Linux Device Drivers

Introduction:

This 5-day course gives developers the knowledge to design, write, and debug Linux device drivers.

At Course Completion

At the end of the course, students should be able to implement a fully-featured Linux device driver including:

- Understand the driver models for Linux
- Use the kernel source for Linux driver development
- Implement standard Open, Close, Read, and Write driver functions
- Perform interrupt-driven I/O
- Perform Direct Memory Access (DMA) I/O
- Debug a device driver using standard tools
- Provide for installation of a device driver

Prerequisites

Before taking this course, students should have the following skills:

- C Programming Language competency
- Experience with C development environment
- Some knowledge of device driver development on other systems (HW and/or SW)
- User-level experience with Linux

Chapter 1: Introduction to Linux Kernel

- Kernel vs. User Mode Protection
- Virtual Memory Management
- Symmetric Multiprocessing
- I/O Operations in Linux
- Linux Kernel Map

Chapter 2: Hardware Terminology

- Introduction to hardware terminology
- Device registers and addressing
- Character vs. Block Devices
- Interrupts
- DMA
- Device memory
- Buses

Chapter 3: Linux Driver Architecture

- User Requests

- Driver request handlers
- Driver Events
- Module init and exit
- printk
- Device registration
- Announcing entry points
- Unified Device Model

Chapter 4: First Driver Code

- Build environment for a driver
- Runtime environment for a driver
- Write "core driver code" (Hello World driver)
- Lab: First Driver Code

Chapter 5: Handling User Requests

- The open, close, read, write semantics
- Accessing the user buffer
- File position
- Associating information with a device
- Lab: Read/Write Loopback

Chapter 6: Working with Real Hardware

- Using I/O port addresses
- Accessing device registers
- The parallel port HW
- Lab: Parallel Port Loopback

Chapter 7: Advanced I/O Operations

- ioctl implementation
- Blocking I/O
- poll and select
- Asynchronous notifications
- Controlling device access

Chapter 8: Interrupt Driven I/O

- Describe the sequence of events for a Programmed I/O Device operation
- Describe the use of an Interrupt Handler Routine
- /proc/interrupts
- Examine data transfer routines
- USB Interrupt packets – not the same thing
- Lab: Writing an Interrupt Handler

Chapter 9: Memory Management

- Virtual Address Translation
- Page Faults
- Uniform vs. Non-uniform Memory Access

- Physical Memory Management
- Nodes, Zones, and Pages
- Accessing "high" memory

Chapter 10: DMA

- DMA Device Concepts
- mmapsystem for user space
- kiobuf
- VMA structure
- mmap
- Allocating DMA memory
- Lab: A DMA walk-thru

Chapter 11: Debugging Linux Drivers

- A Linux Panic
- Hard vs. Soft panics
- Hard panic causes (aieee!)
- Deciphering a trace
- Soft panic troubleshooting (oops)
- Using kdb
- Lab: Debugging Exercises

Chapter 12: Synchronization Issues in Drivers

- Race conditions in a driver

- Linux kernel synchronization methods
- Atomic operations
- Spinlocks
- Kernel mutexes
- Big kernel lock
- Lab: Synchronization Exercise

Chapter 13: Timers

- Kernel time
- jiffies
- Stalling execution of code
- Task queues and tasklets
- Kernel timers
- Lab: Handling device timeouts

Chapter 14: Block Drivers

- Block driver operation
- Registering a block device
- Block device operations
- Request queues
- Lab: Block driver

Chapter 15: USB Drivers

- USB Terminology
- USB within sysfs
- The USB request block
- Registering USB drivers
- Bulk transfers