Windows Low Level Programming

Course Summary Table

Duration:	5 Day
Target Audience:	Experienced developers
Objectives:	• Understand the main mechanisms and components of the windows OS • Write user-mode programs leveraging the Windows API • Use WinDbg and Visual Studio to debug processes and kernel code • Understand driver development fundamentals • Write kernel-mode drivers
Pre Requisites:	• Basic knowledge of OS concepts and architecture • Power-user level working with Windows • Excellent C knowledge (basic C++ knowledge is recommended)
Hardware setup:	• Windows 10 or 11 x64 (any SKU) • Visual Studio 2019 + latest updates (must include the C++ workload) • (optional) Visual Studio 2022 • Windows 11 SDK (at least the Debugging tools for Windows) • Windows 11 Driver Kit (WDK) • Sysinternals suite (from www.sysinternals.com) • PDF reader

Instructor: **Pavel Yosifovich**

Syllabus

- Module 1: Windows System Architecture
  - Overview
  - Tools
  - Processes
  - Virtual Memory
  - threads
  - User mode vs. Kernel mode
  - Architecture Overview
  - System Calls
  - Introduction to WinDbg
  - Summary

- Module 2: Windows API Foundation
  - Windows APIs
  - Using Visual Studio

    - Common Types and Conventions
    - Working with Strings
    - 64-bit vs. 32-bit development
    - Kernel Objects
    - Working with Handles
    - Sharing Objects
    - Object Names
    - Summary

- Module 3: Processes
    - Process creation
    - The main function(s)
    - Creating processes
    - Process termination
    - Enumerating processes
    - DLLs
    - Summary

- Module 3: Threads
    - Thread basics
    - Creating threads
    - Thread Priorities
    - Thread Scheduling
    - Thread Stacks
    - Hooking
    - Summary

- Module 4: Memory
    - Process address space
    - Process memory counters
    - Reserving and committing memory
    - The heap manager
    - Memory Mapped Files
    - Summary

- Module 5: The I/O System and Device Drivers
    - I/O System overview
    - Device Drivers
    - The Windows Driver Model (WDM)
    - The Kernel Mode Driver Framework (KMDF)
    - Other device driver models
    - Driver types
    - Software drivers
    - Driver and device objects
    - I/O Processing and Data Flow
    - Accessing files and devices
    - Asynchronous I/O
    - Summary

- Module 6: Kernel programming basics
    - C++ in a kernel driver
    - Creating a driver project
    - Building and deploying
    - The kernel API
    - Strings
    - Linked Lists
    - Kernel Memory Pools
    - The DriverEntry function
    - The Unload routine
    - Installation
    - Summary
    - Labs: create a simple driver; deploy a driver

- Module 7: Building a complete driver and Client
    - Creating a device object
    - Exporting a device name
    - Building a driver client
    - Driver dispatch routines
    - Introduction to I/O Request Packets (IRPs)
    - Completing IRPs
    - Accessing user space buffers
    - Handling *DeviceIoControl* calls
    - Testing the driver
    - Debugging the driver
    - Using WinDbg with a virtual machine
    - The driver verifier
    - Lab: open a process for any access; zero driver; debug a driver

- Module 8: Kernel Mechanisms
    - Interrupt Request Levels (IRQLs)
    - Deferred Procedure Calls (DPCs)
    - Exceptions
    - Structured Exception Handling
    - System Crash
    - Thread Synchronization
    - Spin Locks
    - Work Items
    - Summary

- Module 9: Process and Thread Notifications
    - Process creation/destruction callback
    - Specifying process creation status
    - Thread creation/destruction callback
    - Notifying user mode
    - Writing a user mode client
    - User/kernel communication
    - Summary

- Module 10: Object and Registry Notifications (if time permits)
    - Process/thread object notifications
    - Pre and post callbacks
    - Registry notifications
    - Performance considerations
    - Reporting results to user mode
    - Summary