

Windows System Programming Using C/C++

Using low-level Windows OS API for maximum performance, security, extensibility, flexibility

This course examines how to use the Windows C API to design and develop advanced systems-level software. The Windows C API has matured and gained a rock-solid quality reputation with modern features such as threads, symmetric multi-processing, system-wide object model, powerful networking, asynchronous I/O and Unicode. The Windows C API is the common programmatic interface shared by all implementations of the Windows OS family. There are some differences in how it behaves on each OS, but it's possible to create a single EXE/DLL to run on all OSes. The focus of this course is the API of Windows [10 (19H1)| Server 2019].

Important features in the areas of the registry, file systems, security and auditing are discussed. We cover the many techniques available for inter-process communication (e.g. pipes, mailslots, RPC and WinSock). Applications may be made run-time extensible by the configurable loading of DLLs.

Developers seeking extra performance and more flexible low-level control over OS system calls will benefit from writing their system-level application code in C/C++ and this course tells them what they need to know to quickly become productive.

Contents of One-Day Training Course	
<p>Target Audience System architects and experienced developers who need to create advanced systems using the Windows C API.</p> <p>Prerequisites Attendees must have some previous experience of system-level programming</p>	<p>The Windows Platform Windows Architecture Windows SDK Issues of interest to app system developers</p> <p>Overview of Windows C API Applicability of Windows C API Important OS functionality Hardware issues Win32/Win64 on various OSes How to determine the underlying OS “Write to the API, not the OS”</p> <p>General Architecture Major OS components and subsystems Layout of OS and apps on file system Common Windows C datatypes & headers An application developer’s view of the OS</p> <p>Memory Management Flat memory structure Various types of alloc APIs Sharing memory / private heaps</p> <p>File System File APIs Directory handling Memory mapped files</p> <p>Security & Auditing How to programmatically interact with: Security descriptors Security attributes SIDs and tokens ACLs and ACEs Privileges WinStations Desktops</p>
	<p>Registry Complete coverage of the Registry APIs Handling configuration data</p> <p>SEH Structured exception handling Exception handlers Termination handlers</p> <p>DLLs Comparison of DLLs & EXEs DLL functions / variables / memory Explicit loading of DLLs using: LoadLibrary FreeLibrary GetProcAddress</p> <p>C Run-Time Library (CRT) Using the CRT with Win32/Win64 How the CRT is layered above the OS API</p> <p>Inter-Process Communication Anonymous vs. Named pipes IPC using pipes & mailslots Distributing functionality using RPC</p> <p>Windows Networking APIs How to programmatically talk to the net Available APIs and when to use which Managing network connections with WNet</p> <p>WinSock & WinInet A protocol-independent API Relationship to Berkeley sockets WSA functions Coding client and server applications Blocking and non-blocking comms Advanced WinSock Socket options /out-of-band data</p>