

ITMC 2074

Course Syllabus

1. Name of Course: MCTS Windows Internals: Advanced Troubleshooting and Debugging

2. Number of Clock Hours: 48

3. Course Description:

This course provides students with the knowledge and skills to debug and troubleshoot low-level Windows applications. The course provides a detailed view of the Windows operating system architecture, its key components, undocumented interfaces and core interactions through the use of debugging utilities, inspection tools and exercises. MOC 50155A.

Prerequisites: MCSA/MCITP or 2 years experience with Windows 2003-2008 Platform.

4. Course Objectives

After completing this course, students will gain the skills to:

- Use debugging tools to troubleshoot user-mode and kernel-mode issues.
- Appreciate the relationships between system components and their interactions with each other.
- Improve application performance by understanding how the system behaves internally.
- Take the MCTS: Windows Internals examination #70-660.

5. Rationale:

Upon completion of this course, students will have a better understanding of Microsoft Windows Internals, the most popular operating system in use today.

6. Required Materials:

Windows® Internals, Fifth Edition, by Mark E. Russinovich and David A. Solomon with Alex Ionescu, Microsoft Press, ISBN 13: 9780735625303. Available at Amazon, Barnes & Noble, Borders

7. Evaluation

Those who participate in class discussions, complete course lab work, and miss no more than three class meetings will be awarded 5.6 continuing education units.

8. Course Outline

Module 1: Computer Architecture Refreshment

This module explains how the modern computer is constructed and how programs execute on this computer architecture.

Lessons

- Computer Architecture: CPU, Registers, Cache, Memory, Input and Output Devices
- Just Enough Assembly Language to Get By
- Program Execution Model: Compilation, Linkage, Loading, Execution

After completing this module, students will be able to:

- Explain the various components of which a modern computer is comprised.
- Describe the execution model of a program.

Module 2: C Programming Basics

This module explains how to develop programs in the C programming language and exposes concepts such as variables, data structures, stack vs. heap allocation and function execution.
Lessons

- Variables and Data Structures
- Functions and Stack: Calling Functions
- Dynamic Memory Allocation

Lab : Computer Architecture and C Programming

- Calling functions
- Tracing through assembly code

After completing this module, students will be able to:

- Analyze and explain simple programs in the C programming language.
- Understand the concept of complex data structures, dynamic memory allocation and function execution.
- Read basic procedures in x86 assembly language.

Module 3: Windows History and Characteristics

This module explains how to approach the Windows operating system using various APIs including Win32, and the motivation, history and design goals of Windows.
Lessons

- The Evolution of Windows: Windows 1.0 through Windows Vista
- Back to 1989: Windows NT Design Goals
- Windows Versions and Editions
- Windows Programming Interfaces: Win32, COM, .NET

After completing this module, students will be able to:

- Explain the primary differences between the various Windows versions and product editions.
- Appreciate the design goals of Windows NT as they are threaded into the Windows operating system.
- Understand the pros and cons of Windows programming interfaces.

Module 4: Win32 Programming Primer

This module explains in brief how the operating system implements windows, message passing between windows, kernel objects and handles, processes and threads and synchronization mechanism. This module serves as an exposition to provide a common ground for examining system internals.

Lessons

- Introduction to Win32 API Paradigms
- Windows and Window Procedures
- Message Loops and Window Messages
- Kernel Objects and Handles
- Creating, Using and Sharing Objects
- Processes and Threads
- Synchronization Mechanisms
- Structured Exception Handling
- An Introduction to NT File System (NTFS)
- Dynamic Link Libraries (DLLs) and Loading
- Inter-Process Communication

Lab : Basic Win32 Concepts

- Theoretical questions about Win32 basic concepts

After completing this module, students will be able to:

- Describe Win32 APIs and understand how to properly use them.
- Understand basic concepts in the Win32 programming model.

Module 5: Tools

This module explains how to use troubleshooting, inspection and debugging tools to examine the internals of the operating system and troubleshoot Windows applications. This module exposes the tools only briefly – crash analysis is covered in Modules 15-16.

Lessons

- Sysinternals Tools – Process Explorer, Process Monitor
- Resource Kit and Support Kit Tools
- Debugging Tools for Windows
- Understanding Debugging Symbols
- Kernel Debuggers

Lab : Symbols, Process Explorer and WinDbg

- Setting up debugging symbols using environment variables
- Configuring symbols in Process Explorer and inspecting thread stacks
- First steps with WinDbg local kernel debugging

Lab : Virtual Kernel Debugging (Optional)

- Setting up kernel debugging with a virtual machine as the target

After completing this module, students will be able to:

- Use troubleshooting tools to understand why applications misbehave.
- Inspect the system's inner workings through a kernel debugger.
- Configure debugging-related behavior for user-mode applications.

Module 6: System Architecture

This module explains how to classify Windows system components and processes, how hard-wired system processes interact and what architectural decisions were made in the design of Windows.

Lessons

- Kernel-Mode vs. User-Mode Execution
- Primary System Components
- Types of User-Mode Processes
- Hard-Wired System Processes

After completing this module, students will be able to:

- Explain why the operating system and CPU provide the distinction between kernel-mode and user-mode execution privilege rings.
- Appreciate the relationships between the primary system components.
- Understand the purpose of each user-mode process type including system services, environment subsystems and standard applications.
- Understand the importance of each hard-wired system processes and their relationship hierarchy.

Module 7: Startup and Shutdown

This module explains how to troubleshoot Windows startup and shutdown and what components are involved in the startup and shutdown process.

Lessons

- Windows Startup – from Setup to Shell
- Windows Shutdown – Applications, Services and Drivers

After completing this module, students will be able to:

- Troubleshoot the Windows startup process, including boot configuration, startup processes, last known good configuration and safe mode.
- Troubleshoot the Windows shutdown process, including unresponsive UI applications, unresponsive console applications and services and unresponsive drivers.

Module 8: System Mechanisms

This module explains how to use exception handling, interrupt dispatching, deferred procedure

calls, asynchronous procedure calls and other kernel mechanisms properly.

Lessons

- Trap and Interrupt Dispatching
- Interrupt Request Levels (IRQL)
- Exception Handling in User-Mode and Kernel-Mode
- Deferred Procedure Calls (DPC)
- Asynchronous Procedure Calls (APC) in User-Mode and Kernel-Mode
- The LPC Facility
- System Worker Threads

Lab : Understanding System Mechanisms

- Examining interrupt processing using the kernel profiler (kernrate.exe)
- Using the Performance Monitor system performance data collector set to obtain statistics on DPCs, interrupts, network activity, file system access and memory usage
- Using the Reliability Monitor to analyze system stability, driver installations, application failures and Windows failures

After completing this module, students will be able to:

- Understand the various types of structured exception handling and when each should be used.
- Understand the implications of the IRQL on system stability and responsiveness.
- Use APCs for delayed asynchronous processing in user-mode applications.
- Inspect the behavior of various system mechanisms on a live system.

Module 9: Process and Thread Internals

This module explains how to create operating system processes and threads and how to inspect their internal data structures in a kernel debugger.

Lessons

- Process Data Structures: Kernel, Executive, Environment Subsystem
- Process Creation Flow
- Thread Data Structures: Kernel, Executive, Environment Subsystem
- Thread Creation Flow

Lab : Examining Processes and Threads

- Determining whether a thread is a UI thread or not using the system service table pointer in the thread's kernel data structure

After completing this module, students will be able to:

- Understand the flow of process and thread creation.
- Inspect the behavior of threads and processes on a system using a kernel debugger.

Module 10: Thread Scheduling

This module explains how to properly interact with the operating system with regard to thread creation, thread priorities and multiprocessing on a multi-processor system. This module focuses on priority boosts, quantum boosts, thread processor affinity and other specific optimizations for multi-processor scheduling.

Lessons

- Thread Scheduling in a Preemptive Multitasking Operating System
- Thread Execution States and Transitions
- Dispatcher Database
- Quantum Length, Tuning and Boosts
- Thread Priority and Priority Boosts
- Thread Scheduling on a Multi-Processor System
- Thread CPU Affinity and Non-Uniform Memory Access (NUMA)

After completing this module, students will be able to:

- Tune the system by changing the foreground quantum boost, the quantum length and other parameters.
- Schedule threads for execution on multi-processor system and carefully assign CPU affinities to threads if necessary.
- Understand the implications of multi-processor CPU scheduling and NUMA considerations.

Module 11: Synchronization Mechanisms

This module explains how to use synchronization mechanisms to control thread execution and access to shared resources, and how to diagnose deadlocks and other problems arising from the use of synchronization mechanisms.

Lessons

- Concurrency and the Need for Synchronization
- Kernel Synchronization: IRQL, Spinlocks and Queued Spinlocks
- Executive Synchronization: Dispatcher Objects and Wait Blocks
- Waiting for and Signaling Dispatcher Objects
- Tracing Dispatcher Objects and Wait Chain Traversal

After completing this module, students will be able to:

- Choose the most appropriate synchronization mechanism for the task.
- Appreciate how dispatcher objects become signaled.
- Trace through dispatcher objects and wait blocks in a kernel debugger.
- Use Wait Chain Traversal to diagnose deadlocks and wait chains in user-mode applications.

Module 12: The Memory Manager

This module explains how to interact with the Windows Memory Manager in order to allocate, reserve and commit memory, share memory between processes, protect memory and lock

pages into physical memory.

Lessons

- Virtual Memory and Paging
- Allocating Memory: Reserve, Commit, Heap
- Address Space Layout: User-Mode and Kernel-Mode , 32-bit and 64-bit
- Virtual Address Translation, Translation Look-Aside Buffer (TLB)
- Protecting Memory
- Locking Pages into Memory, Address Windowing Extensions (AWE)
- System Memory Pools and Look-Aside Lists
- Working Set Management: Fetch, Placement and Replacement Policies
- Page Frame Number (PFN) Database

After completing this module, students will be able to:

- Allocate memory using the most appropriate mechanism for the task, including reserving and committing memory on demand.
- Share memory between processes and protect memory from being read, written or executed improperly.
- Allocate physical memory directly to overcome virtual memory limitations on 32-bit systems.
- Tune working set management for applications to achieve less paging and better performance.

Module 13: The I/O System

This module explains how to examine the various components comprising the Windows I/O system and briefly reviews the basic infrastructure of Windows device drivers, the power manager and the Plug-and-Play (PNP) manager.

Lessons

- The I/O Manager, Power Manager and Plug-and-Play Manager
- Device Driver Structure
- I/O Data Structures: File Objects, Driver Object, Device Object
- I/O Flow: I/O Request Packets
- A Glimpse Towards Windows Driver Foundation (WDF)

After completing this module, students will be able to:

- Examine I/O data structures to identify drivers, devices and files.
- Trace the flow of I/O request packets through the I/O system.
- Appreciate the role of power management and Plug-and-Play in the Windows I/O system.

Module 14: The Cache Manager

This module explains how to use the Windows file system cache manager to achieve better performance for applications which rely heavily on the file system.

Lessons

- Operating System File Caching vs. CPU Caching
- Cache Structure: Cache Control Blocks, Private Control Blocks
- Cache Operation: Read and Write, Fast I/O
- Controlling the Cache Manager: Hints to CreateFile

After completing this module, students will be able to:

- Provide hints to the cache manager to achieve better performance for applications which rely on the file system.
- Understand the structure and operation of the file system cache and the fast I/O path.

Module 15: User-Mode Debugging

This module explains how to debug live user-mode applications and examine crash dumps or hang dumps obtained from user-mode applications.

Lessons

- Understanding the Role of Debugging Symbols
- Generating Dump Files: Crash and Hang Scenarios
- Debugging Application Crashes
- Debugging Application Hangs and Deadlocks

Lab : User-Mode Dumps and Debugging

- Analyzing a simple crash in an application by generating and opening a crash dump
- Detecting a deadlock condition between critical sections in the debugger
- Pinpointing a problem with file system and registry access by an application
- Predicting compatibility issues with future versions of Windows (Application Compatibility Toolkit)
- Tracing through a handle misuse scenario

After completing this module, students will be able to:

- Obtain dumps of running processes for later analysis in hang or crash scenarios.
- Debug common issues such as crashes, hangs, deadlocks and invalid handles.

Module 16: Kernel-Mode Debugging

This module explains how to debug kernel-mode dumps obtained from a crashed or hung Windows system.

Lessons

- Blue Screen of Death: When Does the Operating System Crash?
- Manually Obtaining a Dump of the System
- Debugging Crashes
- Using Driver Verifier to Pinpoint Faulting Drivers

Lab : Kernel-Mode Dumps

- Opening a minidump and finding the faulty driver
- Crashing the system and finding the faulty driver using Driver Verifier to reproduce the problem

After completing this module, students will be able to:

- Obtain dumps of the entire system for later analysis.
- Debug common crashes and understand the reasons for Windows crashes.
- Use Driver Verifier to detect faulty drivers.

Module 17: What's Different in 64-Bit Windows?

This module explains how to port 32-bit applications to 64-bit Windows, how to execute 32-bit and 64-bit applications side-by-side and how to take advantage of the 64-bit Intel/AMD processor architecture.

Lessons

- 64-Bit Processor Architecture (AMD64)
- Windows on Windows 64 (WOW64) Architecture
- File System and Registry Redirection and Virtualization
- Performance Improvements on 64-Bit Windows

After completing this module, students will be able to:

- Port existing 32-bit applications to 64-bit Windows or execute them natively using WOW64.
- Troubleshoot common issues with 32-bit applications on 64-bit Windows.

Additional Reading

To help you prepare for this class, review the following resources:

- Andrew S. Tanenbaum, "Structured Computer Organization," Prentice Hall (1998)
- Jeffrey Richter, "Programming Applications for Microsoft Windows," Microsoft Press (1999)
- Jeffrey Richter, "Windows via C/C++," Microsoft Press (2007)

Before attending this course, students must have:

- Familiarity with the Windows operating system on a power-user level.
- Familiarity with operating systems concepts (multiprocessing, thread scheduling, virtual memory) is preferable but not a must.
- Working knowledge of the C or C++ programming language – exempt from Modules 1-2 of the course.
- Proficiency in Win32 programming – exempt from Modules 3-4 of the course.