L-1 Linux Systems Programming

Duration: 3 days	
Pre-requisites	
Mandatory	***Preferable***
Course L-0 “Linux CLI and Shell Scripting” *plus*	Usage experience on the Linux platform
Course C-1 “Programming in C”	Shell scripting
Below is the TOC (Table Of Contents) document: it presents the (approximate) Day-wise Coverage with topic names	

Day 1

Module 1 : Introduction

Introduction to Linux
The Birth of Linux: Linus Torvalds' historic post
The GNU project
Licensing : The GNU [L]GPL

OS Architecture
- Processor Protection Levels
- Monolithic Kernel

Module 2 : LINUX Programming Model - Process Management

Process Privileges
- Internal View
- Process Credentials
- Setuid / Setgid programs

Process Execution
- Executing a process
- The exec family of system calls

Lab Assignment

Process creation with the fork system call
- Working with fork()
- Waiting for children

Lab Assignment

Optimizations
- COW
- vfork(2)

Orphans and Zombies
Lab Assignment

Advanced Signalling with POSIX sigaction
Signals in LINUX
Replacing Signal Handlers

POSIX: The sigaction() system call
-signal mask
-signal flags for special behaviour

Day 2

Module 2 : LINUX Programming Model - Process Management (contd.)

Re-entrant Safety Issues

Executing critical code: Blocking signal(s) from delivery to a process when required

Using the (POSIX.1) SA_NODEFER flag
Sending signals to other processes

Lab Assignments

Module 3 : Memory

Internals: An Introduction to Virtual Memory Concepts
Process (Virtual) Image

Common Memory Issues
Memory Leaks
malloc() and fork()
OS Memory Overcommit Behaviour
(Internal view of process image layout as seen by the kernel)

Lab Assignment: Demonstrate an application that "leaks" memory. Can you find tools on Linux that can help you combat this common (and devestating!) software issue? Hint: google.

Module 4 : Memory Management Debugging

Available opensource tools
Valgrind
Introduction
Valgrind Quick Start
Installation
Debugging session
Comparison
Assignments

Module 5 : System Information (/proc) and Performance Monitoring

What is /proc ?
A Map of /proc
Shell scripts to monitor procfs – system and process

Performance Monitoring
- vmstat, dstat
- perf
- top, iotop, iftop
- More tools (blog article)

Day 3

Module 6 : Multithreading on LINUX with POSIX Threads

Introduction to threading
Differences - processes and threads

Thread Management
- Creating Threads
- Terminating Thread Execution
- Example: Pthread Creation and Termination
- Passing Arguments to Threads
- Thread Identifiers
- Joining Threads
- Detaching / Joining Threads
- Example: Joining Threads

Synchronization Issues
- Mutex Variables
  - Mutex Variables Overview
  - Creating / Destroying Mutexes
  - Locking / Unlocking Mutexes
  - Example: Using Mutexes
- Condition Variables

Signal issues
POSIX Scheduling models
- Pthread / syscall APIs to specify scheduling model and thread priority

Pthreads most FAQ

Lab Assignments

Module 7 : IPC

UNIX IPC Mechanisms
- Overview
- SysV IPC Architecture
- Comparison

Sockets - LINUX Network Programming

Introduction – the OSI model
TCP Connection Oriented Socket Calls
Connectionless Socket Calls
Creating a Socket
Binding an Address
UNIX Domain, Internet Domain

Establishing a Connection
listen, accept, connect
Transferring Data

Complete application source walkthrough:
cfingr.c and sfingr.c - a client/server application written using TCP sockets for implementing a simple LINUX finger application.

Lab Assignment : Re-implement the above client/server application as a concurrent multithreaded server (as opposed to a concurrent multi-process server architecture).

Wrap Up.
