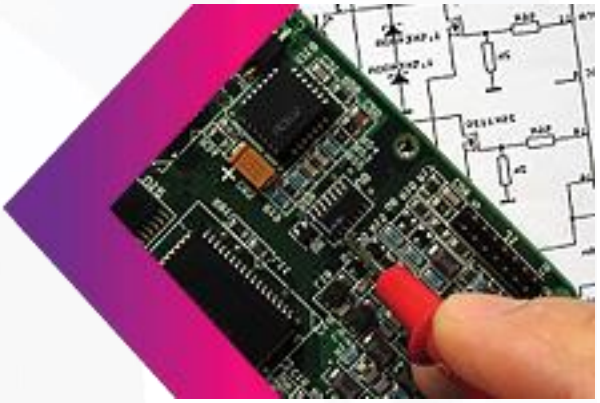
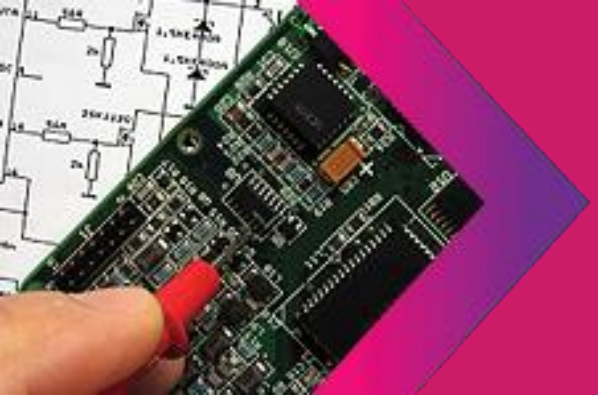
EMERTXE

"Here to take you beyond"

LINUX DEVICE DRIVERS
Weekend Workshop





NOW FOR SIMPLIFIED SOLUTIONS

Linux Device Drivers – Weekend workshop

- **Objectives:**
  - ✓ To get you started with writing device drivers in Linux
  - ✓ Provide real time hardware exposure
  - ✓ Equip you with real-time tools, debugging techniques and industry usage
  - ✓ Special focus on character drivers. Hardware access for ioports and iomem
  - ✓ Develop USB device drivers
  - ✓ Implement Interrupt Handling and Bottom Halves
  - ✓ Become familiar with Block device drivers and PCI driver
  - ✓ Learn how to debug the kernel

- **Overview:**
  This workshop provides a practical overview about the structure of device drivers, in-depth information about the interface between the device driver and the rest of the Linux kernel, and various practical exercises to develop and test device drivers for major devices in a Linux environment. It also covers the initiators and detailed insights of writing a Device Driver in Linux, focused and aligned towards the industry perspective.

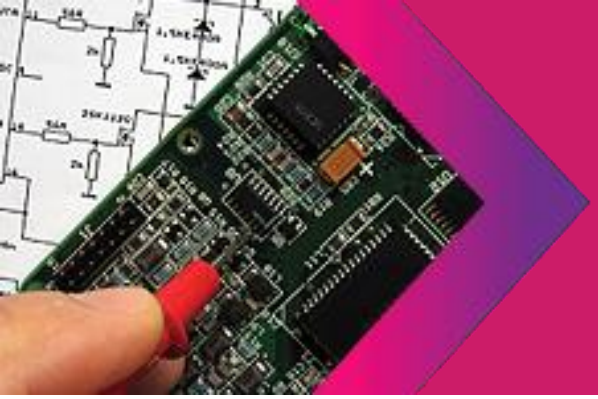
  Ideally designed for working professionals to gain device drivers competency by attending weekend sessions.

- **Duration:**
  4 days (Two weekends)

- **Platform:**
  OS - Ubuntu Linux
  Hardware - Emertxe Linux Device Drivers learning kit SDK
  Kernel version – 3.x/4.x

- **Delivery method:**
  Workshop based approach delivered in a fast-track

- **Pre-requisites:**
  - ✓ Good C & Programming Skills
  - ✓ Basic Hands-On Linux Usage
  - ✓ Good to have - Understanding of basic File related system calls



- **Detailed course contents:**

Day-1:

✓ **Introduction**
- Linux Driver Ecosystem
- The Kernel Source Organization
- Driver Development Environment
- Knowing the commands
- Writing your first driver
- /proc files
- /boot contents

✓ **Character Drivers:**
- Major & Minor Numbers
- Registering & Unregistering
- Device Files & Device Classes
- File Operations & its related Kernel Data Structures
- Special Focus on open, release, read, write
- Memory Access in Kernel Space
- Multiple minors
- Module arguments

Day-2:

✓ **Hardware Access Mechanisms:**
- System Memory
- Device Memory
- I/O Ports
- Getting familiar with “Emertxe's LDD Kit”
- Memory allocation
- ioctl system call
- ***Writing a character driver for a serial based device***

✓ **Interrupts**
- IRQs & their Registration
- IRQ Handling & Control
- Soft IRQs
- Top & Bottom Halves
- ***Illustration of Interrupt handler of Keyboard device (i8042 controller)***

✓ **Time keeping, Delays, and deferred work**
- Time since bootup
- Knowing the Current Time
- Delaying Execution
- Kernel Timers

- Tasklets
- Workqueues
- ***Illustration of these concepts using demo drivers***

Day 3:

✓ **Concurrency**
- Concurrency and Its Management
- Semaphores and Mutexes
- Spinlocks
- Alternatives to Locking
- Other Synchronization mechanisms
- ***Illustration of these concepts using demo drivers***

✓ **USB Drivers**
- USB Device Layout
- USB Driver Layout
- USB Core & Sysfs
- USB Driver Registration
- USB Device Hot-plug-ability
- URB & its Operations
- Special Focus on Control & Bulk Transfers
- Loading the firmware for Emertxe LDD kit.
- ***Implementation of USB driver for CDC-ACM***

Day 4:

✓ **Block Drivers**
- Driver Registration
- Disk Drive Registration
- Block Device Operations & its related Kernel DS
- Request Queues & their Processing
- Creating partitions
- Implementation of memory based block driver
- Testing the block driver using dd command
- Associating a file system driver with block driver (mounting)

✓ **PCI Driver**
- PCI Interface
- PCI Configuration Space
- PCI Driver Registration
- PCI Device Access & Operations

- ✓ **Debugging**
  - Debugging Support in the Kernel
  - Debugging by Printing
  - Debugging by Querying
  - Debugging by Watching
  - Debugging using kdb

- **Hands-on session details:**

  - ✓ The Driver specific Commands & Tools

  - ✓ Setting up the Driver Development Environment
    - Understanding Kernel's Build System
    - Writing your Makefile
    - Adding a driver to kernel

  - ✓ Writing your “first” Driver

  - ✓ Writing various Character Drivers
    - Null Driver
    - Memory Based Driver
    - Multiple minors
    - Module parameters

  - ✓ UART (Hardware) Based Drivers
    - Char drivers using UART to apply char driver concepts
    - Implementing ioctl function

  - ✓ Understanding the USB Ecosystem
    - Walk through of Procfs & Sysfs in relation of USB
    - Understanding the USB Device entries
    - Interfacing with the USB Core
    - Reading USB cdc adm class specification

  - ✓ Writing a USB Driver
    - USB Driver & Device Registration
    - Hot-plug-ability: probe and disconnect
    - Bulk Transfers & Various System Calls
      USB driver for peripherals on Emertxe's LDD kit

  - ✓ Interrupt implementation using deferred execution

  - ✓ Concurrency management using mutex, semaphores

  - ✓ Memory based Block Driver

- ✓ Implementation a char based PCI driver
- ✓ Using kdb debugger
- ✓ Creating proc entries for a driver

- **Workshop Highlights:**

  - *Platform: x86-based*
  - *Kernel: Version 3.x / 4.x*
  - *Drivers: For Real Hardware*
  - *Special Focus: Live hands-on with Character and USB "Device" Drivers*
  - *Kernel debugging using various techniques*

Emertxe Information Technologies Private Ltd
#83, 1st Floor,
Farah Towers, MG Road,
Bangalore – 560001

T: +91 809 555 7 333 (M) +91 80 4128 9576 (L)
E: training@emertxe.com

www.emertxe.com