

Inhaltsverzeichnis

Einleitung	XV
Die Windows-Plattformen von Heute	XVI
Der Kernel von Windows 2000	XVI
Der Kernel von Windows 98	XVII
Der Kernel von Windows CE	XVII
Die Windows-Plattformen von Morgen (64-Bit-Windows 2000)	XVIII
Neues in dieser vierten Auflage	XIX
Dieses Buch ist fehlerfrei	XXI
Informationen zu Begleit-CD und Systemanforderungen	XXI
Korrekturen, Kommentare und Hilfe	XXII
Danksagung	XXII

Teil I

Pflichtlektüre	1
-----------------------------	----------

1 Fehlerbehandlung	3
---------------------------------	----------

Definition eigener Fehlercodes	7
Die Beispielanwendung <i>ErrorShow</i>	8

2 Unicode	15
------------------------	-----------

Zeichensätze	16
Einzelbyte- und Doppelbyte-Zeichensätze	16
Unicode: Der 16-Bit-Zeichensatz	17
Gründe für die Verwendung von Unicode	18
Windows 2000 und Unicode	18
Windows 98 und Unicode	18
Windows CE und Unicode	19
Zusammenfassung	20
Anmerkung zu COM	20
Einbindung von Unicode in eigene Programme	20
Unterstützung von Unicode in der C-Laufzeitbibliothek	21
In Windows definierte Unicode-Datentypen	23
Unicode- und ANSI-Funktionen unter Windows	24
String-Funktionen von Windows	25
Anwendungen für ANSI und Unicode entwickeln	26
Unicode-String-Funktionen von Windows	27
Ressourcen	30
Textdateien	30
Umwandlung zwischen Unicode und ANSI	31

3	Kernel-Objekte	37
	Was ist ein Kernel-Objekt?	37
	Zugriffszähler	38
	Sicherheit	38
	Die Handle-Tabelle der Kernel-Objekte eines Prozesses	41
	Anlegen eines Kernel-Objekts	41
	Schließen eines Kernel-Objekts	43
	Gemeinsame Verwendung von Kernel Objekten über Prozessgrenzen hinweg	44
	Vererbung von Objekt-Handles	44
	Benannte Objekte	49
	Duplizieren von Objekt-Handles	53

Teil II

Anwendungen 59

4	Prozesse	61
	Die erste eigene Windows-Anwendung	62
	Der Instanz-Handle eines Prozesses	66
	Der Handle der vorhergehenden Instanz eines Prozesses	68
	Die Befehlszeile eines Prozesses	68
	Die Umgebungsvariablen eines Prozesses	69
	Die Affinität eines Prozesses	73
	Der Fehlermodus eines Prozesses	73
	Das aktuelle Laufwerk und Verzeichnis eines Prozesses	74
	Die Betriebssystemversion	76
	Die Funktion <i>CreateProcess</i>	79
	<i>pszApplicationName</i> und <i>pszCommandLine</i>	80
	<i>psaProcess</i> , <i>psaThread</i> und <i>bInheritHandles</i>	82
	<i>fdwCreate</i>	84
	<i>pvEnvironment</i>	86
	<i>pszCurDir</i>	86
	<i>psiStartInfo</i>	87
	<i>ppiProcInfo</i>	91
	Beenden eines Prozesses	93
	Rückkehr der Startfunktion des primären Threads	93
	Die Funktion <i>ExitProcess</i>	93
	Die Funktion <i>TerminateProcess</i>	95
	Sämtliche Threads im Prozess enden	96
	Das Ende eines Prozesses	96
	Untergeordnete Prozesse	97
	Ausführung eigenständiger untergeordneter Prozesse	99
	Auflistung der im System laufenden Prozesse	99
	Die Beispielanwendung <i>ProcessInfo</i>	100

5	Auftragsobjekte	117
	Festlegen der Beschränkungen für die Prozesse eines Auftrags	120
	Einbinden eines Prozesses in einen Auftrag	128
	Terminieren aller Prozesse innerhalb eines Auftrags	129
	Abrufen von Auftragstatistiken	129
	Auftragsbenachrichtigungen	132
	Die Beispielanwendung <i>JobLab</i>	135
6	Grundlagen der Thread-Programmierung	155
	Situationen, die für den Einsatz von Threads sprechen	156
	Situationen, in denen der Einsatz von Threads nachteilig ist	158
	Die erste eigene Thread-Funktion	159
	Die Funktion <i>CreateThread</i>	160
	<i>psa</i>	161
	<i>cbStack</i>	161
	<i>pfnStartAddr</i> und <i>pvParam</i>	162
	<i>fdwCreate</i>	163
	<i>pdwThreadID</i>	163
	Beenden eines Threads	164
	Normale Beendigung der Thread-Funktion	164
	Die Funktion <i>ExitThread</i>	164
	Die Funktion <i>TerminateThread</i>	165
	Vorgänge beim Beenden eines Prozesses	166
	Vorgänge beim Beenden eines Threads	166
	Interna der Thread-Aktivierung	167
	Anmerkungen zur C/C++-Laufzeitbibliothek	170
	Zu den Folgen einer Verwendung von <i>CreateThread</i> an Stelle	
	von <i>_beginthreadex</i>	179
	Zwei C/C++-Laufzeitbibliotheksfunktionen, die Sie nicht verwenden sollten	179
	Zur Identität von Prozessen und Threads	180
	Umwandeln eines Pseudohandles in einen echten Handle	181
7	Thread-Ablaufsteuerung, Prioritäten und Affinitäten	185
	Anhalten und Fortsetzen von Threads	187
	Anhalten und Fortsetzen von Prozessen	188
	Zeitlich begrenztes Unterbrechen	190
	Temporärer Wechsel zu einem anderen Thread	190
	Ausführungszeiten eines Threads	191
	Klarstellungen zum Thread-Kontext	194
	Thread-Prioritäten	199
	Ein abstrakte Darstellung der Prioritäten	200
	Programmieren von Prioritäten	204
	Dynamische Anhebung der Prioritätsstufe eines Threads	207
	Beeinflussen der Scheduler-Funktion für Vordergrundprozesse	209
	Die Beispielanwendung <i>Scheduling Lab</i>	210
	Affinitäten	217

8 Thread-Synchronisation im Benutzermodus	225
Atomarer Variablenzugriff unter Verwendung der Interlocked-Funktionen	226
Cache-Zeilen	232
Erweiterte Methoden der Thread-Synchronisation	234
Eine Synchronisationstechnik, die vermieden werden sollte	235
Kritische Abschnitte	236
Zur Feinmechanik kritischer Abschnitte	240
Kritische Abschnitte und Spinlocks	243
Kritische Abschnitte und Fehlerbehebung	244
Nützliche Tipps und Techniken	245
9 Thread-Synchronisation mit Kernel-Objekten	249
<i>Wait-Funktionen</i>	252
Nebeneffekte erfolgreichen Wartens	254
Ereignisobjekte	256
Die Beispielanwendung <i>Handshake</i>	260
Kernel-Objekte für programmierbare Zeitgeber	267
Verknüpfung von programmierbaren Zeitgebern mit APC-Einträgen	270
Schwachstellen von programmierbaren Zeitgebern	272
Semaphorobjekte	274
Mutex-Objekte	276
Herrenlose Mutex-Objekte	278
Mutex-Objekte und kritische Abschnitte im Vergleich	279
Die Beispielanwendung <i>Queue</i>	280
Tabellarische Übersicht der Objekte für die Thread-Synchronisation	289
Weitere Funktionen für die Thread-Synchronisation	290
Asynchrone Ein-/Ausgabe bei Geräten	290
<i>WaitForInputIdle</i>	291
<i>MsgWaitForMultipleObjects(Ex)</i>	292
<i>WaitForDebugEvent</i>	292
<i>SignalObjectAndWait</i>	293
10 Toolkit zur Thread-Synchronisation	295
Implementierung eines kritischen Abschnitts: Das Optex-Objekt	295
Die Beispielanwendung <i>Optex</i>	298
Erstellen thread-sicherer Datentypen und inverser Semaphoren	308
Die Beispielanwendung <i>InterlockedType</i>	312
SWMRG (Single Writer/Multiple Reader Guard)	320
Die Beispielanwendung <i>SWMRG</i>	322
Implementierung einer <i>WaitForMultipleExpressions</i> -Funktion	329
Die Beispielanwendung <i>WaitForMultipleExpressions</i>	331
11 Thread-Pooling	345
Szenario 1: Asynchroner Aufruf von Funktionen	347
Szenario 2: Aufruf von Funktionen in zeitgesteuerten Intervallen	349
Die Beispielanwendung <i>TimedMsgBox</i>	353
Szenario 3: Funktionen aufrufen, wenn einzelne Kernel-Objekte signalisiert werden	356
Szenario 4: Aufruf von Funktionen, wenn asynchrone E/A-Anforderungen abgeschlossen sind	359

12 Pseudo-Threads	361
Verwendung von Pseudo-Threads	361
Die Beispielanwendung <i>Counter</i>	364
Teil III	
Speicherverwaltung	373
13 Die Speicherverwaltung von Windows	375
Der virtuelle Adressraum eines Prozesses	375
Wie Windows den virtuellen Adressraum partitioniert	376
Die Partition für NULL-Zeigerzuweisungen (Windows 2000 und Windows 98) ...	377
Die Partition für die Kompatibilität mit MS-DOS/16-Bit-Windows-Anwendungen (nur Windows 98)	378
Benutzermodus-Partition (Windows 2000 und Windows 98)	378
Die geschützte 64-KByte-Partition (nur Windows 2000)	380
Gemeinsam genutzte MMF-Partition (nur Windows 98)	380
Kernel-Modus-Partition (Windows 2000 und Windows 98)	381
Bereiche innerhalb eines Adressraums	381
Bereitstellung von physischem Speicher innerhalb eines reservierten Bereichs	382
Physischer Speicher und die Auslagerungsdatei	383
Nicht zur Auslagerungsdatei gehörender physischer Speicher	385
Schutzattribute	387
Zugriff mittels Copy-On-Write	388
Spezielle Flags im Zusammenhang mit Schutzattributen	389
Grundlegende Zusammenhänge	389
Zusätzliche Unterteilung der Bereiche	393
Unterschiede des Adressraums unter Windows 98	398
Die Bedeutung der Datenausrichtung	403
14 Virtueller Speicher in der Praxis	409
Systeminformationen	409
Die Beispielanwendung <i>SysInfo</i>	411
Statistik des virtuellen Speichers	417
Die Beispielanwendung <i>VMStat</i>	418
Ermitteln des Zustands eines Adressraums	423
Die Funktion <i>VMQuery</i>	425
Die Beispielanwendung <i>VMMMap</i>	431
15 Virtueller Speicher in Anwendungen	443
Reservierung eines Adressbereichs	444
Bereitstellung von Speicher	446
Reservieren eines Adressbereichs und gleichzeitige Bereitstellung von Speicher	447
Wann sollte physischer Speicher bereit gestellt werden?	447
Freigabe eines Adressraums und des dafür bereit gestellten physischen Speichers ...	450
Wann sollte physischer Speicher freigegeben werden?	451
Die Beispielanwendung <i>VMAlloc</i>	452
Ändern von Schutzattributen	461

Den Inhalt physischen Speichers zurücksetzen	462
Die Beispielanwendung <i>MemReset</i>	463
AWE (Address Windowing Extensions) (nur Windows 2000)	466
Die Beispielanwendung <i>AWE</i>	470
16 Der Stack eines Threads	479
Der Stack eines Threads unter Windows 98	483
Funktion der C/C++-Laufzeitbibliothek zum Überprüfen des Stacks	485
Die Beispielanwendung <i>Summation</i>	487
17 Speicherbasierte Dateien	495
Speicherbasierte EXE- und DLL-Dateien	496
Statische Daten werden nicht von mehreren Instanzen einer EXE- oder DLL-Datei gemeinsam verwendet	497
Statische Daten werden von mehreren Instanzen einer EXE- oder DLL-Datei gemeinsam verwendet	500
Die Beispielanwendung <i>AppInst</i>	506
Speicherbasierte Dateien	510
Methode 1: Eine Datei, ein Puffer	510
Methode 2: Zwei Dateien, ein Puffer	511
Methode 3: Eine Datei, zwei Puffer	511
Methode 4: Eine Datei, kein Puffer	512
Verwendung speicherbasierter Dateien	512
Schritt 1: Erzeugen oder Öffnen eines Dateiobjekts	512
Schritt 2: Erzeugen eines internen Dateiabbildungsobjekts	514
Schritt 3: Abbildung des Dateiinhalts in den Adressraum des Prozesses	517
Schritt 4: Entfernen einer Dateiansicht aus dem Prozessadressraum	520
Schritt 5 und 6: Schließen des Dateiabbildungsobjekts und des Dateiobjekts	521
Die Beispielanwendung <i>FileRev</i>	522
Bearbeiten einer umfangreichen Datei mithilfe speicherbasierter Dateien	531
Speicherbasierte Dateien und Kohärenz	532
Abbildung einer Datei an eine bestimmte Adresse	533
Details zur Implementierung von speicherbasierten Dateien	535
Gemeinsame Datennutzung unter Prozessen mithilfe speicherbasierter Dateien	537
Speicherbasierte Dateien als Bestandteil der Auslagerungsdatei	538
Die Beispielanwendung <i>MMFShare</i>	539
Speicherbasierte Dateien mit wenig bereit gestelltem Speicher	544
Die Beispielanwendung <i>MMFSparse</i>	546
18 Speicherverwaltung mit Heaps	561
Der standardmäßige Heap eines Prozesses	562
Gründe für die Erstellung zusätzlicher Heaps	563
Schutz von Komponenten	563
Effiziente Speicherverwaltung	564
Lokaler Zugriff	565
Verhindern von Overhead durch Thread-Synchronisation	566
QuickFree	566
Erstellen eines zusätzlichen Heaps	566
Zuweisen eines Speicherbereichs auf dem Heap	568

Ändern des Umfangs eines reservierten Bereichs	569
Ermitteln des Umfangs eines reservierten Bereichs	570
Freigabe eines Speicherbereichs	570
Abbau eines Heaps	571
Verwendung von Heaps in C++	571
Weitere Heap-Funktionen	575

Teil IV

Dynamische Link-Bibliotheken **579**

19 DLL-Grundlagen **581**

DLLs und der Adressraum eines Prozesses	582
Überblick	584
Erstellen eines DLL-Moduls	587
Die Bedeutung des Exportierens	589
Erstellung von DLLs zur Verwendung mit anderen Compilern als Visual C++	591
Erstellen des ausführbaren Moduls	592
Die Bedeutung des Importierens	593
Ausführen des EXE-Moduls	595

20 DLL-Techniken **599**

Explizites Laden von DLL-Modulen und Binden von Symbolen	599
Das DLL-Modul explizit laden	601
Explizites Entfernen des DLL-Moduls	602
Explizites Verknüpfen mit einem exportierten Symbol	604
Die Einstiegs-/Ausstiegsfunktion der DLL	605
DLL_PROCESS_ATTACH	606
DLL_PROCESS_DETACH	608
DLL_THREAD_ATTACH	610
DLL_THREAD_DETACH	611
Das Serialisieren der Aufrufe von <i>DllMain</i>	611
<i>DllMain</i> und die C/C++-Laufzeitbibliothek	614
Verzögertes Laden von DLLs	615
Die Beispielanwendung <i>DelayLoadApp</i>	619
Funktionsweiterleitungen	625
Bekannte DLLs	626
DLL-Installationsverzeichnis ändern	627
Festlegen der Basisadressen von Modulen	628
Binden von Modulen	635

21 Thread-lokaler Speicher **639**

Dynamischer thread-lokaler Speicher	640
Dynamischen thread-lokalen Speicher verwenden	642
Statischer thread-lokaler Speicher	644

22 Überwinden von Prozessgrenzen	647
Injizieren einer DLL: Ein Beispiel	648
Injizieren einer DLL mittels Registrierung	650
Injizieren einer DLL mittels Windows-Hooks	652
Das Dienstprogramm DIPS	653
Injizieren einer DLL mittels Remote-Threads	665
Die Beispielanwendung <i>Inject Library</i>	669
Die Bibliothek <i>ImgWalk.dll</i>	677
Injizieren einer DLL mittels trojanischer DLL	679
Injizieren einer DLL mittels Debugger	680
Injizieren von Code mittels einer speicherbasierten Datei unter Windows 98	680
Injizieren von Code mit <i>CreateProcess</i>	680
API-Hooking: Ein Beispiel	681
API-Hooking durch Überschreiben von Code	682
API-Hooking durch Ändern des Importabschnitts eines Moduls	683
Die Beispielanwendung <i>LastMsgBoxInfo</i>	686

Teil V

Strukturierte Ausnahmebehandlung

703

23 Endebehandlung

705

Beispiel für einen Endehandler	706
<i>Funcenstein1</i>	707
<i>Funcenstein2</i>	707
<i>Funcenstein3</i>	709
<i>Funcfurter1</i>	710
Zwischendurch ein Quiz: <i>FuncaDoodleDoo</i>	710
<i>Funcenstein4</i>	712
<i>Funcarama1</i>	713
<i>Funcarama2</i>	713
<i>Funcarama3</i>	714
<i>Funcarama4</i> : Die letzte Grenze	715
Einige Anmerkungen zum <i>finally</i> -Block	717
<i>Funcfurter2</i>	717
Die Beispielanwendung <i>SEHTerm</i>	719

24 Ausnahmebehandlung und Softwareausnahmen

723

Beispiele für Ausnahmefilter und -handler	724
<i>Funcmeister1</i>	724
<i>Funcmeister2</i>	725
EXCEPTION_EXECUTE_HANDLER	727
Einige nützliche Beispiele	728
<i>Global unwinds</i>	730
<i>Global Unwinds</i> abbrechen	733
EXCEPTION_CONTINUE_EXECUTION	734
Verwenden Sie EXCEPTION_CONTINUE_EXECUTION vorsichtig	735
EXCEPTION_CONTINUE_SEARCH	736

<i>GetExceptionCode</i>	738
Speicherausnahmen	738
Ausnahmen von Ausnahmen	739
Debugging-Ausnahmen	739
Ganzzahlausnahmen	739
Gleitkommaausnahmen	739
<i>GetExceptionInformation</i>	742
Softwareausnahmen	746
25 Unbehandelte Ausnahmen und C++-Ausnahmen	749
<i>Just-In-Time-Debugging</i>	751
Unterbinden des Ausnahmemeldungsfelds	753
Erzwingen der Prozessbeendigung	753
Umhüllen einer Thread-Funktion	753
Umhüllen aller Thread-Funktionen	754
Automatischer Aufruf des Debuggers	754
Aufruf von <i>UnhandledExceptionFilter</i>	755
<i>UnhandledExceptionFilter</i> intern	755
Ausnahmen und der Debugger	757
Die Beispielanwendung <i>Spreadsheet</i>	760
C++-Ausnahmen und strukturierte Ausnahmen	772
Strukturierte Ausnahmen mit C++ abfangen	773
Teil VI	
Nachrichtenbehandlung	777
26 Fensternachrichten	779
Die Nachrichtenwarteschlange von Threads	780
Bereitstellung von Nachrichten in der Nachrichtenwarteschlange eines Threads	782
Senden von Nachrichten an Fenster	783
Aktivieren eines Threads	789
Die Flags für den Warteschlangenstatus	789
Einlesen von Nachrichten aus der Warteschlange eines Threads	791
Aktivieren von Threads mit Kernel-Objekten oder Warteschlangenstatusflags	795
Datenaustausch mithilfe von Nachrichten	797
Die Beispielanwendung <i>CopyData</i>	800
Wie Windows mit ANSI/Unicode-Zeichen und Zeichenfolgen umgeht	805
27 Hardware-Eingabemodell und lokaler Eingabezustand	807
Der <i>Raw-Input-Thread</i> (RIT)	807
Lokaler Eingabezustand	809
Tastatureingaben und Eingabefokus	810
Verwaltung des Mauszeigers	814
Verbinden von Eingabewarteschlangen und lokalem Eingabezustand	816
Die Beispielanwendung <i>LISLab</i>	818
Die Beispielanwendung <i>LISWatch</i>	834

Teil VII	
Anhang	843
A Die Entwicklungsumgebung	845
Die Header-Datei <i>CmnHdr.h</i>	845
Windows-Version	846
Unicode	846
Windows-Definitionen und Warnungen der Stufe 4	846
Hinweise während des Übersetzens	847
Die Makros <i>chINRANGE</i> und <i>chDIMOF</i>	847
Das Makro <i>chBEGINTHREADEX</i>	848
Erweiterung der Funktion <i>DebugBreak</i> für x86-Plattformen	849
Erstellen von Codes für Softwareausnahmen	850
Das Makro <i>chMB</i>	850
Die Makros <i>chASSERT</i> und <i>chVERIFY</i>	850
Das Makro <i>chHANDLE_DLGMSG</i>	850
Das Makro <i>chSETDLGICONS</i>	850
Inline-Funktionen zur Prüfung der Betriebssystemversion	850
Prüfung, ob das Hostsystem Unicode unterstützt	851
Linker auf die Startfunktion (<i>w</i>) <i>WinMain</i>	851
B Nachrichtenanalysemakros, Makros für untergeordnete Steuerelemente und API-Makros	857
Analysemakros für Nachrichten	858
Makros für untergeordnete Steuerelemente	860
API-Makros	861
Stichwortverzeichnis	863
Über den Autor	885