Linux System Programming

Course 5911 – 40 Hours

Overview

This course explores the system-level programming interfaces provided by Linux to develop applications and Embedded Linux Systems.
The course covers all the services provided by the C-Library to the application layer: accessing and handling files, creating processes and threads, IPC and many more

On Completion, Delegates will be able to

- Understand the Linux programming model
- Create and manage processes and threads
- Using synchronization objects and IPC effectively
- Debugging applications

Who Should Attend

The course is designed for developers who need to make their first steps in Linux development

Prerequisites

Delegates should have a working knowledge with C programming language

Course Contents

Linux for Programmers

- Linux standards and compatibility
- System calls
- Privilege mode – how does it works
- Utilities and system limits
- How to get help – man pages, commands help
- Tools – make , cross compile tools

Working with files

- File descriptors
- Opening and closing files
- Reading and writing files
- File status and the inode
- Mapping files
- dup and dup2
- Creating directories
- FILE* APIs
- Symbolic links
- Poll/select/epoll
- Passing file descriptors between processes

Introduction to Processes

- What is in a process?
- Memory protection
- Process ids, and parent relationship
- Startup and shutdown
- Resource limits
- The process address space
- Virtual and physical memory
- The Linux scheduler
- Creating process – fork/vfork
- Priorities
- Wait* , signalfd and multiplexing
- Fork and exec*
- Process return value
- Adoption
- Zombies and solutions
- System(3)
- Process groups, sessions
- Best practices and common patterns

Posix Threads

- Thread overview
- Creating and controlling threads
- Threads attributes
- Canceling threads
- Thread local storage

Memory allocation

- malloc
- mmap
- Custom allocators
- Obstacks
- Allocating on the stack (alloca(3) and C99 stack allocation)

Synchronization Objects

- Posix mutex
- Mutex attributes
- Futex
- Priority inheritance
- Using reader-writer locks
- Using condition variables
- Avoiding deadlocks
- POSIX semaphores
- POSIX Shared memory
- SYS V IPC – and why we should avoid it
- Barriers and multicore issues
- Atomic operations

Pipes and FIFOs

- Creating pipes
- Working with pipes
- Popen(3)
- Named pipes (FIFOs)
- Working with named pipes

Signals

- What is a signal?
- Signal vs Interrupt
- Reacting to a signal
- Signal masks and signal sets
- Writing a signal handler
- signalfd
- Alarm, and interval timers
- How signal implemented (user/kernel)

Sockets

- Domain and socket types
- Using datagram sockets
- Using stream sockets
- UNIX domain socket APIs
- Internet domain sockets
- Hosts, addresses, and ports
- Byte ordering
- IPv4 and IPv6
- Internet domain socket APIs
- Raw sockets
- Direct access to network device

IO Subsystem

- IO overview
- Kernel page cache
- IO Schedulers
- ionice , ioprio_set
- Using O_DIRECT

Zero Copy

- Overview
- sendfile
- pipe
- splice
- tee
- vmsplice
- Other tools

Application debugging

- Toolchain overview
- Pre-processing info
- Symbols, name mangling
- Debugging information
- ELF format and tools
- Using GDB
- Debugging with eclipse
- DDD
- Remote debugging
- GDB automation
- Debugging with core dumps
- Writing fault handlers
- Other tools for debugging, tracing and profile