

Hands-on course , 4
day(s)
Ref : LDI

Pre-requisites

A good knowledge and practice of Linux or another UNIX-like system associated with experience with C language programming are required for this training session.

Next sessions

Linux kernel and device drivers programming

OBJECTIVES

During this session you will study the Linux kernel internals. This will enable you to write better applications, optimize your Linux system or participate to Linux kernel enhancement. You will learn how to write device drivers and to integrate them with the Linux 2.6 and 3.x driver framework to provide hotplug and power management.

1) Linux kernel programming overview

2) Essential Linux kernel interface

3) Interfacing with the Virtual File System

4) Interfacing with the hardware

5) The Linux Driver Framework

6) Linux Kernel Subsystems

Case study

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

1) Linux kernel programming 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.

Workshop

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.

Workshop

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.

Workshop

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.

Workshop

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.

Workshop

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.

Workshop

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