Real-Time Linux with *PREEMPT_RT* training

On-site training, 2 days

Latest update: December 06, 2021

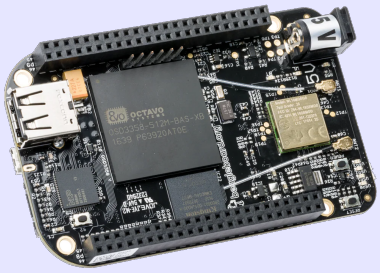
Title	**Real-Time Linux with *PREEMPT_RT* training**
Training objectives	• Be able to understand the characteristics of a real-time operating system • Be able to download, build and use the *PREEMPT_RT* patch • Be able to identify and benchmark the hardware platform in terms of real-time characteristics • Be able to configure the Linux kernel for deterministic behavior. • Be able to develop, trace and debug real-time user-space Linux applications.
Duration	**Two** days - 16 hours (8 hours per day).
Pedagogics	• Lectures delivered by the trainer: 50% of the duration • Practical labs done by participants: 50% of the duration • Electronic copies of presentations, lab instructions and data files. They are freely available at bootlin.com/doc/training/preempt-rt.
Trainer	Maxime Chevallier `https://bootlin.com/company/staff/maxime-chevallier/`
Language	Oral lectures: English Materials: English.
Audience	Companies and engineers interested in writing and benchmarking real-time applications and drivers on an embedded Linux system.
Prerequisites	• **Knowledge and practice of UNIX or GNU/Linux commands**: participants must be familiar with the Linux command line. Participants lacking experience on this topic should get trained by themselves, for example with our freely available on-line slides at bootlin.com/blog/command-line/. • **Minimal experience in embedded Linux development**: participants should have a minimal understanding of the architecture of embedded Linux systems: role of the Linux kernel vs. user-space, development of Linux user-space applications in C. Following Bootlin's *Embedded Linux* course at bootlin.com/training/embedded-linux/ allows to fulfill this pre-requisite. • **Minimal English language level: B1**, according to the *Common European Framework of References for Languages*, for our sessions in English. See bootlin.com/pub/training/cefr-grid.pdf for self-evaluation.

Required equipment	**For on-site sessions at our customer location, the customer must provide**: • Video projector • One PC computer on each desk (for one or two persons) with at least 8 GB of RAM, and Ubuntu Linux 20.04 installed in a **free partition of at least 30 GB** • Distributions others than Ubuntu Linux 20.04 are not supported, and using Linux in a virtual machine is not supported. • **Unfiltered and fast connection to Internet**: at least 50 Mbit/s of download bandwidth, and no filtering of web sites or protocols. • **PC computers with valuable data must be backed up** before being used in our sessions.
Certificate	Only the participants who have attended all training sessions, and who have scored over 50% of correct answers at the final evaluation will receive a training certificate from Bootlin.
Disabilities	Participants with disabilities who have special needs are invited to contact us at *training@bootlin.com* to discuss adaptations to the training course.

Hardware in practical labs

The hardware platform used for the practical labs of this training session is the **BeagleBone Black** board, which features:

- An ARM AM335x processor from Texas Instruments (Cortex-A8 based), 3D acceleration, etc.
- 512 MB of RAM
- 2-4 GB of on-board eMMC storage
- USB host and device
- HDMI output
- 2 x 46 pins headers, to access UARTs, SPI buses, I2C buses and more.



Day 1

Lecture - Introduction to Real-Time behaviour and determinism

- Definition of a Real-Time Operating System
- Specificities of multi-task systems
- Common locking and prioritizing patterns
- Overview of existing Real-Time Operating Systems
- Approaches to bring Real-Time capabilities to Linux

Lecture - The *PREEMPT_RT* patch

- History and future of the *PREEMPT_RT* patch
- Real-Time improvements from *PREEMPT_RT* in mainline Linux
- The internals of *PREEMPT_RT*
- Interrupt handling: threaded interrupts, softirqs
- Locking primitives: mutexes and spinlocks, sleeping spinlocks
- Preemption models

Lab - Building a mainline Linux Kernel with the *PREEMPT_RT* patch

- Downloading the Linux Kernel, and applying the patch
- Configuring the Kernel
- Booting the Kernel on a BeagleBone Black

Lecture - Hardware configuration and limitations for Real-Time

- Interrupts and deep firmwares
- Interaction with power management features: CPU frequency scaling and sleep states
- DMA

Lecture - Tools: Benchmarking, Stressing and Analyzing

- Benchmarking with *cyclictest*
- System stressing with *stress-ng* and *hackbench*
- The Linux Kernel tracing infrastructure
- Latency and scheduling analysis with *ftrace*, *kernelshark* or *LTTng*

Lab - Tools: Benchmarking, Stressing and Analyzing

- Usage of benchmarking and stress tools
- Common benchmarking techniques
- Benchmarking and configuring the BeagleBone Black Wireless

Day 2

Lecture - Kernel infrastructures and configuration

- Good practices when writing Linux kernel drivers
- Scheduling policies and priorities: *SCHED_FIFO*, *SCHED_RR*, *SCHED_DEADLINE*
- CPU and IRQ Affinity
- Memory management
- CPU isolation with *isolcpus*

Lecture - Real-Time Applications programming patterns

- POSIX real-time API
- Thread management and configuration
- Memory management: memory allocation and memory locking, stack
- Locking patterns: mutexes, priority inheritance
- Inter-Process Communication
- Signaling

Lab - Debugging a demo application

- Make a demo userspace application deterministic
- Use the tracing infrastructure to identify the cause of a latency
- Learn how to use the POSIX API to manage threads, locking and memory
- Learn how to use the CPU affinities and configure the scheduling policy

Questions and Answers

- Questions and answers with the audience about the course topics
- Extra presentations if time is left, according what most participants are interested in.