability
  - correctness proof of implementation (not just a model of it!)
  - general purpose
  - $\leq$ 10% performance loss

seL4 – Iterative Design Process

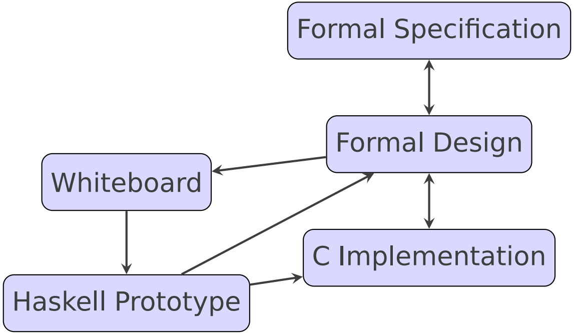
Two Sides

- OS development: highly optimized
- Formal reasoning: abstract representation

seL4 – Iterative Design Process

Two Sides

- OS development: highly optimized
- Formal reasoning: abstract representation



seL4 – Proofs

Functional correctness

- C implementation is correct w.r.t. abstract model
- free of deadlocks, livelocks, buffer overflows
- free of arithmetic overflows/exceptions, use of uninitialized variables, etc.

seL4 – Proofs

Functional correctness

- C implementation is correct w.r.t. abstract model
- free of deadlocks, livelocks, buffer overflows
- free of arithmetic overflows/exceptions, use of uninitialized variables, etc.

Translation correctness

- binary code correct w.r.t. C implementation
- all properties hold for binary code!

seL4 – Proofs

Security enforcement

- abstract model enforces CIA security confinements
  - confidentiality: data cannot be read without permission
  - integrity: data cannot be changed without permission
  - availability: data always available

seL4 – Proofs

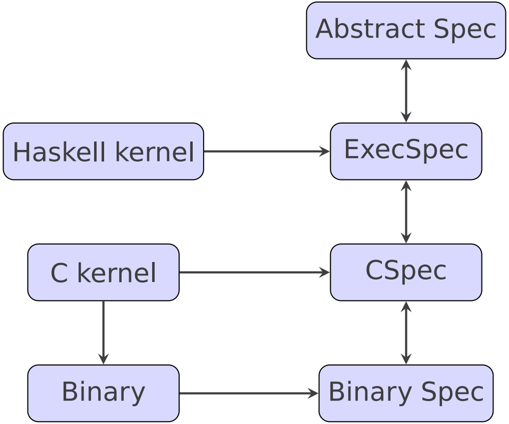
Security enforcement

- abstract model enforces CIA security confinements
  - confidentiality: data cannot be read without permission
  - integrity: data cannot be changed without permission
  - availability: data always available

Core concept: capabilities

- mechanism to grant access to specific resources in the system (object reference + access rights)
- capability is required for any operation on a kernel object
- kernel keeps track of everything in capability derivation tree
- allows for reasoning about information flow

seL4 – Refinement



seL4 – Current State

ARM 32-bit

- fully verified

seL4 – Current State

ARM 32-bit

- fully verified

RISC-V 64-bit

- fully verified

seL4 – Current State

ARM 32-bit

- fully verified

RISC-V 64-bit

- fully verified

x86 64-bit

- functional correctness only

seL4 – Current State

ARM 32-bit

- fully verified

RISC-V 64-bit

- fully verified

x86 64-bit

- functional correctness only

kernel size highly HW dependent

- almost twice as large for x86

seL4 – Limitations

Still unverified

- boot code
- details of MMU/cache
- multi-core execution

seL4 – Limitations

Still unverified

- boot code
- details of MMU/cache
- multi-core execution

Ongoing

- time protection
  - cause: competition for HW resources
  - using micro-architectural features: *hidden* by ISA (HW-SW contract)
  - Ariane (64-bit RISC-V, ETH-Zürich) allows flushing of micro-architectural state

seL4 – Building a secure System

seL4 core platform

- seL4 API very low-level, architecture-dependent, not user-friendly
- ease of development and deployment, portability
- correct use of seL4 mechanisms
- retain performance
- target HW: embedded SoC

seL4 – Building a secure System

seL4 core platform

- seL4 API very low-level, architecture-dependent, not user-friendly
- ease of development and deployment, portability
- correct use of seL4 mechanisms
- retain performance
- target HW: embedded SoC

HENSOLDT Cyber’s TRENTOS

- secure OS built on top of seL4
- important parts verified (secure boot, key store, ...)

seL4 in Action

DARPA HACMS (High-Assurance Cyber Military Systems)

- development of highly hack-resilient aerial and ground vehicles

seL4 in Action

DARPA HACMS (High-Assurance Cyber Military Systems)

- development of highly hack-resilient aerial and ground vehicles

(https://en.wikipedia.org/wiki/Boeing_AH-6)

Thank you for your attention!

seL4 Microkernel

David Föger