Trainings & Workshops

- Live Training Photos
- Online Training
- SAN Trainings
- SAN Training I
- SAN Administration II
- Linux Developers Trainings
- Advanced C Training
- Linux System Programming
- Multithreaded Programming
- Linux Network Programming
- Linux Kernel Internals Training
- Linux Device Drivers Basics
- Linux and C Debugging Training
- Linux Kernel Debugging Training

Training on Linux Device Drivers Programming

« Prev Next »

Title

SF007 – Linux Device Drivers Training

Course Overview

Many Linux professionals would like to write device drivers in Linux, but don’t know how to learn and understand the essentials of writing a driver.

Trainings

Linux Administration Training

Linux Internals & Debugging

Advanced Bash Scripting

download the books (Rubini et al), pdf documents and materials on writing drivers, but fail to understand those driver code. What is required at this point is a systematic approach towards learning the architecture of linux device driver model and how to interface the driver with the linux kernel as well as to the hardware device. There are thousands of device drivers in Linux kernel and are normally characterized as Character drivers, Block drivers, Network drivers and Bus drivers. Furthermore, these bus device drivers can be of various types

Part Time Internships @ Home

1-Day Campus Ambassador

Content Developer

Content Writer

Programmer

Digital Marketer

Linux Device Driver Trainings

Linux Network Device Drivers
Linux PCI Device Drivers
Linux USB Device Drivers
Linux Video Device Drivers
Linux Audio Device Drivers
Linux I2C Device Drivers
DataCenter Trainings
Storoge Protocol Trainings
Bus Protocol Trainings
Network Protocol Trainings

drivers, HDMI drivers, I2C drivers, Uart drivers and a lot more. Furthermore, we have Audio (ALSA) and Video Drivers in the kernel as a separate subsystem. Besides, we have pure software drivers as well as virtual device drivers in Linux as well as older Unixes (Solaris/HPUX /AIX/BSD) that supports many essential and advanced features.

So, how does one master so many device drivers on Linux? The ideal approach is to learn one device driver at a time. Take it as a project on Linux and complete that driver

Linux Kernel Developer
Linux Driver Developer
Linux Network Developer
SAN Developer

fun (and lot of hard work). Right?

Our **Linux device driver training course** helps people learn design and develop one such driver – Virtual Character device driver, on a standard desktop PC architecture (on an x86/x8-64 hardware platform). Every participant will be writing substantial code from scratch and complete that as a project in the training session.

This intensive training course transforms an IT-Professional or a Student into a Linux Device Driver & Kernel Developer. The participant will develop a deep understanding

Linux kernel as well as various devices; Participant will also learn other kernel subsystems and skills necessary to do efficient programming in kernel mode in Linux.

Course Highlights

- Linux Device Driver Training will be delivered by our Founder/ Director who is an Expert with 20+ years of experience in Linux Kernel and SAN software development.
- The course

model so that participant can have a deep understanding of kernel modules & Linux device driver framework as well as kernel mode programming practices.

- Participant will be writing an advanced memory based device driver from scratch that not only teaches techniques to write an efficient driver, but also to races, Linux kernel hangs & oops leading to kernel crash

Course Delivery

- Lectures, Classroom Discussions and Lab Exercises
- 30% Theory, 70% Lab
- Location: Sanfoundry Institute, Bangalore, India

Labs-Assignments

Lab1 – Identification of major and minor numbers for various popular (reserved) devices.
Lab2 – Writing simple kernel module with command line

allocated IO-ports, IO-memory & IRQs on your system/laptop.
Lab4 – Writing a memory based character device driver (DLKM Kernel Module) of fixed size (/dev/sanfd0).
Lab5 – Writing an advanced memory based character device driver of dynamic size (/dev/sanfd_dynamic).
Lab6 – Writing /dev/sanfd_zero device driver (reading any sized data from this device returns zero-filled data).
Lab7 – Writing /dev/sanfd_null device driver (ala bit-bucket / black-hole driver).
Lab8 – Implementation of ioctls – RESET (it should reset

return the current size of the device), EXPAND X (will expand the size of /dev/sanfd_dynamic device by X bytes).
Lab9 – Writing a userspace program to get the device size.
Lab10 – Writing a userspace program to expand the dynamic device size by 1MB and verify the working of the driver.
Lab11 – Writing user-space code to parallelly generate load on the devices, generate race conditions and implement locks in the driver to fix all the issues.

Test the working of all the devices of the driver as follows.

device driver class” > /dev/sanfd0

1b. Verify the output by issuing “cat /dev/sanfd0”

2a. dd if=/dev/sanfd0 of=mydata count=1 bs=512 – verify the output & size of mydata

2b. dd if=/dev/sanfd0 of=mydata count=1 bs=1M – verify the output & size of mydata file

2c. dd if=/dev/sanfd0 of=mydata – verify the output & size of mydata file

3a. dd if=/dev/zero of=/dev/sanfd0 count=1 bs=512 – verify the behavior of the driver

3b. dd if=/dev/zero of=/dev/sanfd0 count=1 bs=1M if=/dev/zero of=/dev/sanfd0 – verify the behavior of the driver

/dev/sanfd_dynamic

1. echo "welcome to sanfoundry's device driver class" > /dev/sanfd_dynamic

2a. dd if=/dev/sanfd_dynamic of=mydata count=1 bs=512 – verify the output & size of mydata
2b. dd if=/dev/sanfd_dynamic of=mydata count=1 bs=1M – verify the output & size of mydata file
2c. dd if=/dev/sanfd_dynamic of=/dev/null – Observe the behavior of the system
3a. dd if=/dev/zero

behavior of the driver
3b. dd if=/dev/zero of=/dev/sanfd_dynamic count=1 bs=1M – verify the behavior of the driver
3c. dd if=/dev/zero of=/dev/sanfd_dynamic – verify the behavior of the system

/dev/sanfd_zero
1. dd if=/dev/sanfd_zero of=zerodata count=1 bs=512 – verify the output & size of zerodata file
2. dd if=zerodata of=/dev/sanfd_zero count=1 bs=512 – verify the behaviour

Pre-Requisites

- Sound knowledge of C

knowledge of Linux/Unix Systems Programming

Target Audience
IT Professionals and/or Students who want to be a serious Linux Device Driver & Kernel Developer on Linux based enterprise and embedded platforms

Fee, Schedule & Registration
Click Here for Linux Device Drivers course training schedule, fee and registration information or if you are interested in Class-room training @ Bangalore.
Click Here if you are a Foreign National,

Course Outline

- What is Kernel
- Linux System Architecture
- Linux Software Architecture
- Basic Kernel Services
- Linux Kernel Code
- What is a Device Driver
- Classes of Devices
- Device Driver Classification
- Concept of a Module
- Fundamental Concepts
- Kernel Module Vs Application
- Namespace
- Major & Minor Numbers
- Reserved Major Numbers
- Module Parameters
- Loading/Unloading Modules
- Current Process Information
- Kernel Memory Allocations
- Driver Entry Points
- Driver Switch Tables
- Module Init & Exit
- Ioctl command numbers
- Capabilities & Restricted Operations
- Driver Usage Count
- Kernel Synchronization Mechanisms
- Introduction to Race Conditions
- Sources of Race Conditions
- Preemption – User & Kernel
- Preemption APIs
- Interrupt Handling APIs
- Semaphores
- Binary & Counting Semaphores
- Reader Writer Semaphores
- Semaphore APIs
- Mutexes
- Spinlocks
- Spinlock APIs
- Atomic Operations
- Seqlocks
- Seqlock APIs

Drivers
- File Structure
- File Operations Structure
- Driver-User Data Transfer
- Driver-Kernel Communication
- Driver-Device Communication
- Device File Creation
- Device File Control Operations

- Deadlock Prevention
- Waitqueues
- Rules for Sleeping
- Waitqueue APIs
- Linux Kernel Tree
- Linux Source Code
- Linux Kernel Configuration

« Prev - Linux/Unix System Programming Training Course
» Next - C Training – Advanced C Programming Course

Recommended Posts:

1. Java Programming Examples
2. C Programming Examples without using Recursion
3. C Programming Examples on Puzzles & Games
4. Java Programming Examples on String Handling
5. 100+ Java Android Programming Examples
6. Home
7. C# Programming Examples on Files
8. C

on Bitwise Operations
9. Java Programming Examples on File Handling
10. Linux Command Tutorials with Examples and Explanations
11. Testimonials
12. Best Reference Books – Technology, Engineering and Sciences
13. Training Classes on C, Linux & SAN – Group Photos
14. C Programming Examples

on LINQ

16. C Programming Examples on File Handling
17. About
18. Linux Questions and Answers
19. Virtualization Internals Training – Linux KVM, QEMU, VMware, VirtualBox
20. Training on Kernel Level Snapshot Software Design & Implementation on Linux

veteran with 20+ years @ Cisco & Wipro, is Founder and CTO at Sanfoundry. He is Linux Kernel Developer & SAN Architect and is passionate about competency developments in these areas. He lives in Bangalore and delivers focused training sessions to IT professionals in Linux Kernel, Linux Debugging, Linux Device Drivers, Linux Networking, Linux Storage, Advanced C Programming, SAN Storage Technologies, SCSI Internals & Storage Protocols such as iSCSI & Fiber Channel. Stay connected with him @ LinkedIn