COURSE DESCRIPTION

The Linux Device Drivers course provides engineers with a fast, cost-effective way to acquire the skills necessary to develop, deploy, and debug their own customized Linux device drivers in the Wind River® Linux environment.

After this course, participants will be able to perform the following:

- Develop and manipulate Linux kernel modules
- Develop Linux device drivers for the various types of devices supported in Linux
- Describe the workings of the different kernel subsystems and how they impact the structure of a device driver
- Debug Linux device drivers

PRODUCTS SUPPORTED

- Wind River Linux 8.0
- The following targets are available:
  - QEMU simulated target (Intel® x86-64)

COURSE FORMAT

- This three-day expert-led course consists of lectures and lab sessions.
- Students gain hands-on experience and receive personal guidance from expert Wind River instructors.
- Specific questions are addressed.

AUDIENCE

- Anyone new to device driver development in Linux
- Linux application developers who are looking to gain more insight into how the Linux kernel works
- Developers interested in the interface between the Linux kernel and device drivers

PREREQUISITE SKILLS

- Familiarity with makefiles and GNU toolchain
- Understanding of various methods used to deploy and debug Linux-based applications in a cross-development environment
- C or C++ programming experience on Linux/UNIX

PREREQUISITE COURSES

Introduction to Linux

RELATED COURSES

- Wind River Linux and Workbench Essentials
- Wind River Linux BSP Development

SYLLABUS

Day 1

INTRODUCTION TO LINUX DEVICE DRIVERS

- Linux architecture overview
- Linux device driver overview
- Device driver types
- Linux device model
- In-tree vs. out-of-tree driver development
LINUX DEVICE DRIVERS

LINUX KERNEL SOURCE CODE
- Source code organization
- The kernel configurator
- The kernel build system
- Working with kernel patches
- LAB: Getting Started with the Wind River Linux Lab Environment

INTRODUCTION TO LINUX KERNEL MODULES
- Overview
- Anatomy of a kernel module
- Module licensing
- Building modules
- Installing modules
- Managing modules
- Module parameters
- LAB: Developing Kernel Modules

CHARACTER DEVICE DRIVERS
- Driver lifecycle
- Major and minor numbers
- Character driver entry points
- Blocking operations
- Controlling a device
- Querying read/write ability
- Restricting operations
- LAB: Developing a Character Device Driver

Day 2
MANAGING MEMORY IN THE LINUX DEVICE DRIVERS
- How Linux manages memory
- Allocating memory with kmalloc()
- Page-based memory allocation
- Manipulating memory
- Memory-mapped I/O
- Accessing user space memory
- Implementing the mmap operation
- LAB: Managing Memory in Kernel Code

CONCURRENCY IN LINUX DEVICE DRIVERS
- Concurrency
- Race conditions
- Locking primitives
- Deadlock
- Atomic variables
- LAB: Managing Concurrency in Kernel Code

MANAGING TIME IN LINUX DEVICE DRIVERS
- Measuring time in the kernel
- Delaying execution
- Deferring execution
- LAB: Managing Execution of Driver Code

HANDLING INTERRUPTS IN LINUX DEVICE DRIVERS
- How interrupts work
- IRQs
- Top half processing
- Bottom half processing
- LAB: Implementing Interrupt Handlers

DEBUGGING LINUX DEVICE DRIVERS
- Debugging by printing
- Debugging by querying
- Debugging by observation
- Using a kernel debugger
- LAB: Implementing Debugging Strategies in Kernel Code
- LAB: Configuring KGDB
- LAB: Kernel Debugging with GDB

Day 3
LINUX PCI DEVICE DRIVERS
- PCI configuration space
- Identifying devices
- Matching devices and drivers
- Driver registration
- Probe() function
- Memory and I/O regions
- DMA
- Remove() function
- LAB: Driving Devices over PCI

Linux USB Device Drivers
- USB architecture
- Matching devices and drivers
• Driver registration
• Probe function
• Communicating with USBs
• LAB: Driving Devices over USB

Block Device Drivers
• Driver lifecycle
• Major and minor numbers
• Block driver entry points
• Processing requests
• Controlling a device
• LAB: Developing a Block Device Driver

Linux Network Device Drivers
• Overview of network devices
• Driver registration
• Network driver entry points
• Controlling interfaces
• Packet transmission
• Interrupt-driven packet reception
• NAPI drivers
• LAB: Developing a Network Device Driver

GLOBAL REACH OF WIND RIVER EDUCATION SERVICES
With more than 30 years of device software experience, Wind River provides education services in every region of the world. Our private classes can be tailored to your needs by adding or removing topics from multiple courses. If you have more specific project challenges, Wind River Mentoring provides coaching by experienced engineers to help you integrate Wind River solutions into your environment. And when you’re too busy to attend a whole class, our On-Demand Learning options provide around-the-clock access to advanced and specialized topics. All of our education services are led by expert engineers who are closely connected to the Wind River technical community for access to specific expertise.