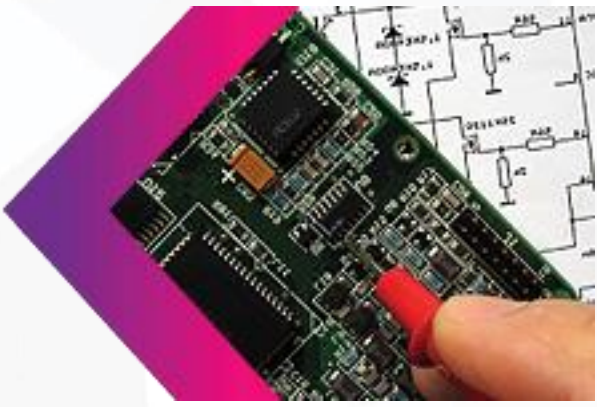
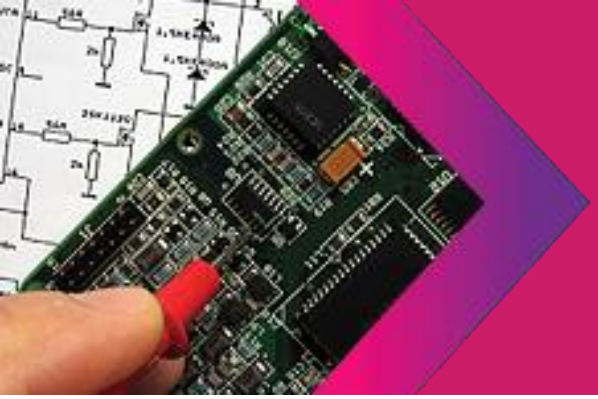
EMERTXE

"Here to take you beyond"

# LINUX INTERNALS & NETWORKING

## Weekend Workshop





**NOW** FOR **SIMPLIFIED** SOLUTIONS

# Linux Internals & Networking – Weekend workshop

- **Objectives:**
  - ✓ To get you started with writing system programs in Linux
  - ✓ Build deeper view about soft interrupts and how Kernel handles it
  - ✓ Usage of synchronous and asynchronous communication mechanisms in Linux
  - ✓ Linux Kernel system call implementation
  - ✓ POSIX thread library usage
  - ✓ Race conditions and thread safe coding practices
  - ✓ Internetworking concepts using TCP/IP
  - ✓ Network packet analysis using Wireshark
  - ✓ Protocol development using TCP/IP socket programming

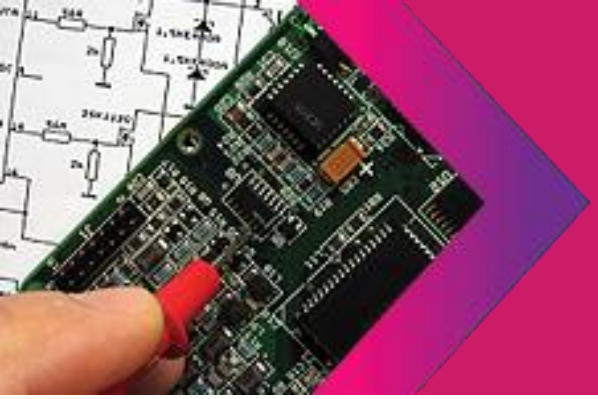
- **Overview:**
  This workshop provides a practical overview Linux internals for writing highly optimized system programs in Linux environment. It gives complete understanding of the Operating system concepts and Linux internals (Interfaces, API's and system calls) and helps the audience to move to the next level of programming by considering other factors in the system. This module is industrial aligned and provides ample practical classes to provide good exposure to Linux programming.

- **Duration:**
  4 days (Two weekends)

- **Platform:**
  Linux (Fedora / Mandriva / Ubuntu)

- **Delivery method:**
  Workshop based approach delivered in a fast-track

- **Pre-requisites:**
  - ✓ Good C & Programming Skills
  - ✓ Basic Hands-On Linux Usage
  - ✓ Fundamentals of Operating systems
  - ✓ Embedded working experience would be a plus





- **Detailed course contents:**

## Day-1:

✓ **Introduction**
- Introduction to Linux & Open source
- GPL, LGPL licensing
- Introduction to various flavors to Linux
- Using the command line interface
- Components of Linux

✓ **System Calls**
- User and Kernel Space
- Introduction to System Calls
- System Calls in Detail
- Strace – Tracing system calls

✓ **Process**
- Introduction to Process
- Process vs. Program
- Process States
- Creating Process
- Process termination
- Special case of processes

## Day-2

✓ **Inter Process Communication (IPC)**
- Introduction to IPC
- Pipe
- FIFO
- Shared Memory
- Advantages and Disadvantages of various IPC mechanisms
- Application use cases

✓ **Signals**
- Introduction to Signals
- Default disposition of Signals
- Handling the Signals
- Signal Related Functions

✓ **Threads**
- Introduction to Threads

- Creating Thread
- Data handling with Thread
- Types of Threads – Thread Attributes
- Thread Cancellation
- Threads vs. Process

## Day-3

✓ **Introduction to Synchronization**
- Introduction to race conditions
- Critical section
- Priority inversion
- Deadlock
- Atomicity & Mutual exclusion
- Solutions to race condition

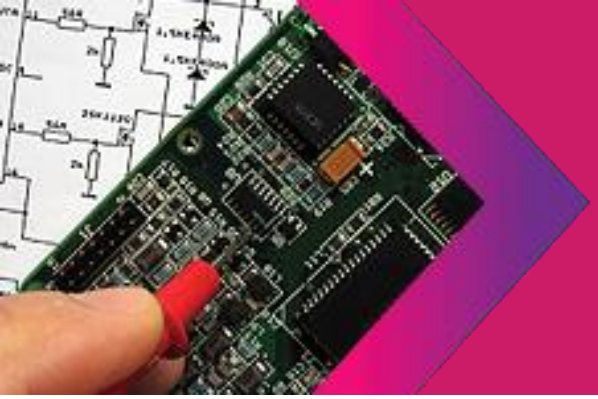
✓ **Thread Synchronization**
- Race condition in multi-threaded applications
- Writing thread safe code
- Mutex
- POSIX Semaphores
- Usage of Binary semaphores and Mutex

✓ **Process synchronization**
- Race condition in multi-process applications
- Limitations of shared memory
- Semaphore (System-V) implementation

✓ **Internetworking with TCP/IP**
- OSI and TCP/IP models
- Addressing in TCP/IP
- IPv4 and IPv6 differences
- TCP three-way handshake
- Network packet analysis in Linux
- Networking commands in Linux

## Day-4

✓ **Linux Network programming using Sockets**
- Socket APIs
- Iterative and Concurrent servers
- Client-server implementation using sockets
- TCP and UDP sockets
- Synchronous I/O using select()
- xinetd daemon in Linux





- ✓ **Process Management**
  - Scheduler and scheduling algorithms
  - Real time OS
  - RTOS characteristics

- ✓ **Memory management**

  - Memory Management Unit (MMU) introduction
  - Virtual memory – Paging & Page fault
  - MMU concepts – Relocation, Protection, Sharing, Logical and physical organization

## • Hands-on session details:

- ✓ System call tracing using strace
- ✓ Timer system call and usage of man pages for programming using system calls
- ✓ Creation of new process using fork(), exec() and system()
- ✓ Tracing Linux process tree structure and establishing parent-child relationship
- ✓ Creation of orphan & zombie process
- ✓ IPC message passing using combination of pipe and shared memory
- ✓ Asynchronous event handling using signal IPC
- ✓ Iterative and current client-server implementation using TCP/UDP
- ✓ Network live packet capturing and analyzing TCP/IP header fields
- ✓ Multi-thread application and data handling using POSIX thread library
- ✓ Race condition & solving in multi-thread & multi-process scenario

## • Workshop Highlights:

- *Platform: x86-based*
- *Kernel: 3.x / 4.x*
- *Optimization and use-case examples*
- *Choosing right IPC mechanism for your multi-tasking application – How?*
- *Writing thread-safe code*
- *Special Focus: Application scenarios in Embedded systems*

Emertxe Information Technologies Private Ltd
#83, 1st Floor,
Farah Towers,
MG road,
Bangalore - 560001

T: +91 809 555 7 333 (M), +91 80 4128 9576 (L)
E: training@emertxe.com