Windows7 Memory Management

Von Thomas Kestler, elevato GmbH

Einleitung

Windows 7 ist ein modernes und komplexes Betriebssystem. Auch das Memory Management ist sehr interessant und spannend. Es hat durchaus Ähnlichkeit mit dem Memory Management von UNIX/LINUX.

Leider geistern viele Mythen um das Thema Memory Management und häufig werden Begriffe falsch verwendet. Microsoft selbst hat mehrfach die Bezeichnung der Kennzahlen im Task Manager und deren Berechnung geändert, so dass man immer genau fragen muss, auf welches Betriebssystem die Angaben beziehen. Die Angaben des Task Managers reichen nicht zur Beurteilung der Speichernutzung aus.

Dieser Artikel fasst die wichtigsten Grundlagen und Begriffe zusammen.

Virtual Memory

Alle modernen Prozessoren unterstützen Virtual Memory, d. h. der virtuelle Speicher eines Prozesses scheint zusammenhängend, obwohl der physikalisch verteilt oder (noch nicht) existent ist. Windows ist ein Demand-Paging-Operation-System, heißt, einzelne Speicherseiten werden je nach Bedarf eingelagert oder ausgelagert. Eigentlich ist es sogar so, dass ein Prozess mit virtuellem Speicher gestartet es, bei dem keine virtuelle Speicherseite mit dem RAM verbunden ist und beim ersten Zugriff erfolgt ein Fault, der zum Verbinden (Laden) der Seite im RAM führt. Virtuelles Memory kann aber nicht nur auf RAM gemappt sein, sondern auch auf Dateien, was es auf einfache Art ermöglicht, Dateien als zusammenhängenden Speicherbereich zu bearbeiten.

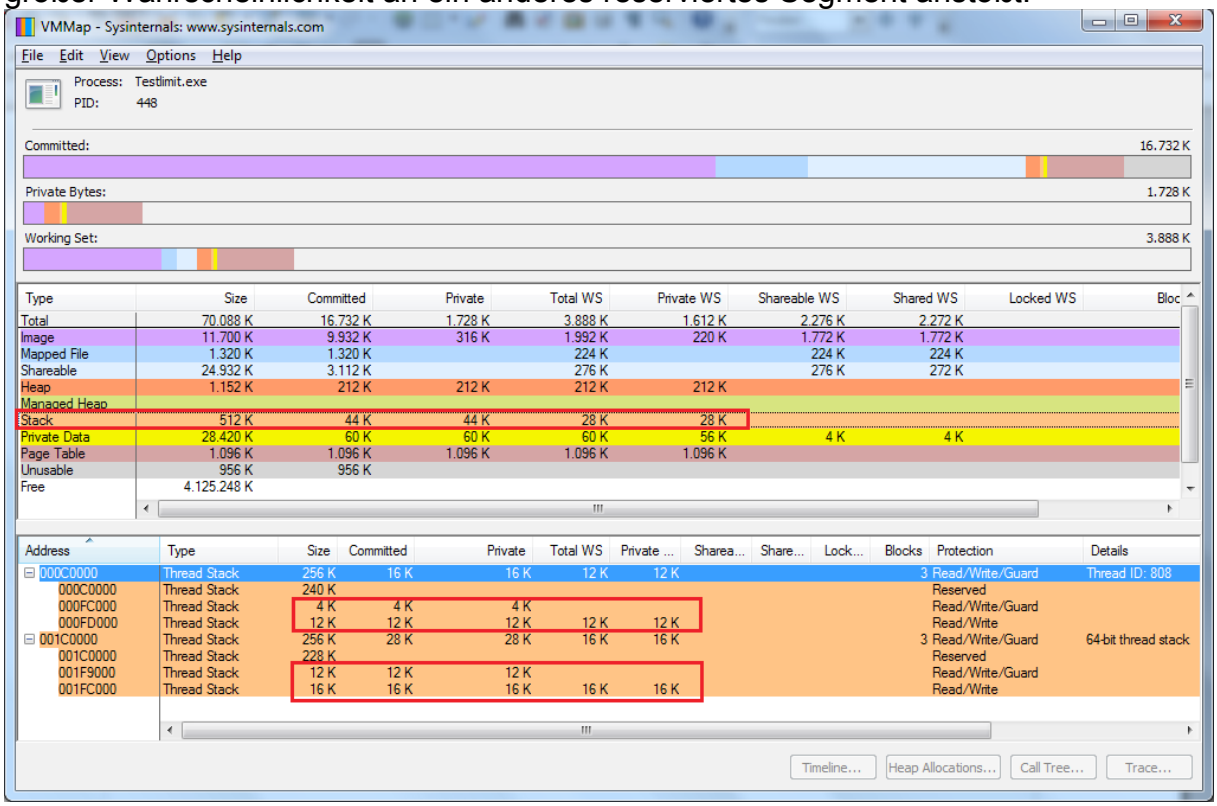
Der virtuelle Adressraum für einen Prozess bei Windows 7 (64 Bit) beträgt 8TB, ist also riesig groß. Ebenso groß ist der virtuelle Adressraum des Systems (Kernelspace). Jeder Prozess hat seinen eigenen virtuellen Adressraum. Die PageTables (PFN-Database) legen pro Prozess fest, wie virtuelle Seiten auf phsikalische Seiten oder Dateien gemappt sind. Jede Speicherseite ist 4KB groß, es gibt aber auch sog. Large Pages. Allokationen müssen immer auf einer 64KB-Grenze liegen, daher führen mehrere malloc()-Aufrufe zu 4KB zu einer Fragmentierung des virtuellen Speichers. Bei 32-Bit-Systemen konnte die Fragmentierung durchaus dazu führen, dass kein genügend großer, zusammenhängender Speicherbereich mehr für einen malloc() mit z. B. 256 MB mehr frei war. Bei 64 Bit ist die Wahrscheinlichkeit dafür sehr gering, weil der Adressraum eben so riesig ist.

Die virtuellen Speicherbereiche können ungenutzt (Free), privat (Private: Heap, Stack) oder gemeinsam nutzbar sein (Shareable: Code, Memory Mapped Files).

Reserved Memory

Ein Prozess kann zusammenhängende Bereiche reservieren. Die wichtigsten Bereiche sind wohl Heap und Stack. Ein Prozess wird standardmäßig mit einem Stack von 512 KB gestartet, d. h. es ist ein Bereich von 512KB für den Stack reserviert. Allerdings ist davon nur eine Seite initial zugesagt (commited). Je mehr Stack benötigt wird, desto mehr des reservierten Speichers wird zugesagt (Guard Page Mechanismus). Einmal zugesagt, gibt der Prozess den Stack nicht mehr frei.

Größer als 512 KB kann der Stack aber nicht werden, weil das virtuelle Segment mit großer Wahrscheinlichkeit an ein anderes reserviertes Segment anstößt.

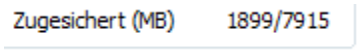


Das obige Bild zeigt zwei Stack-Segmente mit je 256 KB Größe: eines für 32-bit und eines für 64-bit (der Kernel ist ja 64-bit und Systemcalls benötigen eine 64-bit-Stack). Von den 512 KB sind 44KB zugesagt, letztlich aber nur 28KB im aktiven WorkingSet (siehe unten). Die Guard Page liegt zwischen aktivem Stack und reserviertem Stack und sorgt für die dynamische Erweiterung des Stacks.

System Commit Limit

Das Betriebssystem kann den Prozessen nur soviel Speicher zusagen (Commit), wie es im RAM und Paging-File zusammen aufnehmen kann. Will ein Prozess darüber hinaus mehr virtuellen Speicher anfordern, erhält er einen Fehler. Damit wird klar, dass die Größe des Paging-File nicht nach einer starren Formel berechnet werden kann, sondern vom Anwendungsfall abhängt. Man sollte daher PF eher zu groß wählen und dann nach einiger Zeit anhand des maximalen Commit Peak verkleinern.

Virtuelle Seiten, die auf DLLs oder EXE-Dateien gemappt sind, werden nicht auf das System Commit Limit angerechnet, sie müssen ja auch nicht in das PF ausgelagert werden, da sie eh schon auf Platte liegen (das System weiß wo die Daten sind). Angerechnet werden dagegen private Seiten und (potenziell) schreibbare Seiten, sowie Systemseiten.



Der TaskManager unter Windows 7 zeigt das aktuelle System Commit und nach dem „/“ das System Commit Limit an.

Shared Memory

Führen mehrere Prozesse das gleiche Programm oder gleiche DLLs aus, so können sie sich die Speicherseiten teilen. Für den Code kein Problem, sogar Datensegmente können geteilt werden, solange, bis ein Prozess eine solche Seite beschreiben will, Dann kommt das Copy-On-Write-Flag zum Tragen: Der Schreibvorgang löst einen Fault aus, die physikalische Seite wird kopiert und dann dem schreibenden Prozess als read/write eingehängt.

Der allergrößte Teil des virtuellen Speichers besteht aus (read-only) Seiten, die mit anderen Prozessen geteilt werden. Bei einem Terminal-Server wird z. B. Outlook 20 mal ausgeführt, es wird aber kaum mehr physikalischer Speicher belegt, als bei nur einer Ausführung.

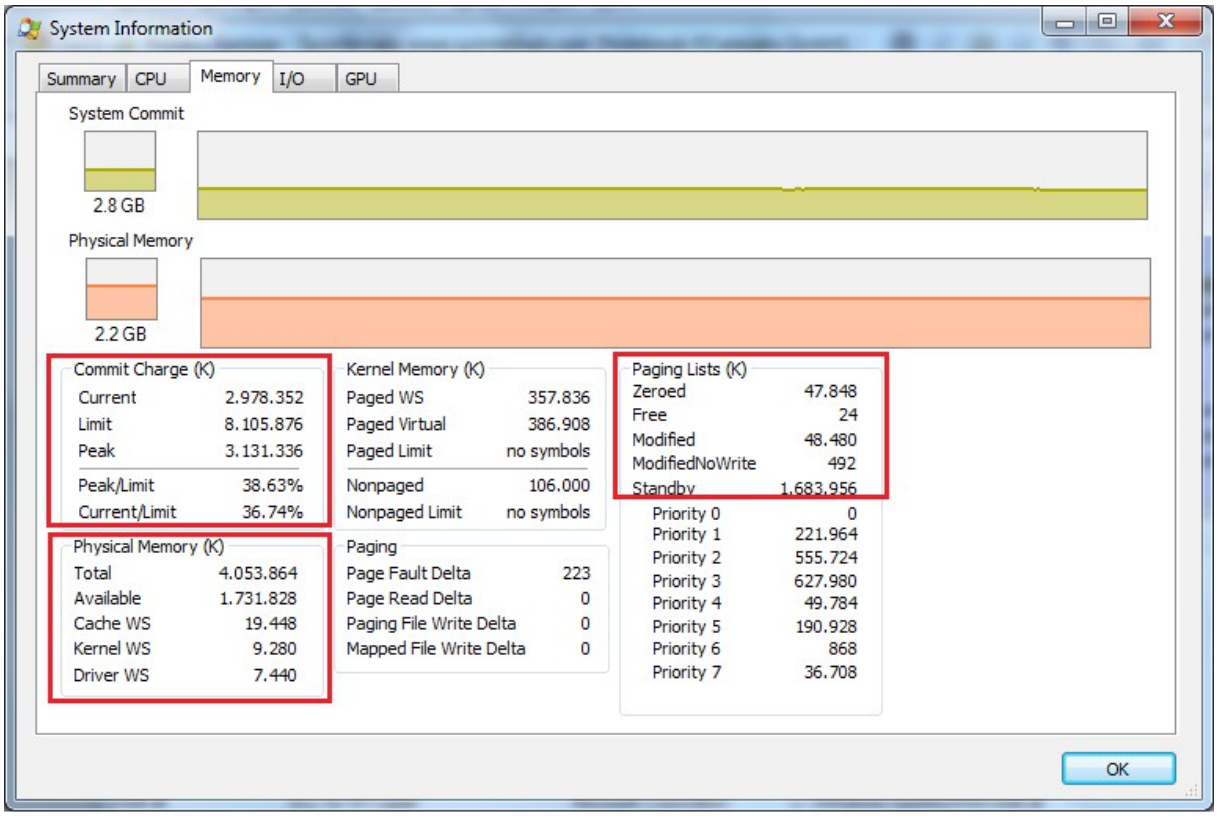
Aus UNIX kennt man Shared Memory als Mittel zur Interprozesskommunikation. In Windows kann man Shared Memory ganz einfach durch Mapping einer temporären Datei schaffen. Andere Prozesse mappen die gleiche Datei und können direkt lesend und schreibend zugreifen. In den Weblinks befindet sich ein Link auf MSDN, wo auch Beispiele dafür zu finden sind.

Vmmap

Das Tool Vmmap aus der Sysinternals-Suite ermöglicht die genaue Analyse des virtuellen Adressraumes eines Prozesses. Ein Screenshot ist ja bereits weiter oben zu sehen.

Process Explorer

Der Process Explorer aus der Sysinternals-Suite kann genutzt werden, um das System Commit Peak zu ermitteln.



Zum System Commit Limit gehören übrigens auch die Seiten im Paged Pool und Nonpaged Pool. Letzterer enthält Seiten, die garantiert nie austransferiert werden.

Physical Memory

Da das RAM immer viel kleiner ist als der virtuelle Adressraum, muss das System den kostbaren Speicher effizient verwalten. Dies ist die Aufgabe des Memory Managers. In Windows 7 wurde hier auf dem bestehenden Konzept seit NT und Vista aufgebaut und verfeinert. Im wesentlichen gilt das hier gesagte aber auch für Vista, XP und NT.

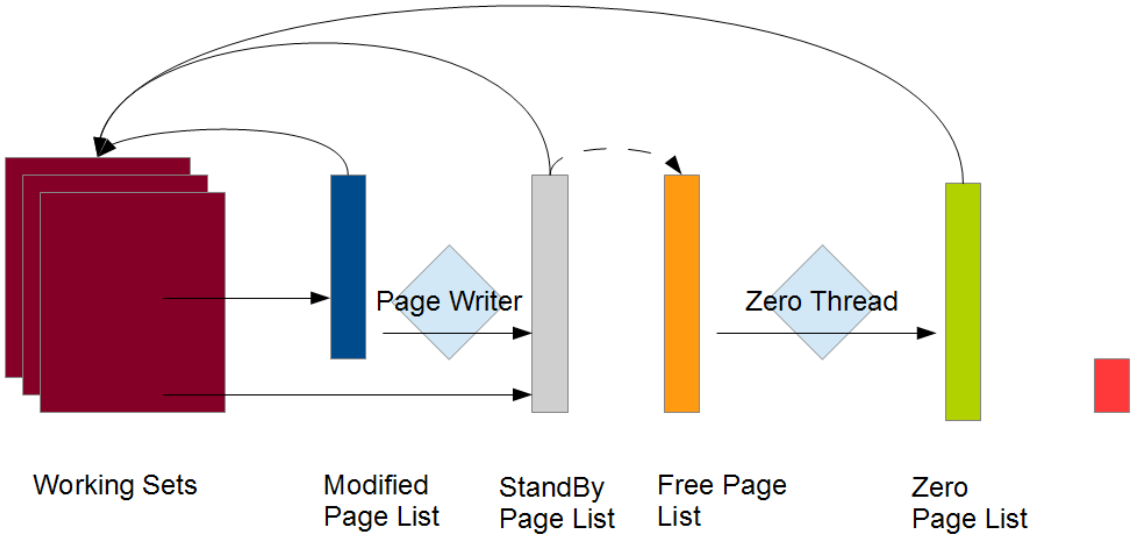
Listen

Der Memory Manager verwaltet das RAM in verschiedenen Listen:

- WorkingSets – Jeder Prozess hat ein Working Set aus den ihm zugeordneten physikalischen Seiten im RAM. Dies ist die einzige Schnittstelle von virutellem und physikalischem Speicher!
- Modified Page List – Seiten, die aus dem Working Set herausfallen, jedoch noch modifiziert sind und daher in das PF oder die gemappte Datei geschrieben werden müssen

- StandBy Page List - Seiten, die aus dem Working Set herausfallen und nicht modifiziert werden. Seiten aus der Modified-Liste kommen nach dem Austransferieren hierher
- Free Page List – Seiten die garantiert nicht mehr gebraucht werden, weil letzte Referenz daruf entfallen ist (letzter nutzender Prozess beendet). Diese enthalten aber noch Daten und müssen vor Freigabe an einen anderen Prozess noch genullt werden. Das System kann diese Seiten bereits für Leseoperationen nutzen, da sie dann mit dem Dateiinhalt überschrieben werden.
- Zero Page List – Seiten, die der Zero-Thread aus der Free List entnommen und genullt hat. Diese stehen jetzt zur Nutzung bereit.

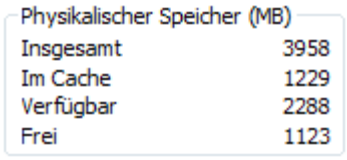
Weil die Zusammenhänge dieser Listen so wichtig für das Verständnis sind, habe ich die folgende Skizze angefertigt:



Das Herausfallen der am wenigsten benutzten Seiten aus den Working Sets der Prozesse wird als Trimming bezeichnet, siehe Weblink zu Vortrag auf der PDC09. Seiten, die modifiziert wurden, gehen zunächst in die Modified Page Liste und werden über den Page Writer Thread austransferiert, um dann in die StandBy Page List umgehängt zu werden. Diese Seiten können direkt wieder in das Working Set übernommen werden, wenn ein Prozess diese Seite zugreift (Soft Fault, kein Disk Access nötig). Nur wenn diese Seite inzwischen auf die Free Page List umgehängt wurde (weil Freier Speicher knapp wird), dann muss das System die Seite von der Platte nachladen (Hard Fault).

Seiten, die ein Prozess mit Lock sperrt, können allerdings nicht per Trimming freigegeben werden. Da keine besondere Privilegien zum Sperren von Speicherbereichen erforderlich sind, besteht hier das Potenzial das gesamte System lahm zu legen (böswillig, Denial of Service).

Der TaskManager in Windows 7 zeigt folgende Informationen zum Physical Memory an:



Hier die Bedeutung der Kennzahlen[1]:

- Insgesamt (Total): Installiertes RAM
- Im Cache (Cached): StandBy + Modified + WokingSets
- Verfügbar (Available): Free + Zero
- Frei (Free): Free Page List

Die Aussage von „Im Cache“ ist ziemlich nutzlos, StandBy + Modified wäre wohl sinnvoller gewesen.

AWE und Large Pages

AWE steht für Address Windowing Extension und ermöglicht direktes Einblenden von RAM in den virtuellen Adressraum, vorbei an den oben beschriebenen Listen. Gerade Datenbanksysteme nutzen dieses Feature. Dieses RAM kann für Demand-Paging genutzt werden, steht also dem Anforderer exklusiv zur Verfügung.

Large Pages sind ein weiteres Konzept für große Server-Anwendungen: Bei 4KB-Seiten wird der Verwaltungsaufwand in der PageTable (PFN Database) relativ hoch, wenn wirklich viel RAM verwaltet werden soll. Daher können auch große

Super Fetch

Jetzt hätten wir das Memory Management schon fast verstanden, wenn da nicht noch ein Störfeuer namens Super Fetch ins Spiel käme: Super Fetch lädt auf Verdacht Dateien in den Speicher, von denen es vermutet, dass diese bald gebraucht werden könnten. Dazu führt Super Fetch über einen längeren Zeitraum Heuristiken über die Dateizugriffe und lädt die Daten in sein Working Set, um diese gleich wieder frei zu geben (StandBy Page List). Mit relativ hoher Wahrscheinlichkeit erfolgt ein Repurpose dieser Seiten (Soft-Fault), so dass dann beim Zugriff kein Plattenzugriff mehr nötig ist.

Dies führt dann dazu, dass man kaum Seiten in der Zero Page List finden wird, da diese alle auf der StandBy Page List landen. Natürlich können auch die Seiten der StandBy Page List für neue Anforderungen.

Super Fetch findet jedoch nicht statt auf Servern (2008 R2) und bei Boot von einer SSD-Platte (weil SSD-Zugriff fast so schnell wie RAM ist).

Rammap

Mit dem Tool Rammap aus der Sysinternals-Suite können die einzelnen Listen genau beobachtet werden.

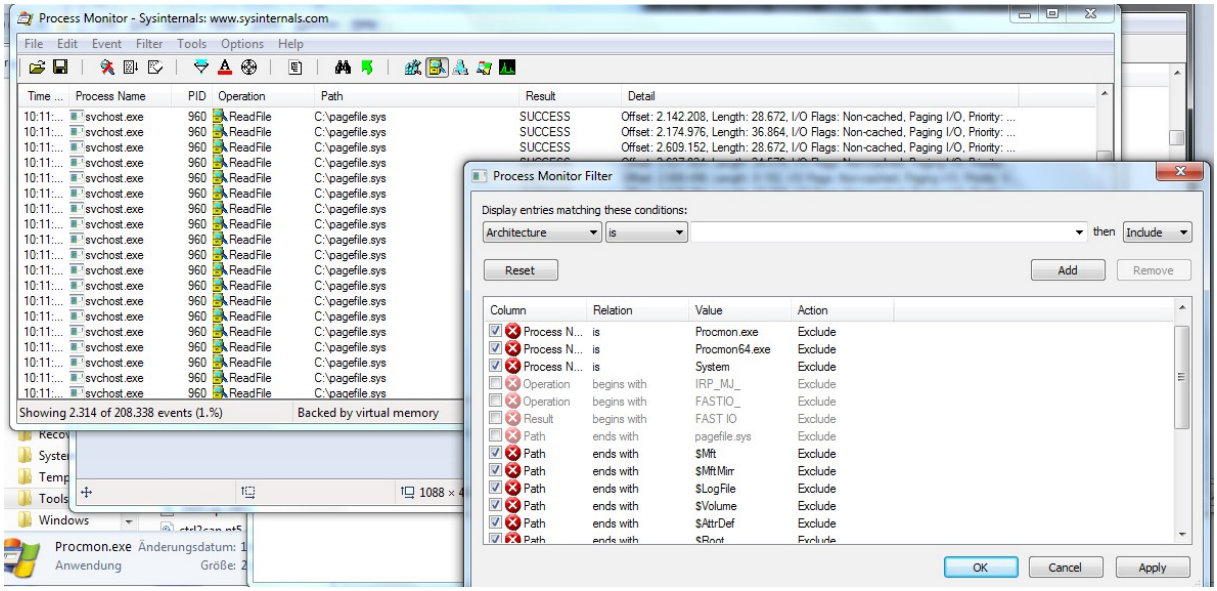
Problemanalysen

Meist beschäftigt man sich mit dem Thema Memory Management weil man Performance-Probleme festgestellt hat und den vorhandenen Hauptspeicher als

1 Quelle: Russinovich, Slide zu Tech Ed NA 2011

Flaschenhals (Bottleneck) vermutet. Eine solche Vermutung lässt sich mit Windows-Bordmitteln aber kaum wirklich belegen. Die Sysinternals-Tools sind hier wirklich Gold wert.

Solche Analyse müssen zunächst immer in einem eingeschwungenen Zustand stattfinden, der möglichst der realen Situation gleicht. Ein Mangel an RAM kann am ehesten anhand hoher Paging-Aktivität identifiziert werden. Perfmon kann hier helfen oder Process Monitor:



File-System-Cache

Bei der Betrachtung des Memory Management stellt sich die Frage nach dem File-System-Cache. Windows speichert Dateizugriffe im Hauptspeicher zwischen, um die Dateizugriffe zu beschleunigen. Anders als bei frühen UNIX-Systemen, wo ein fester Anteil des RAM für den sogenannten Buffer-Cache reserviert wurde, funktioniert der File-System-Cache in Windows (ab NT) deutlich eleganter: Im virtuellen Kernel-Adressraum ist ein Bereich für den File-System-Cache reserviert. Dieser wird wie jeder andere virtueller Speicher dynamisch über WorkingSets auf physikalisches RAM gemappt und über Demand-Paging verwaltet. Das heißt aber auch, dass dem Cache Seiten gestohlen werden, wenn das System Mangel an physikalischem Speicher feststellt.

Ein sehr detaillierter Beitrag zum Thema Cache-Manager (noch zu WinNT) von Mark Russinovich findet sich in den Weblinks.

Zusammenfassung

Das Memory Management in Windows 7 ist sehr ausgefeilt und komplex. Ohne Einigung auf die Begriffe und Kenntnis der Details sind Analysen nicht möglich. Leider hat auch Microsoft hier keine einheitliche Terminologie geschaffen und Mark Russinovich hat hier gute Vorarbeit geleistet, deshalb sollte man sich an die von Mark eingeführte Terminologie halten.

Man sollte sich hüten aus einzelnen Werten Rückschlüsse zu ziehen, insbesondere beim Task Manager ist Vorsicht angebracht, da die Werte je nach Windows-Version eine andere Bedeutung haben.

Mark Russinovich empfiehlt deshalb, dass man sich langsam durch Ausprobieren und Beobachten an das Thema heranarbeitet. Es gilt die alte Weisheit: man weiß nichts, es sei den man probiert es aus! Testlimit.exe kann dabei helfen, man kann es als testlimit.zip herunterladen.

Mark Russinovich hat auf der TechEd North America 2011 einen sehr interessanten, zweiteiligen Vortrag zum Thema Memory Management in Windows gehalten (Links am Ende des Artikels). Der Vortrag beschreibt viele Details zu den Themen Virtual Memory und Physical Memory. Nicht zuletzt wegen der zahlreichen Seitenhiebe auf seinen Arbeitgeber ist der Vortrag absolut sehenswert.
Mark Russinovich hat das Buch Windows Internals geschrieben, das in der 5ten Auflage komplett verfügbar ist, die 6te Auflage ist gerade noch in Arbeit und Part I bereits verfügbar.

Weblinks

http://channel9.msdn.com/Events/TechEd/NorthAmerica/2011/WCL405 - Mysteries of Memory Management Revealed,with Mark Russinovich (Part 1 of 2)

http://channel9.msdn.com/Events/TechEd/NorthAmerica/2011/WCL406 - Mysteries of Memory Management Revealed,with Mark Russinovich (Part 2 of 2)

http://download.sysinternals.com/files/TestLimit.zip – Testlimit Testprogramm

http://technet.microsoft.com/en-us/sysinternals/default – Sysinterals Home Page

http://msdn.microsoft.com/en-us/library/windows/desktop/aa366779(v=vs.85).aspx – MSDN About Memory Management

http://msdn.microsoft.com/en-us/library/windows/desktop/aa366878(v=vs.85).aspx – MSDN: Sharing Files and Memory

http://download.microsoft.com/download/7/e/7/7e7662cf-cbea-470b-a97e-ce7ce0d98dc2/mmwin7.pptx – Windows 7 Memory Management, Landy Wang, Microsoft, PDC09

http://blogs.msdn.com/b/tims/archive/2010/10/28/pdc10-mysteries-of-windows-memory-management-revealed-part-one.aspx – Tim Sneath, PDC10: Mysteries of Windows Memory Management Revealed: Part One

http://blogs.msdn.com/b/tims/archive/2010/10/29/pdc10-mysteries-of-windows-memory-management-revealed-part-two.aspx - Tim Sneath, PDC10: Mysteries of Windows Memory Management Revealed: Part Two

http://technet.microsoft.com/en-us/sysinternals/bb963890 – Mark's Blog

http://blogs.technet.com/b/sysinternals/ - Sysinternals Blog

http://www.windowsitpro.com/article/internals-and-architecture/inside-the-cache-manager – Mark Russinovich Blog zu File-System-Cache.

http://msdn.microsoft.com/en-us/library/windows/desktop/aa366720(v=vs.85).aspx – MSDN Large Page Support

---

Thomas Kestler ist Geschäftsführer der elevato GmbH. http://www.elevato.de