INDIA's one & only
INDUSTRIAL Embedded Systems Training Institute
KERNEL MASTERS

Online Training on

Linux Device Drivers

By Kishore Kumar Boddu

Session Highlights:

- Participants will develop a deep understanding the Linux Device Driver Programming with Real Time Examples.
- This course will teach you how to develop device drivers for Linux systems, grounded with a basic familiarity and understanding of the underlying Linux kernel.
- Online Sessions will be an assignment driven model so that participants can have a deep understanding of system programming well as kernel mode programming practices.
- Adds the following skill set to your profile: **Linux Device Drivers, Setup Device Driver Development Environment, The Role of Device Driver, Platform Driver, Bus Driver, Linux Interrupt Handling, Kernel Timers, Kernel Threads, concurrency in kernel etc.**

Prerequisites:

- We assume that attendees are fully fluent in C, data structures
- Should be familiar with Linux/Unix command line.
- Linux System Programming

Audience:

- This session is mainly intended for those looking to start their career in Linux Device programming or for those already working in Device Drivers.

End of the course you'll be able to:

- Create, load, and manage drivers.
- Register for, manage and defer interrupts
- Effectively select and use different locking mechanisms
- Used the different kinds of device drivers used in Linux
- Use the appropriate APIs through which devices (both hardware and software) interface with the kernel.
- And more.

Syllabus Summary: (Detailed agenda in Next page)

1. Linux Kernel & Device Driver Introduction
2. System Call Implementation
3. Module Programming
4. Character Device Drivers
5. Handling Concurrency in the kernel
6. Advanced Character Device Drivers
7. Communicating with Hardware
8. Linux Interrupt Mechanism
9. Kernel Mechanisms
10. Adding a Driver to the kernel Tree

Authored and Compiled By: Boddu Kishore Kumar
Email: kishore@kernelmasters.org
Reach us online: www.kernelmasters.org
Contact: 9949062828

Linux Device Drivers Detailed Agenda

Note: All Sessions are highly interactive hands-on-sessions.

Session 1: Linux Kernel & Device Drivers Introduction

Objective: To understand Linux Kernel source code overview and Device driver's basics. Kernel Configuration, compilation and build system. Device Driver Programmer Responsibilities.

Topics to be covered :

Introduction to Linux Kernel

1.1. Two Roles of Kernel.
1.2. Kernel Programming
1.3. Linux Source tree Overview
1.4. /proc and /sys virtual file system.
1.5. Types of Kernel

Introduction to Device Drivers

1.6. What is Device Driver? Types of Device Drivers
1.7. Classes of Device Drivers
1.8. The Role of the Device Driver
1.9. Configuring, Compiling and Booting the Linux Kernel
1.10. Kernel Configuration
1.11. Booting the kernel.

Hands-On-Session:

- Linux Kernel Configuration and Compilation Process (Create own built kernel)

Session 2: System Call Implementation

Objective: To understand how system call works and how to initialize Interrupt Vector Table in Linux Kernel.

Topics to be covered :

Introduction to Linux Kernel

2.1. How to add a system call in kernel.
2.2. X86_64 Interrupt Vector table Initialization.

Hands-On-Session:

- Add Hello world system call in kernel space.

Session 3: Module Programming

Objective: To learn, how to write a Module Programming, compile a Module, Module Parameters and Module Dependency.

Topics to be covered :

Module Programming

3.1. What is a Kernel Module?
3.2. User mode vs Kernel mode
3.3. Module Template
3.4. Module parameters
3.5. Module Dependency
3.6. Kernel Specific GCC Extensions (__init and __exit)

Hands-On-Session:

- Write a Module Program Template and compile a Module Program
- Write a Module parameter sample Program
- Write a Module Dependency sample program

Session 4: Character Device Drivers

Objective: To Learn, Character Device Framework and How to write a character device driver template program.

Topics to be covered :

4.1. What is Character Device Drivers
4.2. Character Device Drivers Framework
4.3. Major and Minor numbers.
4.4. Implementation of Character Driver.

Hands-On-Session:

- Write a Character Device Driver template.
- How to transfer data from kernel to user space.
- Write a Multiple Character Device Drivers

Session 5: Handling Concurrency in the Kernel

Objective: To Learn, How to use Concurrency Mechanisms in Device Drivers.

Topics to be covered :

5.1. The Need for Atomicity
5.2. Causes of Concurrency in the kernel
5.3. Concurrency in the Kernel
5.4. Spinlocks and Mutexs
5.5. The Semaphore Interface

Hands-On-Session:

- Write a Program to overcome drawback of calculator application use Semaphores.

Session 6: Advanced Character Device Drivers

Objective: To Learn, How to use Advanced Character device drivers like ioctl, and Blocking I/O.

Topics to be covered :

6.1. ioctl
6.2. Blocking I/O
6.3. Asynchronous notification

Hands-On-Session:

- Write a program display Kernel Masters logo using ioctl(), mmap() system calls

Write a program to read Keyboard and mouse data.

Session 7: Communicating with Hardware

Objective: To Learn, How to communicating different types of hardware.

Topics to be covered:

7.1. I/O Ports (I/O Mapped I/O)
  7.1.1. Intel Architecture (Example Parallel Port and Serial Port)
  7.1.2. Inb(), outb() kernel functions
7.2. I/O Memory (Memory Mapped I/O)
  7.2.1. ARM Architecture (AM3358 Processor)
  7.2.2. ioread() and iowrite() kernel functions

Hands-On-Session:

- 8 LED's interface with Parallel Port and blink LED's.

Session 8: Linux Interrupt Mechanisms

Objective: To Learn, How Interrupt Works and how to write Interrupt handler in Linux.

Topics to be covered :

Interrupts Basics:

8.1. Mode Of I/O
- 8.1.1. Programmed Control I/O
- 8.1.2. Interrupt Driver I/O
- 8.1.3. DMA

8.2. How Interrupt Works?

8.3. Device Identification
- 8.3.1. Multiple Interrupt Lines – Multiple Devices
- 8.3.2. Single Interrupt Line – Multiple Devices (PIC)

Linux Interrupt Mechanisms:

8.4. Process context vs Interrupt context
8.5. Installing an Interrupt Handler
8.6. Interrupt Handler Constraints
8.7. Handler arguments and Return Values
8.8. Interrupt Control Methods.
8.9. Disabling and Enabling Interrupts
8.10. Status of the Interrupt system
8.11. Top and Bottom Halves
- 8.11.1. Taskletes
- 8.11.2. Softirqs
- 8.11.3. Workqueus

8.12. Examples
- 8.12.1. Parallel port
- 8.12.2. Keyboard and Mouse

Hands-On-Session:
- Write a Keyboard/Mouse sample interrupt Example.
- Parallel port example

Session 9: Kernel Mechanisms

Objective: To Learn, general tasks and mechanisms that the Linux kernel needs to supply so that other parts of the kernel work effectively together.

Topics to be covered

9.1. Kernel Threads
- 9.1.1. Kernel Threads vs User Threads

9.2. Kernel Timers
- 9.2.1. Delaying Execution

9.3. Tasklets
9.4. Workqueues

Hands-On-Session:
- Create a Kernel Thread

Session 10: Adding a Driver to the Kernel Tree

Objective: To Learn, How Interrupt Works and how to write Interrupt handler in Linux.

Topics to be covered:

10.1. kernel layout for drivers
10.2. Modifying the Makefile
10.3. Adding it to configuration options - the Kconfig file

Hands-On-Session:

- Add hello world driver in kernel space.