Linux kernel and device drivers programing

Hands-on course of 4 days - 28h
Ref.: LDI - Price 2024: €2 550 (excl. taxes)

CASE STUDY
A large numbers of concrete and progressive case studies will enable you to understand the internal of the kernel.

THE PROGRAMME

last updated: 01/2018

1) Linux kernel programing overview

- Getting the sources. Participating to the LKML. Licenses issues. Overview of the Linux kernel.
- Development tools. The GNU C Compiler and associated tools. indexing the kernel.
- A module primer. Loading a linux module. Using printk and dmesg.
- Loading the kernel and modules.

Hands-on work : The "hello world" module. Creation of a patch for the LKML.

2) Essential Linux kernel interface

- Processes and threads. Kernel data structures task_struct, current and the thread_info.
- Memory management. User and kernel memory spaces. UMA, NUMA, Nodes and zones.
- Synchronizations. Thread context and interrupt context. atomic operations.
- The kernel notion of time. Xtime, jiffies and HZ using delays and sleeps.
- Handling signals in the kernel. Sending and receiving signals in the kernel.

Hands-on work : Creation of little modules implementing single kernel functionalities like wait queues, completions, timers, procs interface, kernel threads and signals.

3) Interfacing with the Virtual File System

- Registering with the VFS.
- Essentials VFS callback.
- Extending the registration.

Hands-on work : Creation of a complete driver implementing a pipeline using most of the kernel utilities, memory allocation and timers. Tests using standard UNIX cat(1) command.

4) Interfacing with the hardware

- Accessing memory and devices.
- Managing DMA.
- Interrupt handling.

Hands-on work : Extension of the previous driver with interrupt handling. Creation of a little module mapping physical memory for a dedicated user program.

5) The Linux Driver Framework

- Overview. The Linux 2.6/3.x object oriented interface for drivers Kset, kobjects and kref
- Object description of device access.
- Implication on driver's development.
- Power management.

Hands-on work : Extension of the previous driver with the integration into the Linux driver framework and with power management callbacks.

TRAINER QUALIFICATIONS
The experts leading the training are specialists in the covered subjects. They have been approved by our instructional teams for both their professional knowledge and their teaching ability, for each course they teach. They have at least five to ten years of experience in their field and hold (or have held) decision-making positions in companies.

ASSESSMENT TERMS
The trainer evaluates each participant's academic progress throughout the training using multiple choice, scenarios, hands-on work and more.
Participants also complete a placement test before and after the course to measure the skills they've developed.

TEACHING AIDS AND TECHNICAL RESOURCES
• The main teaching aids and instructional methods used in the training are audiovisual aids, documentation and course material, hands-on application exercises and corrected exercises for practical training courses, case studies and coverage of real cases for training seminars.
• At the end of each course or seminar, ORSYS provides participants with a course evaluation questionnaire that is analysed by our instructional teams.
• A check-in sheet for each half-day of attendance is provided at the end of the training, along with a course completion certificate if the trainee attended the entire session.

TERMS AND DEADLINES
Registration must be completed 24 hours before the start of the training.

ACCESSIBILITY FOR PEOPLE WITH DISABILITIES
Do you need special accessibility accommodations? Contact Mrs. Fosse, Disability Manager, at psh-accueil@ORSYS.fr to review your request and its feasibility.

6) Linux Kernel Subsystems

- Storage. Data structures and implementation of block drivers.
- Network. The sockets and socket buffers: skbuf.
- Input. Data structures and interface with the Linux driver framework.
- USB. Overview, implementation and data structures.

Hands-on work : Implementation of small modules implementing a basic network interface. Implementation of an input driver moving the mouse according to a simple character interface.

DATES

REMOTE CLASS
2025 : 01 Apr, 01 Jul, 23 Sep, 09 Dec