



COURSE BROCHURE

LINUX

DEVICE

DRIVERS



Hands on Training



1:1 Mentoring



**Placement
Assistance**



The Linux Device Drivers Course is designed to help learners understand how the Linux kernel communicates with hardware through device drivers. This course focuses on building strong fundamentals, practical kernel programming skills, and real-world understanding of Linux driver development—without diving too deep into complex kernel internals.

The curriculum starts with Linux kernel fundamentals and gradually moves into kernel modules and character device drivers. Learners gain hands-on experience in writing, loading, and debugging drivers while understanding key kernel concepts such as memory access, synchronization, timing mechanisms, and interrupts.

This course follows a practical, assignment-driven approach, enabling learners to build confidence in Linux kernel programming and prepare for interviews in embedded Linux and system software roles.

By the end of the course, learners will have the technical clarity and confidence required to understand, develop, and explain Linux device drivers.

WHAT IS A LINUX DEVICE DRIVER?

The Linux Device Driver Foundation Course is designed to help learners understand how the Linux kernel communicates with hardware through device drivers. This course focuses on building strong fundamentals, practical kernel programming skills, and real-world understanding of Linux driver development—without diving too deep into complex kernel internals.

The curriculum starts with Linux kernel fundamentals and gradually moves into kernel modules and character device drivers. Learners gain hands-on experience in writing, loading, and debugging drivers while understanding key kernel concepts such as memory access, synchronization, timing mechanisms, and interrupts.

This course follows a practical, assignment-driven approach, enabling learners to build confidence in Linux kernel programming and prepare for interviews in embedded Linux and system software roles.

By the end of the course, learners will have the technical clarity and confidence required to understand, develop, and explain Linux device drivers.

WHY SHOULD YOU CHOOSE RLS EMBEDDED SYSTEMS AS YOUR TRAINER?

INDUSTRY-ORIENTED TRAINING

The Linux device drivers course at RLS Embedded Systems is designed and delivered by experienced industry professionals. The curriculum is aligned with current job requirements and updated regularly to match emerging technologies and industry trends.

EFFECTIVE TRAINING METHODOLOGY AND INFRASTRUCTURE

RLS Embedded Systems follows a practical training approach with strong emphasis on hands-on learning. Learners spend significant time working on lab exercises and real hardware using industry-standard embedded boards, tools, and software environments.

PRACTICAL AND PROJECT-BASED LEARNING

The course includes projects and real-time embedded applications that help learners strengthen programming, debugging, and system-level understanding through practical implementation.

PLACEMENT AND CAREER GUIDANCE

RLS Embedded Systems provides placement assistance and career guidance to help learners prepare for interviews and entry-level embedded software roles through resume support and technical mentoring.

CONTINUOUS MENTORING AND SUPPORT

Personalized mentoring and technical support are provided throughout the course. Additional guidance, mock interviews, and communication skill sessions help learners become industry-ready professionals.

COURSE DETAILS

LINUX KERNEL FUNDAMENTALS

- Introduction to Linux kernel architecture
- Kernel space vs user space
- Responsibilities of the Linux kernel
- Role of device drivers in Linux
- Types of device drivers (overview)
- Understanding /proc and /sys file systems

KERNEL SOURCE & DEVELOPMENT SETUP

- Linux kernel source tree overview
- Kernel configuration basics
- Kernel build process (high-level understanding)
- Where and how drivers are organized in the kernel

KERNEL MODULE PROGRAMMING

- What is a kernel module?
- Module lifecycle (init and exit)
- Writing and compiling kernel modules
- Kernel logging and debugging techniques
- Module parameters
- Best practices in module development

CHARACTER DEVICE DRIVER BASICS

- Introduction to character device drivers
- Major and minor numbers
- Device registration and deregistration
- File operations: open, read, write, close
- Data transfer between user space and kernel space

ADVANCED CHARACTER DEVICE DRIVERS

- ioctl interface and usage
- Blocking and non-blocking I/O
- Asynchronous notification concepts
- Error handling in device drivers

CONCURRENCY & SYNCHRONIZATION

- Why concurrency issues occur in kernel
- Race conditions in device drivers
- Mutexes, spinlocks, and semaphores
- Protecting shared kernel resources

KERNEL TIMING & BACKGROUND EXECUTION

- Kernel delays and timing concepts
- Kernel timers
- Workqueues
- Introduction to kernel threads

INTERRUPTS – CONCEPTUAL & PRACTICAL

- Interrupt basics
- Interrupt context vs process context
- Registering interrupt handlers
- Top half and bottom half concepts
- Safe interrupt handling practices

DRIVER DESIGN & DEBUGGING PRACTICES

- Kernel coding style and guidelines
- Common driver development mistakes
- Debugging kernel crashes and issues
- Improving driver stability and readability

YOCTO FUNDAMENTALS & DRIVER INTEGRATION

- Introduction to Yocto Project
- Embedded Linux build system overview
- Yocto architecture (high-level)
- Understanding layers and recipes
- BitBake basics
- Building a minimal Linux image
- Adding custom packages
- Integrating a kernel module into Yocto build
- Understanding root filesystem generation
- Build a simple image

MINI PROJECT & INTERVIEW PREPARATION

- Design and implementation of a complete character driver
- Explanation of driver flow end-to-end
- Common Linux device driver interview questions
- Resume and project discussion guidance

C PROGRAMMING BASICS

- Introduction to C
- Structure of a C program
- Data types
- Variables and constants
- Input / Output
- Operators
- Control statements
- Looping statements
- Arrays
- Strings
- Functions
- Basic pointers
- Type casting
- Preprocessor directives

COURSE DETAILS

C PROGRAMMING ADVANCED

- Advanced pointers
- Dynamic memory allocation
- Structures
- Unions
- Enumerations
- File handling
- Storage classes
- Command-line arguments
- Bitwise operators
- Recursion
- Data Structures
- Error handling
- Multi-file programming
- Memory management

WHAT YOU WILL LEARN

By the end of this course, learners will be able to:

- Understand Linux kernel and driver architecture
- Write and manage kernel modules
- Develop basic but real Linux device drivers
- Handle synchronization and timing in kernel space
- Confidently explain Linux device drivers in interviews



+91 73822 73072



<https://www.linkedin.com/company/rls-embedded-systems/>



<https://rlseembeddedsystems.in/>



rlseembeddedsystems@gmail.com



+91 73822 73072