b1Online trainings
Global Learning Portal

LINUX Device Driver Course Content:35-40hours

Course Outline

1. Introduction

- Role of the Device Driver
- User space, kernel space
- Kernel version numbering
- Kernel sources
- Licenses

2. Installing, Compiling and Booting theLinux kernel

- Obtain and install the kernel source code
- Kernel Configuration
- Kernel Build
- Kernel Images
- The Bootline
- The Root Filesystem
- Install the new kernel
- Lab session on kernel compilation

3. Linux Kernel Modules Programming Benefits of loadable modules

- Loadable device drivers
- Writing loadable kernel module
- Compiling kernel module
- Loading and Unloading the modules
- Using module tools - insmod,modprobe, rmmod, and lsmod,modinfo
- Passing command line parameters to a loadable module
- Module documentation macros
- Exporting symbols from a loadablemodules
- Stacked/Layered loadable modules
- Lab session on Loadable kernel Modules

4. Memory Management

- Kernel space Vs User space memory allocation
- Allocating and deallocating memory using kmalloc, kfree

- Allocating and deallocating memory using vmalloc, vfree
- Allocating pages using get_free_page, free_page etc
- Lab session - Allocating and deallocating memory in kernelSpace

5. Writing Character Device Driver

- Types of device files
- Major and Minor numbers
- Registering and Unregistering char device driver
- Implementing char device driver methods - open, release, read, write,ioctl etc
- Copying data to/from user space
- Creating device files with mknod
- Flow tracing from user space application to driver
- Lab session on char device driver
- Case Studies - virtual char devices -/dev/null, /dev/zero

6. Kernel Synchronization methods

- Concurrency and Race condition
- Need for synchronization in driver code
- Semaphores and mutexes
- Spinlocks
- When to use what ?
- Lab session on using kernel synchronization APIs
- Case Study - Studing locking code in existing driver

7. Debugging kernel problems

- Debugging support in the kernel
- Debugging by printing
- Debugging using gdb
- Debugging using ksyms and symoops
- Debugging using kgdb

8. I/O ports and interrupts

- Use of I/O ports and IRQs
- Platform dependency issues
- Reading and writing to I/O ports
- Types of I/O - Programmed and Interrupt-driven
- Interrupt Handling
- Restriction on kernel code running in interrupt context
- Registering and Unregistering interrupt handler
- Lab Case study - A parallel port LED Driver

9. Writing Block Device Driver

- Comparision between Char and Block devices

- Registering and Unregistering block device driver
- Implementing block device driver
  methods - open, release, ioctl,check_media_change, revalidate
- Implementing block device driver methods - read and write
- Handling requests and data transfer
- Request queue management
- Lab session - Writing a block device driver
- Case Study - Implementing a ramdisk block driver