Embitude

Let's have fun with Embedded

Online Training on Embedded Linux Kernel Internals
By Pradeep Tewani

Description
The Linux Kernel Programming course provides the insights into the Linux kernel programming for the Embedded Systems. The course focuses on various programming constructs & data structures required for the Linux driver development.. The course starts with the basics of Linux driver & then proceeds to cover the character drivers and thereafter covering linux kernel programming concepts such as process management, synchronization, interrupt management.

Course Objective
The Linux kernel programming course attempts to serve multiple objectives:

- To enable participants to understand the fundamental of Linux device driver
- To enable pariticipants to understand the complete character driver aspects
- To enable participants apply the kernel programming concepts such as synchronization and interrupt management.

Target group:
Professionals looking to get into Linux device drivers development.

Pre-requisite
Knowledge of C & basic knowledge of Linux

Learning Outcome
- Acquaintance with Linux kernel source organization
- Understand Comfortability with Linux kernel module & related commands
- Understand the character driver
- Understand the Linux kernel programming constructs such as kernel threads, synchronization mechanisms & wait queues
- Understand the Linux kernel timing architecture & interrupt management
- Understand the interrupt management & bottom halves

Methology
Every theoretical topic is accompanied by corresponding hands-on/assignment to get the deep understanding of the topic.

Assessment
Assignment Based

+ Session 1: BBB Set up & Introduction to Linux Driver
- Readying BBB for Linux Kernel Internals
- Linux Driver Ecosystem
- Kernel Source organization

Exercises
- Configure & build the kernel
- Writing a simple Linux kernel module

- Statically building the driver into the kernel

+ ***Session 2: Linux Kernel Module***
- Understanding the Kernel module & related commands
- Writing & Building a first Kernel module

+
Session 3: Character Driver Part - 1
- What is Character driver?
- Major & Minor Number
- Registering & Unregistering the driver
- Writing a First Character Driver

Exercises
- Write a simple character driver
- Enchance the driver to register the file operations
-

+ ***Session 4: Character Driver Part - 2***
- Enhance the driver to exchange the data with user space
- Udev & automatic device file creation
- Controlling the GPIOs
- IOCTL

Exercises
- Enhance the driver to exchange the data with user space
- Enabling the autoloading of driver in Embedded Linux system
- Write the driver to control the on-board leds
- Enhance the driver to support the IOCTLS

+ ***Session 5: Kernel Process Management***
- Synhronization Mechanism – Mutex, Semaphores & Spinlocks
- Waiting in Process
- Sleeping & Waking up
- Wait Queues

Exercises
- Write a driver to handle the consumer/producer problem
- Write a driver to demonstrate the usage of spinlocks
- Write a simple linux driver to block the process
- Enhance the driver to use the wait queues

+ ***Session 6: Kernel Timing Management***
- Kernel Timing Architecture
- Ticking in Jiffies
- Kernel Timers

Exercises
- Write a driver to demonstrate the usage of Kernel timers

+ ***Session 7: Interrupt Management & Deferred work***
- What is interrupt?
- Need for interrupts

- How interrupts work?
- Registering an interrupts handler in linux
- Soft IRQ
- Bottom halves – Tasklets & Work Queues

Exercises
- Write a driver to handle the interrupts
- Register the tasklet as the bottom half
- Register the work queue as the bottom half

+ Session 8: Wrap Up
- Conclusion
- Next Steps