As of 18.08.2022

Embedded Linux Architecture: Kernel Driver Development - Live Online Training

Objectives

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.

Participants

Software developers, software architects

Requirements

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

Live-Online-Training

* Price per attendee, in Euro plus VAT

Training code: LE-LIN-AR

Face-To-Face - English

Duration
4 days

Live Online - German

Date	Duration
12.12. – 15.12.2022	4 days
27.03. – 30.03.2023	4 days

Face-To-Face - German

Date	Duration
12.12. – 15.12.2022	4 days
27.03. – 30.03.2023	4 days

Embedded Linux Architecture: Kernel Driver Development - Live Online Training

Content

Linux Kernel Basics

- System interface, privilege levels
- 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).