L-2_Corp Linux Kernel Internals

Duration: 5 days	
Pre-requisites	
Mandatory	***Preferable***
Course L-0 “Linux CLI and Shell Scripting” *plus*	Experience developing Linux apps with 'C', exposure to OS concepts
Course C-1 “Programming in C” *plus*	
Course L-1 “Linux Systems Programming”	
Below is the Outline TOC (Table Of Contents) document: it presents the (approximate) Day-wise Coverage.	

Day 1

Module 1: Linux OS : Fundamental Prerequisites

- OS Architecture
  - Processor Protection Levels
  - Monolithic Kernel

Module 2 : The Linux Kernel Source Tree

- Who makes up the kernel dev community?
- Layout of the kernel source
  - A Brief Tour of the kernel source tree
- Kernel Releases
- Codebase Size

Module 3 : Practical Guide to Writing Kernel Code with LKMs

- Introduction to Loadable Kernel Modules (LKMs)
- Setting up your Test System
- The Hello World Module
- Compiling, the Makefile, Insertion and Removal
- Passing Parameters

Lab Assigments

Day 2

Module 4 : The Process Descriptor

- Timer Interrupt and the Tick Rate (HZ)
- Kernel Mapping of Processes and Threads
- Process Descriptor
  - Task List

Accessing the current task with 'current'
The task_struct

Lab Assigments

Module 5 : Process Creation

Process Creation
The INIT_TASK macro

Processes and Threads on Linux
__clone() and Clone flags
clone, vfork and fork kernel mapping

Kernel implementation of the fork() system call with code walkthrough
Process Context
Error handling and the 'goto'
max_threads
COW semantics

Day 3

Module 6 : The Linux Kernel Scheduler

Introduction
Real-Time and Linux
POSIX Scheduling Policies
Realtime Policies
Pthreads soft-RT example
taskset
chrt
Linux as an RTOS: a mention of the PREEMPT_RT patch
The Scheduling Policy in Action

Scheduler Classes
CFS
CFS Scheduling Concepts
Implementation Details
The rbtree
vruntime Calculation
Scheduling Latency/Period
Scheduling Classes
CFS – Picking the next task
schedule() - the Scheduler entry point
FTrace
Preemption and Context Switching
User Preemption
Kernel Preemption
Dynticks and Tickless Kernels

Module 7 : Linux Memory Management

- Introduction
- VM Basic Working
  - MMU Functional Description
  - MMU Address Translation Process
  - TLB
- Processor/Hardware Components to Improve Memory Latency

Arch-dependent

- Paging
  - Hardware Paging on the x86 / ARM
  - Process Page Tables – PGD and PTE entries
  - ARMv7 MMU Architecture
    - ARMv7-A MMU Architecture
    - Sections and pages
    - Translation Lookaside Buffers (TLB)
    - Protection and memory behavior
  - ARM specific MMU handling with CP15

Day 4

Module 7 : Linux Memory Management (contd.)

Arch-independent

- Linux VM Architecture
- (n-Level) Paging on Linux
- PAE
- 64-bit VM Layout
- Memory Organization
  - NUMA / UMA
  - Zones
- Dynamic Memory Allocation
  - The Buddy System
  - The Slab Allocator
- Memory (De)Allocation API
  - Low-Level API
  - Freeing Pages
  - kmalloc
    - GFP flags and action modifiers
  - kfree
  - vmalloc
  - Slab Cache Interface

Which to Use When

The Memory Map – the page structure
COW Semantics

The Page Cache
VM LRU Page Lists
Notes
Page Reclamation Mechanism
Watermarks
OOM Killer
Invocation
Linux VM Overcommit feature
Page Faults
COW Handling

The effective working of modern glibc malloc on Linux

Process Virtual Memory Mapping – the mm_struct
Memory Regions and VMAs
Viewing VMAs
VMAs on 64-bit
proc: maps, pmap(1), smaps

Appendices
Virtual to Physical Address Translation
Hardware Segmentation on the x86
Translation Lookaside Buffer (TLB) / MMU API
L1 CPU Cache
Kernel Threads

Day 5

Module 10 : Linux x86 / ARM System Call Implementation

System Calls
LINUX x86 and/or ARM Internal System Call Implementation
The sys_call_table
Additional Details
Lab Assignment: Adding your own system call(s)

Audit Project
A mention on the Kprobes infrastructure
Mini Project

Module 11 : Handling Concurrency in the Kernel

The Need for Atomicity
Causes of Concurrency in the kernel

Deadlock Prevention
    Important Guidelines

Concurrency in the Kernel
Spinlocks and Mutexes
The Semaphore Interface
Specialized Locking
    Atomic Operators
    Reader-Writer Locks
    Memory Barriers
Debugging.

Wrap Up
