As of 03.07.2024

Embedded Real-Time Linux from Bootloader to Real-Time System (Linux RTOS) - Face-to-Face Training

Objectives

If you have to build an embedded Linux target - how do you begin? What do you need? How do you get a real-time system? This embedded Linux training focuses on the setup and operation of an embedded Linux system with hard real-time requirements.

In part 1, we start with the embedded board without SW and create the necessary components, from the bootloader to the finished real-time Linux operating system and the first embedded Linux application. You get familiar with the process and can transfer the new knowledge to your individual project and your target architecture.

Part 2 highlights the scheduling and real-time features of the RT preemption patch. Operation and use in real-time relevant application systems as well as useful synchronization mechanisms are covered in detail.

The tools used in the embedded Linux training are available for a wide variety of architectures. Thus, we make sure that you can apply the practical knowledge gained in the embedded Linux training when working with your embedded board.

Participants

The embedded Linux training addresses software developers and software architects.

Requirements

Profound ANSI-C programming knowledge as well as good basic knowledge of Linux

Embedded Real-Time Linux from Bootloader to Real-Time System (Linux RTOS) - Face-to-Face Training

Content

Development Environment

- Cross-development toolchain
- Creating a toolchain, kernel and root filesystem --> build systems
- Comparison of build methods: yocto, buildroot, Debian-based
- Hardware debugging with gdb

Exercises

- Creating a cross-development toolchain for the target Cortex-A8 with the build system buildroot
- Creating a bootable Flash for the target system

Linux Bootloader

- Configuration and installation of u-boot and barebox
- Adaptations to an individual board
- Diagnosis of boot problems

Exercises

- Configuring and creating the bootloader for the embedded board
- Configuring barebox on the target (environment and start scripts)

Linux Operating System - Linux Kernel and Root Filesystem

- Kernel configuration and generation
- Board-specific kernel adaptations
- Flattened device tree (FDT)

- RAM disk, optimized and minimum root FS with busybox
- Start process in the root FS - init daemon
- Daemons (syslog, inetd, web server, dropbear, crond)
- Libraries (uClibC, glibc)

Exercises

- Configuring and creating a Linux kernel with related device tree
- Configuring the bootloader

Flash as Mass Memory

- Unmanaged/ raw flash (NOR, NAND), memory technology devices (MTD)
- Flash filesystems (JFFS2 and UBIFS)
- FTL/ managed flash, ext-FS
- Read-only file systems, squashfs

Exercise

- Creating a root filesystem for the target on the flash

Embedded Systems - Challenges

- System update scenarios
- Boot time optimization in the bootloader, Linux kernel and root filesystem
- Boot time measurements
- Configuration of the embedded system; initial RAM FS
- Reproducible creation process
- Testing and assessing board support packages

Exercises

- Adapting the device tree
- Minimum root filesystem for boot time reduction

Real-Time Linux, RT Preemption Patch

- Definition of real time; OS requirements
- Scheduling classes (deadline and RT tasks, batch tasks)
- Threading of interrupts and SoftIRQs
- Priority inversion, priority inheritance, RT mutex, spinlock
- Comparison of real-time Linux variants: RT patch, xenomai, RTAI

Exercises

- Creating a real-time patched kernel
- Creating real-time and deadline tasks, identifying pitfalls in application

Tracing, Latency Measurement and Timer in Real-Time Linux

- Latencies in the operating system
- Power management and latencies
- Operation of the function trace framework (ftrace), trace-cmd, kernelshark
- Diagnosing real-time behavior, latencies and wakeup times
- High-resolution timer (hrtimer framework)

Exercises

- Tracing of tasks, interrupts and SoftIRQs
- Generation of Last: CPU, interrupt, memory, disk IO
- Case example: Searching for latency problems with GPIO interrupts of an actual driver

Hardware

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

MicroConsult Plus:

- Participants may keep the exercise board to deepen their new knowledge.

FACE-TO-FACE TRAINING

Price *	Duration
3.300,00 €	5 days

As of 03.07.2024

Training code: E-LIN-RTD
* Price per attendee, in Euro plus VAT

Face-To-Face - German

Date	Duration
25.11. – 29.11.2024	5 days
03.02. – 07.02.2025	5 days

Coaching

Our coaching services offer a major advantage: our specialists introduce their expertise and experience directly in your solution process, thus contributing to the success of your projects.

We will be happy to provide you with further information or submit a quotation tailored to your requirements.