Mastering Embedded Linux

This intensive four-day course will teach you the techniques required to build Linux into embedded systems. During the hands-on sessions you will learn about the fundamental elements that underpin all embedded Linux projects: the toolchain, the bootloader, the Linux kernel and the root filesystem. You will create each of these elements from scratch, and discover out how to automate the process using tools such as Buildroot and the Yocto Project. In later sessions you will delve into architectural issues such as file system layout, how to split functions between user and kernel space and real-time programming. You will also find out how to debug and profile code at both the application level and within the kernel so that you will be able to identify bugs and resolve performance bottlenecks.

The theory is backed up with **hands-on labs** where you will get a chance to try out all the concepts covered in the presentations. The course is available on-site or online. On-site training takes place in your offices and will include an ARM-based development board for the labs. For the online version, we use a virtual classroom based on cloud instances and a browser UI, using the QEMU emulator as the target

Duration

4 days

Audience

This course is ideal for software engineers who are familiar with embedded devices but need to apply that knowledge to Linux development, and to those who are familiar with Linux, but want to apply that knowledge to embedded systems

Prerequisites

Essential: good knowledge of the C programming language, since this is used in the programming portions of the course

Desirable: *either* a good background in embedded devices, *or* a reasonable proficiency in Linux command-line tools. Delegates with neither will find the learning curve rather steep

Course materials

All students will receive:

- Electronic copies of the presentations and lab notes
- Worked solutions to the lab sessions

About the trainer

Chris Simmonds has been using Linux in embedded systems for over 15 years. He has been running training courses and workshops in embedded Linux since 2002 and has delivered hundreds of sessions to many well-known companies in the UK, Europe, USA, South America and SE Asia. He is the author of the book “Mastering Embedded Linux Programming”, and is a frequent presenter at open source and embedded conferences, including Embedded Linux Conference and Embedded World. You can see some of his work on the "Inner Penguin" blog at www.2net.co.uk

Enquiries and bookings

Please email training@2net.co.uk

Mastering Embedded Linux

Course outline

Introduction to Embedded Linux

- Linux as an embedded operating system
- Working with open source licenses
- The four elements of embedded Linux: toolchain; bootloader; kernel and root filesystem

Introduction to Buildroot

- Creating a project in Buildroot
- Configuration menus
- Building images for the target
- Testing on the target

The bootloader and kernel

- Booting embedded hardware: initialization; loading Linux; system maintenance
- The U-Boot bootloader: building, configuring and deploying
- The Linux kernel: main-line and vendor kernels; the development cycle
- Configuring and cross-compiling Linux

Root filesystem

- Directory layout
- Important programs: init and the shell
- Using NFS to create a networked root filesystem
- Creating an initial RAM filesystem

Init, device management and logging

- Choices for init: Busybox, SysV, systemd
- Systemd: writing systemd units
- Configuring network interfaces
- Device managers: populating /dev
- Options for the system log daemon

Embedded Linux build systems

- Using Buildroot for small projects
- Adding Buildroot packages and overlays
- Using the Yocto Project for larger projects
- Overview of Yocto meta layers and recipes

Understanding Toolchains

- Choosing the C-library
- The art of cross compiling
- Building static and shared libraries

Device trees

- Device tree syntax
- Modifying a device tree
- Using pinmux to gain access to SoC signals

Accessing Hardware

- Accessing hardware from user space
- GPIO, IIO and I2C

Debugging with GDB

- Remote debugging using gdbserver
- Crash analysis of core dumps

Linux device drivers

- Writing kernel code: kernel modules
- Anatomy of a simple device driver
- Kernel debugging: interpreting an oops
- Debugging using kgdb

Profiling and tracing

- Tools for profiling: top, perf
- Tools for tracing: strace
- Looking for memory access errors with valgrind

File systems and flash memory

- Types of flash memory: NOR, NAND and eMMC
- Choosing the right file system: UBIFS, JFFS2, and EXT4
- Designing a robust storage strategy

Scheduling and Real-time

- Linux scheduling policies
- Kernel preemption and scheduling latencies
- Approaching hard real-time with PREEMPT_RT