Embitude

Let's have fun with Embedded

Embitude's Linux Rapid Mastery Bundle

Description

Linux Rapid Mastery Program – the perfect launchpad for aspiring Linux Driver Developers! This comprehensive course begins with Linux Fundamentals to ensure you're comfortable in the Linux environment. Next, delve into Linux Application Development, gaining essential skills for creating basic applications. After mastering applications, build a strong foundation in Linux Driver Development. Finally, the Linux Kernel Internals module will empower you to write advanced drivers using key programming constructs like synchronization, timing management, and interrupt management.

Course Objective

The Linux Rapid Mastery Bundle attempts to serve multiple Objectives

- To enable participants understand the fundamental of Linux system & getting comfortability with working under Linux enviroment, so as to help them prepare for Linux Device Drivers
- To enable participants with solid Fundamentals of Linux System Programming which would enable them to get the comfortability with end-to-end System.
- To enable participants with solid Fundamentals of Linux Device Drivers

Target Group

Professionals/Students looking to get into Linux Device Drivers

Prerequisites

Knowledge of C Programming

Methodology

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

Package Includes

- Module-1: Linux Fundamentals
- Module-2: Linux Application Development
- Module-3: Linux Drivers
- Module-4: Linux Kernel Internals
- Life Time Access to Recorded Sessions & Assignments
- Life Time Access to Community (Private Whatsapp Group)

- Life Time Access to Inner Circle Meetups (Weekly Calls)
- E-Certificate

Module-1
Linux Fundamentals

Introduction
- The Big Picture
- Roadmap to Linux Driver Developer

Linux & its Architecture
- History of Linux
- OS & its Tasks
- Linux Architecture

Setting up the Linux System
- Various ways to setup the Linux System
  - Windows Subsystem for Linux (WSL)
  - Virtual Machine
  - Dual Boot
  - Cloud-Based Solutions
  - Browser-based Solutions

Linux Usage Basics
- Linux Directory Structure
- System Directories
- The Shell
- File Permissions

Essential Commands in Linux
- Linux Directory Structure
- System Directories
- File & Directory related commands (cp, mv, mkdir, rm etc)
- Zipping/unzipping the directory
- Get the CPU and Memory info of the system
- Symbolic links
- Mounting/Unmounting the partition
- Connecting to the remote system
- Sending a file over the network
- Productivity Hacks for Linux Command-Line

Embedded Linux Components
- Various Components that Constitute Embedded Linux

Toolchain

- What is Toolchain
- Toolchain Components
- GCC & Its Friends

Makefile

- Makefile & Make Utility
- Makefile Components
- Writing Your Makefile

Git Fundamentals

- Git & its need
- Clone & GIT Project
- Commiting the Changes
- Creating the Patches

Editors in Linux

- Various Editors in Linux
- vim Basics
- Editing with vim
- Productivity Hacks for Vim

Module-2
Linux Application Development

System Calls in Linux

- System Call & its Need
- Tracing System Calls
- System Call Execution Flow
- System Calls vs Library Functions
- System Call Examples

Processes in Linux

- Process Overview
- Process Creation & operations
- Waiting for the Process termination
- Zombie Processes

Exercises/Assignments

- Creating a processes
- Exec'ing a process
- Waiting for the child process to terminate
- Creating zombie & orphan process

Signals

- W's of Signals
- Types of Signals
- Signal Examples
- Project Stage-1

Exercises/Assignments

- Registering a signal handler
- Masking a Signal in the Handler

Threads in Linux

- W's of thread
- POSIX Threads & their Internals
- Threads Creation, Operations & Usages
- Thread Joining
- Thread Cancellation

Exercises/Assignments

- Write a program to create the thread
- Write a program to demonstrate the usage of pthread_join
- Write a program to cancel the thread

+ Session 7: Synchronization in Linux

- Synchronization Overview
- Synchronization Mechanisms

Exercises/Assignments

- Write a program to solve the consumer/producer problem

Module-3
Linux Drivers

Introduction to Linux Drivers

- Driver & its Role
- Linux Driver Ecosystem

Linux Kernel

- Downloading Linux Kernel Source
- Kernel Source Code Organization

Linux Kernel Module & related Commands

- Understanding the Linux Kernel Module & related commands
- Writing & Building a first Kernel module

Character Driver Part - 1

- What is Character driver?
- Major & Minor Number
- Registering & Unregistering the driver
- Writing a First Character Driver

Exercises/Assignments

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

Character Driver Part - 2

- Enhance the driver to exchange the data with user space
- Udev & automatic device file creation
- IOCTL

Exercises/Assignments

- Enhance the driver to exchange the data with user space
- Enhance the driver to support the IOCTLS

Module-4
Linux Kernel Internals

Kernel Synchronization

- Synhronization Mechanism – Mutex, Semaphores & Spinlocks

Exercises/Assignments

- Write a driver to handle the consumer/producer problem
- Write a driver to demonstrate the usage of spinlocks

+ Session 6: Kernel Timing Management

- Kernel Timing Architecture
- Ticking in Jiffies
- Kernel Timers

Exercises/Assignments

- Write a driver to demonstrate the usage of Kernel timers

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/Assignments

- Register the tasklet as the bottom half
- Register the work queue as the bottom half