



# Real-Time Linux with *PREEMPT\_RT* training

On-line seminar, 3 sessions of 4 hours

Latest update: May 06, 2024

<b>Title</b>	<b>Real-Time Linux with <i>PREEMPT_RT</i> training</b>
<b>Training objectives</b>	<ul style="list-style-type: none"><li>• Be able to understand the characteristics of a real-time operating system</li><li>• Be able to download, build and use the <i>PREEMPT_RT</i> patch</li><li>• Be able to identify and benchmark the hardware platform in terms of real-time characteristics</li><li>• Be able to configure the Linux kernel for deterministic behavior.</li><li>• Be able to develop, trace and debug real-time user-space Linux applications.</li></ul>
<b>Duration</b>	<b>Three</b> half days - 12 hours (4 hours per half day)
<b>Pedagogics</b>	<ul style="list-style-type: none"><li>• Lectures delivered by the trainer, over video-conference. Participants can ask questions at any time.</li><li>• Practical demonstrations done by the trainer, based on practical labs, over video-conference. Participants can ask questions at any time. Optionally, participants who have access to the hardware accessories can reproduce the practical labs by themselves.</li><li>• Instant messaging for questions between sessions (replies under 24h, outside of week-ends and bank holidays).</li><li>• Electronic copies of presentations, lab instructions and data files. They are freely available at <a href="https://bootlin.com/doc/training/preempt-rt">https://bootlin.com/doc/training/preempt-rt</a>.</li></ul>
<b>Trainer</b>	Maxime Chevallier <a href="https://bootlin.com/company/staff/maxime-chevallier/">https://bootlin.com/company/staff/maxime-chevallier/</a>
<b>Language</b>	Oral lectures: English, French. Materials: English.
<b>Audience</b>	Companies and engineers interested in writing and benchmarking real-time applications and drivers on an embedded Linux system.



<b>Prerequisites</b>	<ul style="list-style-type: none"><li>• <b>Knowledge and practice of UNIX or GNU/Linux commands:</b> 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 <a href="http://bootlin.com/blog/command-line/">bootlin.com/blog/command-line/</a>.</li><li>• <b>Minimal experience in embedded Linux development:</b> 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 <i>Embedded Linux</i> course at <a href="http://bootlin.com/training/embedded-linux/">bootlin.com/training/embedded-linux/</a> allows to fulfill this pre-requisite.</li><li>• <b>Minimal English language level: B1</b>, according to the <i>Common European Framework of References for Languages</i>, for our sessions in English. See <a href="http://bootlin.com/pub/training/cefr-grid.pdf">bootlin.com/pub/training/cefr-grid.pdf</a> for self-evaluation.</li></ul>
<b>Required equipment</b>	<ul style="list-style-type: none"><li>• Computer with the operating system of your choice, with the Google Chrome or Chromium browser for videoconferencing.</li><li>• Webcam and microphone (preferably from an audio headset)</li><li>• High speed access to the Internet</li></ul>
<b>Certificate</b>	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.
<b>Disabilities</b>	Participants with disabilities who have special needs are invited to contact us at <a href="mailto:training@bootlin.com">training@bootlin.com</a> to discuss adaptations to the training course.



## Hardware platform for practical labs

### STMicroelectronics STM32MP157D Discovery Kit 1 board

- STM32MP157D (dual Cortex-A7) processor from STMicroelectronics
- USB powered
- 512 MB DDR3L RAM
- Gigabit Ethernet port
- 4 USB 2.0 host ports
- 1 USB-C OTG port
- 1 Micro SD slot
- On-board ST-LINK/V2-1 debugger
- Arduino compatible headers
- Audio codec, buttons, LEDs



## Half 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

### Demo - 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 the target hardware

### Lecture - Hardware configuration and limitations for Real-Time

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

## Half day 2

### Lecture - Tools: Benchmarking, Stressing and Analyzing

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

### Demo - Tools: Benchmarking, Stressing and Analyzing

- Usage of benchmarking and stress tools
- Common benchmarking techniques
- Benchmarking and configuring the hardware platform



## 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*

## Half day 3

---

### 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

### Demo - 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.