# Designing Server Platforms for POSIX And Linux 5 Using C/C++

## Installer, Daemons/Worker Process, Pipelines, Networking, Config, Performance, Syslog, Systemd, Patterns, HA/HT, Project

Server platforms consist of a mixture of multiple processes and threads, working in a co-ordinated manner, to provide some service to numerous clients on remote machines. These platforms must be flexible, extensible, configurable, scaleable and controllable. A pipeline architecture allows extensible processing of messages. Many techniques are available for flexible inter-process communication. Service processes are the best way to deliver long-lived non-GUI functionality.

Building server platforms for Linux is the logical choice for future-oriented projects.

If your team consists of senior developers experienced with Linux and C/C++ and you are tasked with developing a high-quality server platform on Linux, then this is the ideal course to get all team members up to speed on what is needed. We recommend developing as much as possible using POSIX APIs (for portability), with careful use of additional APIs where it makes sense. This course covers design concepts, important POSIX APIs, plenty of code samples and a chance to have architectural questions answered. It explores extra features (such as Systemd and syslog) that will distinguish your team's platform from the competition.

### Target Audience

System architects and senior software engineers who need to create advanced server platforms for Linux using C/C++.

### Prerequisites

General Linux system programming and especially multithreading experience.

### *Contents of One-Day Training Course*

#### Designing Server Platforms

Multiple Processes/Threads
Central Service (Manager) / Worker Processes / GUI+CLI Admin Processes
Variety of threading architectures

#### Daemon Process

Overview of systemd
Install+config of daemon
Security right
Controlling worker processes

#### Platform Installation

How best to install server apps
Installer formats and server extensions

#### Platform Configuration

Rich config choices (role of /etc)
Web garden/web-farm layout
Config changes without restarting

#### Pipelines

Processing paths for messages
Sequential and non-sequential steps
Structure of pipeline (handlers&modules)
Pipeline context

#### Dynamically loading .so

Dynamically loading shared libraries
Updating a server's shared libraries without having to restart it

#### Networking

High performance sockets design
Eliminating buffer copying
Async I/O

#### Use of Apache Server API

Building custom Apache HTTP Sever module to link it to your daemon process

#### Performance

Detecting bottlenecks
Tuning performance
Developing with performance in mind

#### Monitoring

Building server platform with sysadmins in mind – what capabilites do they need?
How to adding monitoring to your platform

#### Syslog

Logging architecture
Event tracing APIs

#### Designing Platform Security

Leveraging Linux security features
Defense in depth platform security
Secure communication with remote clients

#### Design Patterns for Server Platforms

Patterns to satisfy competing demands
Pooling, tuning, managing
Sharing, distributing
Extending, scheduling

#### High Availability / High Throughput

Hardware for high-availability/throughput
ccNUMA, Interconnects
SAN, DAS, NAS
Clustering concepts
Design for high availability/throughput

#### Project

Overview of development of part of a sample large server platform for Linux
Focus as much as possible on POSIX APIs
Portability and use of OS-specific APIs