Advanced Linux Programming and Administration

Prerequisites: Familiarity with basic Linux commands and experience with a programming language (such as C, Python, or shell scripting)

Duration: 4 Days (8 Hrs/Day)

Target Audience: Developers, DevOps engineers, advanced system administrators, and IT professionals interested in Linux programming and advanced system management.

Course Objective: This advanced course is tailored for developers and IT professionals looking to deepen their Linux knowledge. It focuses on programming in Linux, managing processes, inter-process communication (IPC), threading, and signal handling. The course includes hands-on labs to develop, debug, and optimize programs in a Linux environment.

Lab Requirement: Koenig DC

Module 1 - Text Editors and Basic Administration

Introduction to Text Editors

- ‘vim’, ‘gedit’, ‘nano’

System Administration Basics

- Rebooting and shutting down
- Managing software packages (‘apt-get’, ‘yum’, ‘dnf’)
- Networking basics (‘ifconfig’)
- Network File System (NFS)

Monitoring and Performance Tuning

- Introduction to ‘top’, ‘htop’, ‘iostat’, ‘vmstat’, and system tuning

Lab: Editing tasks with different text editors

Lab: Installing and managing software packages

Lab: Configuring basic network settings and monitoring system performance

Module 2 - The Bash Shell and Scripting

Command history and navigation

Environmental variables and ‘PATH’

Customizing the prompt

Startup files (‘.profile’, ‘.bashrc’)

Aliases and basic shell scripting

Functions in Bash Scripts

- Writing reusable functions in scripts for modularity

Lab: Writing and debugging shell scripts

Module 3 - Remote Access and Network Tools

Secure Shell (SSH)

File Transfer Protocol (FTP)

Lab: Connecting to remote servers via SSH

Lab: Transferring files with FTP and setting up remote desktop access

Module 4 - Programming with Processes, Signals, and Threads

Programming with Processes

- ‘getpid()’, ‘getppid()’
- Forking and executing processes (fork/exec idiom)
- Handling process termination (‘wait()’, ‘sigchld’)
- Managing zombie processes

Signals

- Introduction to signal handlers
- Signal safety and implementation
- Using ‘kill’, ‘raise’, ‘sigaction’, and ‘sigqueue’
- Real-Time Signals

KOENIG
step forward

- Implementing real-time signal handling in applications

Module 5 - POSIX Threads (pthreads)

Thread creation and attributes

Managing detached threads and thread cancellation

Using thread-specific data

Synchronization with mutexes, semaphores, and condition variables

Lab: Writing C programs for process management

Lab: Implementing signal handling and real-time signal processing

Module 6 - Inter-Process Communication and Time Management

Inter-Process Communication (IPC)

- Pipes and FIFOs
- POSIX semaphores
- Message queues
- Shared memory
- Sockets (Network & Unix domain)

Time Management

- Current time management
- Real-time and process time
- Using timers in programs
- Time Synchronization in Networks
- Understanding and configuring NTP for time synchronization

Lab: Implementing IPC mechanisms in programs

Lab: Working with time and timers, and setting up NTP for network time synchronization