



oD1 - Embedded Linux

Installing Linux on Embedded Systems

Objectives

- Understanding the architecture of the Linux system
- Create and use a cross-development toolchain
- Learn how to install Linux on your hardware
- Explore the Linux system architecture
 - Booting Linux
 - Initializing the system
- Install existing packages on the target
- Learn how to install Linux on flash chips

Labs are conducted QEMU ARM-based board

We use a recent version of Kernel

Prerequisite

- Good C programming skills (see our L2 course)
- Preferably knowledge of Linux user programming (see our D0 course)

Course environment

- Theoretical course
 - PDF course material (in English)
 - Course dispensed using the Teams video-conferencing system
 - The trainer to answer trainees' questions during the training and provide technical and pedagogical assistance through the Teams video-conferencing system
- Practical activities
 - Practical activities represent from 40% to 50% of course duration
 - One Online Linux PC per trainee for the practical activities
 - The trainer has access to trainees' Online PCs for technical and pedagogical assistance
- Downloadable preconfigured virtual machine for post-course practical activities

Duration

- Total: 12 hours
- 2 sessions, 6 hours each (excluding break time)
- From 40% to 50% of training time is devoted to practical activities
- Some Labs may be completed between sessions and are checked by the trainer on the next session

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Course Outline

First Session

Linux overview

- Linux
 - History
 - Version management
- Linux architecture and modularity
- Linux system components
- The various licenses used by Linux (GPL, LGPL, etc)

Cross compiling toolchains

- Pre-compiled toolchains
- Toolchain generation tools
 - Crosstool-ng
 - Buildroot
- Manual toolchain compilation

Exercise: Creating a toolchain with crosstool-ng

Linux tools for embedded systems

- Bootloaders (UBoot, Redboot, barebox)
- C libraries (glibc, eglibc, uClibc)
- Embedded GUIs
- Busybox
- Embedded distributions

The U-Boot bootloader

- Introduction to U-Boot
- Booting the board through U-Boot
 - Booting from NOR
 - Booting from NAND
 - Booting from eMMC
 - Multistage Boot
- Secure Boot
- U-Boot environment variables
 - User-defined variables
 - Predefined variables
 - Variables substitution
- The U-Boot minimal shell
- U-Boot main commands
 - Booting an OS
 - Accessing flash chips
 - Accessing file systems (NFS, FAT, EXT_x, JFFS2...)
- The full U-Boot shell
 - Script structure
 - Control flow instructions (if, for...)

Exercise: Booting the board on NFS, using pre-existing images

Second Session

Creating the embedded Linux kernel

- Downloading stable source code
 - Getting a tarball
 - Using GIT
- Configuring the kernel
- Compiling the kernel and its modules
 - Modules delivered in-tree
 - Out-of-tree modules
- Installing the kernel and the modules
- The Linux BSP overview
 - Structure
 - Device Drivers
 - Device Tree

Exercise: Configuring and compiling a target kernel for the target board

Creating a root file system

- Packages
 - Various package build systems (autotools, CMake, ...)
 - Cross-compiling a package
- The all-in-one applications
 - Busybox, the basic utilities
 - Dropbear: encrypted communications (ssh)
- Manually building your root file system
 - Device nodes, programs and libraries
 - Configuration files (network, udev, ...)
 - Installing modules
 - Looking for and installing the needed libraries
 - Testing file system consistency and completeness

Exercise: Configuring and compiling Busybox and Dropbear

Exercise: Creating a minimal root file system using busybox and dropbear

The Linux Boot

- Linux kernel parameters
- The Linux startup sequence
- Various initialization systems
 - busybox init
 - system V init
 - systemd
- Automatically starting an embedded system

Exercise: Boot Linux automatically starts a user application

Updating Linux

- Linux update solutions
 - SWUpdate
 - Mender

Appendix

Embedded file systems

- Storage interfaces
 - Block devices
 - MTD
- Flash memories and Linux MTDs
 - NOR flashes
 - NAND flashes
 - ONENAND flashes
- The various flash file system formats
 - JFFS2, YAFFS2, UBIFS
- Read-only file system
 - CRAMFS, SQUASHFS
- Standard Linux file systems
 - Ext2/3/4, FAT, NFS
- Ramdisks and initrd
 - Creating an initramfs
 - Booting through an initramfs
- Choosing the right file system formats
- Flashing the file system

Exercise: Building an initrd root file system