The Linux Device Drivers Development course provides engineers with a fast, cost-effective way to acquire the knowledge necessary to build, load, and debug Linux device drivers in a cross-development environment.

After this course, students will be able to:

- Understand the Linux boot process and role of the bootloader and BSP
- Prepare to develop Linux I/O drivers, serial drivers, and network drivers
- Debug Linux device drivers

Products Supported

- General Purpose Platform, Linux Edition 1.4

Who Should Attend

- New and experienced device driver writers
- Application Programmers interested in seeing what goes on at the hardware level
- Senior engineers who want to decide on a final production image of their product
- Experienced developers interested in the interfaces between drivers and Linux

Prerequisite Skills

- Functional knowledge of Linux
- One year programming, including experience with structures, pointers, pointers to structures, typedefs, macros, and bitwise operators
- Experience using binary and mutual exclusion, semaphores, message queues, pipes, and managing multiple tasks
- Experience debugging target code
- Ability to use and modify makefiles to build executable images

Prerequisite Courses

- Introduction to Linux
- Introduction to Real-Time Programming

Related Courses

- RTOS to Linux Migration Basics
- Workbench for Linux
- General Purpose Platform, Linux Edition (Workbench/Linux)
- Linux Board Bring-Up and Board Support Package Development

Course Format

- Our four-day instructor-led courses consist of lectures and lab sessions
- Students gain hands-on experience and receive personal guidance from expert Wind River instructors
- Students examine details of the Workbench environment, focusing on the most commonly used areas
- Specific questions are addressed
- Lab sessions allow hands-on application of course concepts

Global Reach of Wind River Education Services

- 4,000 students per year
- 400 classes delivered per year
- 34 instructors worldwide
- Access to 250 subject matter experts
- 24 training centers worldwide
- 20+ years of device software experience

On-site courses are conducted at your location and include the use of preconfigured laptops and target boards, plus shipping and travel costs. Visit education.windriver.com for registration and schedule information.
Topics

- Review of Linux boot process and the role of the boot loader
- Building and accessing modules
- Linux Device driver debugging techniques
- Using Timer services and memory management techniques for developing device drivers
- Overview of interrupt handling when implementing a Linux device driver
- Steps necessary to develop character and network device drivers
- Techniques for Debugging Device Drivers
- Overview of hot-plug support, power management and use of sysfs

Agenda

Day 1
- Review Linux Architecture
  - Linux Architecture
  - Linux Device Driver Development Model
- Booting Linux
  - Linux Boot/Initialization Process
- Introduction to Device Drivers
  - Device Types
  - Tracking a Driver Call in Linux
  - Dynamic vs Static Drivers
- Kernel and Modules
  - Kernel Configuration Tools
  - Building and Configuring the Kernel
  - Loading the Kernel
  - Adding a Driver
  - Building and Installing Modules
  - Loading and Unloading Modules
  - Passing Parameters to Modules
  - Kernel Versioning
- Memory Management
  - Linux Memory Model
  - Kernel Address Space
  - Caching
  - Allocating Memory in User and Kernel Space
  - Mapping a Device

Day 2
- Character Drivers
  - What is a character device driver?
  - Device Special Files
  - Driver Registration
  - File Operations
  - IOCTL Operation
- User Space To Kernel Space Interfaces
  - Data Flow
  - Using /procfs file system
  - Using relayFS
- Blocking I/O
  - Blocking vs Non Blocking I/O
  - Wait Queues and Sleep
  - Race Conditions and Locks
  - Using Barriers
  - Semaphore Usage
  - Poll and Select

Day 3
- Kernel Space Timer Services
  - Timer Interrupts - HZ and jiffies
  - Delaying Execution
  - Using Task Queues
  - Kernel Space Timers
- Interrupt Handling
  - Linux and Interrupts
  - IRQ Action Table
  - Register, Enabling and Disabling Interrupts
  - Minimizing Interrupt Latency
- Network Device Drivers
  - Linux Network Driver Overview
  - Network Device Structure
  - Network Driver Functions
  - Network Stack Functions
  - Socket Buffers
  - Transmit and Receive Logic Flow
  - Interrupt Handling
  - MII Transceivers (PHY)

Day 4
- Block Device Drivers
  - Role of Block Drivers
  - Block Driver API
- LSP and Driver Debugging Techniques
  - printk
  - Debugging using /proc and ioctl
  - Debugging w/ LEDs
  - Watching System Calls via strace
  - Linux Trace Toolkit (LTT)
  - Kernel Panic
  - Debugging System Faults
  - Using KGBB
  - Using Hardware JTAG/BDM Tools
- Extension to the Linux Driver Model
  - Hotplug
  - Power Management Review
  - The 2.6 Linux Driver Model
  - SysFS overview