Linux Kernel Internals and Development

Course LFD420 Four Days Instructor-led Hands-on

Introduction

Learn how to develop for the Linux kernel. In this instructor-led course you?ll learn how Linux is architected, the basic methods for developing on the kernel, and how to efficiently work with the Linux developer community. If you are interested in learning about the Linux kernel, this is the definitive course on the subject.

This course is taught in partnership with the Linux Foundation.

At Course Completion

Upon successful completion of this course, students will gain the following skills and knowledge as you deploy, test, secure, manage, optimize, and troubleshoot a cloud solution.

- Prepare to deploy cloud solutions
- Deploy a pilot project.
- Test a pilot project deployment.
- Design a secure network for cloud deployment.
- Determine CPU and memory sizing for cloud deployments.
- Determine storage requirements for cloud deployments.
- Plan Identity and Access Management for cloud deployments.
- Analyze workload characteristics to ensure successful migration to the cloud.
- Secure systems to meet access requirements.
- Maintain cloud systems.
- Implement backup, restore, and business continuity measures.
- Analyze cloud systems for required performance.
- Analyze cloud systems for anomalies and growth forecasting.
- Troubleshoot deployment, capacity, automation, and orchestration issues.
- Troubleshoot connectivity issues.
- Troubleshoot security issues

Prerequisites

To make the most of this course, students should::

Be proficient in the C programming language, basic Linux (UNIX) utilities such as ls, grep and tar, and be comfortable with any of the available text editors (e.g. emacs, vi, etc.) Experience with any major Linux distribution is helpful but not strictly required.

Contact ISInc for more information at 916.920.1700 or by visiting our website at http://www.isinc.com

Student Materials

The student kit includes the following:

- Comprehensive digital workbook

Course Outline

Module 1: Introduction

- Objectives
- Who You Are
- The Linux Foundation
- Copyright and No Confidential Information
- The Linux Foundation Training
- Certification Programs and Digital Badging
- Linux Distributions
- Platforms
- Preparing Your System
- Using and Downloading a Virtual Machine
- Things Change in Linux and Open Source Projects
- Documentation and Links

Module 2: Preliminaries

- Procedures
- Kernel Versions
- Kernel Sources and Use of git
- Labs

Module 3: How to Work in OSS Projects **

- Overview on How to Contribute Properly
- Know Where the Code is Coming From: DCO and CLA
- Stay Close to Mainline for Security and Quality
- Study and Understand the Project DNA
- Figure Out What Itch You Want to Scratch
- Identify Maintainers and Their Work Flows and Methods
- Get Early Input and Work in the Open
- Contribute Incremental Bits, Not Large Code Dumps
- Leave Your Ego at the Door: Don't Be Thin-Skinned
- Be Patient, Develop Long Term Relationships, Be Helpful

Module 4: Kernel Architecture I

- UNIX and Linux **
- Monolithic and Micro Kernels

Contact ISInc for more information at 916.920.1700 or by visiting our website at http://www.isinc.com

- Object-Oriented Methods
- Main Kernel Components
- User-Space and Kernel-Space
- Kernel Programming Preview
- Error Numbers and Getting Kernel Output
- Task Structure
- Memory Allocation
- Transferring Data between User and Kernel Spaces
- Object-Oriented Inheritance Sort Of
- Linked Lists
- String to Number Conversions
- Jiffies
- Labs

Module 5: Modules

- What are Modules?
- A Trivial Example
- Compiling Modules
- Modules vs Built-in
- Module Utilities
- Automatic Module Loading
- Module Usage Count
- The module struct
- Module Licensing
- Exporting Symbols
- Resolving Symbols **
- Labs

Module 6: Kernel Architecture II

- Processes, Threads, and Tasks
- Process Context
- Kernel Preemption
- Real Time Preemption Patch
- Dynamic Kernel Patching
- Run-time Alternatives **
- Porting to a New Platform **
- Labs

Module 7: Kernel Initialization

- Overview of System Initialization
- System Boot
- Das U-Boot for Embedded Systems**

Contact ISInc for more information at 916.920.1700 or by visiting our website at http://www.isinc.com

- Kernel Startup

Module 8: Kernel Configuration and Compilation

- Installation and Layout of the Kernel Source
- Kernel Browsers
- Kernel Configuration Files
- Kernel Building and Makefiles
- initrd and initramfs
- Labs

Module 9: System Calls

- What are System Calls?
- Available System Calls
- How System Calls are Implemented
- Adding a New System Call
- Labs

Module 10: Kernel Style and General Considerations

- Coding Style
- kernel-doc **
- Using Generic Kernel Routines and Methods
- Making a Kernel Patch
- sparse
- Using likely() and unlikely()
- Writing Portable Code, CPU, 32/64-bit, Endianness
- Writing for SMP
- Writing for High Memory Systems
- Power Management
- Keeping Security in Mind
- Mixing Userand Kernel-Space Headers **
- Labs

Module 11: Race Conditions and Synchronization Methods

- Concurrency and Synchronization Methods
- Atomic Operations
- Bit Operations
- Spinlocks
- Seqlocks
- Disabling Preemption
- Mutexes
- Semaphores
- Completion Functions

Contact ISInc for more information at 916.920.1700 or by visiting our website at http://www.isinc.com

- Read-Copy-Update (RCU)
- Reference Counts
- Labs

Module 12: SMP and Threads

- SMP Kernels and Modules
- Processor Affinity
- CPUSETS
- SMP Algorithms Scheduling, Locking, etc.
- Per-CPU Variables **
- Labs

Module 13: Processes

- What are Processes?
- The task_struct
- Creating User Processes and Threads
- Creating Kernel Threads
- Destroying Processes and Threads
- Executing User-Space Processes From Within the Kernel
- Labs

Module 14: Process Limits and Capabilities **

- Process Limits
- Capabilities
- Labs

Module 15: Monitoring and Debugging

- Debuginfo Packages
- Tracing and Profiling
- sysctl
- SysRq Key
- oops Messages
- Kernel Debuggers
- debugfs
- Labs

Module 16: Scheduling

- Main Scheduling Tasks
- SMP
- Scheduling Priorities
- Scheduling System Calls
- The 2.4 schedule() Function **

Contact ISInc for more information at 916.920.1700 or by visiting our website at http://www.isinc.com

- O(1) Scheduler **
- Time Slices and Priorities
- Load Balancing
- Priority Inversion and Priority Inheritance **
- The CFS Scheduler
- Calculating Priorities and Fair Times
- Scheduling Classes
- Scheduler Details
- Labs

Module 17: Memory Addressing

- Virtual Memory Management
- Systems With and Without MMU and the TLB
- Memory Addresses
- High and Low Memory
- Memory Zones
- Special Device Nodes
- NUMA
- Paging
- Page Tables
- page structure
- Kernel Samepage Merging (KSM) **
- Labs

Module 18: Huge Pages

- Huge Page Support
- Transparent Huge Pages
- libhugetlbfs
- Labs

Module 19: Memory Allocation

- Requesting and Releasing Pages
- Buddy System
- Slabs and Cache Allocations
- Memory Pools
- kmalloc()
- vmalloc()
- Early Allocations and bootmem()
- Memory Defragmentation
- Labs

Module 20: Process Address Space

- Allocating User Memory and Address Spaces
- Locking Pages
- Memory Descriptors and Regions
- Access Rights
- Allocating and Freeing Memory Regions
- Page Faults
- Labs

Module 21: Disk Caches and Swapping

- Caches
- Page Cache Basics
- What is Swapping?
- Swap Areas
- Swapping Pages In and Out
- Controlling Swappiness
- The Swap Cache
- Reverse Mapping **
- OOM Killer
- Labs

Module 22: Device Drivers**

- Types of Devices
- Device Nodes
- Character Drivers
- An Example
- Labs

Module 23: Signals

- What are Signals?
- Available Signals
- System Calls for Signals
- Sigaction
- Signals and Threads
- How the Kernel Installs Signal Handlers
- How the Kernel Sends Signals
- How the Kernel Invokes Signal Handlers
- Real Time Signals
- Labs