

WIND RIVER EDUCATION SERVICES

TRAINING

COURSE CATALOG



OCTOBER 2026

TRAINING COURSE CATALOG

INSTRUCTOR-LED COURSES

VXWORKS RECOMMENDED LEARNING PATH	3
HELIX PLATFORM CERT EDITION RECOMMENDED LEARNING PATH	19
INTEL SIMICS SIMULATOR RECOMMENDED LEARNING PATH	24
WIND RIVER CLOUD PLATFORM RECOMMENDED LEARNING PATH	36
WIND RIVER LINUX RECOMMENDED LEARNING PATH	43
ELXR	56
ZEPHYR	58

ELEARNING SUBSCRIPTIONS

SUBSCRIPTION PACKAGES	60
-----------------------------	----

RECOMMENDED LEARNING PATH

VxWorks is the industry-leading real-time operating system (RTOS) designed for mission-critical embedded systems.

Learn to build, run, and secure high-performance, mission-critical embedded systems that require real-time determinism, low latency, and strict safety certifications.

01. VXWORKS AND WORKBENCH ESSENTIALS

- 4 days
- Instructor-led / eLearning
- Introductory course that teaches how to develop real-time applications with VxWorks and Workbench.

02. DEVELOPING VXWORKS BSPS AND DEVICE DRIVERS

- 3 days
- Instructor-led / eLearning
- Develop and integrate VxWorks BSPs, kernel components, and device drivers to enable, configure, and support custom hardware platforms.

03. DEVELOPING SYMMETRIC MULTIPROCESSING APPLICATION IN VXWORKS

- 2 days
- Instructor-led / eLearning
- For application developers
- Migrate applications from the uniprocessor (UP) to the SMP environment and optimize application performance using parallel design techniques.

04. DEBUGGING AND SYSTEM DIAGNOSTICS WITH VXWORKS

- 2 days
- Instructor-led
- Investigate runtime issues, crashes, exceptions, memory corruption, synchronization problems, and performance bottlenecks in embedded real-time systems.

To book a training or learn more, contact
training@windriver.com

VXWORKS AND WORKBENCH ESSENTIALS

The course provides a practical introduction to developing real-time applications with VxWorks and Wind River Workbench. Participants learn to configure and manage projects, develop and debug applications, work with tasks, events, semaphores, inter-task communication, memory, interrupts, and real-time processes, and use Workbench tools to analyze and troubleshoot VxWorks systems.

ABOUT THIS COURSE

After this course, you will be able to:

- Design and develop real-time applications in kernel and user modes
- Debug, build, and test real-time applications in a target host development environment with Workbench and VxWorks

COURSE TYPE & DURATION

- Instructor-led training
- Four days

SUPPORTED VERSION

- VxWorks 22.06 and later
- Wind River Workbench 4
- Earlier product releases (topics may vary)

PREREQUISITE

- One year of C programming
- Basic understanding of operating systems and debugging techniques

To book training or learn more, contact
training@windriver.com

DAY ONE

01. GETTING STARTED WITH VXWORKS

- Product overview
- Workbench 4 features
- VxWorks features

02. VXWORKS TARGETS AND CONNECTIONS

- Hardware target configuration
- Booting the hardware target
- Workbench tools architecture
- Configuring and connecting the TCF server
- QEMU - Processor Simulation
- Wind River Simics - a true hardware simulation

LAB: Getting Started with VxWorks

03. MANAGING PROJECTS IN WORKBENCH

- VxWorks installation directory structure
- VxWorks layers and package management
- VxWorks source build projects
- VxWorks image projects
- Configuring the kernel
- The vxprj utility
- ROMFS
- Configuring application projects
- Importing and exporting
- Building projects

LAB: Managing VxWorks Projects

04. VXWORKS KERNEL SHELL

- Kernel shell configuration
- Kernel shell commands and help
- Kernel shell usage
- Command-line history and editing
- Object module loader

LAB: Using the VxWorks Kernel Shell

VXWORKS AND WORKBENCH ESSENTIALS (CONT.)

DAY TWO

05. DEBUGGING IN WORKBENCH

- Debugger overview
- Application mode and stop mode debugger
- Setting breakpoints
- Downloading code
- Attaching to running tasks
- Attaching to system

LAB: Debugging with Workbench

LAB: Debugging VxWorks in Stop Mode

06. REAL-TIME MULTITASKING

- Introduction to real-time requirements
- Task states in VxWorks
- Context switching
- Spawning new tasks
- Task control routines
- Task information
- Alternative POSIX support
- System tasks

LAB: Working with VxWorks Tasks

07. VXWORKS EVENTS

- VxWorks events overview
- Task event register
- Event handling in VxWorks
- Receiving events
- Sending events
- Other eventLib routines
- Usage caveats

LAB: Working with VxWorks Events

DAY THREE

08. SEMAPHORES

- Semaphore types
- Synchronization issues
- Binary and counting semaphores
- VxWorks events and semaphores
- Mutual exclusion issues
- Mutex semaphores
- Deletion and inversion safe mutex semaphores
- Other preemption locks for tasks and ISRs

LAB: Using Semaphores

09. INTER-TASK COMMUNICATION

- Overview of communication methods
- Shared memory and data structures
- Message queues
- Creating, sending, and receiving messages
- Message queue events
- Pipes in VxWorks

LAB: Communicating between VxWorks Tasks

10. REAL-TIME PROCESSES

- MMU use
- Memory allocation and tasks
- VxWorks component support for RTPs
- RTP execution and termination
- Debugging RTPs
- Shared library usage
- Shared data usage
- Public and private objects
- RTP application security
- Design considerations

LAB: Working with Real-Time Processes

VXWORKS AND WORKBENCH ESSENTIALS (CONT.)

DAY FOUR

11. VXWORKS MEMORY

- Physical memory layout
- Virtual memory layout
- Heap memory allocation
- Virtual memory allocation
- Examining memory

LAB: Managing VxWorks Memory

14. SYSTEM VIEWER

- System Viewer overview
- Configuring System Viewer
- Collecting and displaying event data
- User-defined events
- Additional analysis views

LAB: Using System Viewer

12. EXCEPTIONS, INTERRUPTS, AND TIMERS

- Exception handling and signals
- Installing signals to handle exceptions
- Interrupt service routine basics
- Interrupt handling example
- ISR guidelines
- Timing and the system clock
- Watchdog timers
- Polling
- The auxiliary clock

LAB: Using VxWorks Exceptions, Interrupts, and Timers

15. ANALYSIS TOOLS

- Overview
- System browser
- CPU profiler
- Memory analyzer

LAB: Using VxWorks Analysis Tools

13. ERROR DETECTION AND REPORTING

- Error reporting features
- Error reporting capabilities
- Error reporting framework
- Persistent memory manager
- Configuring ED&R in the kernel
- Error records and log
- Extending ED&R recording

LAB: Using VxWorks Error Detection and Reporting

16. (VXWORK 7 CERT ONLY) VXWORKS 7 CERTIFICATION

- Introduction to DO-178
- Background and definitions
- DO-178 and software safety
- DO-178 processes and objectives
- Software verification
- VxWorks 7 Cert Edition considerations

LAB: Deploying a VxWorks Cert System

To book training or learn more, contact
training@windriver.com

VXWORKS

DEVELOPING VXWORKS BOARD SUPPORT PACKAGES AND DEVICE DRIVERS

The VxWorks® Board Support Packages and Device Drivers course provides engineers with a fast, cost-effective way to acquire the knowledge necessary to port VxWorks to customer hardware.

ABOUT THIS COURSE

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

- Create a flattened device tree file
- Boot VxWorks with U-Boot boot loader
- Create and make add-on VxWorks kernel components available in the kernel configuration editor
- Implement a VxWorks I/O system driver and access I/O devices from both user and kernel space
- Implement and integrate a timer driver
- Implement PCI Controller Driver Routines
- Initialize, configure, and integrate END drivers

COURSE TYPE & DURATION

- Instructor-led training
- Three days

SUPPORTED VERSION

- VxWorks 22.09
- Wind River Workbench 4

PREREQUISITE

- C programming, including experience with structures, pointers, pointers to structures, typedefs, macros, and bitwise operators
- Functional knowledge of Linux or Windows host operating system.
- Experience using the Workbench debugger to debug target code

To book training or learn more, contact
training@windriver.com

DAY ONE

01. INTRODUCTION TO VXWORKS BSPS

- BSP overview
- BSP high level architecture

LAB: Getting Started with VxWorks BSPs and Device Drivers

02. FLATTENED DEVICE TREE

- Flattened device tree overview
- Flattened device tree syntax
- Flattened device tree initialization
- Device discovery

LAB: Porting a BSP to Support a Board Variant

03. BOOT LOADERS FOR VXWORKS

- VxWorks Boot Loaders Overview
- U-Boot for PowerPC, ARM and RISC-V
- Boot Loaders for Intel
- VxWorks Boot Application

LAB: Using Boot Loaders for VxWorks

04. VXWORKS BSP INFRASTRUCTURE

- BSP infrastructure overview
- BSP board layer
- Processor support library (PSL)
- Board subsystem
- Quick steps for BSP development

05. KERNEL INITIALIZATION SEQUENCE

- Pre-kernel initialization overview
- Pre-kernel early debug options
- Kernel initialization overview
- kernelInit() function
- usrRoot() function
- Hardware initialization

LAB: Browsing VxWorks Initialization Code

DEVELOPING VXWORKS BOARD SUPPORT PACKAGES AND DEVICE DRIVERS (CONT.)

DAY TWO

06. COMPONENT DESCRIPTION LANGUAGE

- Language and file essentials
 - Adding components examples
- LAB: Adding VxWorks Components**

07. VXWORKS I/O INTERFACE

- VxWorks I/O system
 - I/O system structure
 - User space and I/O facilities
- LAB: Creating a VxWorks I/O Device Driver**

08. VXBUS AND DEVICE DRIVERS FUNDAMENTALS

- VxBus architecture
 - The VXB_DRV structure
 - Device driver methods
 - VxBus initialization sequence
 - Device discovery
 - Resource management
 - Dealing with device interdependency
 - Driver debug aids
 - Using I/O System with VxBus-Compatible Drivers
- LAB: Integrating a VxWorks I/O Device Driver with VxBus**

09. SERIAL DRIVERS

- Serial drivers overview
 - High level serial driver architecture
 - Serial driver data structures
 - Header files
 - Serial driver routines
 - VxBus driver methods
 - Configuring and initializing serial drivers
 - Implementing driver routines
- LAB: Creating a Serial Driver**

DAY THREE

10. TIMER DRIVERS

- Timer Drivers overview
 - VxBus driver methods
 - Timer driver routines
 - Header files
 - Configuring and initializing timer drivers
 - Integrating timer drivers
 - Implementing driver routines
- LAB: Examining Timer Drivers Implementations**

11. PCI DRIVERS

- PCI controller driver description
- PCI bridge driver description
- Data structures
- Header files
- Configuring and initializing PCI controller and bridge drivers
- Implementing PCI controller driver routines

12. VXWORKS NETWORK DRIVERS

- Network drivers overview
 - VxBus driver methods
 - END
 - Network framing
 - Receiving packets
 - Sending packets
 - Contexts of execution
 - MDIO
- LAB: Adding Polling Mode to the VxBus Network Driver**

To book training or learn more, contact
training@windriver.com

DEVELOPING SYMMETRIC MULTIPROCESSING APPLICATIONS IN VXWORKS

The VxWorks® Symmetric Multiprocessing course presents several methods to optimize application performance using parallel design techniques. Issues encountered in migrating applications to parallel design are discussed in detail. Specifics of creating and migrating applications to Wind River VxWorks symmetric multiprocessing (SMP) technology are also addressed.

ABOUT THIS COURSE

After this course, you will be able to:

- Migrate applications from the uniprocessor (UP) to the SMP environment.
- Analyze concurrency using debug tools.
- Perform run-time analysis of applications in the UP vs. SMP environment.

COURSE TYPE & DURATION

- Instructor-led training
- Two days

SUPPORTED VERSION

- VxWorks 7
- Wind River Workbench 4
- Wind River Simics 4.8

PREREQUISITE

- One year of C programming
- Basic understanding of operating systems and debugging techniques

To book training or learn more, contact
training@windriver.com

DAY ONE

01. INTRODUCTION TO SMP

- History of multi-core and multiprocessing
- Overview of SMP
- Other multi-core configurations

LAB: Getting Started with SMP

02. VXWORKS SMP ARCHITECTURE

- Overview of SMP architecture
- Cache and cache coherency
- The sequential memory model
- Mutual exclusion
- Spinlocks and deadlocks
- Memory barriers
- Development challenges

03. VXWORKS SMP CONFIGURATION

- VxWorks SMP components
- Software and hardware requirements
- BSP

LAB: Configuring VxWorks for SMP

DEVELOPING SYMMETRIC MULTIPROCESSING APPLICATIONS IN VXWORKS (CONT.)

DAY TWO

04. VXWORKS SMP PROGRAMMING

- Spinlocks
- Read/write semaphores
- Task CPU affinity
- Interrupt CPU affinity
- Atomic operations
- Memory barriers
- POSIX thread barriers
- CPU information and management
- Uniprocessor incompatibilities

LAB: Synchronizing Data in an SMP Environment

LAB: Synchronizing Data with Core Affinity and Core Reservation

LAB: Synchronizing with Message Queues

LAB: Synchronizing with Semaphores

05. DEBUGGING AND ANALYSIS TOOLS

- Multi-core debugging overview
- Breakpoints
- Multiple context debugging
- System viewer and analysis tools
- Kernel shell debugging

LAB: Working with Workbench Debugger

06. INTRODUCTION TO SOFTWARE PARALLELISM

- SMP limits
- Parallel software design
- Implementing a parallel programming model
- Threading
- Parallelism examples
- Portable parallel programming APIs

07. UNIPROCESSOR TO SMP MIGRATION

- Migration guidelines
- The three-step migration plan
- Step 1: Update to current VxWorks version
- Step 2: Migrate to SMP API
- Step 3: Optimize for SMP

LAB: Comparing the Performance of Single Core and Multi-core Processors

08. VXWORKS SMP SCHEDULER

- Overview
- VxWorks UP scheduler
- VxWorks SMP
- VxWorks SMP scheduler

To book training or learn more, contact
training@windriver.com

DEBUGGING AND SYSTEM DIAGNOSTICS WITH VXWORKS

The VxWorks® Debugging and System Diagnostics equips experienced VxWorks developers with practical debugging and diagnostic skills for investigating runtime issues, crashes, exceptions, memory corruption, synchronization problems, and performance bottlenecks in embedded real-time systems. Students learn to leverage VxWorks debugging tools including Dynamic printf, kprintf, Core Dumps, WRDBG, ED&R, and System Viewer to efficiently diagnose and resolve software failures in both development and deployed environments.

ABOUT THIS COURSE

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

- Use VxWorks debugging and diagnostic tools effectively.
- Apply runtime and post-mortem debugging techniques.
- Diagnose crashes, exceptions, deadlocks, and performance issues.
- Investigate memory corruption and synchronization problems.
- Select the appropriate debugging approach for different scenarios

COURSE TYPE & DURATION

- Instructor-led training
- Two days

SUPPORTED VERSION

- VxWorks 26.03

PREREQUISITE

- Completion of VxWorks Essentials or equivalent experience
- Approximately one or more years of hands-on VxWorks development experience
- Working knowledge of: C/C++ programming
- VxWorks RTPs, tasks, interrupts, and synchronization primitives
- Basic debugging concepts (stack traces, symbols, system logs)
- General understanding of real-time and embedded system concepts

DAY ONE

01. COURSE INTRODUCTION

- Course Introduction

Lab: Getting Started with the Lab Environment

02. USING DYNAMIC PRINTF FOR RUN-TIME DEBUGGING

- dprintf Overview
- Using dprintf API from the kernel shell
- Using dprintf from Workbench debug shell
- Using dprintf from Workbench debugger GUI
- Using dprintf with RTPs
- Hardware-assisted dprintf

Lab: Debugging with dprintf

03. USING KPRINTF FOR LOW-LEVEL DEBUGGING

- Introducing kprintf()
- Enabling and using kprintf() and kputs()
- Using kprintf() with no serial console

Lab: Generating Diagnostic Output with kprintf

04. ANALYZING SYSTEM FAILURES WITH CORE DUMPS

- Purpose of core dumps
- Analyzing kernel exceptions with core dumps
- Analyzing RTP exceptions with core dumps

Lab: Debugging from a Core Dump

To subscribe or learn more, contact
training@windriver.com

DEBUGGING AND SYSTEM DIAGNOSTICS WITH VXWORKS (CONT.)

DAY TWO

05. DEBUGGING APPLICATION USING THE WRDBG SHELL

- Introduction to WRDBG
- Getting started with WRDBG
- Essential debugging operations

Lab: Using the WRDBG Shell

06. INVESTIGATING EXCEPTIONS WITH ED&R

- Introduction to ED&R
- Configuring ED&R
- Understanding exceptions
- ED&R error records

Lab: Debugging an Application Exception

- Memory corruption concepts

Lab: Debugging a Memory Scribbler

07. ANALYZING RUN-TIME BEHAVIOR WITH SYSTEM VIEWER

- Introduction to system viewer
- Using system viewer for application investigation

Lab: Investigating an Application Issue Using System Viewer

- Programmatic trace collection

Lab: Using System Viewer Programmatically

- Post-mortem analysis

Lab: Using System Viewer in Post-Mortem Mode

To book training or learn more, contact

training@windriver.com

VXWORKS

MIGRATING QNX APPLICATIONS TO VXWORKS

The QNX to VxWorks Migration course enables students to port RTOS applications from QNX 6.x to VxWorks 7.x. The class will explore the differences between the two products and how to perform certain tasks in VxWorks.

ABOUT THIS COURSE

After this course, you will be able to:

- Identify the differences between QNX and VxWorks.
- Port QNX applications to VxWorks.
- Develop new VxWorks applications.

COURSE TYPE & DURATION

- Instructor-led training
- Two days

SUPPORTED VERSION

- VxWorks

PREREQUISITE

- Experience programming with C/C++.
- Experience programming applications with QNX.
- Experience with embedded systems development is beneficial but not mandatory.

To book training or learn more, contact
training@windriver.com

DAY ONE

01. COURSE INTRODUCTION

- Identify course objectives
- Student introductions

02. INTRODUCTION TO VXWORKS

- Product overview
- Workbench 4 features
- VxWorks features

03. UNDERSTAND THE DIFFERENT ARCHITECTURES

- Comparison between QNX and VxWorks
- Architecture and kernel differences
- Using multiple ECUs
- Development tools and IDEs
- Supported languages and APIs

Lab Exercise: Getting Started with VxWorks

Lab Exercise: Managing VxWorks Projects

04. MIGRATION STRATEGY

- Incremental vs big-bang approach
- Review of existing QNX applications
- Timers
- Code porting

Lab Exercise: Porting Applications from QNX Neutrino to Wind River VxWorks

MIGRATING QNX APPLICATIONS TO VXWORKS (CONT.)

DAY TWO

05. TESTING AND VALIDATION

- Core dumps
- Debugger overview
- Application mode and stop mode debugger
- Setting breakpoints
- Downloading code
- Attaching to running tasks
- Attaching to a system

Lab Exercise: Debugging with Workbench

Lab Exercise: Debugging VxWorks in Stop Mode

06. PERFORMANCE TUNING

- Establish performance benchmarks
- Profile the application
- Optimize application code
- Kernel configuration and system parameters
- Continuous testing and monitoring

**Lab Exercise: Performance Tuning a Temperature
Sensor Application**

To book training or learn more, contact
training@windriver.com

VXWORKS

MIGRATING FROM VXWORKS 6.9 TO VXWORKS 7

Explore the key features of Workbench 4 and VxWorks 7, including changes to project management, debugging, analysis tools, memory management, BSPs, and drivers. This course highlights the important differences from VxWorks 6.9 and prepares developers to work effectively in the VxWorks 7 environment..

ABOUT THIS COURSE

After this course, you will be able to:

- Identify the available features of Workbench 4 and VxWorks 7
- Identify the changes in VxWorks 7 project management
- Differentiate between application mode debugging and stop mode debugging
- Configure and use VxWorks 7 analysis tools
- Understand VxWorks 7 memory layouts and create custom memory pools
- Explain the difference between a VxWorks 6.9 BSP and a VxWorks 7 BSP, as well as the difference between drivers

COURSE TYPE & DURATION

- Instructor-led training
- Two days

SUPPORTED VERSION

- VxWorks 7
- Wind River Workbench 4

PREREQUISITE

- Proficiency with VxWorks 6.x

To book training or learn more, contact
training@windriver.com

DAY ONE

01. GETTING STARTED WITH VXWORKS 7

- Product overview
- Workbench 4 features
- VxWorks 7 features

02. VXWORKS TARGETS AND CONNECTIONS

- Hardware target configuration
- Booting the hardware target
- Workbench tools architecture
- Configuring and connecting the TCF server
- VxWorks Simulator – a high-level VxWorks simulator
- Simics – a true hardware simulation

03. MANAGING PROJECTS IN WORKBENCH

- Introduction to projects and workspaces
- VxWorks 7 installation and directory structure
- VxWorks layers and package management
- The wrtool utility
- VxWorks source build projects
- VxWorks image projects
- Configuring the kernel
- ROMFS
- Configuring application projects
- Importing and exporting
- Building projects

LAB: Managing VxWorks 7 Image Projects

MIGRATING FROM VXWORKS 6.9 TO VXWORKS 7 (CONT.)

04. DEBUGGING VXWORKS IN WORKBENCH

- Debugger overview
- Application mode and stop mode
- Setting breakpoints
- Downloading code
- Attaching to running tasks
- Attaching to a system

LAB: Debugging VxWorks 7 in Stop Mode

07. VXWORKS MEMORY

- Introduction
- Physical memory layout
- Virtual memory layout
- Heap memory allocation
- Virtual memory allocation
- Examining memory

LAB: Managing VxWorks Memory

DAY TWO

05. SYSTEM VIEWER

- System Viewer overview
- Configuring System Viewer
- Collecting and displaying event data
- User-defined events
- Additional analysis views

06. ANALYSIS TOOLS

- Overview
- System Browser
- CPU Profiler
- Memory Analyzer

LAB: Using VxWorks Analysis Tools

08. MIGRATING BSPS AND DEVICE DRIVERS TO VXWORKS 7

- VxWorks 6.9 vs. VxWorks 7 BSPs
- Migrating to a VxWorks 7 BSP
- Reusing a VxWorks 6.9 BSP in pre-SR0600 VxWorks 7
- VxWorks 6.9 vs. VxWorks 7 device drivers
- Probing and configuring devices
- Methods, device resources, and driver structure
- Migrating a legacy VxBus device driver

LAB: Migrating Device Drivers to VxWorks 7

To book training or learn more, contact
training@windriver.com

WIND RIVER DIAB

WIND RIVER DIAB COMPILER ESSENTIALS

Learn how to improve application performance and reduce memory footprint while reducing the compliance burden for developing safety-certified products

ABOUT THIS COURSE

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

- Create programs in an embedded environment
- Install and configure Diab Compiler
- Configure a target and driver and compiler options
- Use the Wind River Compiler Linker
- Implement safety and security
- Understand Diab Compiler error messages

COURSE TYPE & DURATION

- Instructor-led training
- Two days

SUPPORTED VERSION

- Wind River Diab Compiler

PREREQUISITE

- Knowledge of the Linux operation system and its basic administrative commands
- A basic understanding of C or C++ programming

To book training or learn more, contact
training@windriver.com

DAY ONE

01. INTRODUCTION TO EMBEDDED SOFTWARE

- Real-time architectures
- Real-time operating systems
- AUTOSAR
- Partitioning software into tasks
- Scheduling tasks

Lab Exercise: Identifying Component Tasks and Scheduling

02. DIAB COMPILER

- What is a software compiler?
- Components of a compiler
- Diab compiler 7.x

03. DIAB COMPILER INSTALLATION

- Prerequisites
- Installing Diab Compiler
- Verifying the installation

Lab Exercise: Verifying the Diab Compiler Installation

04. USING DIAB COMPILER

- Diab Compiler tools suite
- Configuring a target
- Configuring driver options
- Configuring compiler options

Lab Exercise: Using the QEMU Emulator

Lab Exercise: Optimizing Diab Compiler

WIND RIVER DIAB COMPILER ESSENTIALS (CONT.)

DAY TWO

05. USING DIAB COMPILER

- Diab Compiler tools suite
 - Configuring a target
 - Configuring driver options
 - Configuring compiler options
- Lab Exercise: Using the QEMU Emulator**
Lab Exercise: Optimizing Diab Compiler

06. DIAB LINKER

- The role of Diab linker
 - The Diab linker process
 - Linking in embedded systems
 - The dld command
- Lab Exercise: Linking with Diab Linker**

07. DIAB COMPILER SAFETY AND SECURITY

- Stack smashing protection
 - Levels of protection
 - Responsibilities
 - Detect memory leaks
- Lab Exercise: Implementing Stack Smashing Protection**

08. DIAB COMPILER ERROR MESSAGES

- Diab Compiler error messages
 - Understanding error messages
 - Compiling for optimization and debugging
- Lab Exercise: Diab Compiler Error Messages**

To book training or learn more, contact
training@windriver.com

HELIX PLATFORM

RECOMMENDED LEARNING PATH

The Wind River Helix Virtualization Platform (HVP) is a Type 1 safety-certifiable hypervisor that consolidates multiple operating systems and mixed-criticality workloads onto a single multi-core processor. Learn to run safety-critical and general-purpose applications side-by-side securely across aerospace, defense, automotive, industrial, and medical edge systems.

01. HELIX PLATFORM CERT EDITION ESSENTIALS

- 4 days
- Instructor-led / eLearning
- Develop safety-critical applications with HVP.

02. HELIX PLATFORM INTERMEDIATE TOPICS

- 1 day
- Instructor-led
- Customize Helix Platform Hypervisor to accelerate the development and deployment of safety-critical systems running on different guest operating systems

To book training or learn more, contact
training@windriver.com

HELIX VIRTUALIZATION PLATFORM

HELIX PLATFORM CERT EDITION ESSENTIALS

This course provides hands-on training for developing, configuring, and debugging safety-critical systems using Helix Platform Cert Edition. Participants learn how to create ARINC 653-based systems, configure partitions and schedules, implement inter-partition and intra-partition communication, use health monitoring, and debug multi-partition applications using Wind River Workbench tools..

ABOUT THIS COURSE

After this course, participants will be able to:

- Describe the architecture and key concepts of Helix Platform Cert Edition and ARINC 653.
- Create, configure, build, and deploy Hypervisor and guest projects.
- Configure partitions, schedules, memory, devices, and communication channels using XML.
- Develop ARINC 653 applications using partitions, processes, and APEX APIs.
- Implement communication and synchronization using ports, buffers, blackboards, semaphores, events, and shared memory.
- Configure health monitoring and error management services.
- Debug and analyze multi-partition applications using Workbench Debugger and System Viewer.

COURSE TYPE & DURATION

- Instructor-led training
- Four days

SUPPORTED VERSION

- Helix Virtualization Platform Cert Edition 24.03

PREREQUISITE

- One year of C or C++ programming experience
- Familiarity with the Helix Platform hypervisor

To book training or learn more, contact
training@windriver.com

DAY ONE

01. GETTING STARTED WITH HELIX PLATFORM CERT EDITION

- Helix Platform Cert Edition overview
- ARINC 653 overview
- The specification
- System overview
- Creating a system
- Booting and connecting host and target
- XML editors
- Hands-on lab
- Key references

LAB: Getting Started with Helix Platform

02. HVP CERT EDITION HYPERVISOR ARCHITECTURE

- Architectural design
- Memory translations
- Virtual machines and devices
- Booting Helix Platform Cert Edition

03. HVP CERT EDITION HYPERVISOR CONFIGURATION AND BUILD

- Architectural design
- Host tools
- Memory pools
- Project Types
- Booting and loading
- Key references

LAB: Developing a Helix Platform System

HELIX PLATFORM CERT EDITION ESSENTIALS (CONT'D)

DAY TWO

04. PARTITIONS

- Partitions introduction
- Essentials
- Key references

05. PROCESSES

- Time in ARINC 653
- Introduction to processes
- Essentials
- Key references

LAB: Scheduling Helix Platform Processes

06. INTRA-PARTITION COMMUNICATION

- Intra-partition communication introduction
- Essentials
- Key references

LAB: Communication Within a Partition

08. HEALTH AND ERROR MANAGEMENT

- Introduction to health monitoring
- Process-level error handling
- Partition and module-level error handling
- Key references

LAB: Monitoring the Health of Processes

09. SYSTEM ARCHITECTURE

- Concept review
- Partition OS space
- Module OS space
- HVP Cert Edition multi-core support
- Key references

10. HVP CERT EDITION SHARED MEMORY

- Shared memory features
- Technical details
- System configuration
- Key references

LAB: Creating a Multi-Core Helix Platform System

DAY THREE

07. INTER-PARTITION COMMUNICATION

- Inter-partition communication introduction
- Essentials
- Hands-on lab
- Key references

LAB: Communicating Between Partitions

HELIX PLATFORM CERT EDITION ESSENTIALS (CONT'D)

DAY FOUR

11. HVP CERT EDITION DEBUGGING

- Debugger overview
- Debug access architecture
- VxWorks guest debugging configuration
- System mode debugging configuration
- Debugging details
- Hands-on lab
- Key references

LAB: Debugging Guests

12. SYSTEM VIEWER IN HVP CERT EDITION

- Introduction
- Configuring System Viewer
- Settings for application mode connections
- Settings for system mode connections
- Instrumenting a VxWorks guest
- Using System Viewer
- Key references

LAB: Using System Viewer

To book training or learn more, contact

training@windriver.com

HELIX VIRTUALIZATION PLATFORM

HELIX PLATFORM CERT EDITION INTERMEDIATE TOPICS

Helix Platform Intermediate Topics course focuses on the internal architecture and operation of the Helix Platform Cert Edition Hypervisor, including BSP structure, system boot sequence, hypervisor command-line debugging, device driver models, virtualization interfaces, and health monitoring mechanisms. Participants gain a system-level understanding of how Helix Platform Cert Edition components interact to support partitioned, multicore, safety-certifiable systems, with particular emphasis on customization, diagnostics, and extensibility for deployed platforms.

ABOUT THIS COURSE

After this course, participants will be able to:

- Understand core Helix Platform virtualization concepts, including system initialization, hypervisor roles, and manager-based architecture
- Explain the structure and responsibilities of Hypervisor and Guest OS BSPs
- Describe and analyze the Helix Platform Cert Edition boot process across multicore systems
- Use the Helix Platform Cert Edition hypervisor debug shell and Hypervisor Interface (HVIF) APIs for system inspection and diagnostics
- Understand Helix Platform Cert Edition device models (pass-through, emulated, shared, and paravirtualized) and recommended I/O architectures
- Explain ARINC 653 health monitoring concepts and configure or extend health and error management using custom handlers

COURSE TYPE & DURATION

- Instructor-led training
- One day

SUPPORTED VERSION

- Wind River Helix Virtualization Platform Cert Edition 26.03

PREREQUISITE

- One year of C or C++ programming experience
- Familiarity with the Helix Platform hypervisor
- VxWorks and Workbench Essentials
- Helix Platform Cert Edition Essentials

To book training or learn more, contact
training@windriver.com

01. HELIX PLATFORM CERT EDITION BSP STRUCTURE AND BOOT SEQUENCE

- Hypervisor and managers
- BSP structure
- Project types
- Runtime configuration and host tools
- Hypervisor boot sequence
- Key references

LAB: Using the Hypervisor Debug Shell

02. HELIX PLATFORM CERT EDITION DEBUG SHELL

- Hypervisor debug shell environment
- Hypervisor debug shell interface API
- Key references

LAB: Using the Hypervisor Debug Shell

03. HELIX PLATFORM CERT EDITION DEVICE DRIVERS

- Device types
- I/O model
- VxWorks hypervisor interface API
- Key references

LAB: Using a Custom Manager

04. CUSTOM HANDLER FOR HEALTH AND ERROR MANAGEMENT

- Health monitoring review
- Adding a custom event handler
- Key references

LAB: Using a Custom Health Monitoring Handler

RECOMMENDED LEARNING PATH

Intel Simics Simulator is a full-system simulator that lets developers run, test, and debug software on virtual models of hardware instead of needing the physical hardware itself. Learn to develop, test, and validate software earlier without waiting for physical hardware, reducing development time, cost, and risk.

01. INTEL SIMICS SIMULATOR 6 ESSENTIALS

- 2 days
- Instructor-led / eLearning
- Start, configure, and customize a simulated target in Simics

02. DEVICE MODELING WITH INTEL SIMICS SIMULATOR 6

- 2 days
- Instructor-led
- Create device models in DML, compile devices and test them for correct functionality

03. SYSTEM MODELING WITH INTEL SIMICS SIMULATOR 6

- 1 day
- Instructor-led
- Create hierarchical component models, compile components and test them

To book training or learn more, contact
training@windriver.com

INTEL SIMICS SIMULATOR

INTEL SIMICS SIMULATOR 6 ESSENTIALS

Learn to run and control full-system simulations and debug target software using Intel Simic Simulator 6. Participants learn to start and configure simulated targets, use command-line and Python scripting, connect targets through virtual networks, and apply capabilities such as reverse execution, checkpointing, breakpoints, tracing, logging, and code coverage to analyze system behavior.

ABOUT THIS COURSE

After this course, participants will be able to::

- Start a simulated target in Simics and load target software
- Leverage Simics to improve product development, testing, and integration processes
- Efficiently debug target software using Simics features such as reverse execution, checkpointing, breakpoints, and control and inspection
- Configure and customize Simics virtual target systems using script parameters and command-line scripting
- Connect a Simics simulated target to another target via virtual Ethernet networking
- Use Simics features such as tracing, logging, and code coverage to analyze the behavior and performance of a target system

COURSE TYPE & DURATION

- Instructor-led training
- Two days

SUPPORTED VERSION

- Intel Simics 6

PREREQUISITE

- Some experience with embedded software development
- Some experience with object oriented programming
- Some debugging experience

To book training or learn more, contact
training@windriver.com

DAY ONE

01. SIMICS OVERVIEW

- Overview
- Architecture
- Directory structure and project

02. USING ECLIPSE

- Introduction
- The Simics project
- Perspectives and specific views
- Other views
- Starting Simics

LAB: Getting Started with Simics Eclipse

03. SIMICS COMMAND LINE

- Simics commands
- Scripting
- Script walkthrough

LAB: Using the Simics Command Line

LAB: Creating Script Branches (Optional)

04. SIMICS TARGET STRUCTURE

- Terminology
- Inspecting the configuration
- Checkpoints

LAB: Exploring the Target Structure

INTEL SIMICS SIMULATOR 6 ESSENTIALS (CONT.)

05. SIMICS LOGGING AND TRACING

- Logging vs. tracing
- Logging
- Tracing
- Controlling log/trace output

LAB: Logging and Tracing in Simics

LAB: Performing Advanced Logging and Tracing in Simics (Optional)

06. SIMICS CLASSICAL DEBUGGING

- Overview
- Built-in debugger
- External debuggers
- CLI debugger

LAB: Performing Classical Debugging in Simics

DAY TWO

07. SIMICS ADVANCED DEBUGGING

- Reverse execution
- Advanced breakpoints
- Simulator breakpoints
- OS awareness
- Code coverage
- Magic breakpoints

LAB: Performing Advanced Debugging in Simics

08. MOVING DATA IN AND OUT OF A SIMICS TARGET

- Using memory images for data exchange
- Loading files into memory
- Host as a network resource
- Simics agent
- Simics file system (Linux/Solaris targets only)

LAB: Moving Simics Data

LAB: Moving Simics Data with SimicsFS (Optional)

09. SIMICS NETWORKING

- Simulated networking without the host
- Simulated networking infrastructure
- Real networking host connections
- Real networking port forwarding
- Real networking Ethernet bridging

LAB: Networking in Simics

10. SIMICS MULTI-TARGET

- Terminology
- Synchronizing multiple targets in Simics
- The multi-machine accelerator
- The multi-core accelerator

LAB: Simulating Multiple Targets

SPECIALIZED TOPICS

11. SIMICS SIMULATION PERFORMANCE

- Terminology
- Overview
- Benchmarking
- Performance tuning

LAB: Benchmarking Simics Performance

12. SIMICS NETWORKING DETAILS

- Ethernet link details
- Service node details
- Moving files with the service node
- Real networking details
- Tracing network traffic

INTEL SIMICS SIMULATOR

DEVICE MODELING WITH INTEL SIMICS SIMULATOR 6

Learn how to use the Simics Device Modeling Language (DML) to describe hardware devices like counters, DMA engines, or interrupt controllers.

ABOUT THIS COURSE

After this course, participants will be able to:

- Understand and follow the general device modeling process
- Create device models in DML
- Compile devices and test them for correct functionality

COURSE TYPE & DURATION

- Instructor-led training
- Two days

SUPPORTED VERSION

- Wind River Simics Model Builder for Simics 6
- Wind River Simics Model Builder for Simics 5
- Wind River Simics Model Builder for Simics 4.8

PREREQUISITE

- Some experience with embedded software development
- Python programming experience
- Basic understanding of simulation

To book training or learn more, contact
training@windriver.com

DAY ONE

01. THE DEVICE MODELING PROCESS

- What to model
- A quick discussion on abstraction
- The modeling process
- A note about performance

02. SIMICS DEVICE MODELING TOOLS

- The Workspace Setup tool
- The DMLC compiler
- The makefile
- Eclipse integration

Lab: Creating a Skeleton Device Model

03. DML BASICS

- Introduction
- Fundamentals
- Parameters, methods, attributes, templates and local storage
- Banks, registers, fields, arrays, and groups
- Connects, interfaces, implements, ports

Lab: Preparing the Device Skeleton

Lab: Adding Attributes, Constants, Templates, and Methods

Lab: Adding Banks, Registers, and Simple Side Effects

Lab: Adding Connections, Interfaces, and Ports

DEVICE MODELING WITH INTEL SIMICS SIMULATOR 6 (CONT.)

DAY TWO

04. DEVICE TESTING

- Overview
- A test suite
- Creating devices within a Simics session
- Eclipse integration
- Hands-on

Lab: Testing Your Device

05. ADVANCED DEVICE MODELING TOPICS

- Events
- Miscellaneous statements and expressions
- Interfacing to C
- Simics CLI commands

Lab: Handling Exceptions and Timed Events

Lab: Implementing Device Commands

To book training or learn more, contact

training@windriver.com

INTEL SIMICS SIMULATOR

SYSTEM MODELING WITH INTEL SIMICS SIMULATOR 6

Create hierarchical component models, compile components and test them using Intel Simics Simulator 6.

ABOUT THIS COURSE

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

- Follow the general component modeling process
- Create hierarchical component models in Python
- Compile components and test them for correct functionality, including Simics features like checkpointing and reverse execution

COURSE TYPE & DURATION

- Instructor-led training
- One day

SUPPORTED VERSION

- Wind River Simics Model Builder for Simics 6
- Wind River Simics Model Builder for Simics 5
- Wind River Simics Model Builder for Simics 4.8

PREREQUISITE

- Some experience with embedded software development
- Python programming experience
- Basic understanding of simulation

To book training or learn more, contact
training@windriver.com

DAY ONE

01. THE SYSTEM MODELING PROCESS

- Terminology
- Process
- Utilities
- Eclipse integration

LAB: Planning a Simics system model

LAB: Preparing the modeling environment

02. BASICS, DEVICE INFORMATION, AND ATTRIBUTES

- Component basics
- Getting device information
- Component attributes

LAB: Creating a component

03. HANDLING SLOTS, CREATING COMPONENTS, AND MANUAL TESTING

- Handling slots
- Compiling and creating components
- Manually testing components

LAB: Adding slots and testing manually

SYSTEM MODELING WITH INTEL SIMICS SIMULATOR 6 (CONT.)

04. CONNECTORS

- Connector concept
- Connector behavior
- Connector implementation
- Connector slots

LAB: Adding and using a connector

05. GUI INTEGRATION AND MACHINE SCRIPTS

- GUI integration
- Machine script concept
- Machine script creation

LAB: Modeling a board—the first iteration

06. COMMON MODELING TASKS

- Configuring memory maps
- Connecting devices
- Debugging component code

LAB: Modeling a board—the second iteration

07. TEST AUTOMATION

- Test suites
- Eclipse integration

LAB: Automating test execution

To book training or learn more, contact
training@windriver.com

RECOMMENDED LEARNING PATH

Intel Simics Simulator is a full-system simulator that lets developers run, test, and debug software on virtual models of hardware instead of needing the physical hardware itself. Learn to develop, test, and validate software earlier without waiting for physical hardware, reducing development time, cost, and risk.

01. INTEL SIMICS SIMULATOR 7 ESSENTIALS

- 4 days
- Instructor-led / eLearning
- Develop safety-critical applications with HVP.

02. DEBUGGING AND PERFORMANCE TUNING WITH INTEL SIMICS SIMULATOR 7

- 1 day
- Instructor-led / eLearning
- Customize Helix Platform Hypervisor to accelerate the development and deployment of ARINC 653 safety-critical systems running on different guest operating systems

To book training or learn more, contact
training@windriver.com

INTEL SIMICS SIMULATOR

INTEL SIMICS SIMULATOR 7 ESSENTIALS

Learn to run and control full-system simulations and debug target software using Intel Simic Simulator 6. Participants learn to start and configure simulated targets, use command-line and Python scripting, connect targets through virtual networks, and apply capabilities such as reverse execution, checkpointing, breakpoints, tracing, logging, and code coverage to analyze system behavior.

ABOUT THIS COURSE

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

- Start a simulated target in the Intel Simics Simulator and load target software
- Leverage the simulator to improve product development, testing, and integration processes
- Efficiently debug target software using checkpoints and breakpoints
- Use simulator features such as tracing, logging, and code coverage

COURSE TYPE & DURATION

- Instructor-led training
- Two days

SUPPORTED VERSION

- Intel Simics 7

PREREQUISITE

- Some experience with embedded software development
- Some experience with object oriented programming
- Some debugging experience

To book training or learn more, contact
training@windriver.com

DAY ONE

01. INTRODUCTION TO INTEL SIMICS SIMULATOR

- Overview

02. INTEL SIMICS SIMULATOR ARCHITECTURE

- Intel Simics Simulator architecture
 - Simulator packages
- LAB: Intel Simics Simulator Basics**

03. BEHAVIORAL, FIRMWARE, AND DESIGN/ARCHITECTURE MODELS

- Modelling variants
- Model advantages and disadvantages

04. INTEL SIMICS SIMULATOR CLI

- Introduction
 - CLI Basics
 - The CLI Help System
- LAB: Using the Command-Line Interface**

05. COMMAND-LINE SCRIPTING

- Scripting in the CLI
 - Script branches
 - Paths in the Intel Simics Simulator
 - Starting the Simulator from shell
- LAB: Scripting in the CLI**

INTEL SIMICS SIMULATOR 7 ESSENTIALS (CONT.)

06. PYTHON IN THE INTEL SIMICS SIMULATOR

- Python in Intel Simics Simulator
- LAB: Using Python with the CLI**

07. INTEL SIMICS SIMULATOR OBJECT MODEL

- Objects
- Inspecting objects
- Modules

08. INTEL SIMICS SIMULATOR COMPONENTS

- Components
- Inspecting the configuration

DAY TWO

09. TARGET SETUP AND PARAMETERS

- Introduction to target setup
- Target and target parameters system
- Classic machine scripts (simics)
- Decl-style parameters

LAB: Setting Up the Target System

10. MEMORY MAPPINGS AND SPACES

- Memory spaces
- Inspecting memory contents
- Inspecting device registers
- QSP machine example

LAB: Inspecting Components and Memory

11. LOGGING AND SERIAL CONSOLE CAPTURE

- Logging
 - Serial console output
- LAB: Logging in the Intel Simics Simulator**

12. TRACE COMMANDS, HAPS, AND NOTIFIERS

- Trace commands
 - Haps and notifiers
- LAB: Tracing with the Intel Simics Simulator**

13. INSTRUMENTATION

- Intel Simics Simulator instrumentation
- Statistics tools
- Device register coverage
- Tracer tools

LAB: Using Instrumentation Tools

14. INTEGRATED SOURCE-LEVEL DEBUGGER

- Debugging overview
- Integrated command-line debugger

LAB: Basic Debugging the Intel Simics Simulator

15. EXTERNAL DEBUGGERS

- Overview
- Non-intrusive debug
- Agent-based debug

To book training or learn more, contact
training@windriver.com

INTEL SIMICS SIMULATOR

DEBUGGING AND PERFORMANCE TUNING WITH INTEL SIMICS SIMULATOR 7

Learn about advanced breakpoints, working with the target operating system, understanding simulator timing and performance metrics, working with simulator images, and networking between systems.

ABOUT THIS COURSE

After this course, participants will be able to:

- Identify different kinds of breakpoints and their use
- Debug a target operating system with a simulator
- Explore how the simulator handles time and list performance considerations
- Back up, restore, and work with simulator images
- Network between simulator instances and external interactions

COURSE TYPE & DURATION

- Instructor-led training
- One day

SUPPORTED VERSION

- Intel Simics 7

PREREQUISITE

- Some experience with embedded software development
- Some experience with object oriented programming
- Some debugging experience
- Intel Simics Simulator 7 Essentials

To book training or learn more, contact
training@windriver.com

DAY ONE

01. BREAKPOINTS

- Advanced breakpoints
- Simulator breakpoints
- Magic breakpoints
- X86-specific

LAB: Debugging the Host Operating System

02. OPERATING SYSTEM AWARENESS

- Introduction
- Configuration
- Usage

03. SIMULATOR TIMING AND VIRTUAL TIME

- Virtual time
- Multiple target processors
- Real-time mode

04. SIMULATOR PERFORMANCE

- Performance overview
- Host machine
- Performance technology
- Performance technology: multithreading
- Performance pitfalls
- Performance analysis

DEBUGGING AND PERFORMANCE TUNING WITH INTEL SIMICS SIMULATOR 7 (CONT.)

05. IMAGE OBJECTS AND CRAFF FILES

- Introduction
- CRAFF file format
- Image objects and files
- craff utility

10. NETWORK TRACING AND THE SERVICE NODE

- Tracing network traffic
 - Service node services
- LAB: Analyzing Multiple Instances and Network Traffic**

06. GETTING DATA INTO AND OUT OF THE TARGET

- Disk images
 - Removable drives
 - Loading and saving binaries
 - Network
 - Intel Simics agent and SimicsFS
- Lab: Debugging Driver Initialization**

To book training or learn more, contact
training@windriver.com

07. THE INTEL SIMICS AGENT

- Introduction
- Using the agent

08. CHECKPOINTING AND RECORDING

- Checkpointing
- Inspecting the configuration
- Inspecting the configuration
- Checkpointing in action
- Persistent state
- Session checkpoints

09. NETWORKING BASICS

- Simulating networks
 - Connecting to the real network
- LAB: Networking in Single Instances**

WIND RIVER CLOUD PLATFORM

RECOMMENDED LEARNING PATH

Wind River Cloud Platform is an enterprise-grade private cloud solution built on open-source technologies like StarlingX, Kubernetes, and OpenStack. Learn to manage ultra-low latency, high-performance, and distributed edge or core cloud infrastructure at scale.

01. WIND RIVER CLOUD PLATFORM ADMINISTRATION ESSENTIALS

- 4 days
- Instructor-led / eLearning
- Manage cluster deployment, deploy software updates, modify platform configurations and perform daily cluster operations

02. WIND RIVER CONDUCTOR ESSENTIALS

- 1 day
- Instructor-led / eLearning
- Create and deploy plugins, blueprints, and workflows, configure deployment secrets and geographic settings, and execute workflows on a live Wind River Cloud Platform cluster

03. WIND RIVER ANALYTICS ESSENTIALS

- 1/2 day
- Instructor-led / eLearning
- Use Wind River Analytics to create alerts, collect metrics, and access logs.

To book training or learn more, contact
training@windriver.com

WIND RIVER CLOUD PLATFORM

WIND RIVER CLOUD PLATFORM ADMINISTRATION ESSENTIALS

Learn to deploy, manage, and operate Wind River Cloud Platform through real-world use cases, from initial deployment through day-two operations.

ABOUT THIS COURSE

After this course, participants will be able to:

- Perform daily cluster operation and management tasks
- Manage cluster deployment
- Deploy software updates
- Modify platform configurations
- Examine operation logs and monitor a system

COURSE TYPE & DURATION

- Instructor-led training
- Four Days

SUPPORTED VERSION

- Wind River Cloud Platform

PREREQUISITE

- Familiarity using the Linux OS and its CLI (available from third-party resources such as the Linux Foundation)
- An understanding of networking fundamentals (Ethernet, switching, and routing)
- Familiarity with the fundamentals of virtualization, containers, and Kubernetes orchestration (available from third-party resources such as the Linux Foundation)

To book training or learn more, contact
training@windriver.com

DAY ONE

01. WIND RIVER CLOUD PLATFORM OVERVIEW

- Wind River Cloud Platform infrastructure
- High-Level architecture, scalability, and availability

Lab Exercise: Accessing Wind River Cloud Platform Using Proxmox Virtual Environment

02. SOFTWARE INSTALLATION

- Software installation

Lab Exercise: Installing Wind River Cloud Platform from an ISO Image

03. BOOTSTRAPPING THE CONTROLLER

- Downloading Docker images and running Ansible

Lab Exercise: Bootstrapping the Controller

04. AIO SIMPLEX CONFIGURATION

- Required and optional networks
- Configuring networks using the CLI
- AIO simplex hardware and settings

Lab Exercise: Activating the System Controller

DAY TWO

05. AIO DUPLEX INSTALLATION

- Installing and configuring a standby controller
- AIO duplex hardware

Lab Exercise: Setting up a Standby Controller

WIND RIVER CLOUD PLATFORM ADMINISTRATION ESSENTIALS (CONT.)

06. WORKER HOST SETUP

- Installing and Configuring a Worker Host
- Lab Exercise: Adding a Worker Host**

07. STORAGE OPTIONS

- Storage Options

08. ADDING SUBCLOUDS

- Subcloud Details
 - Installing Subclouds
- Lab Exercise: Installing and Beginning to Bootstrap a Subcloud**
- Lab Exercise: Completing Bootstrapping, and Configuring the Subcloud**

09. ADMINISTRATION OPTIONS

- Three options: CLI, API and GUI
- Horizon UI setup

DAY THREE

10. TECHNICAL DETAILS

- StarlingX History
- Cloud Platform Architecture
- BIOS Requirements

11. DEPLOYMENT MANAGER

- The Deployment Manager
- The Deploy Tool

12. APPLYING PATCHES AND UPGRADES

- Update Orchestration
- Space Reclamation
- Rolling Back Software Updates

13. UPDATING SYSTEM FUNCTION COMPONENTS

- Upgrading Kubernetes
 - Updating the Deployment Manager
- Lab Exercise: Scheduling Nodes and Controlling Pod Replicas**

14. FAULT MANAGEMENT

- Fault Management Overview
 - Logs, Alarms, and Events
 - Sensor Monitoring
 - The Collect Tool
- Lab Exercise: Monitoring Alarms and Events**

15. SUBCLOUD GROUP MANAGEMENT

- Purpose and Usage of Subclouds
- Lab Exercise: Managing Subcloud Groups**

16. CERTIFICATE ADMINISTRATION

- Certificate Overview
 - Certificate Architecture
 - Certificate Management
- Lab Exercise: Creating and Renewing Certificates**

WIND RIVER CLOUD PLATFORM ADMINISTRATION ESSENTIALS (CONT.)

DAY FOUR

17.

ROLE-BASED ACCESS CONTROL

- Authentication and Authorization
- Use Cases

**Lab Exercise: Applying an RBAC
Authentication and Authorization Policy**

18.

BACKUP AND RESTORE

- System Backup
- System Restore
- Backup and Restore Using the
DCManager CLI

**Lab Exercise: Backing Up a Cloud
Platform System**

19.

(OPTIONAL) OPERATIONAL TASKS

- Troubleshooting
- Host Labelling

20.

(OPTIONAL) SUBCLOUD REHOMING

- Distributed cloud
- Rehoming
- Procedure

21.

(OPTIONAL) REPLACING HARDWARE COMPONENTS

- Replacing a Host Using
Deployment Manager
- Replacing a NIC
- Replacing a Disk
- Replacing an Accelerator

22.

OPTIONAL MODULE

- Demo: Patching the Central
Cloud

To book training or learn more, contact
training@windriver.com

WIND RIVER CONDUCTOR

WIND RIVER CONDUCTOR ESSENTIALS

Wind River Conductor is an orchestration and automation platform that centrally deploys, scales, and manages applications and cloud infrastructure across distributed environments. Learn to reduce operational complexity and manual effort by automating and standardizing cloud operations across many sites.

ABOUT THIS COURSE

After this course, participants will be able to:::

- Install Wind River Conductor
- Use the Wind River Conductor CLI
- Install the Wind River Cloud Platform Plugin and Blueprint
- Execute Wind River Cloud Platform Workflows

COURSE TYPE & DURATION

- Instructor-led training
- One Day

SUPPORTED VERSION

- Wind River Conductor

PREREQUISITE

- A familiarity with the Linux OS and its CLI
- An understanding of IP networking fundamentals (addressing and routing)
- A familiarity with the fundamentals of virtualization, containers, and Kubernetes orchestration

To book training or learn more, contact
training@windriver.com

DAY ONE

01. WIND RIVER CONDUCTOR OVERVIEW

- Conductor: cloud orchestration software
- Understanding cloud orchestration
- Conductor features and benefits
- Conductor terminology
- History of Conductor

Optional Lab: Installing Deployment Manager

02. INSTALLING CONDUCTOR MANAGER

- Components of Conductor
- Manager, CLI and Plugins
- Types of plugins
- Installing and verifying Conductor Manager
- Licensing Conductor

Lab: Installing Wind River Conductor

03. INSTALLING AND USING THE CLI

- Pre-requisites for installing CLI
- CLI Installation and verification
- Relating blueprints, workflows and plugins
- Using CLI for plugins and deployments

Lab: Using the Conductor CLI

04. USING CLOUD PLATFORM PLUGINS

- Conductor sites, tenants, security, secrets
- Types of plugins and blueprints
- WRCP Plugins download and installation
- Configuring plugins and blueprints
- TOSCA concepts: nodes, templates, workflows

Lab: Installing the Cloud Platform Plugin and Blueprint

WIND RIVER CONDUCTOR ESSENTIALS (CONT.)

05. UNDERSTANDING WORKFLOWS

- Structure of a blueprint
- Node types and built-in nodes
- Built-in and plugin specific workflows
- Executing different workflows
- WRCP settings for workflows

Lab: Executing Cloud Platform Workflows

06. WORKFLOW VARIATIONS

- Day 0, 1, and 2 operations
- Managing components: LLD (Low Level Design) Document
- CIQ: Customer Information Questionnaire and Jinja2 files
- Suspend, resume, schedule and restarting workflows
- Custom workflows

Lab: Collecting and Transferring Subcloud Logs

Lab: Installing a Subcloud

To book training or learn more, contact

training@windriver.com

WIND RIVER ANALYTICS

WIND RIVER ANALYTICS ESSENTIALS

Wind River Analytics collects and analyzes data across distributed cloud environments to provide operational visibility, insights, dashboards, and alerts. Learn to identify issues using Wind River Analytics, and make faster, data-driven decisions to keep cloud environments optimized and efficiently managed.

ABOUT THIS COURSE

After this course, participants will be able to:

- Understand the architecture of Wind River Analytics
- Install Analytics
- Use Analytics dashboards
- Review alerts and create custom alerts
- Use and customize log and metric collections

COURSE TYPE & DURATION

- Instructor-led training
- Half Day

SUPPORTED VERSION

- Wind River Analytics 24.09

PREREQUISITE

- Familiarity using the Linux OS and its CLI (available from third-party resources such as the Linux Foundation)
- An understanding of networking fundamentals (Ethernet, switching, and routing)
- Familiarity with the fundamentals of Cloud Platform

To book training or learn more, contact
training@windriver.com

DAY ONE

01. INTRODUCTION TO ANALYTICS

- Installation
- LAB: Installing Wind River Analytics
- LAB: Installing Wind River Analytics on a Subcloud

02. USING WIND RIVER ANALYTICS

- Dashboards
- Log and metric collection
- Machine learning jobs

LAB: Monitoring with Wind River Analytics

WIND RIVER LINUX

RECOMMENDED LEARNING PATH

Wind River Linux is a commercial, production-grade embedded Linux development platform and distribution builder. Learn to build, deploy, and manage secure embedded systems and intelligent edge devices without maintaining a “roll-your-own” open-source distribution.

01. WIND RIVER LINUX ESSENTIALS

- 3 days
- Instructor-led / eLearning
- Configure and use components of Wind River Linux using command line technique exclusively.

02.

DEVELOPING LINUX BOARD SUPPORT PACKAGES AND DEVICE DRIVERS

- 4 days
- Instructor-led
- Develop, deploy, and debug customized Linux device drivers and BSP.

DEVELOPING APPLICATIONS WITH WIND RIVER LINUX

- 4 days
- Instructor-led
- Develop, build, test, debug, and deploy Linux applications on Wind River Linux using industry-standard development workflows, advanced Linux features, and target-host integration techniques.

LINUX REAL-TIME ESSENTIALS USING PREEMPT_RT

- 2 days
- Instructor-led
- Develop and deploy real-time Linux applications while analyzing system behavior, optimizing determinism, and leveraging the PREEMPT_RT kernel for low-latency performance.

SELINUX ESSENTIALS

- 2 days
- Instructor-led
- Develop complex policies for securing Linux-based devices using SELinux.

To book training or learn more, contact
training@windriver.com

WIND RIVER LINUX

WIND RIVER LINUX ESSENTIALS

This course teaches developers how to build, customize, deploy, and maintain embedded Linux systems using Wind River Linux LTS. Participants learn essential workflows for application development, kernel customization, target management, package management, debugging, and build system customization using Yocto Project technologies, BitBake, SDKs, and QEMU simulation.

ABOUT THIS COURSE

After this course, participants will be able to:

- Build and customize Wind River Linux platforms and target images.
- Develop and debug Linux applications and kernel modules.
- Configure, deploy, and manage target system
- Integrate software packages, recipes, layers, and templates into a Wind River Linux build environment.

COURSE TYPE & DURATION

- Instructor-led training
- Three days

SUPPORTED VERSION

- Wind River Linux LTS 25
- The following targets are available: QEMU simulated target (Intel® x86-64)

PREREQUISITE

- A basic understanding of operating systems and debugging techniques.
- A working knowledge of Linux and Linux command-line usage.
- Familiarity with makefiles and software build processes.
- Approximately one year of C or C++ programming experience on Linux/UNIX systems.

To book training or learn more, contact
training@windriver.com

DAY ONE

01. INTRODUCTION TO EMBEDDED LINUX

- Overview of Linux
- Linux boot process
- Linux user space
- Cross development
- Open source software licenses

LAB: Getting Started with the Wind River Linux Lab Environment

02. INTRODUCTION TO WIND RIVER LINUX

- Wind River Linux platform
- Creating a build environment
- Build environment structure
- Building target images
- Optimizing builds
- Package revision management

LAB: Managing a Build Environment

LAB: Customizing the Target Image

03. TARGET MANAGEMENT

- Hardware targets
- Cross development workflow
- Deploying to hardware targets
- Simulating a target with QEMU

LAB: Managing Simulated Targets

04. APPLICATION DEVELOPMENT

- Application development workflow
- Migrating applications to a build environment
- Application debugging

WIND RIVER LINUX ESSENTIALS (CONT.)

DAY TWO

05. APPLICATION DEVELOPMENT (CONT'D)

LAB: Building Applications
LAB: Debugging Applications with GDB
LAB: Debugging a Program Crash with GDB
LAB: Profiling CPU Usage
LAB: Profiling Memory Usage
LAB: Analyzing Code Coverage

06. KERNEL DEVELOPMENT

- Configuring the kernel
- Building the kernel
- Kernel modules
- Kernel debugging

LAB: Developing the Kernel
LAB: Managing Kernel Modules
LAB: Developing Kernel Modules
LAB: Configuring KGDB
LAB: Debugging the Kernel with GDB

DAY THREE

07. SOFTWARE MANAGEMENT

- Overview
- Build lifecycle
- Managing packages
- Managing file system trees
- Integrating new software
- Recipes

LAB: Managing Packages
LAB: Patching Packages
LAB: Writing a Recipe
LAB: Integrating Applications with the devtool Tool

08. LAYERS AND TEMPLATES

- Layers
- Anatomy of a layer
- Templates
- Layer and template processing

LAB: Creating Layers
LAB: Managing Layers and Templates

To book training or learn more, contact
training@windriver.com

WIND RIVER LINUX

DEVELOPING LINUX DEVICE DRIVERS AND BOARD SUPPORT PACKAGES

Learn how to develop Linux kernel modules and device drivers for embedded systems using Wind River Linux. The course covers kernel modules, character, block, network, PCI, and USB drivers, along with memory management, concurrency, interrupts, debugging, and BSP creation and customization. Hands-on labs reinforce concepts through practical development and debugging exercises.

ABOUT THIS COURSE

After this course, participants will be able to:

- Explain Linux kernel architecture and driver fundamentals.
- Build, configure, and debug Linux kernel modules.
- Develop character, block, network, PCI, and USB device drivers.
- Manage memory, concurrency, and interrupts in kernel code.
- Use debugging techniques including printk, debugfs, and KGDB.
- Create, customize, patch, and validate Wind River Linux BSPs

COURSE TYPE & DURATION

- Instructor-led training
- Four days

SUPPORTED VERSION

- Wind River Linux LTS 21
- The following targets are available: QEMU simulated target (Intel® x86-64)

PREREQUISITE

- Working knowledge of Linux or UNIX systems.
- Basic understanding of embedded systems concepts.
- Experience programming in C.
- Familiarity with the GNU toolchain and Makefiles.

DAY ONE

01. INTRODUCTION TO LINUX DEVICE DRIVERS

- Linux architecture overview
- Linux device driver overview
- Device driver types
- Linux device model

02. 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

LAB: Managing Simulated Targets from the Command Line

03. INTRODUCTION TO LINUX KERNEL MODULES

- Overview
- Anatomy of a kernel module
- Module licensing
- Building modules
- Installing modules
- Managing modules
- Module parameters

LAB: Managing Kernel Modules

LAB: Developing Kernel Modules

To book training or learn more, contact
training@windriver.com

DEVELOPING LINUX DEVICE DRIVERS AND BOARD SUPPORT PACKAGES (CONT.)

04. CHARACTER DEVICE DRIVERS

- Overview
- Driver Life Cycle
- Major and Minor Numbers
- Character driver entry points
- Blocking operations
- Controlling a device
- Querying the read/write ability
- Restricting operations

LAB: Developing a Character Device Driver

DAY TWO

05. 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

06. CONCURRENCY IN LINUX DEVICE DRIVERS

- Concurrency
- Race conditions
- Locking primitives
- Deadlock
- Atomic variables

LAB: Managing Concurrency in Kernel Code

LAB: Managing Execution of Driver Code

07. HANDLING INTERRUPTS IN LINUX DEVICE DRIVERS

- How interrupts work
- IRQs
- Interrupt handlers

LAB: Implementing Interrupt Handlers

08. 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 THREE

09. LINUX PCI DEVICE DRIVERS

- PCI configuration space
- Identifying devices
- Matching devices and drivers
- Driver registration
- The `probe()` function
- Memory and I/O regions
- DMA
- The `remove()` function

LAB: Driving Devices over PCI

DEVELOPING LINUX DEVICE DRIVERS AND BOARD SUPPORT PACKAGES (CONT.)

DAY FOUR

10. LINUX USB DEVICE DRIVERS

- USB architecture
- Matching devices and drivers
- Driver registration
- The probe() function
- Communicating with the device

LAB: Driving Devices over USB

11. BLOCK DEVICE DRIVERS

- Driver Life Cycle
- Major and Minor Numbers
- Block Layer Architecture
- Block Driver Entry Points
- Processing Requests
- Controlling a Device

LAB: Developing a Block Device Driver

12. NETWORK DEVICE DRIVERS

- Overview of Network Devices
- Driver Registration
- Network Driver Entry Points
- Controlling Interfaces
- Packet Transmission
- Packet Reception

LAB: Developing a Network Device Driver

13. WIND RIVER LINUX BSP OVERVIEW

- The role of a Wind River Linux BSP
- The Wind River Linux BSP structure
- Setting up the build environment
- Configuring and patching a kernel
- Configuring user space

14. CREATING WIND RIVER LINUX BSPS

- BSP development overview
- Starting from scratch
- Starting with third-party code
- Enabling supported boards
- Cloning BSPs
- Packaging Wind River Linux BSPs

LAB: Creating a BSP

LAB: Customizing the BSP

LAB: Patching the Kernel in the BSP Layer

15. ADDITIONAL BSP CONSIDERATIONS

- BSP documentation
- Boot loaders
- Legal requirements
- Validating a BSP

LAB: Validating the Kernel

To book training or learn more, contact
training@windriver.com

WIND RIVER LINUX

DEVELOPING APPLICATIONS WITH WIND RIVER LINUX

Learn how to develop, build, debug, package, and deploy embedded Linux applications using Wind River Linux. Through lectures and hands-on labs, you will gain practical experience with cross-compilation, application debugging, package integration, multi-threading, memory management, file I/O, inter-process communication (IPC), synchronization, and libraries.

ABOUT THIS COURSE

After this course, participants will be able to:

- Build and deploy Linux applications for embedded targets.
- Configure and manage Wind River Linux development environments.
- Cross-compile, test, and debug applications in a target-host environment.
- Integrate applications into Wind River Linux using packages, recipes, layers, and templates.
- Develop multi-process and multi-threaded applications.
- Implement memory management, file I/O, IPC, synchronization, and library techniques in Linux applications.

COURSE TYPE & DURATION

- Instructor-led training
- Four days

SUPPORTED VERSION

- Wind River Linux LTS 18
- The following targets are available: QEMU simulated target (Intel® x86-64)

PREREQUISITE

- Basic knowledge of Linux or UNIX command-line environments.
- Basic understanding of embedded systems and debugging concepts.
- Familiarity with Makefiles and software build tools.
- At least one year of C or C++ programming experience.

To book training or learn more, contact
training@windriver.com

DAY ONE

01. INTRODUCTION TO EMBEDDED LINUX

- Overview of Linux
- Linux boot process
- Linux user space
- Cross development
- Open source software licenses

LAB: Getting Started with the Wind River Linux Lab Environment

02. INTRODUCTION TO WIND RIVER LINUX

- Overview
- Wind River Linux Platform
- Creating a build environment
- Build environment structure
- Building target images
- Optimizing builds
- Package revision management

LAB: Managing a Build Environment from the Command Line

LAB: Customizing the Target Image

03. TARGET MANAGEMENT

- Hardware targets
- Cross development workflow
- Deploying to hardware targets
- Simulating a target with QEMU

LAB: Managing Simulated Targets from the Command Line

DEVELOPING APPLICATIONS WITH WIND RIVER LINUX (CONT.)

DAY TWO

04. APPLICATION DEVELOPMENT

- Application development workflow
 - Migrating applications to a build environment
 - Application debugging
- LAB: Building Applications from the Command Line**
LAB: Debugging Applications with GDB
LAB: Debugging a Program Crash with GDB
LAB: Profiling CPU Usage
LAB: Profiling Memory Usage
LAB: Analyzing Code Coverage

05. SOFTWARE MANAGEMENT

- Overview
 - Build lifecycle
 - Managing packages
 - Integrating new software
 - Recipes
- LAB: Managing Packages**
LAB: Patching Packages
LAB: Writing a Recipe
LAB: Integrating Applications with the devtool Tool

06. LAYERS AND TEMPLATES

- Layers
 - Anatomy of a layer
 - Templates
 - Layer and template processing
- LAB: Creating Layers**
LAB: Managing Layers and Templates

DAY THREE

07. LINUX PROCESSES AND THREADS

- Processes
 - Process lifecycle
 - Threads
 - Thread lifecycle
 - Scheduling
- LAB: Multi-tasking in a Linux Application**
LAB: Managing the Scheduler

08. MEMORY MANAGEMENT

- How memory is managed
 - Allocating and releasing memory
 - Anonymous memory mapping
 - Manipulating memory
 - Common memory problems
- LAB: Allocating Memory**
LAB: Sharing Memory

09. FILE INPUT/OUTPUT

- Overview of files in Linux
 - Basic I/O
 - Buffered I/O
 - Memory-mapped I/O
 - Exchanging binary data
- LAB: Reading and Writing Data**
LAB: Implementing Advanced File I/O
LAB: Working with Directories

DEVELOPING APPLICATIONS WITH WIND RIVER LINUX (CONT.)

DAY FOUR

10. INTER-PROCESS COMMUNICATIONS

- Overview
- Files
- Pipes and FIFOs
- Message queues
- Shared memory
- Sockets
- Signals

LAB: Communicating with Pipes

LAB: Communicating with FIFOs

LAB: Communicating with Message Queues

11. SYNCHRONIZATION

- Overview of concurrency
- File locks
- Semaphores
- Mutexes
- Futexes
- Condition variables
- Deadlocks

LAB: Synchronizing Threads

12. LIBRARIES

- Overview of libraries
- Static libraries
- Shared libraries
- Dynamically loaded libraries
- Common libraries
- Library tools

LAB: Creating Libraries

LAB: Working with Dynamically Loaded Libraries

To book training or learn more, contact
training@windriver.com

WIND RIVER LINUX

LINUX REAL-TIME ESSENTIALS USING PREEMPT_RT

Develop practical skills for designing, optimizing, and evaluating Linux real-time systems using the PREEMPT_RT kernel. This course covers real-time fundamentals, Linux scheduling and synchronization mechanisms, PREEMPT_RT internals, real-time application development techniques, and performance analysis using industry-standard tracing and benchmarking tools. Through hands-on labs, participants configure real-time kernels, measure latency, tune system performance, and analyze real-time behavior.

ABOUT THIS COURSE

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

- Explain real-time concepts, latency, jitter, and determinism.
- Describe Linux scheduling, priorities, processes, threads, and synchronization mechanisms.
- Understand the architecture and behavior of the PREEMPT_RT kernel.
- Develop and optimize Linux applications for real-time performance.
- Configure and tune Linux platforms for improved real-time responsiveness.
- * Measure, analyze, and troubleshoot real-time performance using tools such as ftrace, cyclictest, latencytop, and lmbench.

COURSE TYPE & DURATION

- Instructor-led training
- One day

SUPPORTED VERSION

- Linux kernel versions 2.6 and newer
- Yocto Project Linux
- Wind River Linux 9

PREREQUISITE

- Basic knowledge about the Linux operating system and Linux application or platform development
- Basic experience using Linux command-line tools and utilities.

To book training or learn more, contact
training@windriver.com

DAY ONE

01. REAL-TIME FUNDAMENTALS

- Real-time basics
- Latency and jitter
- Kernel preemption

02. PROCESSES AND SYNCHRONIZATION

- Linux processes and threads
- Atomic operators
- Memory fence
- Spinlocks
- Semaphores
- Mutexes
- Futexes
- CPU variables
- Read/write locks
- Read-copy update

03. SCHEDULING AND PRIORITIES

- Scheduling
- O(N) scheduler
- CFS scheduler
- Scheduling policies
- RT scheduling
- Process priorities

LINUX REAL-TIME ESSENTIALS USING PREEMPT_RT (CONT.)

04. INTERNALS OF THE PREEMPTIVE_RT OPTION

- Preemption modes
- The PREEMPT_RT patch
- Priority inversion and inheritance
- IRQ threads
- IRQ thread development guidelines

05. DEVELOPING LINUX REAL-TIME APPLICATIONS

- Real-time application development
- Memory management
- Thread management
- Scheduling management
- Application development guidelines

06. LINUX REAL-TIME TOOLS

- Latency measurement
- Overall system measurement
- System real-time fine-tuning
- System benchmarking
- System tracing

- **LAB: Configuring and Building a Wind River Linux Preemptable Kernel**
- **LAB: Optimizing Memory Access Performance in an Application**
- **LAB: Evaluating Wind River Linux Kernels with ftrace**
- **LAB: Evaluating a Real-time System with cyclicttest**
- **LAB: Tuning the Platform for Real-Time Performance**
- **LAB: Manipulating Scheduling Policies and Priorities**

To book training or learn more, contact
training@windriver.com

WIND RIVER LINUX

SELINUX ESSENTIALS

This course provides a practical introduction to Security-Enhanced Linux (SELinux) and its role in implementing mandatory access control on Linux systems. Participants learn SELinux architecture, security contexts, type enforcement, role-based access control, and MLS/MCS concepts. Through lectures and hands-on labs, participants gain experience configuring SELinux, managing users and roles, customizing security policies, creating policy modules, and troubleshooting access denials using SELinux auditing and analysis tools.

ABOUT THIS COURSE

After this course, participants will be able to:

- Explain SELinux architecture, terminology, and security models.
- Describe the relationship between DAC, MAC, RBAC, and MLS/MCS security mechanisms.
- Manage SELinux modes, policies, users, roles, and security contexts.
- Use SELinux command-line tools to configure and administer SELinux systems.
- Analyze and modify file and process security contexts.
- Configure SELinux policy behavior using Booleans and policy settings.
- Create, build, deploy, and maintain custom SELinux policy modules.
- Troubleshoot SELinux access denials using audit logs and policy analysis tools.

COURSE TYPE & DURATION

- Instructor-led training
- Two days

SUPPORTED VERSION

- OpenEmbedded/Yocto Project-based distributions
- Wind River Linux LTS
- The following targets are available: QEMU simulated target (Intel x86-64)

PREREQUISITE

- A basic understanding of operating system concepts and file permissions.
- Familiarity with the Linux operating system
- One year of experience working with and/or administering a Linux/UNIX system

To book training or learn more, contact
training@windriver.com

DAY ONE

01. INTRODUCTION

- SELinux Background Information
- DAC, MAC, RBAC
- MAC Implementations
- Linux Security: LSM Model
- SELinux Security Context Introduction

LAB: Getting Started with SELinux

02. SELINUX AND MAC

- Type Enforcement (TE) Security Model
- SELinux Users and RBAC
- MLS and MCS Security Models

LAB: Using SELinux Commands

03. SELINUX SECURITY CONTEXT

- What's a Security Context?
- Policy and Access Control
- Type Enforcement
- RBAC
- MLS
- Domain transitions

LAB: Understanding Security Contexts

04. SELINUX AND SECURITY POLICY

- The Reference Policy
- SELinux Booleans
- Policy Versions
- Wind River Linux Policy

LAB: Using Boolean to Customize Security Policies

SELINUX ESSENTIALS (CONT.)

DAY TWO

05. SELINUX IDENTITY AND ROLES

- SELinux users and Linux users
- Default identities
- SELinux and PAM
- Role switching
- Granting sensitivity and category to users

LAB: Managing SELinux Users

06. SELINUX CONFIGURATION

- Configuration Files
- Logging and the Audit Daemon
- Configuring the SELinux mode

LAB: Configuring an SELinux System

07. CREATING AND MANAGING SELINUX POLICY

- The Policy Development Environment
- Policy Configuration Files
- SELinux Policy Language
- Compiling Security modules
- Loading and Testing Security Modules
- Making Changes Persist Across Boots

LAB: Creating SELinux Modules

08. SELINUX TROUBLESHOOTING

- Root cause analysis
- Silent Denials
- Solutions
- Using Permissive Modes and Domains

LAB: Using Audit Tools for Troubleshooting

To book training or learn more, contact
training@windriver.com

ELXR ESSENTIALS

This course introduces eLxr, an enterprise-grade Linux distribution designed for intelligent edge computing. Through lectures and hands-on labs, participants learn how to install and configure eLxr Server and eLxr Edge, manage and build packages, patch software, develop kernels and kernel modules, build images using containers, manage OSTree-based systems, and deploy applications on Kubernetes-based edge clusters.

ABOUT THIS COURSE

After this course, participants will be able to:

- Install and configure eLxr Server and eLxr Edge systems.
- Manage, build, and patch software packages using Debian-based tools.
- Build and customize Linux kernels and kernel modules.
- Create and build eLxr images using Docker-based build containers.
- Use OSTree and apt-ostree to manage eLxr Edge systems.
- Create and manage edge clusters, and deploy applications using Kubernetes

COURSE TYPE & DURATION

- Instructor-led training
- Two days

SUPPORTED VERSION

- eLxr 12.9.0.0
- The following targets are available: QEMU simulated target (Intel® x86-64) and Virtual Machine Manager

PREREQUISITE

- Basic understanding of operating systems and debugging techniques
- Functional knowledge of Linux
- One year of C or C++ programming experience on Linux

To book training or learn more, contact
training@windriver.com

DAY ONE

01. INTRODUCTION TO WIND RIVER ELXR

- eLxr overview
- Open source and community
- Near edge solution
- eLxr use cases
- Getting started using eLxr

Lab Exercise: Getting Started

Lab Exercise: Installing the eLxr Server

02. VIRTUALIZATION AND HYPERVISOR

- Virtualization basics
- Hypervisor concept Linux virtualization stack

03. PACKAGE MANAGEMENT

- Package overview
- Building packages
- Managing packages

Lab Exercise: Managing Packages

Lab Exercise: Building Packages

04. PATCHING PACKAGES

- Patching overview
- Using quilt to patch packages
- Using git to patch packages

Lab Exercise: Patching Packages

05. APPLICATION DEVELOPMENT

Lab Exercise: Debugging Applications with GDB

ELXR ESSENTIALS (CONT.)

06. KERNEL DEVELOPMENT

- Configuring the kernel
- Building the kernel
- Kernel modules

Lab Exercise: Developing the Kernel

Lab Exercise: Developing Kernel Modules

07. WORKING WITH CONTAINERS

- Docker overview
- Building the docker eLxr Image
- Using the container to configure and build eLxr images

Lab Exercise: Using a Container to Build eLxr Images

10. ELXR EDGE CLUSTER

- Overview fo the edge cluster
- Creating the eLxr cluster
- Managing the eLxr cluster
- Creating a website on the eLxr cluster

Lab Exercise: Creating the Edge Cluster

Lab Exercise: Managing the Edge Cluster

Lab Exercise: Deploying the Kubernetes Dashboard

To book training or learn more, contact
training@windriver.com

DAY TWO

08. ELXR EDGE COMPUTING OVERVIEW

- Overview of the edge computing
- Installing eLxr edge
- Installing cloud-native eLxr edge

Lab Exercise: Installing eLxr Edge

Lab Exercise: Installing Cloud-Native eLxr Edge

09. MANAGING ELXR EDGE

- OSTree overview
- Managing the eLxr edge
- Managing the eLxr edge with apt-ostree

Lab Exercise: Managing the eLxr Edge

ZEPHYR

ZEPHYR ESSENTIALS

This course provides the foundational knowledge for application developers to understand the basics of the Zephyr RTOS to get up and running.

ABOUT THIS COURSE

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

- Explain features and benefits of Zephyr RTOS
- Configure the Zephyr development environment
- Build simple Zephyr applications
- Discuss components of Zephyr projects
- Utilize the Zephyr subsystems and APIs
- Explore multi-threading capabilities
- Explain kernel settings and device trees

TYPE & DURATION

- Instructor-led training
- Three days

SUPPORTED VERSION

- Zephyr 3.7.0
- Visual Studio Code

PREREQUISITE

- C programming, including experience with structures, pointers, pointers to structures, typedefs, macros, and bitwise operators
- Functional knowledge of Linux host operating system.
- Experience using the Visual Studio Code

To book training or learn more, contact
training@windriver.com

DAY ONE

01. COURSE OVERVIEW

- Class Introduction
- Prerequisites
- What to Expect
- Learning Objectives

02. ZEPHYR OVERVIEW

- History of Zephyr
- Zephyr key features and benefits
- The Zephyr project
- Real time OS concepts
- Zephyr use cases and scenarios
- Zephyr 2.x versus Zephyr 3.7

03. INSTALLING ZEPHYR

- Prerequisite components
- West Utility
- Zephyr SDK
- Workspaces and projects
- Application Development vs Platform Development

04. DEVELOPMENT ENVIRONMENT

- Application development options
- Manual setup
- IDE Options
- Zephyr Plugins
- Vendor Specific Environments

Lab: Setting up the Zephyr Environments

05. APPLICATION STRUCTURE

- Types of applications
- Application structures
- Key Files: prj.config, cmake.list
- Kernel Configuration options

Lab: Running the Emulator

ZEPHYR ESSENTIALS (CONT.)

06. DEVICE CONFIGURATION

- Board Details
- Device tree structure
- overlay file
- Device tree specification
- Customizing the device tree

Lab: Changing Boards

DAY THREE

11. MULTI-THREADED APPLICATIONS

- User and System threads
- Thread Lifecycle
- Thread Priority and states
- Single thread Applications
- Multi Thread Applications

Lab: Working with Basic Threads

DAY TWO

07. ZEPHYR API BASICS

- Identify the different Zephyr APIs
- API explanation
- Zephyr Modularity

Lab: Using the LED Sample

12. DATA STRUCTURES AND QUEUES

- Linked Lists
- Ring Buffers
- Work Queues
- System Queues

Lab: Working with User Queue

Lab: Working with Work Queue

08. ZEPHYR GPIO BASICS

- Device driver API
- Types of interfaces
- GPIO configuration
- GPIO API

Lab: Working with GPIO

13. THREAD SYNCHRONIZATION

- Semaphore Types (Binary, counting, mutex)
- Synchronization Issues
- Threads and Semaphores

Lab: Working with Semaphores

Lab: Working with Mutexes

09. THE WEST UTILITY

- West Installation
- West Tool Details
- West Options

Lab: Managing Projects

14. ZEPHYR SCHEDULING

- Scheduling in Zephyr
- Time slicing
- Preemptive and cooperative scheduling

Lab: Working with Ring Buffer

10. REAL-TIME OS CONCEPTS

- The Main Thread and Idle loop
- Time slicing
- Scheduling basics
- Priority based / cooperative option
- Static / dynamic scheduling

15. CUSTOM DEVELOPMENT

- Custom header files
- Linking external libraries
- Development recommendations
- Best practices

Lab: Organizing code

SUBSCRIPTION PACKAGES

Empower your teams with unlimited access to high-quality technical training through our flexible eLearning subscription offerings.

Whether you need comprehensive access to our full catalog or focused training for a specific product, we offer subscription packages designed to meet your needs. Learners can access content on demand and progress at their own pace, making it easy to fit training into busy schedules. Many courses include interactive hands-on labs that reinforce learning through practical experience. With over 160 hours of content available, our subscriptions help organizations accelerate onboarding, reduce the impact of staff turnover, increase product proficiency, and drive greater productivity across teams

ALL CONTENT STREAM

All Content Stream is our most comprehensive subscription package, offering access to the complete Wind River eLearning catalog.

Ideal for organizations with learners across multiple teams and disciplines, this package provides unlimited access to all available training content.

EDGE PRODUCTS STREAM

Edge Products Stream provides access to all eLearning content for the Wind River Edge portfolio, including VxWorks, Helix, Simics, Wind River Linux, eLxr, the Diab Compiler, and Zephyr. Ideal for teams focused on edge software development,

VXWORKS AND HVP STREAM

VxWorks and HVP Stream is for companies that want focused training on the world's most trusted RTOS, VxWorks, along with training on the Helix Virtualization platform.

WIND RIVER CLOUD PLATFORM STREAM

WRCP Stream includes all content on Wind River Cloud Platform, Conductor, Analytics, and Openstack.

WIND RIVER LINUX AND ELXR STREAM

Wind River Linux and eLxr Stream offers access to all content on Wind River Linux and eLxr.

To subscribe or learn more, contact
training@windriver.com

ELEARNING SUBSCRIPTION

VXWORKS & HVP ELEARNING TOPICS

We offer an extensive library of VxWorks and HVP eLearning content covering topics from foundational concepts to advanced development and system design. Our catalog is continuously expanded and updated to ensure learners have access to the latest technologies, best practices, and product capabilities.

TOPICS COVERED

- BSP Infrastructure
- Configuring and Building Hypervisor Systems
- Component Description Language
- Debugging in Workbench
- Debugging SMP Systems
- Deploying a VxWorks 653 System
- Error Detection and Reporting
- Exceptions, Interrupts, and Timers
- Flattened Device Tree
- Health and Error Management
- Inter-Partition Communication
- Inter-Task Communication
- Intra-Partition Communication
- Kernel Initialization Sequence
- Managing Projects in Workbench
- Multi-core Systems
- Optimizing HVP Systems
- Parallel Software Design Theory
- QNX to VxWorks - Different Architectures
- QNX to VxWorks - Migration Strategy
- QNX to VxWorks - Testing and Validation
- QNX to VxWorks - Performance Validation
- Real-Time Multitasking
- Real-Time Process
- Shared Memory
- SMP Architecture
- Targets and Connections
- Troubleshooting HVP Systems
- Uni-Processor to Symmetric Multi-Processor Conversion and Incompatibilities
- VxBus and Device Drivers Fundamentals
- VxWorks Analysis Tools
- VxWorks Boot loaders
- VxWorks Events
- VxWorks I/O Interface
- VxWorks Kernel Shell
- VxWorks Memory
- VxWorks Network Drivers
- VxWorks Semaphores
- VxWorks Serial Drivers
- VxWorks System Viewer
- VxWorks Timer Drivers

To subscribe or learn more, contact
training@windriver.com

ELEARNING SUBSCRIPTION

WRCP ELEARNING TOPICS

Elearning content for WRCP package covers Wind River Cloud Platform, Conductor, and Analytics. Our catalog is continuously expanded and updated to ensure learners have access to the latest technologies, best practices, and product capabilities.

TOPICS COVERED

- AIO Duplex Configuration
- AIO Simplex Installation
- AIO Simplex Configuration
- Bootstrapping the Controller
- Cloud Administration
- Cloud Management
- Cloud Monitoring
- Components of Cloud Platform
- Dashboards and Alerts
- Dcmanager Utility
- Distributed Clouds and Edge Clouds
- Events and Alarms
- Exploring Storage Options
- Horizon UI
- Installing Base Images
- Installing the Standby Controller
- Installing Subclouds
- Installing Wind River Analytics
- Installing Worker Nodes
- Logging Details
- Network Details
- Personalities -- Controller, Worker, Storage
- Role Based Access Control
- Setting up Storage / Ceph
- Standard Configuration
- Standby Controller Features
- Starting and Verifying the Controller
- Subcloud Features and Configuration
- System Backup and Restore
- Worker Node Features and Configuration
- Working with Dashboards and Alerts
- WRCP Installation

To subscribe or learn more, contact
training@windriver.com

ELEARNING SUBSCRIPTION

INTEL SIMICS SIMULATOR ELEARNING TOPICS

ELearning for Simics covers both Intel Simics Simulator version 6 and 7. Learn to develop, run, test, and debug embedded applications on simulated hardware using Intel Simics Simulator.

TOPICS COVERED

- Debugging techniques
- Getting Started with Simics
- Logging and Tracing
- Moving Data In and Out of Simics
- Networking in Simics
- Simics Command Line
- Simulation Performance
- Target Structure
- Timing and Synchronization

To subscribe or learn more, contact
training@windriver.com

ELEARNING SUBSCRIPTION

WIND RIVER LINUX ELEARNING TOPICS

Learn to build, customize, integrate, and deploy secure embedded Linux systems and applications using Wind River Linux and Yocto Project-based development tools.

TOPICS COVERED

- Application Debugging Techniques
- Application Development
- Application Integration Workflows
- Build Optimization Techniques
- Building, Debugging, and Installing Linux Kernel Updates
- Configuring the Build Environment
- Configuring the Root File System
- Configuring Templates
- Creating and Managing Layers
- Creating a Bootable Image
- Creating a Platform Project
- Creating an SDK
- Customizing the Linux Kernel
- Debugging Applications
- Deploying Target Applications
- Developing Cross-Platform Applications
- Developing Kernel Modules
- Developing Recipes
- Embedded Linux Troubleshooting
- Generating Target Images
- Introduction to Embedded Linux
- Introduction to Wind River Linux
- Kernel Debugging Techniques
- Kernel Development
- Layers and Templates
- Managing Linux Packages
- Open Source Software Integration
- Package Revision Management
- QEMU Target Simulation
- Software Management
- Yocto Project Build Workflows

To subscribe or learn more, contact
training@windriver.com

ELEARNING SUBSCRIPTION

ELXR ELEARNING TOPICS

Learn to build, deploy, and manage secure, high-performance Linux systems for enterprise, cloud, and edge environments using eLxr

TOPICS COVERED

- Introduction to eLxr
- Package Management
- Patching Packages
- Working with Containers
- Developing Real-Time Applications
- Deploying eLxr Edge
- Managing Edge System with OS Tree
- Building and Managing Edge Clusters

To subscribe or learn more, contact
training@windriver.com

CONTACT US

2026 WIND RIVER EDUCATION SERVICES

education-inquiry@windriver.com