

“Weekend Workshop on Linux Device Drivers” by Anil Pugalia

Days 1 & 2

+ **Session 1: Linux Kernel Overview**

- W's of Linux Kernel
- Kernel Source Organization
- Kernel Image & Kernel Arguments
- Kernel Configuration & Building

+ **Session 2: Introduction to Linux Drivers**

- W's of Linux Drivers
- Ecosystem of Linux Drivers: Verticals & Horizontals
- Drivers, Modules, and the "Device" Drivers
- Driver Development Environment
- Building & Using Drivers: The First Driver

+ **Session 3: Kernel Debugging & Profiling**

- Various Debugging Techniques
- Various Profiling Techniques
- Hacking & Testing Options

+ **Session 4: Character Drivers**

- Name vs Number: Major & Minor numbers
- Registration & the Cleanups
- Kernel Data Structures & File Operations
- Linux Device Model & Bus Architectures

+ **Session 5: Low-level Accesses**

- Memory Access
- Hardware Access

+ **Session 6: 'Kernel C' Programming**

- Fundamentals of Kernel Programming
- Concurrency & Synchronization
- Time Management
- Delays & Timers

Days 3 & 4

+ **Session 7: USB Drivers**

- Device & Driver Layout
- USB Core
- Driver & Device Registration
- URB & its Functionalities

+ **Session 8: Interrupts**

- Interrupts & IRQs
- Soft IRQs
- Exceptions
- Board coupled Porting

+ **Session 9: Block Drivers**

- Driver & Device Registration
- Kernel Data Structures & Device File Operations
- Request Queue Ecosystem

+ **Session 10: File System Modules**

- Virtual File System
- The Five Operation Sets
- Interaction with the Block Device

+ **Session 11: Wrap Up**

- Conclusion
- What Next?

Caution: All sessions are highly interactive & hands-on with hardware

Hands-On Details

+ **Linux Kernel**

- Source Code Browsing
- Configuring & Building the Kernel

+ **Introduction & First Driver**

- Setting up the environment
- Writing, Building, Using the First Driver

+ **Writing Character Drivers**

- Registering the Character Driver
- Various file operations including read, write, ioctl
- Automatic creation of device file nodes

+ **Doing Low-level Accesses in an x86-based Linux system**

- Accessing the “DOS” days video ram
- Accessing the hardware through Serial interface to control an LED array
- Accessing the hardware through USB interface

+ **'Kernel C' Programming**

- Using the various kernel primitives

+ **Coding USB Driver**

- Registering the USB Driver
- Auto-probing & detection of a USB device
- USB operations with control & interrupt endpoints
- In/Out operations for on-board LED status
- USB2Serial Device Driver

+ **Understanding the Block Drivers**

- Registering the Block Drivers
- Experiments with a RAM-based Block driver
- Creating Partitions and Formatting them
- Coding for the EEPROM Block over a USB interface

+ **Decoding the File System Module**

- Designing a File System
- Coding for a minimal File System
- Mounting the File System through our Block Driver
- Experiments with various File System operations