Advanced Linux Training

Prerequisites:

1) C programming and compilation using GCC
2) Working knowledge in Linux Environment

Lab Setup:

1) Ubuntu 20.04 installation on VM or physical system
   RAM: 4GB – 8GB
   Hard disk space: 50GB (minimum)
   Capable of connecting to internet.
2) Following packages should be installed on Ubuntu.
   - build-essential
   - vim
   - tree
   - htop
   - openssh-client
   - linux-tools-common
   - linux-tools-generic
   - linux-tools-`uname -r`
   - qemu
   - qemu-utils
   - qemu-kvm
   - virt-manager
   - libvirt-daemon-system
   - libvirt-clients
   - bridge-utils
   - qemu-system-arm
   - valgrind

Contents

1) Linux System Programming
    - Programming languages
    - GCC compilation stages
    - GCC compilation optimization techniques
    - Profile guided optimization.
    - Bin utils
    - Introduction to Linux
    - Linux filesystem hierarchy
    - X86 Assembly Language
    - Arm Assembly Language
    - RISCV Assembly Language
    - Encryption techniques
    - Linux Binary Analysis - Tools
    - ELF analysis
    - Static library
    - Dynamic library
    - Error handling
    - Asserts
    - Linux process Tracing – Ptrace
    - Linux Anti Debugging
    - Process management commands
    - File operations
    - Profiling
    - Code coverage
    - Doxygen
    - Process management
    - Thread Management
    - IPC
    - Process scheduling
    - Thread Scheduling
    - User and Groups
    - Memory management

- OOM Killer
- Temp Files
- Timers
- FUSE
- Filesystem Type
- FUSE FAT Filesystem
- GIT
- SED
- AWK
- Shell
- Cmake
- Makefile
- CPU isolation
- CPU Pinning
- UIO Driver

2) Linux Kernel programming
    - Introduction to Linux kernel
    - Kernel space and user space
    - Classification of Driver
    - Linux kernel versioning
    - Linux kernel source tree
    - Configuring kernel
    - Boot Sequence
    - Rootfs
    - Initramfs
    - Starup init program
    - Storage strategy
    - UML Kernel
    - Qemu Kernel
    - Syscalls
    - Kernel helper functions
    - Kernel logs
    - Printk
    - Module programming
    - Export symbols

- Procfs, sysfs, debugfs and seqfile
- Timers
- Thread
- Bottom halves, softirq
- Memory allocation
- Syncronization
- Signals
- Barriers
- Data structure in kernel
- Interrupts, threaded irq
- Character driver old and new
- Misc driver
- Device tree
- Platform Driver
- Kernel module context
- Idle Thread
- GPIO Driver
- input driver
- I2C Driver
- SPI Driver
- I2C driver user space
- PWM
- UART
- RTC
- Watchdog driver
- Regmap API
- IIO Driver
- USB Driver
- PCI Driver
- Block Driver
- Process management
- RT Linux

3) Linux Debugging
   - Application debugging
     - ✓ Tracing

- ✓ Application Debugging using GDB
- ✓ Address sanitizer
- ✓ thread Sanitizer
- ✓ Valgrind
- ✓ Segfault debugging
- ✓ syslog

- Kernel debugging
  - ✓ Ftrace
  - ✓ Perf
  - ✓ KGDB
  - ✓ KDB
  - ✓ LTTng
  - ✓ eBPF
  - ✓ Crash dump
  - ✓ JTAG Debugging
- Network debugging
  - ✓ Wireshark
  - ✓ TCP dump
  - ✓ eBPF

4) Linux Network programming
   - Application programming
     - ✓ Introduction
     - ✓ Network configuration commands
     - ✓ Socket Example
     - ✓ Casting
     - ✓ IPV6 addressing
     - ✓ Connection Sequence
     - ✓ OSI model and TCP/IP model
     - ✓ Case study-1
     - ✓ IP Headers
     - ✓ TCP Headers
     - ✓ UDP Headers
     - ✓ Socket options
     - ✓ UDP
     - ✓ Name and address conversion
     - ✓ Advanced IO
     - ✓ Domain Sockets

- ✓ Raw Sockets
- ✓ Datalink Access
- ✓ IP Tables
- ✓ Routing
- ✓ Netfilter
- ✓ DHCP
- ✓ DNS
- ✓ NTP

- Kernel programming
  - ✓ Network hardware (MAC , PHY)
  - ✓ Network drivers
  - ✓ Stack analysis and debugging
  - ✓ ARP
  - ✓ ICMP
  - ✓ IPV4
  - ✓ add new protocol