

CodeMachine Course Catalog

© CodeMachine Inc. 1999-2010. All rights reserved.

www.codemachine.com

info@codemachine.com

Course List

Windows Internal Architecture and Troubleshooting

For IT Professionals, System Administrators, Support Engineers and Field Engineers

5 Days (Lecture and Hands-On Labs)

Windows User Mode Internals, Debugging and Dump Analysis

For Application Developers, Support Engineers and Software QA Engineers

5 Days (Lecture and Hands-On Labs)

Windows Kernel Mode Internals, Debugging and Dump Analysis

For Driver Developers, Support Engineers and Software QA Engineers

5 Days (Lecture and Hands-On Labs)

Developing and Debugging Windows KMDF Drivers

For Driver Developers, Support Engineers and Software QA Engineers

3 Days (Lecture and Hands-On Labs)

Developing and Debugging Windows Network Drivers

For Driver Developers, Support Engineers and Software QA Engineers

3 Days (Lecture and Hands-On Labs)

Developing and Debugging Windows Filter Drivers

For Driver Developers, Support Engineers and Software QA Engineers

3 Days (Lecture and Hands-On Labs)

Developing and Debugging Windows Driver Model (WDM) Drivers

For Driver Developers, Support Engineers and Software QA Engineers

3 Days (Lecture and Hands-On Labs)

Course Details

On following pages ...

Windows Internal Architecture and Troubleshooting

5 Days (Lecture and Hands-On Labs)

Target Audience

IT Professionals, System Administrators, Support Engineers and Field Engineers

Description

This course teaches the architecture and internals of the Windows operating system with emphasis on using various tools to troubleshoot common problems and identify offending components on production systems. It helps attendees understand the behind the scenes working of the Windows operating system and troubleshoot various common failures that occur during the operation system.

The hands-on lab familiarizes attendees with the troubleshooting and performance analysis tools (including the debugger) and how to effectively use them to investigate the state of the system, identify common problem symptoms and isolate faulty components.

Pre-Requisites

Attendees should be familiar with basic operating system concepts and have hands-on experience using the Windows. This course does NOT require attendees to have a developer (programming) background.

Attendees who have developer background should attend the Windows User Mode Internals, Debugging and Dump Analysis or the Windows Kernel Mode Internals, Debugging and Dump Analysis courses.

Topics

Platform Architecture

OS Components
User Mode vs. Kernel Mode
CPU Support
Symmetric Multiprocessing
IRQLs, Interrupts and DPCs
Virtualization

Debugging Tools

Debugging Tools for Windows
Performance Analysis Tools
Performance Monitoring Tools
Profiling Tools
SysInternals Tools

Processes and Threads

Processes
Threads
Sessions
System Service
Thread States
Thread Priorities
Thread Scheduling
Thread Pools
Synchronization
User Mode Scheduling (UMS)

Boot-up and Shutdown

Boot Components
OS Kernel and Boot-Drivers
User Mode Startup
Service Startup and Shutdown
User Session Creation & Teardown

Objects and Handles

Object Name Space
Session Name Space
Symbolic links
Handle Tables
Objects
Reference Counting

Memory Manager

Physical Memory, PAE & NUMA
Virtual Memory
Process Virtual Address Space
Reserved and Committed Memory
Address Windowing Extension (AWE)
Process Heaps
Thread Stacks
Working Set
Shared Memory
System Virtual Address Space
Sessions Space
File System Cache
Page Tables (PTEs)
Page States
Pools

Services

Services Architecture
Service Control Manager
SVCHost
Service Security
Window Stations and Desktops
Session Isolation

Security

Security Identifiers (SID)
Tokens
Impersonation
Security Descriptors
Rights & Privileges
Mandatory Integrity Levels
User Account Control (UAC)
Logon Process & Authentication

Devices and Drivers

Device Hierarchy
Boot & Critical Devices
Driver Staging
INF, PNF & CAT Files
Driver Signing
System & Device Power States
Sleep and Hibernation
Remote Wakeup

Dump Generation & Analysis

Dump Generation
Debugger Configuration
!analyze -v
Register Contexts
Hang vs. Crash Dumps
Analyzing Process State
Analyzing System State
Identifying Faulting Modules

Windows User Mode Internals, Debugging and Dump Analysis

5 Days (Lecture and Hands-On Labs)

Target Audience

Application Developers, Support Engineers and Software QA Engineers

Description

This course teaches architecture and internals of the Windows operating system with emphasis on production debugging of user mode applications and services. It helps attendees understand the behind the scenes working of the Windows operating system and debug common crashes and hangs that occur during user mode code execution.

The hands-on lab familiarizes attendees with the debugging and instrumentation tools, relevant debugger extension commands, interpretation of the command's output to investigate the state of processes, debugging techniques to isolate faulting modules and root cause crashes and hangs caused by applications and services.

Pre-Requisites

Attendees who must be familiar with basic operating concepts like processes, threads, virtual memory, synchronization, object and handles. To get the most value from the course attendees must be familiar with Win32 API and C programming language.

Attendees who do not have a developer (programming) background should attend the Windows Internals and Troubleshooting course.

Topics

Windows Architecture

- Memory Addressing
- CPU & Registers
- User Mode vs. Kernel Mode
- Operating System Components
- Processes and Threads
- Executable Images
- Virtual & Physical Address Space

Debugging Tools

- Debugging Tools for Windows
- Live Debugging Scenarios
- Image File Execution Options
- Gflags for Application Debugging
- Application Verifier
- User-Mode Dump Heap (UMDH)
- SysInternals Tools

Basic Debugging

- Debugger Components
- Debugger Usage
- Symbol Files
- Public & Private Symbols
- Symbol Server
- Symbol Usage
- Basic Data Structures
- Live vs. Post Mortem Debugging

User Mode Memory Dumps

- Structured Exception Handling
- Post-Mortem Debuggers
- First & Second Chance Dumps
- Memory Dump Contents
- Dump Generation Tools
- Windows Error Reporting (WER)
- Kernel Dumps for Application issues

Dump Analysis

- Principles of Debugging
- Common Analysis Steps
- Debugger Contexts
- Analyzing Process State
- Modules Identification Techniques
- Fault Analysis Techniques

Debugging Assembler Code

- CPU Registers
- X86 and X64 Instruction Formats
- Common Instruction Sequences
- Reverse Engineering Techniques
- Navigating Assembler Code
- Debugging Optimized Code

Process and Threads

- Process Resources
- Process and Thread Data Structures
- WOW64 Processes
- API Call Failures
- High CPU Usage
- Unresponsive UI

Process Virtual Address Space

- Reserved and Committed Memory
- Virtual Address Space Layout
- Working Sets
- Address Space Fragmentation
- Address Space Depletion

Objects and Handles

- Process Handle Table
- Handle Duplication
- Handle Leaks
- Invalid Handles

Call Stacks

- User Mode Calls Stacks
- Function Prolog & Epilog
- Calling Conventions
- Frame Pointer Omission (FPO)
- Retrieving Information from Call Stacks
- Issues with X64 call stacks
- Stack Corruption
- Stack Overflows

Process Heaps

- Heap Internals
- Heap Layout
- Low Fragmentation Heap
- Heap Corruption
- Heap Leaks

Synchronization

- Thread Synchronization
- Critical Sections
- Slim Reader Writer Locks
- Intra-Process Deadlocks

Services

- Service Architecture
- Service Registration
- Service Startup
- Service Failures

Remote Procedure Calls (RPC)

- RPC/COM/DCOM Internals
- Marshalling
- RPC Protocols
- RPC Debugging
- RPC Hangs

Windows Kernel Mode Internals, Debugging and Dump Analysis

5 Days (Lecture and Hands-On Labs)

Target Audience

Driver Developers, Support Engineers and Software QA Engineers

Description

This course teaches architecture, internals and of the Windows operating system with emphasis on production debugging of kernel mode drivers. It helps attendees understand the behind the scenes working of the Windows operating system and debug common crashes and hangs that occur during kernel mode code execution.

The hands-on lab familiarizes attendees with the debugging and instrumentation tools, relevant debugger extension commands, interpretation of the command's output to investigate the state of device drivers and the system, debugging techniques to isolate faulting modules and root cause crashes and hangs caused by drivers.

Pre-Requisites

Attendees must be familiar with the basic usage of the debugger (Debugging Tools for Windows). Basic usage includes symbol server, debugger commands for displaying call stacks, data structures, memory contents and system information. To get the most value from the course attendees must be familiar with Windows kernel API and C programming language.

Topics

Kernel Mode Debugging Tools

- Debugging Tools for Windows
- Collecting System Information
- Gflags for Kernel Debugging
- Performance Analysis Tools
- Driver Verifier
- Driver Verifier Logging

Kernel Architecture

- Kernel Mode Components
- System Service Dispatching
- Process Context
- Thread Context
- Exception Handling
- Trap Frames
- Task State Segment (TSS)
- Context Structures
- System Bug-Checks

Dump Generation

- Live Debugging
- Memory Dump Generation
- Memory Dump Types & Contents
- Hang vs. Crash Dumps
- Types of Crashes
- Memory Dump Navigation

Dump Analysis

- Common Analysis Steps
- Register Contexts
- Analyzing System State
- Identifying Faulting Modules
- Hardware Failures
- Access violations
- Assembly language
- Call Stacks

Kernel Mechanisms

- Processor Control Region (PCR)
- IRQLs
- Interrupt Service Routines (ISRs)
- Deferred Procedure Calls (DPCs)
- Asynchronous Procedure Calls (APCs)
- Intermittent CPU Spikes
- High CPU Usage
- System Threads
- Work Items

Memory Manager

- Kernel Virtual Address Space
- Dynamic Kernel Space Management
- SysPTE Depletion
- Kernel Stacks
- Stack Overflows and Double Faults
- Page Table Entries (PTEs)
- Page Frame Number (PFN) Database
- Memory Descriptor Lists (MDLs)
- Direct Memory Access (DMA) Issues
- Pools and Look-aside Lists
- Pool Corruption
- Pool Depletion

Kernel Synchronization

- Dispatcher Objects
- Fast Mutexes and Guarded Mutexes
- ERESOURCES
- Deadlocks
- Spin Locks
- Queued Spin Locks
- Livelocks

I/O Manager

- Driver Architecture & Entry Points
- I/O Manager Data Structures
- I/O Request Packet (IRP) Flow
- IRP Processing
- Synchronous and Asynchronous I/O Processing
- Completion Routines
- Stuck I/O Requests
- Cancel Routines
- I/O Cancellation Hangs

PnP & Power

- Driver, Device Types and Device Nodes
- Device Object Layering
- PnP IRPs and PnP State Transitions
- Device Enumeration
- Device Startup Failures
- Device Manager Error Codes
- System and Device and CPU Power States
- Power IRPs and Power State Transitions
- Idle Power Management
- Remote Wakeup
- Power Watchdog Timeouts

Developing and Debugging Windows KMDF Drivers

3 Days (Lecture and Hands-On Labs)

Target Audience

Driver Developers, Support Engineers and Software QA Engineers

Description

The Kernel Mode Driver Framework (KMDF) which is a part of Windows Driver Foundation (WDF) greatly simplifies development of kernel mode drivers. This course covers development and debugging of kernel mode drivers based on KMDF. Attendees will learn about the architecture of KMDF, debugging facilities provided by KMDF, KMDF objects, key data structures used by KMDF drivers, KMDF USB Drivers, KMDF Bus drivers and how to debug crashes and hangs caused by KMDF drivers.

In the hands-on labs attendees get an opportunity to develop, install, test and debug drivers that exercise the above interfaces on Windows 7 running inside a Virtual Machine as well as analyze kernel mode crash dumps that pertain to these technologies.

Pre-Requisites

Proficiency in "C" programming language

Familiarity with Windows kernel architecture and data structures

Topics

KMDF Architecture

KMDF Drivers
KMDF Library
KMDF Loader
KMDF Versions
KMDF Co-Installer

KMDF Debugging Tools

KMDF Symbols
Debugger Extension
In Flight Recorder (IFR) Log
WDF Verifier
KMDF Status Codes
KMDF Bug Checks

KMDF Objects

KMDF Objects
Object Hierarchy
Object Cleanup
Object Contexts
Object Handles
Execution Level
Synchronization Scope

KMDF Data Structures

Driver, Device, File, Queues, Requests
DPC, Timer, Work Item
Memory, Collections, String, Locks
Interrupt, Resource Lists
DMA Common Buffer
DMA Enabler & Transaction
I/O Targets

KMDF USB Drivers

USB Objects (Devices, Interfaces, Pipes)
USB Device Configuration
USB Requests
USB Continuous Reader
USB Power Management

KMDF Bus Drivers

KMDF Bus Drivers
Structure of a KMDF Bus Driver
Enumeration Models
Reporting Device Arrival and Removal
Debugging Child Lists
Bus Driver Testing and Debugging

Developing and Debugging Windows Network Drivers

3 Days (Lecture and Hands-On Labs)

Target Audience

Driver Developers, Support Engineers and Software QA Engineers

Description

This course covers the development and debugging of various Windows kernel mode network drivers. Attendees will learn about components of the new Windows networking stack (introduced in Windows Vista), kernel mode interfaces available for networking, NDIS6 miniport, protocol, intermediate and light weight filter drivers, kernel mode socket clients using TDI and WSK and transport layer filters using TDI and WFP.

In the hands-on labs attendees get an opportunity to develop, install, test and debug drivers that exercise the above interfaces on Windows 7 running inside a Virtual Machine as well as analyze kernel mode crash dumps that pertain to these technologies.

Pre-Requisites

Proficiency in "C" programming language

Familiarity with Windows kernel architecture and data structures

Topics

Networking Stack

Network Stack Components
Network Stack Interfaces
NDIS Driver Types
TCP/IP Components
RDBSS and Mini Redirectors, CSC
MPR & MUP
User Mode Networking Components
Networking Tools

NDIS Drivers

NDIS Concepts
Miniport Drivers
Protocol Drivers
Adapter Bindings
IM Drivers & Notify Objects
NDIS Internal Data Structures
OIDs & Direct OID Handlers
NDIS PnP Events

NDIS Data Transfers

NDIS Packets and NDIS Buffers
Net Buffers, Net Buffer Lists & NBL
Contexts
Receive Data Path
Send Data Path
Packet Filtering
Packet OOB Data
NDIS Loopback
NDIS Offload Capabilities

TDI

TDI Architecture
Address, Connection & Control Objects
Requests & Events
Clients & Servers
Connection Setup and Teardown
Data Transfers
TDI Issues
TDX Driver

WSK

WSK Architecture
Operational Model
Socket Types
API Model
Event Callbacks
Socket Options
Data Handling
TDI Interface Mapping

Windows Filtering Platform (WFP)

WFP Architecture
WFP Layers, Filters, Sub-layers and Callouts
WFP Registration
WFP Flow Contexts
WFP Traffic Processing
WFP Traffic Injection

Developing and Debugging Windows Kernel Filter Drivers

3 Days (Lecture and Hands-On Labs)

Target Audience

Driver Developers, Support Engineers and Software QA Engineers

Description

This course covers the development and debugging of drivers that take advantage of the various filtering and interception technologies provided by the windows kernel. Attendees will learn about SSDT hooking, network filtering at the packet level and at the transport level, file system filtering, registry filtering, process/thread creation and access interception and DLL/driver load interception.

In the hands-on labs attendees get an opportunity to develop, install, test and debug drivers that exercise the above interfaces on Windows 7 running inside a Virtual Machine as well as analyze kernel mode crash dumps that pertain to these technologies.

Pre-Requisites

Proficiency in "C" programming language

Familiarity with Windows kernel architecture and data structures

Topics

Legacy Hooking

- SSDT Hooking
- Driver Dispatch Table Hooking
- Kernel Data Structure Hooking
- Import Table Hooking
- Detour Hooking

NDIS Light Weight Filters

- NDIS 6.0 Architecture
- Filter Driver Types
- Filter Driver States
- Net Buffer Lists (NBL)
- Net Buffers (NB)
- Data Transfer Operations
- OID Handling
- Filter Driver Installation

OS Supported Hooking

- x64 Kernel Patch Protection
- Object Access Interception
- Process and Thread Interception
- Module Load Interception

File System Mini-Filters

- Filter Manager Architecture
- Context Management
- Name Management
- I/O Operation Processing
- Handling Create Operations
- File Sharing and Attributes
- Handling Renames and Deletes
- Transactions

Registry Filters

- Registry Filtering Model
- Registry Key Contexts
- Registry Path Processing
- Registry Access Monitoring & Modification

Windows Filtering Platform (WFP)

- WFP Architecture
- WFP Layers, Filters, Sub-layers and Callouts
- WFP Registration
- WFP Flow Contexts
- WFP Traffic Processing
- WFP Traffic Injection

Developing and Debugging Windows Driver Model Drivers

3 Days (Lecture and Hands-On Labs)

Target Audience

Driver Developers, Support Engineers and Software QA Engineers

Description

This course covers the development and debugging of drivers using the native kernel API i.e. the Windows Driver Model (WDM). Attendees will learn about authoring and validating .INF files to install WDM drivers, synchronous and asynchronous IRP processing, building and dispatching IRPs, participating in PnP and Power state transitions and interfacing with PCI hardware devices.

In the hands-on labs attendees get an opportunity to develop, install, test and debug drivers that exercise the above interfaces on Windows 7 running inside a Virtual Machine as well as analyze kernel mode crash dumps that pertain to these technologies.

Pre-Requisites

Proficiency in "C" programming language

Familiarity with Windows kernel architecture and data structures

Topics**Installing Drivers**

- .INF Files
- Driver Installation
- Class and Interface GUIDs
- Driver Signing
- Driver Ranking
- Setup API
- Plug and Play Notifications
- Driver Installation Tools

I/O Request Processing

- Win32 I/O API
- Dispatch I/O Routines
- Types of I/O Transfers
- Synchronous I/O Processing
- Asynchronous I/O Processing
- IRP Queuing Mechanisms
- I/O Cancellation
- Cleanup Routines

Driver Layering

- Attaching to Device Objects
- I/O Request Forwarding
- I/O Completion Routines
- Pre & Post Processing Requests
- Building I/O Requests

Plug and Play

- Device Enumeration
- PnP Device States
- PnP IDs
- Handling Device Arrival
- Handling Resource Assignment
- Handling Query and Cancel IRPs
- Handling Resource Rebalancing
- Handling Device Removal

Power Management

- System & Device Power States
- Power Management Support Routines
- Handling System & Device Power State Changes
- Idle State Power Management
- Handling Remote Wakeup

Hardware Interfacing

- Hardware Resources
- I/O Ports
- Adapter Memory
- Interrupt Handling
- Deferred Procedure Calls
- Timers
- Work Items
- Bus Master DMA