Contents

Foreword . . . . . *xv*
Introduction . . . . . *xvii*
Who Should Read This Book xviii
Assumptions xviii
Organization of This Book xviii
Conventions in This Book xix
System Requirements xx
Code Samples xxi
Installing the Code Samples xxi
Running the Code Samples xxii
Acknowledgments xxvi
Errata & Book Support xxvii
We Want to Hear from You xxvii
Stay in Touch xxviii

PART I A BIT OF BACKGROUND

Chapter 1 Software Development in Windows 3

Windows Evolution 3
Windows Release History 3
Supported CPU Architectures 4
Windows Build Flavors 5
Windows Servicing Terminology 6
Windows Architecture 7
Kernel Mode vs. User Mode 8
User-Mode System Processes 9
User-Mode Application Processes 10
Low-Level Windows Communication Mechanisms 13
Windows Developer Interface 16

Developer Documentation Resources 16
WDM, KMDF, and UMDF 17
The NTDLL and USER32 Layers 18
The Win32 API Layer 18
The COM Layer 19
The CLR (.NET) Layer 25
Microsoft Developer Tools 28
The Windows DDK (WDK) 29
The Windows SDK 29
Summary 30

PART II DEBUGGING FOR FUN AND PROFIT

Chapter 2 Getting Started 33

Introducing the Debugging Tools 34
Acquiring the Windows Debuggers Package 34
Acquiring the Visual Studio Debugger 38
Comparing the WinDbg and Visual Studio Debuggers 38
User-Mode Debugging 39
Debugging Your First Program with WinDbg 39
Listing the Values of Local Variables and Function Parameters 47
Source-Level Debugging in WinDbg 52
Symbol Files, Servers, and Local Caches 53
Caching Symbols Offline for WinDbg 55
Troubleshooting Symbol Resolution Issues in WinDbg 56
Name Decoration Considerations 57
Getting Help for WinDbg Commands 58
Kernel-Mode Debugging 60
Your First (Live) Kernel Debugging Session 61
Setting Up a Kernel-Mode Debugging Environment Using Physical Machines 67
Setting Up a Kernel-Mode Debugging Environment Using Virtual Machines 73

Diagnosing Host/Target Communication Issues 76
Understanding the KD Break-in Sequence 77
Controlling the Target in the Kernel Debugger 78
Setting Code Breakpoints in the Kernel Debugger 81
Getting Help for WinDbg Kernel Debugging Commands 83
Summary 83

Chapter 3 How Windows Debuggers Work 85

User-Mode Debugging 85
Architecture Overview 86
Win32 Debugging APIs 87
Debug Events and Exceptions 88
The Break-in Sequence 91
Setting Code Breakpoints 93
Observing Code Breakpoint Insertion in WinDbg 93
Kernel-Mode Debugging 98
Architecture Overview 98
Setting Code Breakpoints 100
Single-Stepping the Target 100
Switching the Current Process Context 101
Managed-Code Debugging 103
Architecture Overview 103
The SOS Windows Debuggers Extension 106
Script Debugging 112
Architecture Overview 112
Debugging Scripts in Visual Studio 114
Remote Debugging 116
Architecture Overview 116
Remote Debugging in WinDbg 117
Remote Debugging in Visual Studio 121
Summary 123

Chapter 4 Postmortem Debugging 125

Just-in-Time Debugging 125
Your First JIT Debugging Experiment 126
How Just-in-Time Debugging Works 128
Using Visual Studio as Your JIT Debugger 132
Run-Time Assertions and JIT Debugging 138
JIT Debugging in Session 0 139

Dump Debugging 139
Automatic User-Mode, Crash-Dump Generation 139
Analyzing Crash Dumps Using the WinDbg Debugger 143
Analyzing Crash Dumps in Visual Studio 150
Manual Dump-File Generation 151
"Time Travel" Debugging 153
Kernel-Mode Postmortem Debugging 153

Summary 157

Chapter 5 Beyond the Basics 159

Noninvasive Debugging 159

Data Breakpoints 162
Deep Inside User-Mode and Kernel-Mode Data Breakpoints 163
Clearing Kernel-Mode Data Breakpoints 165
Execution Data Breakpoints vs. Code Breakpoints 166
User-Mode Debugger Data Breakpoints in Action: C++ Global Objects and the C Runtime Library 168
Kernel-Mode Debugger Data Breakpoints in Action: Waiting for a Process to Exit 170
Advanced Example: Who Is Changing a Registry Value? 172

Scripting the Debugger 176
Replaying Commands Using Debugger Scripts 176
Debugger Pseudo-Registers 178
Resolving C++ Template Names in Debugger Scripts 180
Scripts in Action: Listing Windows Service Processes in the Kernel Debugger 181

WOW64 Debugging 183
The WOW64 Environment 183
Debugging of WOW64 Processes 184
Windows Debugging Hooks (GFLAGS) 187
Systemwide vs. Process-Specific NT Global Flags 187
The GFLAGS Tool 188
The *!gflag* Debugger Extension Command 191
Impact of User-Mode Debuggers on the Value of the NT Global Flag 193
The Image File Execution Options Hooks 193
Summary 194

Chapter 6 Code Analysis Tools 195

Static Code Analysis 195
Catching Your First Crashing Bug Using VC++ Static Code Analysis 196
SAL Annotations 199
Other Static Analysis Tools 202
Runtime Code Analysis 206
Catching Your First Bug Using the Application Verifier Tool 206
A Behind-the-Scenes Look: Verifier Support in the Operating System 209
The *!avrf* Debugger Extension Command 214
The Application Verifier as a Quality Assurance Tool 217
Summary 217

Chapter 7 Expert Debugging Tricks 219

Essential Tricks 220
Waiting for a Debugger to Attach to the Target 220
Breaking on DLL Load Events 222
Debugging Process Startup 227
Debugging Child Processes 234

More Useful Tricks 245
Debugging Error-Code Failures 245
Breaking on First-Chance Exception Notifications 252
Freezing Threads 253
Kernel-Mode Debugging Tricks 255
Breaking on User-Mode Process Creation 255
Debugging the Startup of User-Mode Processes 259
Breaking on DLL Load Events 260
Breaking on Unhandled SEH Exceptions 262
Freezing Threads 262
Summary 265

Chapter 8 Common Debugging Scenarios, Part 1 267

Debugging Access Violations 267
Understanding Memory Access Violations 268
The *!analyze* Debugger Extension Command 269
Debugging Heap Corruptions 271
Debugging Native Heap Corruptions 271
Debugging Managed (GC) Heap Corruptions 281
Debugging Stack Corruptions 291
Stack-Based Buffer Overruns 291
Using Data Breakpoints in Stack Corruption Investigations 294
Reconstructing Call Frames from Corrupted Stacks 295
Debugging Stack Overflows 297
Understanding Stack Overflows 297
The *kf* Debugger Command 299
Debugging Handle Leaks 300
A Handle Leak Example 300
The *!htrace* Debugger Extension Command 302
Debugging User-Mode Memory Leaks 307
Detecting Resource Leaks Using the Application Verifier Tool . . . 307
Investigating Memory Leaks Using the UMDH Tool 310
Extending the Strategy:
A Custom Reference Stack-Trace Database 314

Debugging Kernel-Mode Memory Leaks 316
Kernel Memory Basics 316
Investigating Kernel-Mode Leaks Using Pool Tagging 318
Summary 322

Chapter 9 Common Debugging Scenarios, Part 2 323

Debugging Race Conditions 323
Shared-State Consistency Bugs 324
Shared-State Lifetime Management Bugs 330
DLL Module Lifetime-Management Bugs 340
Debugging Deadlocks 343
Lock-Ordering Deadlocks 344
Logical Deadlocks 348
Debugging Access-Check Problems 352
The Basic NT Security Model 353
Windows Vista Improvements 358
Wrapping Up 362
Summary 363

Chapter 10 Debugging System Internals 365

The Windows Console Subsystem 365
The Magic Behind *printf* *366*
Handling of Windows UI Events 373
Handling of the Ctrl+C Signal 374
Anatomy of System Calls 380
The User-Mode Side of System Calls 381
The Transition into Kernel Mode 383
The Kernel-Mode Side of System Calls 385
Summary 387

PART III OBSERVING AND ANALYZING SOFTWARE BEHAVIOR

Chapter 11 Introducing Xperf 391

Acquiring Xperf 391

Your First Xperf Investigation 396

Devising an Investigation Strategy 397

Collecting an ETW Trace for the Scenario 397

Analyzing the Collected ETW Trace 399

Xperf's Strengths and Limitations 411

Summary 412

Chapter 12 Inside ETW 415

ETW Architecture 416

ETW Design Principles 416

ETW Components 417

The Special NT Kernel Logger Session 418

Configuring ETW Sessions Using Xperf 419

Existing ETW Instrumentation in Windows 422

Instrumentation in the Windows Kernel 422

Instrumentation in Other Windows Components 426

Understanding ETW Stack-Walk Events 431

Enabling and Viewing Stack Traces for Kernel Provider Events . . . 432

Enabling and Viewing Stack Traces for User Provider Events 434

Diagnosing ETW Stack-Trace Issues 436

Adding ETW Logging to Your Code 441

Anatomy of ETW Events 441

Logging Events Using the ETW Win32 APIs 445

Boot Tracing in ETW 449

Logging Kernel Provider Events During Boot 450

Logging User Provider Events During Boot 452

Summary 455

Chapter 13 Common Tracing Scenarios 457

Analyzing Blocked Time 458
The *CSwitch* and *ReadyThread* ETW Events 459
Wait Analysis Using Visual Studio 2010 461
Wait Analysis Using Xperf 467
Analyzing Memory Usage 473
Analyzing High-Level Memory Usage in a Target Process 474
Analyzing NT Heap Memory Usage 475
Analyzing GC Heap (.NET) Memory Usage 481
Tracing as a Debugging Aid 490
Tracing Error Code Failures 490
Tracing System Internals 494
Summary 502

Chapter 14 WinDbg User-Mode Debugging Quick Start 505

Starting a User-Mode Debugging Session 505
Fixing the Symbols Path 505
Fixing the Sources Path 506
Displaying the Command Line of the Target Process 507
Control Flow Commands 507
Listing Loaded Modules and Their Version 508
Resolving Function Addresses 509
Setting Code (Software) Breakpoints 509
Setting Data (Hardware) Breakpoints 510
Switching Between Threads 511
Displaying Call Stacks 511
Displaying Function Parameters 512
Displaying Local Variables 513
Displaying Data Members of Native Types 513

Navigating Between Call Frames 514
Listing Function Disassembly 515
Displaying and Modifying Memory and Register Values 516
Ending a User-Mode Debugging Session. 518

Chapter 15 WinDbg Kernel-Mode Debugging Quick Start 519

Starting a Kernel-Mode Debugging Session 519
Switching Between CPU Contexts 519
Displaying Process Information 520
Displaying Thread Information. 521
Switching Process and Thread Contexts 522
Listing Loaded Modules and Their Version 523
Setting Code (Software) Breakpoints Inside Kernel-Mode Code 524
Setting Code (Software) Breakpoints Inside User-Mode Code. 525
Setting Data (Hardware) Breakpoints 525
Ending a Kernel-Mode Debugging Session 526

Index *527*

About the Author *561*