

Embedded Linux Architecture: Kernel Driver Development - Live Online Training

Ziele - Ihr Nutzen

How do I develop a kernel driver? What do I have to consider in embedded and real-time systems?

The development of high-performance drivers requires a basic understanding of the kernel architecture, and this is the focus of our training.

It provides an overview of the kernel structure and then highlights the components that are relevant to embedded systems. Attendees thus get a complete view of the operating system which is a prerequisite for the professional development of drivers.

The exercise comprises the basic structure of a kernel driver which is step by step extended with the mechanisms discussed in the training. At the end of the training, you have developed an entire driver and can develop drivers in your project.

Teilnehmer

Software developers, software architects

Voraussetzungen

This training requires a knowledge level as accomplished with our training "Embedded Real-Time Linux".

Live Online Training

* Preis je Teilnehmer, in Euro zzgl. USt.

Anmeldecode: LE-LIN-AR

Präsenz-Training - Englisch

Dauer

4 Tage

Live-Online - Deutsch

Dauer

4 Tage

Präsenz-Training - Deutsch

Termin

Dauer

26.10. – 29.10.2026 4 Tage

Embedded Linux Architecture: Kernel Driver Development - Live Online Training

Inhalt

Linux Kernel Basics

- System interface, privilege levels

© MicroConsult Academy GmbH

Weitere Trainings auf www.microconsult.de. Änderungen vorbehalten.

Alle Preise sind Nettopreise pro Person zzgl. gesetzlicher USt.

Kontakt: info@microconsult.de, Tel. +49 (0)89 450617-71

- Virtual filesystem, address spaces
- Device driver classes (character, block, net)
- Kernel modules

Character Device Driver

- Implementation of the file interface
- Device nodes
- Udev daemon
- Hardware access; register, IO memory, DMA
- /proc and /sys filesystem; use in kernel driver

Scheduling

- Scheduling classes
- Processes and threads, kernel threads
- Wait queue; interruptible sleep

Interrupts

- Interrupt service routine
- Secondary reactions (softIRQ, tasklet, kernel timer)
- High-resolution timer (hrtimer)

Synchronization Mechanisms

- Atomic variables
- Preemption lock, interrupt lock
- Ring buffer, kernel FIFO
- Semaphore, mutex, RW semaphore
- Completion
- Spinlock, RW lock, sequence lock
- Diagnosis of locking problems

Memory Management

- Memory protection, memory management unit (MMU)
- Memory types, DMA, high memory
- Physical memory management
- SLAB allocator, kernel malloc
- Data exchange with userspace, memory mapping

Hardware

- All exercises are performed on a phyBOARD with ARM Cortex-A8 (AM-335x) using freely accessible open source tools (remote access).