Embedded Linux, Kernel and Driver Programming

17 – 12 Sept given by Doulos Blackbox offices, Limerick

Objectives

Setting up a development environment
Mastering kernel development and debug tools
Discovering multi-core programming in the Linux kernel
Programming IOs, interrupts, timers and DMA
Installing and integrating drivers inside Linux kernel
Managing synchronous and asynchronous IOs and ioctl
Writing a complete character driver
Mastering kernel debugging techniques.
Discovering the specificities of USB drivers
Labs are conducted on quad Cortex/A9-based "Sabrelite" target boards from NXP
We use a recent (4.x) Linux kernel.

Course environment

Printed course material (in English)
One Linux PC for two trainees.
One target platform (i.MX6 Sabrelite) for two trainees

Prerequisite

Good C programming skills
Preferably knowledge of Linux user programming

Course Outline

First Day

Cross development toolchains 1 hour

Pre-compiled toolchains
Toolchain generation tools
Exercise: *Installing a pre-compiled toolchain*

The Linux Boot 2 hours

Booting Linux using u-boot
Linux kernel parameters
The Linux startup sequence
The SystemV initialization system
Exercise: *Boot Linux through the network using TFTP and NFS*

Creating the embedded Linux kernel 1 hour

Downloading stable source code
Configuring the kernel
Compiling the kernel and its modules
Installing the kernel and the modules
Exercise: *Configuring and compiling a target kernel for the target board*

Linux kernel programming and debugging 3 hours

Development in the Linux kernel

Memory allocation
Linked lists
Debug tools
Exercise: *Writing the "hello world" kernel module*
Exercise: *Adding a driver to kernel sources and configuration menu*
Exercise: *Using module parameters*
Exercise: *Writing interdependent modules using memory allocations, reference counting and linked lists*

Second Day

Kernel multi-tasking 2.5 hours

Task handling
Concurrent programming
Kernel threads
High Resolution Timers
Exercise*: Fixing race conditions in the previous lab with mutexes*

Linux Kernel Tracing and Profiling 2.5 hours

The Kernel tracing infrastructure
Debugging the kernel using traces
Exercise: *Analyze kernel behavior using static and dynamic traces*
Performance monitoring in the Linux kernel
Exercise: *Use perf to analyse kernel and program performances*

Introduction to Linux drivers 2 hours

Accessing the device driver from user space
Driver registration
Exercise: *Step by step implementation of a character driver: •driver registration (major/minor reservation) and device special file creation (/dev)*

Third Day

Driver installation and device access 2 hours

Kernel structures used by drivers
Opening and closing devices
Exercise: *Step by step implementation of a character driver: •Implementing open and release*

Driver I/O functions 2.5 hours

Data transfers
Controlling the device
Mapping device memory
Exercise: *Step by step implementation of a character driver:*
•Implementing read and write •Implementing ioctl •Implementing mmap

Synchronous and asynchronous requests 2.5 hours

Task synchronization
Synchronous request
Asynchronous requests
Exercise: *implementation of a pipe-like driver: •implementing waiting and waking •adding nonblocking, asynchronous and multiplexed operations (O_NONBLOCK, SIGIO, poll/select)*

Fourth Day

Input/Output and interrupts 2 hours

Memory-mapped registers
Interrupts
Gpios
Exercise: *Polling gpio driver with raw register access*
Exercise: *Interrupt-based gpio driver with raw register access*
Exercise: *gpio driver using the gpiolib*

Busses 2.5 hours

The Platform bus
The PCI bus
Exercise: *implementing a platform driver*

Linux Driver Model 2.5 hours

Linux Driver Model Architecture
Hot plug management
Writing udev rules
Exercise: *Writing a custom class driver*
Exercise: *Writing a misc driver*

Fifth Day

DMA 2 hours

Direct Memory Access
DMA programming
Memory barriers

USB Drivers 4 hours

The USB bus
User view of the USB bus and devices
USB device drivers
Exercice: *Writing a basic usb device driver using URBs*
Exercice: *Writing an usb device driver using synchronous request management*

Embedded file systems 1 hour

Storage interfaces
Flash memories and Linux MTDs
The various flash file system formats: JFFS2, YAFFS2, UBIFS
Flashing the file system
Exercise: *Building JFFS2, YAFFS2 or UBIFS file systems*