Embedded Linux Device Drivers

Kernel device drivers expose the underlying hardware to the rest of the system. Devices running embedded **Linux** or **Android** almost always need to interface with novel configurations of hardware, and so it is often necessary for engineers working in the embedded and Android space to be familiar with the device driver infrastructure within the Linux kernel.

This course contains the background information necessary to be able to write, configure and debug device driver code. Throughout, there is an emphasis on the techniques that are applicable to embedded systems such as platform independence, cross development and efficient use of resources. All the lab exercises are cross-compiled and tested on an ARM based development board, the BeagleBone Black.

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:

- A printed copy of the presentations and lab notes
- Worked solutions to the lab sessions
- A free copy of the trainer's book, “Mastering Embedded Linux Programming”

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 or call +44 (0)1962 869003

Embedded Linux Device Drivers

Course outline

Developing for embedded Linux

- Getting and configuring a cross toolchain
- Linux bootloaders: U-Boot

The Linux kernel

- Where to get the kernel source
- Board support packages
- Kernel configuration options
- Building and loading a kernel

Kernel modules

- Writing a module and compiling "out of tree"
- Loading and testing on the target
- Passing parameters
- Modules and the GPL license

Device driver basics

- Types of device driver
- Different ways for applications to interact with the driver
- Building and testing a simple character device driver

Kernel debugging

- Review of printk and the magic sysrq key
- Interactive debugging using kgdb
- What to do when the board won't boot

Sysfs and proc

- Classes of driver: /sys/class
- Driver attributes
- Extending the proc file system

Wait queues

- Waiting for things to happen
- Sleeping and waking

Kernel locking

- Mutual exclusion using kernel mutexes
- Spinlocks
- Atomic operations

Time and timers

- Getting the system time: high resolution timers
- Delays and sleeps
- Kernel timers

Interrupts

- Installing an interrupt handler
- Synchronising using using spin_lock_irq
- Deferred processing using tasklets and work queues

Accessing hardware

- Overview of device trees and how to extract information from them
- Mapping device memory: ioremap
- DMA buffers: coherent and stream mappings
- Memory barriers

Sharing memory

- mmap: sharing device registers
- mmap: sharing DMA buffers
- get_user_pages: how to access user memory