

C/C++ Toolchain

Understand, Organize, Build, Write, Document, Test, Optimize, Verify, Debug, Package, Explore, Deploy

Lots of things have to happen correctly in order to efficiently transform source code on developers' computers into commercial digital products in use by customers around the world.

After writing the source, the code has to be built, unit tested, documented for developers, stored in repositories, optimized, debugged, the performance monitored, the source style has to be verified, libraries and components packaged & applications deployed.

For C/C++ developers, a toolchain is needed with appropriate tools for each of these tasks.

The C/C++ toolchain often gets less attention than the languages themselves or high-profile libraries (such as STL) but it is vital for highly productive developers to have a deep understanding of the C/C++ toolchain and what it can offer. As projects get larger and there is pressure to deliver updates in shorter time frames but with higher quality expectations, successfully leveraging the C/C++ toolchain and all its rich capabilities makes all the difference.

This course explores a range of useful developer tools that when used together achieves just that.

Contents of One-Day Training Course	
<p>Target Audience Software engineers who wish to more fully explore the toolchain options available for C/C++ projects</p> <p>Prerequisites Experience of C and/or C++ programming. At least some experience of the C/C++ toolchain.</p>	<p>Tour of C/C++ Toolchain Range of tools needs for high-productivity high-quality C/C++ programming</p> <p>Intro to CMake CMake is the popular cross-platform C/C++ build system Relationship to native build systems Defining the build process with CMake Handling the build & source directory tree</p> <p>VIM Options for editing code across platforms General use of VIM VIM for C/C++ source editing</p> <p>Intro to Unit Testing Benefits of unit tests & overview Designing tests and automatic evaluation Intro to Google C++ Testing Framework</p> <p>Advanced Unit Testing More detailed look at Google C++ Testing Framework and how it can be successfully used in large C/C++ projects Mocking, code coverage, performance</p> <p>Code Documentation Doxygen auto-generating documentation Professional finish</p> <p>Lint, etc. Making sure code complies with styling and other requirements Use of lint and other specialist tools</p> <p>Larger Source Trees Organizing large C/C++ source trees Header files, libs, main</p> <p>C/C++ Compilers The three major compilation toolkits are GCC, LLVM and Visual C++ compiler plus there are many smaller toolkits</p> <p>LLVM Tour of LLVM project CLang Optimizer Standard library</p> <p>Language Service for C/C++ Using LLVM to build C/C++ code Command-line options Compilation passes</p> <p>Debugging Native debugger Extracting run time information Symbol management</p> <p>Package Management Unlike other languages, C/C++ does not have popular package management systems (e.g. no C/C++ equivalent of npm or nuget) We look at what are the packaging options available to C/C++ apps (e.g. chocolatey)</p> <p>Deployment Approaches to packaging C/C++ apps for different platforms Deciding where & how to place assets</p> <p>Real-world Toolchain Usage Exploring the toolchain setup for a large open-source C/C++ project</p> <p>Project Correctly setting up toolchain usage for our own enterprise projects</p>