Stand 17.05.2024

Embedded-Linux-Schulung: Embedded-Echtzeit-Linux vom Bootloader bis zum Realtime-System (Linux-RTOS) - Präsenz-Training

**Ziele -
Ihr Nutzen**

Sie haben die Aufgabe, ein Embedded-Linux-Target aufzubauen?

Wie fange ich damit an? Was benötige ich dazu?

Wie komme ich zu einem echtzeitfähigen System?

Der Aufbau und die Funktionsweise eines Embedded-Linux-Systems mit harten Echtzeiteigenschaften stehen im Mittelpunkt der Embedded-Linux-Schulung.

Im 1. Teil beginnen wir beim Embedded-Board ohne SW und erstellen die notwendigen Komponenten vom Bootloader bis zum fertigen Echtzeit-Linux-Betriebssystem und der ersten Embedded-Linux-Anwendung.

Sie bekommen den "roten Faden" der Vorgehensweise und transferieren das erlernte Wissen auf Ihr individuelles Projekt und Ihre Zielarchitektur.

Im 2. Teil erlernen Sie das Scheduling und die Echtzeitfähigkeit vom RT Preemption Patch.

Die Funktionsweise sowie der Einsatz in echtzeitrelevanten Anwendungssystemen wird detailliert behandelt sowie auf zweckmäßige Synchronisierungsmechanismen eingegangen.

In der Übungsaufgabe werden alle Komponenten basierend auf frei zugänglicher Open-Source-Software erstellt und eingesetzt.

Die Embedded-Linux-Schulung legt großen Wert darauf, dass die verwendeten Tools für eine breite Auswahl an Architekturen verfügbar sind.

Dadurch können Sie die praktischen Fähigkeiten der Embedded-Linux-Schulung in modifizierter Form für Ihr Embedded-Board einsetzen.

Teilnehmer

Die Embedded-Linux-Schulung richtet sich an Software-Entwickler und Software-Architekten.

Voraussetzungen

Sichere Programmierkenntnisse in ANSI-C sowie gute Linux-Grundlagenkenntnisse.

Embedded-Linux-Schulung: Embedded-Echtzeit-Linux vom Bootloader bis zum Realtime-System (Linux-RTOS) - Präsenz-Training

Inhalt

Entwicklungsumgebung

- Cross-Development Toolchain
- Erstellung von Toolchain, Kernel und Root-Filesystem --> Buildsysteme
- Buildmethoden im Vergleich: yocto, buildroot, Debian-basierend
- Hardware-Debugging mit dem gdb

Übungen:

- Erstellen einer Cross-Development Toolchain für das Target Cortex-A8 mit dem Buildsystem buildroot

- Erstellen eines bootbaren Flashes für das Zielsystem

Linux Bootloader

- Konfiguration und Installation von u-boot und barebox
- Anpassungen an individuelles Board
- Diagnose von Bootproblemen

Übungen:

- Konfigurieren und Erstellen des Bootloaders für das Embedded-Board
- Einrichten von barebox auf dem Target (Environment und Startskripte)

Linux-Betriebssystem - Linux-Kernel und Root-Filesystem

- Kernel-Konfiguration und -Erstellung
- Boardspezifische Kernel-Anpassungen
- Flattened Device Tree (FDT)
- RAM-Disk, optimiertes und minimales Root-FS mit busybox
- Startvorgang im Root-FS - init-Dämon
- Dämonen (syslog, inetd, Webserver, dropbear, crond)
- Bibliotheken (uClibC, glibc)

Übungen:

- Konfigurieren und Erstellen eines Linux-Kernels mit zugehörigem Device-Tree
- Einstellungen im Bootloader vornehmen

Flash als Massenspeicher

- Unmanaged/ Raw-Flash (NOR, NAND), Memory Technology Devices (MTD)
- Flash-Dateisysteme (JFFS2 und UBIFS)
- FTL/ Managed-Flash, ext-FS
- Read-only Filesysteme, squashfs

Übung:

- Erstellen eines Root-Filesystem für das Zielsystem auf dem Flash

Herausforderungen an Embedded-Systeme

- Systemupdate-Szenarien
- Bootzeit-Optimierung im Bootloader, Linux-Kernel und Root-Filesystem
- Messung der Bootzeiten
- Konfiguration des Embedded-Systems; Initial RAM-FS
- Reproduzierbarer Erstellvorgang
- Prüfung und Bewertung von Board Support Packages

Übungen:

- Anpassungen am Device-Tree vornehmen
- Minimalst-Root-Filesystem zur Reduzierung der Bootzeit

Echtzeit-Linux, RT Preemption Patch

- Echtzeit-Definition; Anforderungen an das Betriebssystem
- Scheduling-Klassen (Deadline- und RT-Task, Batch-Aufgaben)
- Threading von Interrupts und SoftIRQ's
- Prioritätsinversion, Prioritätsvererbung, RT-Mutex, Spin-Lock
- Vergleich unterschiedlicher Echtzeit-Linux-Varianten: RT-Patch, xenomai, RTAI

Übungen:

- Erstellen eines Realtime-gepatchten Kernels
- Echtzeit- und Deadline-Tasks erstellen und Fallstricke bei deren Einsatz kennenlernen

Tracing, Latenzmessungen und Timer im Echtzeit-Linux

- Latenzen im Betriebssystem
- Power-Management und Latenzzeiten
- Funktionsweise des Function Trace Frameworks (ftrace), trace-cmd, kernelshark
- Echtzeitverhalten, Latenzen und Wakeup-Zeiten diagnostizieren
- Hochauflösende Timer (hrtimer-Framework)

Übungen:

- Tracing von Tasks, Interrupts und SoftIRQ's durchführen
- Generierung von Last: CPU, Interrupt, Memory, Disk-IO
- Fallbeispiel: Suchen von Latenzproblemen bei GPIO-Interrupts eines realen Treibers

Hardware

- Alle Übungsaufgaben der Embedded-Linux-Schulung werden auf dem phyBOARD mit ARM Cortex-A8 (AM-335x) unter Verwendung frei zugänglicher Open-Source-Tools durchgeführt

Präsenz-Training

Termin	Preis *	Dauer
25.11.2024 – 29.11.2024	3.300,00 €	5 Tage
03.02.2025 – 07.02.2025	3.300,00 €	5 Tage

* Preis je Teilnehmer, in Euro zzgl. USt.

Anmeldecode: LIN-RTD

Präsenz-Training - Englisch

Dauer

5 Tage

Coaching

Unsere Coaching-Angebote bieten den großen Vorteil, dass unsere Experten ihr Wissen und ihre Erfahrungen direkt in Ihren Lösungsprozess einbringen und damit unmittelbar zu Ihrem Projekterfolg beitragen.

Für Ihre Anfrage oder weiterführende Informationen stehen wir Ihnen gern zur Verfügung.