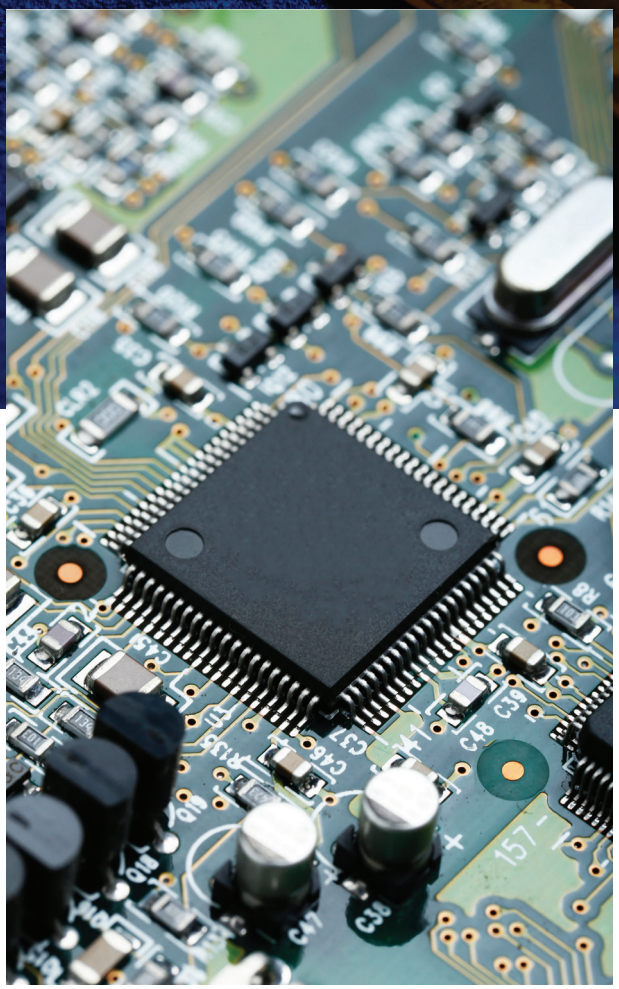
ESSENTIALS OF LINUX SYSTEM PROGRAMMING



Length: 3 Days

Hardware: STM32 MPUs microprocessor devices from STMicroelectronics

Software: Visual Studio Code

Training Board: STM32MP157F-DK2

ENROLL NOW

OVERVIEW

This intensive course covers critical facets of Embedded Linux system programming, including Linux system calls, file operations, signals, process lifecycle, program execution, cross-toolchains, cross-debugging in embedded systems, and crash troubleshooting. Participants will gain practical skills through lectures and hands-on workshops, fostering a profound understanding of Linux system programming.

COURSE OBJECTIVE

After completing this course, you will:

- Understand the fundamentals of Linux system calls and their significance.
- Gain proficiency in file operations, signal handling, and process lifecycle management.
- Acquire practical skills in executing programs and utilizing essential system programming techniques.

TARGET AUDIENCE

Students, software developers, system administrators, and individuals seeking a solid foundation in Linux system programming.

PREREQUISITES

Basic knowledge of the Linux operating system and C programming and Embedded Systems.

TEACHING METHODOLOGY

- **Instructional strategies** include lectures, discussions, demonstrations, and workshop-based learning which are used to convey information, facilitate learning, and promote understanding.
- **Technology integration** Hands-on workshops and exercises to reinforce theoretical concepts. Hands-on workshops and exercises to reinforce theoretical concepts.
- **Learning activities** Integration of Linux environment for practical exercises.

COURSE FRAMEWORK

DAY 1 : *Linux System Calls, File Operations, SSH Deployment and Introduction to libgpiod*

- Introduction to System Programming
- Overview of Linux System Architecture
- Linux System Calls:
  - Understanding System Calls and their importance.
  - Common System Calls in Linux.
- File Systems in Linux:
  - Inodes and directory structures
  - File permissions and attributes.
- Learn cross-toolchains for embedded systems, enabling remote deployment via SSH for efficient testing.
- **Lab exercise:** Hello World Deployment with SSH (Lab1 Manual) Task: Develop a "Hello World" program, deploy it to the target using SSH, and test it.

  Objective: Understand the process of deploying applications remotely using SSH.
- **Lab exercise:** Blinking LED using libgpiod (Lab2 Manual) Task: Modify the C program to blink an LED using libgpiod for basic GPIO operations.

  Objective: Gain hands-on experience with libgpiod and understand GPIO operations.

DAY 2 : *Signals and Interprocess Communication (IPC) Basics*

- Understanding Signals in Linux
- Signal Handling and Signal Sets
- IPC Mechanisms Overview:
  - Pipes
  - Message Queues
- **Lab exercise:** Signal-Driven Processes (Lab3 Manual)
  Task: Implement a program with multiple processes communicating through signals. Use signals for synchronization and process termination.

  Objective: Apply signal handling techniques in Interprocess communication scenarios.

DAY 3 : *Process Lifecycle, Execution, and Other Concepts - GDB*

- Process Creation and Termination
- Process States and Transitions
- Process Synchronization
- Error Handling and Debugging Techniques
- Memory Management in C and Linux
- Executing Programs in Linux:
  - The exec family of functions
  - Environment variables and command-line arguments
- **Lab exercise:** Process Execution and Coordination (Lab4 Manual)
  Task: Create a program that demonstrates process creation, execution, and coordination. Utilize the fork () system call and explore process synchronization.

  Objective: Understand the process lifecycle and coordination mechanisms between parent and child processes.
- **Lab exercise:** Introduction to GDB and Coredump Analysis (Lab5 Manual)
  Task: Design a lab to work with GDB, debug a program, and analyze crashes using coredump.

  Objective: Understand the use of GDB for debugging and analyze crashes by employing coredump analysis.