+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Updated on 02/04/2026

Register

Embedded Linux Development Training

4 days (28 hours)

Overview

Developing drivers under Linux is a critical skill for the industry, where system stability depends on the quality of low-level code. Unlike application development, an error in the kernel is unforgiving. It is therefore essential to master the internal mechanisms of the kernel in order to correctly integrate new devices.

Our training demystifies how the Linux kernel works. You will learn how to navigate the sources, configure and compile a custom kernel, and above all, write your own modules and device drivers. We will place particular emphasis on modern standards such as Device Tree, secure memory management, and synchronization mechanisms (locking), which are essential for ensuring the robustness of your embedded systems.

By the end of the training, you will be able to write a complete driver, manage hardware interrupts, and effectively debug your kernel modules.

Like all our training courses, this one will introduce you to **the latest stable version** of the technology and its new features.

Objectives

- Understand the architecture and subsystems of the Linux kernel.
- Configure (Kconfig) and compile (Kbuild) a custom kernel.
- Develop and integrate kernel modules (LKMs).
- Write a character device driver.
- Master the Device Tree syntax to describe the hardware.
- Manage interrupts and memory access (I/O).

Target audience

- Low-level/embedded C developers
- Linux system engineers

Prerequisites

- Solid command of the C language (pointers, structures, memory management).
- Comfortable with Linux commands and the Shell.
- Knowledge of hardware architecture (processor, bus) is a plus.

Technical prerequisites

- Linux PC or Virtual Machine provided.
- Hardware target: QEMU (emulation) and/or Raspberry Pi for real-world testing.

Embedded Linux development training

[Day 1 - Morning]

Architecture and Build System

- User Space vs. Kernel Space
- Exploring Linux kernel sources (Mainline)
- The Kconfig Configuration System
- The Kbuild build system and Makefiles
- Hands-on workshop: Configuring, compiling, and booting a custom kernel on QEMU.

[Day 1 - Afternoon] First

Kernel Module

- Structure of a module (Loadable Kernel Module)
- Macros
- Out-of-tree compilation
- License and metadata management
- Commands
- Hands-on workshop: Development, compilation, and testing of the "Hello Kernel" module.

[Day 2 - Morning]

Character drivers

- Device file concept (/dev)
- Major and minor numbers
- The structure
- Implementation of functions
- Hands-on workshop: Creating a simple driver and registering the device.

[Day 2 - Afternoon]

Exchange with User Space

- Kernel/User memory segregation
- Transfer functions
- Pointer validation and security
- Use of specific commands
- Hands-on workshop: Controlling the driver from a user application in C.

[Day 3 - Morning]

Device Tree and Platform Drivers

- The role of the Device Tree (DTS, DTB, DTC)
- Describing non-discoverable hardware
- The "Platform Driver" model: Probe and Remove
- Matching via "Compatible strings"
- Hands-on workshop: Modifying the Device Tree to declare a custom virtual device.

[Day 3 - Afternoon]

Access to Hardware (I/O Memory)

- Memory Architecture and I/O Mapping
- Resource allocation
- Virtual mapping
- Register Access Primitives
- Hands-on workshop: Controlling LEDs/GPIOs through direct register manipulation.

[Day 4 - Morning]

Interrupt management

- IRQ operation in Linux
- Registering a handler
- Contextual constraints (no sleep during interrupts)

- Deferred Processing (Bottom Halves): Tasklets and Workqueues
- Hands-on workshop: Managing a simulated interrupt (e.g., push button).

[Day 4 - Afternoon]

Concurrency and Advanced Debugging

- Understanding Race Conditions and Reentrancy
- Locking Primitives: Mutexes vs. Spinlocks
- Debugging Tools
- Analysis of "Kernel Panic" and "Oops"
- Hands-on workshop: Securing the driver with mutexes and crash debugging session.

Target companies

This training course is aimed at both individuals and companies, large or small, wishing to train their teams in new advanced IT technology or to acquire specific professional knowledge or modern methods.

Placement at the start of training

The positioning at the start of the training complies with Qualiopi quality criteria. Upon final registration, the learner receives a self-assessment questionnaire that allows us to assess their estimated level of knowledge of different types of technologies, their expectations and personal objectives for the upcoming training, within the limits imposed by the selected format. This questionnaire also allows us to anticipate certain connection or internal security issues within the company (intra-company or virtual classroom) that could be problematic for the monitoring and smooth running of the training session.

Teaching methods

Practical training: 60% practical, 40% theory. Training materials distributed in digital format to all participants.

Organization

The course alternates between theoretical input from the trainer, supported by examples and discussion sessions, and group work.

Assessment

At the end of the session, a multiple-choice questionnaire is used to verify that the skills have been correctly acquired.

Certification

A certificate will be issued to each trainee who has completed the entire training course.