Linux kernel and device drivers programing

Hands-on course of 4 days - 28h
Ref.: LDI - Price 2025: 2 450 (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: 04/2024

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.

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 : 23 sept., 09 déc.