FORM OF TRAINING | MATERIALS | PRICE | DURATION
--- | --- | --- | ---
Traditional | Hardcopy | 1450 EUR | 4 days
Traditional | CTAB Tablet | 1550 EUR | 4 days
Distance learning | Hardcopy | 1450 EUR | 4 days
Distance learning | CTAB Tablet | 1450 EUR | 4 days

LOCATIONS

Krakow - 5 Tatarska Street, II floor, hours: 9:00 am - 4:00 pm
Warsaw - 17 Bielska Street, hours: 9:00 am - 4:00 pm

TRAINING TERMS

2019-12-10 | 4 days | Warszawa
2020-03-23 | 4 days | Warszawa
2020-06-01 | 4 days | Kraków

TRAINING GOALS:

This 4 days course will give you the step-by-step framework for developing an embedded Linux product. You'll learn the methods used to adapt the Linux kernel and user-space libraries and utilities to particular embedded environments, such as those in use in consumer electronics, military, medical, industrial, and auto industries.

During this course you’ll learn:

- The Linux kernel architecture, emphasizing the essential points relevant to adapting the kernel to a custom embedded platform.
- The techniques for right-sizing the system to meet project constraints
- The multitude of resources available for constructing a cross development environment for embedded projects.
- The options available for populating libraries and application user-spaces to meet the goals and constraints of embedded systems.
- And more.

The information in this course will work with any major Linux distribution.
CONSPECT:

- Introduction
  - Objectives
  - Who You Are
  - The Linux Foundation
  - Linux Foundation Training
  - Linux Distributions
  - Platforms
  - Preparing Your System
  - Things change in Linux
  - Documentation and Links
  - Course Registration

- Preliminaries
  - Linux Distributions
  - Virtual Machine Installation
  - Procedures

- How to Work in OSS Projects **
  - Overview on How to Contribute Properly
  - Study and Understand the Project DNA
  - Figure Out What Itch You Want to Scratch
  - Identify Maintainers and Their Work Flows and Methods
  - Get Early Input and Work in the Open
  - Contribute Incremental Bits, Not Large Code Dumps
  - Leave Your Ego at the Door: Don’t Be Thin-Skinned
  - Be Patient, Develop Long Term Relationships, Be Helpful

- Embedded and Real-Time Systems Concepts
  - Basic Concepts
  - Protection Motivations
  - Off the Shelf (OTS)
  - Embedded Caveats
  - Real Time Operating Systems
  - Real Time Linux
  - Custom Hardware Assistance
  - Resources

- Cross-Development Environments: Goals and Needs
Introduction
Why is it Hard?
Project Goal Considerations
Links to Additional Discussions
Labs

Kbuild System
  Introduction
  Kbuild Makefiles
  Kconfig Basics

Cross-Development Toolchain
  The Compiler Triplet
  Built-in Linux Distribution Cross Compiler
  Linaro
  CodeSourcery
  crosstool-ng
  Buildroot
  OpenEmbedded
  Yocto Project
  Clang
  Labs

Basic Target Development Board Setup
  Objectives of the Lab
  Labs

Booting the Target Development Board from uSD
  Objectives of the Lab
  Labs

Booting a Target Development Board over Ethernet
  Objectives of the Lab
  Labs

Boot loaders and U-Boot
  Boot Code Stages
  Some GPL Boot Loaders
  Das U-Boot
  U-Boot Command Line
  U-Boot Environment
  Labs
- Kernel Configuration, Compilation, Booting
  - Configuring the Kernel for the Development Board
  - Labs
- Device Drivers**
  - Types of Devices
  - Device Nodes
  - Character Drivers
  - An Example
  - Labs
- Device Trees
  - What are Device Trees?
  - What Device Trees Do and What They Do Not Do
  - Device Tree Syntax
  - Device Tree Walk Through
  - Device Tree Bindings
  - Device Tree support in Boot Loaders
  - Using Device Tree Data in Drivers
  - Coexistence and Conversion of Old Drivers
  - Labs
- Target Filesystem Packaging
  - Embedded Filesystem Goals
  - Directories: a Survey
  - Embedded Filesystem Types
- Build Target Root Filesystem
  - Objectives of the Lab
  - Labs
- Root Filesystem Choices
  - SysV init vs. BusyBox init
  - udev vs. BusyBox mdev
  - Systemd
  - C Library Choices
- Configuring uClibc
  - Configuring uClibc for NFS
  - Labs
- Build BusyBox Utility Suite
  - Basic Workings
- Integrated with Buildroot
- Labs

- Kernel Monitoring and Debugging
  - Tracing and Profiling
  - Ftrace, Trace-Cmd, Kernelshark
  - Perf
  - Using perf
  - sysctl
  - SysRq Key and oops Messages
  - Kernel Debuggers
  - Labs

- Right-Sizing
  - Oft-Needed Embedded Components
  - Taking Inventory of Kernel Sizes

- Memory Technology Devices (Flash Memory Filesystems)
  - What are MTD Devices?
  - NAND vs. NOR vs. eMMC
  - Driver and User Modules
  - Flash Filesystems
  - Labs

- Compressed Filesystems
  - SquashFS
  - Deploying in an MTD Partition
  - Labs

- System Upgrades
  - When do we need to update?
  - Update strategies
  - Prebuilt upgrade systems
  - Labs

- Real-Time Extensions
  - Predictability and Preemption and Locks
  - PREEMPT_RT Project
  - Real-Time Checklist

- Closing and Evaluation Survey
  - Evaluation Survey

** These sections may be considered in part or in whole as optional. They contain either background
reference material, specialized topics, or advanced subjects. The instructor may choose to cover or 
not cover them depending on classroom experience and time constraints.

REQUIREMENTS:

The course is primarily intended for experienced developers, programmers, and engineers who are 
interested in learning how to adapt Linux to an embedded system. You should be familiar with basic 
Linux utilities, know the C programming language, and be comfortable developing for Linux or UNIX.

Difficulty level

CERTIFICATE:

The participants will obtain certificates signed by The Linux Foundation.

TRAINER:

Certified The Linux Foundation Trainer.

ADDITIONAL INFORMATION:

Pre-class preparation material will be provided before class.