Windows 10 Device Drivers - Fundamentals

Introduction:

This 5-day course gives developers knowledge of the basic fundamentals of writing a Windows 10 device driver using KMDF and UMDF.

At Course Completion

At the end of the course, students should be able to describe fundamentals of Windows 10 Device Driver development including:

- Describe the Windows 10 architecture & software layers
- Understand the role of WDF (Windows Driver Framework) for KMDF (kernel-mode) and UMDF (user-mode) drivers
- Use the Windows 10 Driver Kit (WDK) for driver development
- Integrate the WDK with Visual Studio 2015 for driver development
- Implement Plug and Play for a device driver
- Understand and implement a Windows Universal Driver
- Implement device driver instrumentation using Windows Management Instrumentation (WMI), Event Tracing for Windows (ETW), and Windows Trace Preprocessor (WPP)
- Describe the fundamentals of power management techniques for Windows 10 drivers
- Implement a filter and layered driver
- Debug a device driver using Visual Studio and WinDbg tools
- Provide for installation of a device driver

Prerequisites

Before taking this course, students should have the following skills:

- C Programming Language competency
- Experience with Microsoft Visual Studio
- User-level experience with Windows 7, 8, or 10

Course Outline

Introduction to the Windows 10 Architecture

- Design Goals
- Windows layers
- Kernel mode vs. user mode
- The I/O subsystem
- Kinds of Win10 Drivers

Windows Driver Architecture

- The Driver Models: Legacy, WDM, WDF, KMDF, UMDF
- Universal Drivers
- The KMDF Model
- KMDF Objects
- Event Handling
- The UMDF Model

Building Drivers with MSBuild

- The Windows 10 Driver Kit
- Integration with Visual Studio
- MSBuild
- Project Types
- **Lab**: First Win10 Driver

Plug and Play with Windows 10

- Driver-centricity
- The role of INF files
- The role of the registry
- Device ID string scoring
- Resource discovery & allocation
- **Lab**: PNP Detection of HW Resources

WDF – KMDF Drivers

- Describe the WDF Model
- Dispatching
- Event Handling
- Creating a KMDF Driver
- **Lab**: Loopback Driver

WDF – UMDF Drivers

- Describe the need for User Mode Drivers
- The UMDF Model
- UMDF Objects
- Event Handling
- Creating a UMDF Driver
- **Lab**: Building a loopback UMDF Driver

Universal Drivers

- Describe the purpose of a Universal Driver
- Limitations of a Universal Driver
- Universal DDI functions
- **Lab**: Building a Universal driver

Windows Management Instrumentation (WMI)

- Overview of WMI and event logging
- The WMI Classes
- Becoming a WMI Provider
- Event Tracing for Windows
- Adding ETW support
- **Lab**: Adding ETW support to drivers

Windows 10 Power Management

- Power management principles
- Power requests
- Power management policies
- Implementing power management in a driver
- **Lab**: Adding power management support to a driver

Windows 10 Layered & Filter Drivers

- What is a filter driver?
- Typical uses of filter drivers
- Using KMDF to write a filter driver
- **Lab**: Writing a filter driver

Analyzing Driver Quality

- Code analysis tools
- Static driver verifier
- Source Code Annotation Language (SAL 2.0)
- Analysis warnings
- Performance Monitoring
- **Lab**: Using driver code analysis

Driver Signing

- Why drivers are signed
- Windows 10 signing rules
- Test signing drivers with Visual Studio
- Test certificates
- Catalog files
- Microsoft App Store (Co-Installer out)
- **Lab**: Signing a driver

Debugging Drivers

- The Windows “Blue Screen of Death”
- Debugging drivers with Visual Studio
- Symbol files
- Crash dumps

- Interactive debugging with Visual Studio
- **Lab**: Debugging a driver

Windows Hardware Quality Assurance

- WHQL requirements for Windows 10
- The WHQL test environment
- Windows 10 Logos
- Windows File Protection
- Driver certification
- Digital signatures

