Hands-On Unix/Linux Systems Programming with C++

Subtitle:

Take the best advantage of the available hardware resources to write system level code in C++ for Unix & Linux systems.

Overview:

C++ is a general-purpose programming language with a bias towards systems programming. It provides ready access to hardware-level resources, efficient compilation, and a versatile approach to higher-level abstractions.

With this comprehensive course, you will understand the benefits of Unix/Linux systems programming with C++. You will gain a firm understanding of the various C++, and POSIX standards, the current POSIX-compliant systems, and the C++ compilers available for them. After a brief refresher on C++ and the STL, you will proceed programming for Unix/Linux Systems. You will explore details on C++'s and POSIX support for synchronous and asynchronous I/O, multi-threading, sockets and inter-process communication. You'll be implementing practical examples of each of these using C++. You will learn the aspects of realtime programming in POSIX. Towards the end, you will be guided through signals and error handling.

By the end of this course, you will be comfortable with programming high-quality systems in C++.

Target Audience:

If you are a developer who has intermediate knowledge of C++ but little to no knowledge of UNIX and Linux system programming and want to learn system programming with C++, then this course is for you.

Essentials of C++ programming and a basic understanding of multi-threading and network communications is assumed. However, no previous experience in systems programming is required.

Key Features:

Learn the essential tasks and constructs in portable Unix systems programming for live, production applications.
Implement these features and functionality in modern C++.
Learn best practices and idioms for C++ when programming Unix.

About the Author:

George Cross is has over 20 years experience programming C++ on Unix systems. He has worked at such companies as Paypal, Cisco, Business Objects, Hewlett Packard and Borland. He holds a Bachelor of Applied Science from Simon Fraser University, 1990.

Approach:

This series is hands-on with practical coding exercises for the professional developer. You will be briefed on key concepts and then dive straight into the implementation details. All code examples will use modern C++ to layer on top of the C-language POSIX System Interface. Sessions are 30 minutes maximum each session a self-contained topic relevant to POSIX programming and a hands-on real coding exercise.

What You Will Learn:

The essential portable system interfaces in Unix.
Modern C++ idioms for programming legacy C interfaces.
File I/O, Multi-threading, Socket programming and IPC
Signals and Error Handling in Unix.
Fundamentals of applying modern C++ to the POSIX API.
Real-time programming on POSIX-compliant systems.
Expert insight to coding production systems for Unix in C++.

Summary of Contents:

1. *Section One: What are the ISO C/C++, POSIX, IEEE, SUSV4 standards?*
2. *Section Two: C++ 17 Review*
3. *Section Three: I/O, Hands-On coding a Large File Reader*
4. *Section Four: Multi-Threading, Hands-On coding a multi-threaded service*
5. *Section Five: Sockets Concepts and Hands-On server*
6. *Section Six: Sockets Hands-On client*
7. *Section Seven: Inter-Process Communications, Hands-On coding a logging facility using shared memory*
8. *Section Eight: Realtime programming on POSIX*
9. *Section Nine: Signals, Error Handling and Exceptions, Hands-On coding and error handling strategy*

Course Roadmap:

Section One - The relationship between ISO C/C++, POSIX, IEEE, and SUSV4 standards (10 minutes)

Section Two - C++17 Review (20 minutes)

- *Type deduction*
  - *Template type deduction*
  - *auto type deduction*

  - *decltype deduction*
- *Rvalue references*
- *Lambda expressions*
- *Concurrency API*

Section Three - I/O (30 minutes)

- *File I/O with C++ I/O streams*
- *POSIX file I/O*
  - *File descriptors*
  - *Standard streams*
  - *STREAMS*
  - *Asynchronous I/O*
- *Hands-On coding a large file reader using AIO*

Section Four - Multi-threading (30 minutes)

- *POSIX threads*
  - *pthreads interface*
    - *mutex*
    - *condition variables*
    - *read-write locks*
    - *memory barriers*
  - *POSIX thread priority and scheduling*
- *C++ async tasks, promises and futures*
- *Atomics and lock-free programming in C++*
- *Hands-On coding a multi-threaded application*

Section Five - Sockets (30 minutes)

- *TCP - connected sockets*
- *UDP - stateless communication*
- *select versus epoll*
- *Hands-On coding a socket server*

Section Six - Hands-On coding a socket client (30 minutes)

- *Sessions management*
- *Watching socket communications from the browser*

Section Seven - IPC (30 minutes)

- *Message queues*
- *Shared memory segments*
- *Semaphores*
- *Hands-On coding a logging facility using shared memory*

Section Eight - Realtime POSIX programming (20 minutes)

- *Considerations for realtime programming*
- *POSIX interface for realtime applications*

Section Nine - POSIX Signals, Error handling (30 minutes)

- *Hands-On coding a POSIX system error-handling strategy in C++*

Requirements:

- *Any Posix compliant operating system. See this list: https://en.wikipedia.org/wiki/POSIX#POSIX-oriented_operating_systems*

- *Any C++ 17 compiler.*