Linux Internals & Kernel Development (ES102)

40 Hours

Outline

This course focuses on the basic elements of the Linux kernel, which allow programmers to build modules and device drivers. The students will gain a general understanding of the basic tools and interfaces needed in order to successfully modify features and develop new aspects of the kernel. During the labs, the students will build a full device driver including hardware access and interrupt handlin.

Target Audience

Programmers and software designers who plan to use the Linux kernel below the application level and to develop kernel space modules and device drivers

Prerequisites

Students should have a working knowledge of Linux user space programming

Objectives

- Explain the core elements of the Linux kernel
- Be able to use the code for modifying and building new modules
- Build complex kernel modules
- Debug a kernel module and a kernel oops
- Explain how the kernel manages memory
- Explain the use of interrupt handlers
- Understand the flow between user space and kernel space
- Understand the network sub system and write network modules

Contents

Introduction to the Linux kernel

- Kernel overview
- What is Open Source
- The system boot process
- Linux kernel history and versions
- What is a device driver
- Kernel configuration and compilation
- Kernel build system – Makefiles and KConfigs
- Kernel command line
- Downloading the source
- Building the kernel

Writing a simple kernel module

- A simple kernel module structure
- Implicit steps of compiling modules
- Using shell commands to manipulate modules
- The kernel log
- Using the printk function

Runtime information

- Passing parameters to the module
- Exporting symbols
- The / proc file system
- Sysfs

Memory Management

- Memory areas
- Memory page frames
- Requesting and releasing page frames
- Allocating contiguous virtual memory area
- The slab and slob allocators
- Memory caches and allocations
- Managing slabs
- Creating and destroying caches
- User space memory access

Implementing a character device file

- The VFS structure
- Initialization and termination
- Opening the device file
- IOCTL
- Implementing base operations

Moving data between kernel and user

- Mapping memory
- Virtual file sistems
- signals
- netlink

Debugging

- Kernel configuration for debugging
- KGDB
- Kdb
- Trace tools

Locking mechanisms

- Locking requirements
- Preemption
- Atomic bit operations
- Interrupt disabling
- Spin lock
- Semaphores

Linux Scheduler

- Process and thread
- Scheduling policies
- Priorities
- Kernel tasks
- task_struct structure
- SMP scheduling

Interrupt handling

- Hardware interrupt handling basics
- Interrupt handler and control
- Low level handling
- Wait queues technique
- Threaded interrupts

Bottom halves

- Differing work
- Using software interrupts
- Tasklets
- Timers & RTC
- Work queues

Network device drivers

- The layer model
- Registration and un-registration
- Socket buffers, allocations and manipulations
- Network headers
- Softnet basics
- Packet reception
- Packet transmission
- NAPI
- Writing a simple dummy device module

- Network queues
- Netlink
- Ip layer and routing
- Network filters

Block Layer and VFS

- Block Layer
- Block device drivers VFS
- IO schedulers
- DMA