Embitude

Let's have fun with Embedded

Embitude's Embedded Proficiency Mode (Silver Membership)

Package Includes

- Module-1: Linux Application Level Proficiency (30 Day Challenge)
- Module-2: Linux Driver Level Proficiency (60 Day Challenge)
- Module-3: Embedded Linux System Level Proficiency (60 Day Challenge)
- Life Time Access to Hackathons with 10 Missions for Live Assignments
- Life Time Access to Community (Private Telegram & Discord Channel)
- Life Time Access to Inner Circle Masterminds (On Weekends)
- Life Time Access to Recordings

Module-1
Linux Application Level Proficiency (30 Day Challenge)

Module Description
This Module provides a deep insight into the Linux operating system concepts. The various user space programming concepts such as process, threads, synchronization, Interprocess communication mechanisms are brought to sharp focus. The usage of programming concepts enables the candidates to design & develop the systems requiring the multitasking capabilities to efficiently manage the system resources.

Module Objective
The Linux Application level Proficiency module attempts to serve multiple objectives:

- To enable participants to effectively apply the OS cocepts to design the multiprocessing & multi-threading systems requiring sycnhronization.
- To enable participants to design the systems requiring the communication across the processes or across the system, thereby sharing the system resources as required.

Target group:
Professionals/Students looking to design & develop the Linux based advance application under the multi-threaded & multi-processing environment.

Pre-requisite
Knowledge of C Programming with comfortability in Linux environment

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

Assessment
Assignment Based

Session 1: Introduction to Linux User Space & System Calls

- Linux System Components
- Need for System Calls
- System Calls and Library Functions

Exercises/Assignments

- System call tracing
- Locking a file and file regions

Session 2: 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

+ ***Session 3: 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
- Blocking a Signal
- Project Stage-1

+ Session 4 & 5: Inter Process Communication

- IPC Overview
- Pipe & Fifo
- Shared Memory
- Process Semaphores
- Project Stage-2 & Stage-3

Exercises/Assignments

- Write a program to demonstrate each of the above IPC mechanism
- Project Stage– 2 & 3

+ Session 6: Threads in Linux

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

- Thread Joining
- Thread Cancellation
- Project Stage - 4

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
- Project Stage - 4

+ Session 7: Synchronization in Linux
- Synchronization Overview
- Synchronization Mechanisms
- Project Stage-5

Exercises/Assignments
- Write a program to solve the consumer/producer problem
- Project Stage-5

+ Session 8: Linux Network Management
- Network Management Overview
- Introduction to Sockets
- Basic Socket Programming

Exercises/Assignments
- Write a programs to communicate between client & server
- Project Stage-6

Module-2
Linux Driver Level Proficiency (60 Day Challenge)

Module Description
The Linux Driver Level Proficiecy Module 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.

Module Objective
The Module 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 nterrupt management.

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

Pre-requisite
Knowledge of C Programming with comfortability in Linux environment

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 halve

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/Assignments

- 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

Exercises/Assignments

- Writing & Building the 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/Assignments

- 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/Assignments

- 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/Assignments

- 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/Assignments

- Write a driver to demonstrate the usage of Kernel timers

+ Session 7 & 8: 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

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

Session 9: Platform Drivers & Device Tree Binary

- Need for Platform Drivers
- Platform Drivers & Platform Device
- Registering a Platform Driver & Device
- Binding the Device with Driver
- Brief on Device Tree Binary (DTB)

Exercises/Assignments

- Writing a simple Platform Driver & Platform Device
- Modifying Device Tree to Swap the LED functionality

Module-3
Embedded Linux System Level Proficiency (60 Day Challenge)

Module Description
This Module intends to provide Complete end-to-end system level understanding. The module includes various enhancements to the assignments completed in the previous modules. Further to this, it includes the Mini Projects to develop the complete system level understanding.

Module Objective
Tembedded Linux System Level Proficiency attempts to multiple objectives:

- To enable participants take their skills to next level by enhancing the assignments
- To enable participant understand the complete end to end system (From user space to Kernel Space) with help of the Mini Projects

Life Time Access to Hackathons

The Hackathons are the live sessions that would fast-track your journey by solving assignments as a part of live sessions. It consists of 10 Missions with each mission focussing on assignments for the particular topic. Participants would be given the assignments which needs to be solved during the live session. Further to this, there would be mutliple Hackathons during the Year and participant can join any on Hackathons with Life Time access

Life Time Access to Inner Circle Masterminds

The idea behind these meetups is to provide the environment conducive to growth. The These are live weekly meetups scheduled for doubt clarification & Knowledge Sharing. This is the recharging event to help you stay focussed on the tasks. Also, these sessions are meant to share the success stories & milestones achieved by members.