# Embedded Linux Development (LFD450)

This instructor-led Embedded Linux Development course will give you the step-by-step framework for developing an embedded Linux product. Starting with the cross-compiler, you'll learn about setting up a development system, boot loaders, the kernel, drivers, device tree, and all the various software and decisions that need to be made when building a user space root filesystem, such as those in use in consumer electronics, military, medical, industrial, and auto industries. Hands-on labs with a RISC-V based emulated development target allow students to practice both coding and building the various parts of the system covered in class.

**Duration:** 5 Days

## Prerequisites for this course

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. Pre-class preparation material will be provided before class.

**Note: ** 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.**

## Outline for this course

Introduction- Objectives
- Who You Are
- The Linux Foundation
- Copyright and No Confidential Information
- Linux Foundation Training
- Certification Programs and Digital Badging
- Linux Distributions
- Platforms
- Preparing Your System
- Things change in Linux
- Documentation and Links

Preliminaries
- Linux Distributions
- Virtual Machine Installation
- Procedures
- Labs

How to Work in OSS Projects **
- Overview on How to Contribute Properly
- Know Where the Code is Coming From: DCO and CLA
- Stay Close to Mainline for Security and Quality
- 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
- Searching Kconfig

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

- Yocto Project
- Clang
- Labs

QEMU
- What is QEMU?
- Why use QEMU?
- Emulated Architectures
- Image Formats
- Labs

Booting a Target Development Board from uSD
- Why do we use uSD cards?
- Getting SW onto a uSD card
- Booting from flash
- Why is using uSD cards a bad idea?
- Labs

Booting a Target Development Board over Ethernet
- Using virtual Hardware
- An easier way to develop
- The Boot Sequence using TFTP and NFSroot
- Objectives of the Lab
- Labs

Boot loaders and U-Boot
- Boot Code Stages
- Some GPL BIOSes
- 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
- Labs

Configuring uClibc
- Configuring uClibc for NFS
- Labs

Another Alternate C-library: musl **
- What is musl?
- Configuring BuildRoot for musl
- 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
- oops Messages
- Kernel Debuggers
- debugfs

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

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

Kernel Architecture Preview
- Linux and UNIX
- Monolithic and Micro Kernels
- Main Kernel Tasks
- User-Space and Kernel-Space

Kernel Source Tree Overview
- Installation and Layout of the Kernel Source
- Kernel Browsers
- Kernel Configuration Files
- Why is it Hard? Part 2

Kernel Programming Preview
- Coding Style
- kernel-doc
- Using Generic Kernel Routines and Methods
- Error Numbers, Printing Kernel Output, syslogd
- Task Structure
- Memory Allocation
- Transferring Data between User and Kernel Space

Modules
- What are Modules?
- A Trivial Example

- Compiling Modules
- Modules vs Built-in
- Module Utilities
- Automatic Module Loading
- Module Usage Count
- Module Licensing
- Exporting Symbols
- Resolving Symbols **
- 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
- An easier way to develop
- The Boot Sequence using TFTP and NFSroot
- Objectives of the Lab
- Labs