L-1_Corp Linux Systems Programming

Duration: 5 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 : Using the GNU Debugger gdb

gdb Commands
Compiling a Program That Is to Be Debugged with gdb
A Typical gdb Session
Useful gdb commands

Lab Assignment: A first Debugging Session with gdb

Core Dumps
Analysing a Core Dump
Article: “Exterminate Bugs Faster with GDB”

Lab Assignments: Three lab assignments follow; in each, the participant is given a buggy program and has to identify and correct it using gdb (and/or other available tools). Hint: One can trace the execution flow of any *process using the powerful [s|l]trace tools.*

Some Important Guidelines for Robust API usage
General Issues
Signal Issues
Sleeping correctly

The Data Display Debugger (ddd)
“The Ten Commandments for 'C' Programmers” – Henry Spencer

"The Top 10 Ways to get *crewe* by the 'C' Programming Language"
- D Dyer

Day 3

Module 4 : 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 5 : Shared Libraries

Introduction
Static & dynamic libraries
Building a Shared Library
ldd, ldconfig

Lab Assignment: Build and demo the usage of a shared library 'mylib.so' that contains the "new" APIs:
- *char * todays_fortune(void);*
- *int is_prime(int num);*

Module 6 : The Linux Scheduler - User-space

Introduction
The userspace API
sched_getaffinity(2), sched_getparam(2), sched_getscheduler(2), sched_setaffinity(2), sched_setparam(2), sched_setscheduler(2)

POSIX Scheduling Model
- SCHED_NORMAL, SCHED_RR, SCHED_FIFO
- Writing a (Userspace) Real-Time Application

Lab Assignment

Day 4

Module 7 : System Information (/proc) and Introduction to 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)

Module 8 : 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

IPC Mechanisms

Module 9 : Pipes

Using Pipes with File Descriptors: pipe()
Using Pipes with File Pointers: popen()
The dup() and dup2() system calls
- How Redirection & Pipes work

A mention on Named Pipes (FIFOs)

Lab Assignment

Day 5

Module 10 : System V IPC Mechanisms

Introduction
System V IPC Identifiers & keys

Message Queues
- Creating/accessing a message queue
- Sending messages to an MQ
- Retrieving messages from an MQ

Synchronization Issues and Why they Matter
Mutexes
Semaphores
- Creating/accessing a semaphore
- The semop() system call
- Case study- file locking using semaphores

Shared Memory
- Creating/accessing a shared memory segment
- Writing/Reading a shmem segment
- Shmem and semaphores

Linux System Limits on SysV IPC
Example programs

Lab Assignment : write a small "library" of convenience routines for using SysV IPC mechanisms effectively.

Module 11 : Memory Mapping

Types of Mappings
File, Anonymous
Private, Shared
Creating a mapping with the mmap() system call
Unmapping
Example

Lab Assignments:

Implement a simple copy program using mmap().

Module 12 : Module 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.
