1. Advanced Linux Programming

Process Management

- What a Process Is
- Process Relationships
- Create a Child Process
- Doing Something Else
- Related execve() Functions
- Wait For a Child
- Changing Priority/Nice
- Real Time Priority

Advance Process Management(Programming with Threads)

- Introducing Threaded Programming
- Applications Suited to Threads
- Building Threaded Programs
- Creating Threads
- Thread Identity
- Synchronizing by Joining
- Stopping Threads
- Synchronizing with Mutexes
- Using Mutexes

Memory Operations

- Allocating/Freeing Memory
- Memory Alignment
- Locked Memory
- Memory Copy/Initialization
- Memory Comparison/Search

File Operations

- Opening/Closing File Descriptors
- File Descriptor I/O
- Repositioning File Descriptors
- Stream/File Descriptor Conversions
- cat using POSIX I/O

Process Scheduling

- Linux's Process Scheduler
- Complete Fair Schuduling
- Process Priority
- Changing Priority/Nice
- Changing Scheduling Policy

Signals

- What Signals Are
- Handling Signals with signal()
- Sending Signals

Interprocess Communication

- Communicating with Pipes
- System Call: pipe()
- Using pipe()
- Named Pipes
- Using Named Pipes
- For Further Reading

Interprocess Communication (IPC)
System V IPC Overview
System V IPC Shared Memory

Computer Network Programming

Introduction to Networking
Need/Uses of Networking
Use of Layered architecture
OSI Protocol layers
Ethernet, Token Ring, Token Bus, FDDI
TCP/IP Stack Internals
User datagram Protocol (UDP)
Transmission Control Protocol (TCP)
Socket concepts
Socket API Interface
Client VS Server
Connectionless and connection oriented client-server communication.
Socket calls for UDP/TCP server/client
Iterative vs concurrent servers
Iterative Connection-less servers (UDP)
Iterative Connection-Oriented servers (TCP)

The GNU C Library and System Calls

GNU C Library - glibc
tools-objdump,file,strace
types of executable
    static executable
    dynamic executable
Building Libraries
Why Use Libraries?
Static Versus Shared
Static Library Benefits
Shared Library Benefits
Creating a Static Library
Using Static Libraries
Creating a Shared Library
Using Shared Libraries
Shared Library Management
ldconfig

2.Linux kernel Programming and Character Device Driver Programming

Kernel Classifications

Monolithic Kernels
Micro Kernels
The User space & Kernel space
Tool Chains, Libraries, The Makefile

Module Programming

The HelloWorld Module
Module Stacking
Module Parameters

- System Calls

The Virtual Filesystem

- Common Filesystem Interface
- Filesystem Abstraction Layer
- VFS Objects and Their Data Structures
  - super block
  - inode block
  - data block
  - boot block

Memory Management

- Kernel High level MMU
- Kernel Low level MMU
- Kernel Memory Allocators
  - slab allocator
  - page allocator
  - fragment allocator
  - pool allocator

Interrupts

- Handling I/O
- I/O Architecture
- I/O Mapped I/O
- Memory Mapped I/O
- Interrupts & Registering Interrupt Handlers
- Interrupt Context vs Process Context

Interrupts Bottom Halves

- Soft irqs
- Tasklets
- Work Queues
- Kernel Data Types

Kernel Synchronization

- Critical Sections, Race Conditions
- Concurrency and its Sources

Mechanisms for Kernel Synchronization

- Semaphores
- Reader/ Writer Semaphores
- Spinlocks
- Reader/ Writer Spinlocks
- Atomic Operations

Memory Allocation in the kernel

Kernel Timers and Time Management

- HZ & Jiffies, Delays
- Kernel Timers

Porc FS

- virtual file systems.
- information about processes
- communication between kernel space and user space
- /proc/interrupts
- /proc/meminfo
- /proc/cpuinfo
- /proc/devices

/proc/ioports

Sys FS

Enumeration of the devices and busses attached to the system
file system hierarchy

Character Drivers and Operations

Registering a System Call
System Call Handler
Service Routines
Character Drivers
Synchronous Driver model

Device Numbers

Major and Minor Numbers
Registering and Unregistering
Static and Dynamic allocations
Important Structures

File Operations

File
Inode
Character Devices

cdev structure

Adding, Allocating, Initializing and Deleting
User Space Applications and Device Driver mapping
Device file operations
Access methods within the driver, open, read, write and close
Advanced Character Drivers
Ioctl implementations
Wait queues and pollings

Accessing Hardware

Accessing I/O Ports
Accessing I/O Memoiry

3.Advance Linux Device Drivers

I.USB Driver

USB Architecture & Protocol
Types of Descriptors
URB structure creation
USB subsystems
USB Driver Layered Architecture
USB Device Drivers
Understanding the USB framework.
Programming the Control Endpoint Zero.
Exchanging the Interrupt Messages
File System Implementation
Virtual File System & its Role
File System Design & Challenges
Kernel File System & and its Operation Sets
Auto-probing & detection of a USB device

II.Block Device Driver

Fundamentals of Block Device Driver

Block drivers Definitions.
Block drivers Registration.
Block device operations.
Linux Block I/O Layer
I/O Schedulers
Block Driver Data Structures and Methods.
How to handle block devices
RAMDISK Device Drivers
RAMDISK-based block device driver.
Using the RAMDISK block device.
Driver registration
Obtaining a gendisk object
Implement the driver's methods.

III.PCI-Network card Drivers

PCI Driver
PCI Architecture & Protocol
PCI Regions & Direct Memory Access
PCI subsystems
PCI Driver Layered Architecture
Porting, Development & Validation of PCI client Driver
Network Device Driver Operations
Network Driver & Device Registrations
Kernel Data Structures & Buffer Management
Programming the PCI
Understanding the x86 processor bus: PCI
PCI Core & Programming the PCI
Finding & Interacting with a PCI Device
Developing the PCI based Network Driver
Programming the Network Device Registers
Implementing the PCI Network Driver
Registering the Network Driver
Buffer Management with skbuffs
Packet Transmission & Reception
Reception using interrupt and poll

4.Porting & Board Bringup Linux- ARM

Toolchain Setup

Introduction to Toolchain
Toolchain Components
Building Toolchain
Toolchain compilation and usage

Bootloader Compilation

Introduction to Bootloader
1 st and 2 nd Stage Bootloader
U-Boot Bootloader Porting
U-Boot Commands Lists
U-Boot Image for Target Board

Clear Understading of Boot Up Sequence

• Getting Started w/ Beagle board
• Embedded Linux System boot up stages
• Beagle board boot up stages

Kernel Configuration
Linux kernel Cross Compilation for Target board
Browsing Linux Kernel Source
Cross-Compilation of Kernel Source
Generating Kernel Image
-uImage
uImage on Target Board
Application development and Cross Compilation
Kernel Procedures
• Booting up the kernel with NFS RootFS

Techniques for Optimizing the Boot up time
• Measuring & Analyzing the boot up time
• Optimization at Kernel space
• Optimization at User space

5.Debugging

What Is My Program Doing?
Source Level Debugging
Invoking gdb
Getting Started with gdb
Examining and Changing Memory
Using gdb with a Running Process
Debugging Libraries - ElectricFence
Debugging with valgrind
Debugging the Kernel
Printk, Traces.
gdb, kgdb.
Proc & Sys File Systems
Timers & Bottom Halves
kernel debugging with dmesg.