

Fallstudie: Windows 2000

20. Juli 2004

1 Die Geschichte

Im Jahre 1981 programmierte Microsoft MS-DOS für den IBM 8088, einer der ersten Personal Computer für den Firmen- und Heimgebrauch. Es gab viele Ähnlichkeiten mit UNIX und war auch von UNIX inspiriert worden. Trotz der vielen Neuerungen bei neuen Versionen blieb es aber ein kommandozeilenorientiertes Betriebssystem.

1985 entwickelte Microsoft - angeregt durch Lisa von Apple - Windows 1.0, das sich allerdings genau wie sein Nachfolger 2.0 als Flop entpuppte. Erst ab 1990 mit der Version 3.0 warf Windows Gewinne ab.

Windows 95 war dann zwar das erste richtig eigenständige Windows-Betriebssystem, war aber noch von DOS 7.0 abhängig und hatte noch große Teile 16-Bit-Code. Windows 98 war prinzipiell nur ein Update, auch wenn der DOS-Teil verringert wurde. Die große Neuerung war letztlich die Integration von Internetfunktionen auf dem Desktop, die allerdings später vom amerikanischen Justizministerium angeprangert wurde. Im Jahr 2000 kam Windows ME (Millennium Edition) auf den Markt. Es war im Kern aber Windows 98; es wurden lediglich zusätzliche Multimedia- und Internetfunktionen eingebaut. Das größte Problem war aber immer noch nicht gelöst: Alle Prozesse teilten sich die Hälfte des für Prozesse verfügbaren Speichers sowie die Unterbrechungsvektoren. Dadurch konnten beim Absturz eines Prozesses alle anderen Prozesse mitabstürzen.

Das grundsätzliche Problem von DOS-basierten Windows-Betriebssystemen erkannte Microsoft schon in den 80er Jahren. Sie beauftragten bereits zu diesem Zeitpunkt ein Entwicklerteam mit der Programmierung eines neuen portierbaren, reinen 32-Bit-Betriebssystems. 1993 war es dann soweit: Windows NT (New Technology) 3.1 kam auf den Markt. Die Versionsnummer sollte mit der aktuellen Versionsnummer vom konventionellen Windows übereinstimmen. Da NT jedoch nicht abwärtskompatibel bezüglich der alten DOS-Programme war, wurde es von den Benutzern nicht angenommen.

Lediglich auf Servern wurde NT zunehmend eingesetzt. Der erste Quantensprung erfolgte 1996 mit dem Upgrade auf NT 4.0. Die Benutzerschnittstelle war mit der von Windows 95 identisch, wodurch den Benutzern der Umstieg erleichtert wurde.

Die Nachfolgeversion von Windows NT 4.0 wurde dann mit Windows 2000 bezeichnet, um einen Namen zu haben, der für Windows 98- und NT 4.0 Benutzer passabel sein sollte. Prinzipiell müsste somit Windows 2000 eigentlich Windows NT 5.0 heißen, da das System komplett auf NT basiert und die interne Versionsnummer ist 5.xxx.yyy.zzz. Nun ist Windows ein echtes 32-Bit-System und unterstützt damit auch bis zu 32 CPUs, die zusammen Prozesse bearbeiten können. So können auch große Serversysteme mit diesem System aus gestattet werden. Eine wichtige Neuerung ist die Trennung von Benutzer- und Betriebssystemraum. So können nicht alle Prozesse auf alle Räume zugreifen und Sicherheitslinien können so überhaupt durchgesetzt werden. Auch moderne Schnittstellen wie Plug and Play, USB, FireWire und IrDA werden nun unterstützt. So werden diese Vorzüge von Windows 9x auch in Windows 2000 eingebaut. Andere Zusätze wie Active Directory, Kerberos und Systemüberwachung spielen für die Sicherheit des Systems eine große Rolle. So kann das Sicherheitsmanagement besser durchgesetzt werden. Die 4 Versionen, die von Windows 2000 ausgeliefert werden sind: Professional, Server, Advanced Server und Datacenter Server. Diese Versionen unterscheiden sich nur von der möglichen Größe des adressierbaren Speicherraums und die Anzahl der gleichzeitig ansteuerbaren CPUs. Alle diese Versionen liegen auf der Installations-CD von Windows 2000 und werden nur nach dem Kriterium ausgewählt, welche Registrierungsnummer eingegeben wird. Der komplette Code von Windows 2000 umfasst ca. 30 Millionen Zeilen C geschriebenen Text. Dies hört sich viel an im Vergleich zu anderen Betriebssystemen, wie z.B. Linux, jedoch ist in diesem Code auch alle Teile von Windows verankert. Um diese Masse an Code zu schreiben sind jeden Tag mehrere hunderte Programmierer für 2000 beschäftigt. So erklärt es sich auch, dass es jeden Tag eine neue Build-Version von dem Betriebssystem Windows 2000 gibt. So werden neue entwickelte Hardwareteile in den Kern eingebunden und neue Technologien fließen in das System ein. Die Build-Nummer von Windows 2000 bei der Veröffentlichung betrug 2195.

2 Windows-2000-Programmierung

2.1 Die Programmierschnittstelle Win32

Windows hat wie jedes andere Betriebssystem eine Programmierschnittstelle. Diese enthält eine Reihe von Unterprogrammen, die das Betriebssystem für ein Programm ausführen kann. Die Win32 API (Application Program-

ming Interface) hat einen großen Unterschied zu anderen Programmierschnittstellen: Der Code der Unterprogramme ist nicht öffentlich. Die API wurde von Version zu Version immer mehr erweitert, die Abwärtskompatibilität wurde aber gewährleistet. So laufen Programme von Betriebssystemen ab Windows 95, die sich streng an die API halten, ohne Änderung. Lediglich bei Windows 3.x wird eine spezielle Bibliothek benötigt, um die 32-Bit-Funktionen auf 16-Bit-Funktionen zu projizieren. Aufgrund der Erweiterungen laufen Programme für ein bestimmtes Betriebssystem nicht auf älteren Betriebssystemen.

Diese Unterprogramme können entweder Adressraum des Programms oder im System Änderungen herbeiführen. Wenn Objekte erzeugt werden, wird ein so genanntes Handle erzeugt. Handles sind eindeutige Identifikatoren für die erzeugten Objekte, mit denen dann weitere Operationen mittels der API ausgeführt werden können. So wird nicht mehr direkt auf die Objekte zugegriffen. Die Handles können nicht an andere Prozesse weitergegeben werden, sondern unter bestimmten Sicherheitsvoraussetzungen dupliziert werden. Auf diesem Weg werden die Objekte insgesamt sicherer. Objekte werden in normale und Kernobjekte unterteilt. Kernobjekte sind z.B. Prozesse, normale Objekte z.B. Fenster.

Die API enthält Steuerungsfunktionen für Prozesse, Threads, Synchronisation, Speicher, Dateisystem, Ein/Ausgabe, Sicherheit und die graphische Benutzerschnittstelle. Besonders die Steuerung für die graphische Benutzerschnittstelle ist immens groß.

2.2 Die Registrierung (Registry)

In alten Windowssystemen, Windows 3.x, wurden alle Systeminformationen in .ini (Initialisierung) Dateien abgespeichert und aus diesen Dateien auch ausgelesen. So wurden wichtige Pfade und Treiber in eine Form gebracht, die einen schnelleren Zugriff auf die Werte gewährleistete. Die Registrierung ist aufgebaut wie ein Dateisystem, jedoch werden in diesen Pfaden keine Dateien abgespeichert, so wie die Initialisierungsdateien, sondern so genannte Values. Diese Values liegen in Verzeichnissen, die keys genannt werden, und können bestimmte Werte beinhalten, wie z.B. die Versionsnummer einer Software oder der Speicherort für Speicherstände von Programmen. Alle Schlüssel, die keys, werden auf der Festplatte gespeichert, bis auf die zwei Schlüssel HARDWARE und PERFORMANCE_DATA. Der Schlüssel HARDWARE wird beim Booten durch den Plug-And-Play-Manager erstellt und dort werden alle gefundenen Hardwarebestandteile abgespeichert, die im Computer gefunden wurden. So können dort Verweise auf Treiber angelegt werden und Programme können dort nach installierter Hardware

oder auch nur angeschlossener Hardware nachschauen. Der Schlüssel PERFORMANCE_DATA wird während dem Systemstart erzeugt und stellt alle Zähler des System anderen Programmen zur Verfügung. Hier wird z.B. angezeigt wie lange das System schon läuft. Die Schlüssel ist so zu sagen dynamisch, da er sich jede Sekunde, in der das System läuft, verändert.

In der Registrierung werden alle wichtigen Informationen des Systems abgespeichert und jeder Programmierer, bzw. jeder Hersteller einer Software, kann hier einen eigenen Schlüssel anlegen und dort Values anlegen, auf die sein Programm zu greifen kann. Es ist auch wichtig, dass es einen Ort gibt, wo Programme Zähler und Timer oder auch Hardware des Systems nachschauen kann um das Programm an die Umgebung anpassen zu können. Es gibt zwei Schlüssel HKEY_LOCAL_MACHINE und HKEY_USERS unter denen eigentlich alle Dienste und Programme ihrer Einstellungen abspeichern können und wo sie Informationen über Hardware und Software bekommen können. Die Schlüssel HKEY_CLASSES_ROOT, HKEY_CURRENT_CONFIG und HKEY_CURRENT_USER sind nur Verweise auf genau diese ersten zwei Schlüssel der Registrierung. Sie wurden angelegt um einen schnelleren Zugriff auf bestimmte Bereiche zu zulassen. In dem Schlüssel HKEY_CLASSES_ROOT stehen die COM-Objekte (Component Object Model), diese Objekte erklären dem Betriebssystem welches Programm zu starten ist, wenn eine bestimmte Dateieindung aufgerufen wurde. So entsteht eine Bindung zwischen einem Programm und seinen Dateien, die es öffnen kann.

Durch bestimmte API-Aufrufe kann ein Programmier die Registrierung ansteuern und Aktionen bewerkstelligen. Er kann nur mit diesen Aufrufen Schlüssel ablegen oder auslesen. Dies ist eine einfache Schnittstelle für Programmieren um mit der Registrierung zu arbeiten.

Sobald Windows 2000 beendet wird, wird die Registrierung in Dateien abgespeichert. Diese Dateien werden Hives genannt und sind für den erfolgreichen Start von Windows von größter Wichtigkeit. Die meisten dieser Hives werden in dem Verzeichnis \winnt\system32\config abgespeichert. Um die Informationen der Registrierung zu speichern oder zu Sichern werden immer atomare Transaktionen durchgeführt. Dies bedeutet, dass einfachste Befehle benutzt werden, die länger dauern, jedoch eine sehr geringe Fehlerquote haben. So kann sichergestellt werden, dass ein Fehler sehr unwahrscheinlich ist und das man einmal getane Fehler einfach wieder beheben kann. Denn wenn die Registrierung bei einer Kopieaktion zerstört werden würde, würde dass eine Neuinstallation alle Software nach sich ziehen.

3 Systemstruktur

3.1 Struktur des Betriebssystems

Das System ist in mehrere Abstraktionsebenen geteilt, die größtenteils in C geschrieben sind. Lediglich die beiden der Hardware am nächsten Ebenen greifen auf Assemblercode zurück. Diese Aufteilung wurde gewählt, um das Ziel der Portierbarkeit erfüllen zu können.

3.1.1 Hardware Abstraction Layer (HAL)

In der untersten Schicht werden die maschinenabhängigen Befehle gespeichert und verarbeitet. Die für die Hardware passende HAL wird bei der Installation einmalig ausgewählt und als `hal.dll` auf der Festplatte gespeichert. Das System muss so nicht direkt auf die Hardware zugreifen, sondern kann abstrakte Befehle bereitstellen. Ein Beispiel wäre die Ein/Ausgabe, wo normalerweise der entsprechende Treiber entscheiden müsste, ob er E/A-Ports benutzt oder im Speicher abbildende E/A. Dies wird aber von HAL übernommen. Der Treiber muss lediglich das Gerät, die Zugriffsart (schreiben/lesen) und die Wortbreite auswählen. Die HAL ist so in der Lage, den Unterschied von zwei Chipsätzen zu verstecken. Die HAL baut auch nach dem Booten die Kommunikation mit dem BIOS auf um zu erfahren, welche Ein/Ausgabegeräte im System installiert sind.

Die HAL stellt z.B. Dienste für DMA, Uhren, Synchronisation und Treiber zur Verfügung. Außerdem werden Gerätereister und -adressen, Unterbrechungen, Spinlocks und anderes verwaltet. Aus Geschwindigkeitsgründen wurde für Multimediaanwendungen ein Teil ausgelagert. Dieser Teil wird mit DirectX bezeichnet.

3.1.2 Die Betriebssystem-Ebene

In der nächst oberen Schicht wird das Wechseln von Threads, speichern von CPU-Registern, löschen des CPU-Cache usw. gehandhabt. Hier würde dementsprechend ein Großteil der Arbeit für die Portierung von einem Pentium- auf ein Alphasystem anfallen. Alle Eingaben der Hardware werden hier analysiert und Unterbrechungsrouitinen werden eingeleitet.

Die Betriebssystem-Ebene wird in allen Prozessen in ihrem Adressraum abgebildet.

In dieser Schicht sind auch Ereignisse und andere Teile des Thread-Management verwoben. So werden hier zum Beispiel mit Kontrollobjekten Unterbrechungsrouitinen eingeleitet und der Zeitablauf festgelegt. Auch Dispatcher-Objekte sind hier integriert, da sie eng mit dem Thread-Management

verbunden sind. Dispatcher-Objekte sind alle Objekte, auf die Threads warten können, wie z.B. Semaphoren, Stoppuhren und Mutexe.

3.1.3 Die Executive

Die Executive ist eine Schicht, die aus 10 Komponenten besteht. Jede dieser Komponenten ist eigentlich eine Ansammlung aus Prozeduren um bestimmte Arbeiten anzubieten und durchzuführen. Die Komponenten sind alle untereinander verwoben und können sich so untereinander aufrufen. Von dieser Möglichkeit machen sie auch oft Gebrauch. So können sie ihre Arbeiten mit den anderen Komponenten synchronisieren und dem Benutzer eine bessere Arbeitsumgebung schaffen. Die Schicht ist architekturunabhängig und kann so ohne viel Aufwand auf andere Prozessoren übernommen werden. Diese Schicht ist in der Datei `ntoskrnl.exe` abgelegt und wird aus dieser Datei gestartet.

1. Der Objekt-Manager

Diese Komponente der Executive-Schicht verwaltet alle Objekte, die dem System bekannt sind wie zum Beispiel Threads, Prozesse, Dateien und Geräte. Der Manager ist für die reibungslose Kommunikation zwischen diesen Objekten zuständig und stellt die Beziehungen zwischen diesen Objekten her. So verwaltete dieser Manager auch den Namensraum für die Objekte, um sie so einzutragen und immer wieder zu finden.

2. Der E/A-Manager

Dieser Ein/Ausgabe-Manager ist die so genannte Heimat aller Gerätetreiber. Das System und die Programme sind auf diesen Manager sehr angewiesen um Geräte für die Ein- und Ausgabe anzusteuern. Er ruft den richtigen Treiber für das Gerät auf, das angefordert wird und steuert ihn an. So bietet er dem System eine Ein/Ausgabe an, die prinzipiell geräteunabhängig ist. Auch Dateisysteme sind eigentlich auch nur Ein- und Ausgabegeräte. In sofern verwaltet dieser Manager auch die FAT und NTFS Dateisysteme, die in unterschiedlichen Treiberdatei aufgeteilt sind. So laufen also alle Dateiaktionen über diesen Manager.

3. Der Prozess-Manager

Wie der Name sagt, behandelt dieser Manager Prozesse und Threads. Er ist für die Erzeugung und Vernichtung von Prozessen und Thread zuständig. Dieser Manager ist der Schlüssel zur Multiprogrammierung. Den nur dieser Manager kann entscheiden wie Prozesse und Threads gestartet und verwaltet werden. So kann er auch mehreren Prozessoren bestimmte Prozesse zuordnen und so das System schneller machen. Auch für das Multitasking ist der Prozess-Manager besonders wichtig.

4. Die Speicherverwaltung

Die Speicherverwaltung implementiert die Speicherarchitektur. Auch die Überwachung der Schutzregeln im Speicher werden durch diese Verwaltung durchgeführt. So wird sicher gestellt, dass nur jeder Prozess seine eigenen Speicherseiten zugreifen kann und nicht auf die Speicherseiten von anderen Prozessen. Sonst könnte ein Prozess auf den Speicher eines anderen Prozesses zugreifen und die Dateien dieses Programms verändern. Die Systemaufrufe für den virtuellen Speicher werden natürlich auch von dieser Speicherverwaltung behandelt und durchgeführt.

5. Der Security-Manager

So wird den ausgeklügelten Sicherheitsmechanismen von Windows Nachdruck verliehen. Dieser Manager stellt auch sicher, dass der Standard der US Regierung erfüllt wird. Dieser Standard hat den Namen U.S. Dept. Of Defense Orange Book C2 Standard. Hier wird zum Beispiel die authentifizierte Anmeldung behandelt, so dass sich Benutzer nicht ohne ein gültiges Passwort in Windows einloggen können. Auch die Zugriffskontrolle ist ein wichtiger Bestandteil des Managers, den durch diese Kontrolle wird gewährleistet, dass jeder Benutzer seine Dateien vor den anderen Benutzern sichern kann. Eine wichtige Tatsache bei der Sicherheit des Speichermanagement ist die Maßnahme, dass virtuelle Seiten mit Nullen gefüllt werden müssen, ehe sie benutzt werden dürfen. So wird eine Speicherseite nicht mit Daten von einem Programm an ein anderes Programm weitergeben, wenn dieses eine neue Seite anfordert.

6. Der Cache-Manager

Hier geschieht die Verwaltung der Blöcke im Cache. Der Cache-Manager arbeitet für alle Dateisysteme auf dem System und liefert alle benötigten Cache-Blöcke. Egal woher er diese Blöcke nehmen muss. So kann er die Blöcke entweder aus dem Cache holen, was sehr schnell geht und dann diese Blöcke einem Prozess zur Verfügung stellen oder er holt die Blöcke von der Festplatte und schreibt sie in den Cache. Er muss sich immer für den schnellsten Weg entscheiden um so den besten Zugriff auf die Blöcke zu ermöglichen. Es versteht sich von selbst, dass dieser Manager immer einen Überblick über die Blöcke haben muss, um seine Entscheidungen treffen zu können.

7. Der Plug-and-Play-Manager

Diese Komponente wird ganz am Anfang des Bootvorgangs gestartet um sich um alle Geräte im System zu kümmern. Der Manager sucht alle Geräte und installiert gegebenenfalls die Treiber, wenn dieser noch nicht installiert ist. Sobald das System hochgefahren ist, kümmert er sich nur noch um bestimmte Geräte und Schnittstellen wie z.B. USB.

Dann schickt er Benachrichtigungen nur noch über Veränderungen an diesen Geräten oder Schnittstellen aus. Beim Booten informiert er das System über neu angeschlossene Hardware. Er kann sich dann sogar selbstständig die passenden Treiber, die meist in .sys Dateien liegen, suchen und installieren.

8. Der Power-Manager

Dieser so genannte Herrscher über die Energieverwaltung überwacht die Nutzung von angeschlossenen Geräten. Er kann das Abschalten von diesen Geräten wie z.B. den Monitor veranlassen, um so Strom zu sparen. Durch diesen Manager versucht das System so wenig Strom wie möglich zu verbrauchen, wenn es gefordert wird. Bei normalen Personal Computern ist dieser Manager längst nicht so wichtig wie bei Laptops oder Notebooks, wo er auf die Batterie achtet. So kann er den Benutzer alarmieren, wenn nur noch wenig Energie in der Batterie verfügbar ist. Dann fordert er Programme auf ihre Daten abzuspeichern und sich selbst herunterzufahren um so weniger Prozesse laufen zu haben und Strom zu sparen.

9. Der Konfigurations-Manager

Der Konfigurations-Manager kümmert sich um die Registrierung. Dort erzeugt und löscht er Einträge und kümmert sich sonst noch um die reibungslose Kommunikation zwischen den Programmen und der Registrierung.

10. Der Local-Procedure-Call-Manager

Dieser komplett auf Effizienz getrimmte Manager bietet eine schnelle Interprozesskommunikation dem System. Hier wird die komplette Kommunikation zwischen den Prozessen durchgeführt, die besonders wichtig ist, damit das System überhaupt schnell funktionieren kann. So wird einem Prozess z.B. mitgeteilt, dass ein anderer Prozess mit seiner Arbeit fertig ist und der erste Prozess nun mit dem Ergebnis des ersten weiterarbeiten kann. Natürlich werden hier auch komplexe Kommunikationen durchgeführt, in der mehrere Prozesse mit einander reden”.

- Das Win32-GDI-Ausführungsmodul

Das GDI (Graphics Device Interface) behandelt Bilddaten für Monitor und Drucker. Zur Leistungssteigerung wurde das GDI mit dem Release von NT 4.0 vom Benutzerraum in den Kern verschoben. Das GDI ist mit Win32 befinden sich in der Datei win32k.sys und sind zusammen größer als der Rest der Executive.

- Die Systemdienste

Die Systemdienste sind die Schnittstelle zur Executive. Sie empfangen API-Befehle und gibt diese an die Executive zur Ausführung weiter.

- Die Checked Builds
Sie führen bei jedem Treiber einen Plausibilitätstest durch, was das System bremst. Dafür sind sie aber eine Hilfe für Autoren von Treibern bei der Fehlersuche.

3.1.4 Die Gerätetreiber

Gerätetreiber steuern entweder einfach E/A-Geräte, oder aber sie übernehmen ganz andere Funktionen wie die Verschlüsselung eines Datenstroms oder die Bereitstellung eines Zugangs zum Kern. Treiber werden in eigenen Dateien abgespeichert, so dass die Datei `ntoskrnl.exe` für alle Systeme gleich ist. Gerätetreiber müssen sehr sorgfältig geschrieben werden, da sie von den Benutzer selbst installiert werden und auf diesem Weg auf den Kern zugreifen können. Der größte Treiber ist der Grafiktreiber.

3.2 Realisierung von Objekten

Objekte sind das wichtigste Konzept der Windows-Architektur. Objekte sind Datenstrukturen im RAM. Alle Objekte werden beim Lauf des Systems erzeugt, dass heißt bei einem Absturz gehen alle Objekte verloren. Nicht mehr gebrauchte Objekte werden aus dem Speicher gelöscht. Ein Zähler gibt die Anzahl der Prozesse an, die auf ein Objekt zugreifen. Wichtige Objekte werden fest im Speicher verankert. Jedes Objekt hat einen Typ. Für jeden Typ gibt es wieder ein Typobjekt. In den Typobjekten gibt es für Standardoperationen wie Öffnen, Schließen und Löschen Zeiger auf den zugehörigen Code. Es gibt Standardtypen, jedoch kann das System neue Typen dynamisch erschaffen. Nachdem ein Objekt erzeugt wurde erhält der Benutzer die Position des Objekts um darauf zureifen zu können. Ein Objekt kann verschiedene Rechte haben, z.B. read-only.

In den Namensraum werden alle Objekte eingetragen. Win2000 hat drei Namensräume, und zwar für Objekte, Registrierung und Dateien. In diesen Räumen schaut man nach, um bestimmte Dinge zu finden. Die Namensräume sind als Verzeichnis realisiert.

3.3 Subsysteme für andere Umgebungen

Windows unterstützt 3 APIs: Win32, POSIX und OS/2. Win32 ist die offizielle Schnittstelle für Windows-Programme. Es wird mit DLL (Dynamic Link Libraries) Dateien gearbeitet. Mehrere Programme können die gleiche DLL verwenden. Es gibt über 800 System-DLLs, die zusammen über 130MB belegen. Die Zahl, der darin enthalten API-Aufrufe liegt bei etwa 13 000. Die POSIX-Umgebung ist für die UNIX-Unterstützung vorgesehen, diese ist allerdings minimal. Sie bietet nur die von der US-Regierung geforderte P1003.1 Funktionalität. Größtenteils wird auf Win32-Funktionen

zurückgegriffen. Für UNIX-Umsteiger bietet Microsoft aber noch das Produkt Interix an. Das OS/2 Subsystem ist ähnlich beschränkt und ist in der Praxis ähnlich nutzlos wie das POSIX-Subsystem.

4 Prozesse und Threads in Windows 2000

4.1 Grundlegende Konzepte

In Windows 2000 gibt es Jobs, Prozesse, Threads und Fiber. Jeder Prozess startet mit einem Thread, er kann aber noch weitere Threads erzeugen. Jeder Prozess hat einen 4GB großen Adressraum (2GB davon belegt immer das Betriebssystem). Ein Job ist eine Zusammenfassung von mehreren Prozessen.

Normalerweise laufen Threads im Benutzermodus. Wenn ein Thread einen Systemaufruf ausführt, wird er aber in den Kernmodus verlagert. Jeder Thread kann 3 Zustände einnehmen (bereit, laufend, blockiert). Ein Thread kann auf alle Objekte seines Prozess zugreifen. Nach der Ausführung werden Threads beendet. Ein Prozess endet, wenn alle seine Threads beendet sind.

Ein Fiber ist ein leichtgewichtiger Thread. Diese Fiber werden nur von Threads erzeugt und werden vom Betriebssystem nicht erkannt und verwaltet.

Da Win2000 multiprozessorfähig ist müssen alle Prozesse bzw. Threads wiedereintrittsfähig sein. Win2000 kann maximal 32 CPUs benutzen, da nur 32 Bit Bitmaps zur Bestimmung der Aktiven CPU zur Verfügung stehen.

4.2 API-Aufrufe zum Job, Prozess, Thread und Fiber Managment

Es gibt nahezu 100 Win32-API-Aufrufe, die sich mit Prozessen, Threads und Fibern beschäftigen. Neue Prozesse werden durch die API-Funktion CreateProcess generiert. Diese Funktion hat 10 Parameter, wovon jeder eine Vielzahl von Optionen hat. Jeder Prozess kann andere Prozesse erzeugen, ohne aber eine Vater-Kind-Beziehung zu erzwingen. Prozesse können den Aufruf CreateThread starten. Diese Funktion hat 6 Parameter. Der Kern erzeugt und verwaltet alle Threads. Threads können untereinander kommunizieren, mit Pipes, Named-Pipes, Mailslots, Sockets, Remote-Procedure-Calls und gemeinsam genutzten Dateien. Zusätzlich gibt es auch mehrere Werkzeuge zur Synchronisation zwischen Threads (Semaphoren, Mutexe, kritische Bereiche und Ereignisse).

4.3 Realisierung von Prozessen und Threads

Die Win32-API-Create-Aufrufe rufen eine Prozedur in der kernel32.dll auf. Erst wird überprüft, ob es sich wirklich um eine Win32-Anwendung handelt. Wenn dies der Fall ist, wird die Funktion NTCreatProcess ausgeführt und schafft eine leere Prozesshülle. Nun wird NTCreatThread bearbeitet und erstellt den ersten Thread. Kernel32.dll gibt nun dem Win32-Subsystem Meldung und die Daten werden in die internen Tabellen des Systems eingefügt, außerdem wird der Cursor auf Sanduhr gestellt. Jetzt ist der Thread lauffähig und ruft das Laufzeitsystem zur Initialisierung. Das Laufzeitsystem setzt die Priorität des Threads fest, teilt den DLLs mit, dass ein neuer Thread gibt und verrichtet andere Arbeiten. Dann startet es den Code des Hauptprogramms. Die Erzeugung eines Threads läuft ähnlich ab. Bei beiden Erzeugungen wird ein Handle zurückgeliefert, mit dem man den Prozess oder Thread z.B. starten, stoppen, töten oder untersuchen kann.

Das Scheduling von Threads wird vom Kern übernommen. Ein aktuelle laufender Thread kann das Scheduling aufrufen, wenn z.B. seine Zeitscheibe abgelaufen ist. Der Scheduler kann auch noch unter 2 anderen Bedingungen aufgerufen werden: Entweder, wenn eine E/A-Operation beendet wird oder wenn eine bestimmte Wartezeit abläuft. Prozesse können das Scheduling durch die PriorityClass des Prozesses oder des Threads beeinflussen. So entstehen 42 Prioritäten für Threads von denen nur 32 vom Betriebssystem erkannt werden. Die Prioritätenliste wird von oben nach unten abgearbeitet. Jeder Thread darf eine Zeitscheibe laufen und wird dann an das Ende der Liste gestellt. Bei mehreren Prozessoren macht jeder Prozessor sein eigenes Scheduling.

Benutzer können lediglich die Prioritäten 0-15 erreichen, die Prioritäten von 16-31 bleiben Administratoren und dem System vorbehalten. Unter bestimmten Bedingungen kann das System die Priorität eines Threads automatisch hoch setzen, aber nicht über 15. Dies kann z.B. der Fall sein, wenn ein Thread eine bestimmte Zeit lang nicht zur Ausführung gekommen ist. Sie wird auch erhöht, wenn ein Thread ein bestimmtes E/A-Gerät anspricht. Der Grad der Erhöhung hängt dann vom Typ des Geräts ab. Diese Erhöhungen gelten jedoch nur für eine Zeitscheibe. Eine Zeitscheibe ist unter Win2000 20ms lang, bei verschiedenen Systemen kann diese Zeit variieren. Normalerweise werden Prioritäten langsam heruntergefahren, außer ein Thread hängt fest und ist bereit, dann darf er 2 Scheiben lang mit der höchsten Priorität arbeiten. Der Idle-Thread verbraucht alle CPU-Zeit, die zur Zeit nicht genutzt wird.

4.4 MS-DOS-Emulation

Soll ein DOS-Programm gestartet werden, startet Windows 2000 zuerst die Win32-Anwendung ntvdm (NT Virtual DOS Machine). Diese Anwendung interpretiert das DOS-Programm. Durch ein so genanntes Trampolin kommen die Aufrufe anstatt zum System zum Emulator. Der Emulator untersucht die Aufrufe und verhindert den direkten Zugriff auf die Ressourcen des Systems. Wenn das Programm Fehler produziert wird es abgebrochen.

4.5 Starten von Windows 2000

Als erstes wird vom BIOS der Boot-Sektor angesprungen, dann wird die Datei ntldr in den Speicher geladen. Danach wird der Loader Ntldr geladen und liest die boot.ini. Die Informationen aus dieser Datei werden nun ausgelesen und verarbeitet. Die Dateien hal.dll, ntoskrnl.exe und bootvid.dll werden ausgewählt und auch gestartet bzw. geladen. Jetzt liest ntldr die Registrierung und gibt die Kontrolle an die Datei ntoskrnl.exe ab. Nach einer allgemeinen Initialisierung des Betriebssystems wird die Executive aufgerufen. Durch die Datei bootvid.dll kann der Bildschirm angesteuert werden und so kann der Balken wachsen und der erste Prozess gestartet werden: smss.exe (der Session-Manager). Dieser startet nun die crss.exe und stellt die Registrierung zusammen. Als letztes erzeugt der Loader noch den Anmelde-dienst winlogon.exe. Jetzt kann der Benutzer sich durch die lsass.exe anmelden. Doch der richtige Anmeldevorgang wird durch msgina.dll behandelt, um den Vorgang einfacher zu verändern, wenn man sich z.B. per Fingerabdruck anmelden möchte. Auch der Vaterprozess aller Dienste (service.exe) startet nun. Dieser Vaterprozess startet alle Programme, die gemäß Registrierung gestartet werden müssen. Letztendlich wird auch die Shell (explorer.exe) gestartet. Und somit läuft das System.

5 Speicherverwaltung

5.1 Grundlegende Konzepte

Jeder Prozess hat 4GB adressierbaren Speicherraum, den er nutzen kann. Die unteren 2GB sind der Benutzerraum und die Oberen sind der Kernraum, in der das Betriebssystem abgebildet ist. Die obersten und untersten 64KB sind ungültig um Programmierfehler vorzeitig zu vermeiden. Denn wenn Programme in diesen Raum springen liegt normalerweise schon ein Fehler in der Programmierung vor oder es ist ein altes DOS-Programm das versucht direkte Hardwareadresse anzuspringen. Im letzten Teil der unteren 2GB sind die Timer und Zeitgeber des Programms abgelegt worden. Normale Programme können nur auf die oberen 2GB, der Kernraum, nur lesend zugreifen, jedoch können Threads, die in den Kernmodus wechseln, so sofort

diesen Bereich adressieren, um Systemaufrufe auszuführen.

Eine virtuelle Speicherseite hat 3 Zustände: frei, reserviert und belegt. Belegte Seiten können nur in Auslagerungsdateien gespeichert werden. Es können maximal 16 Auslagerungsdateien auf 16 Festplatten benutzt werden. Jede Datei kann im Speicher abgebildet und verändert werden. Ihre Auslagerungsdatei ist die Originaldatei, die auf der Festplatte noch gespeichert liegt. Nach dem Zugriff wird die Originaldatei auf der Festplatte überschrieben. Alle Prozesse wird die gleiche Seite, mit der neusten Version der Datei, im Speicher zugewiesen. Eine private Kopie von einer Seite wird an einen Prozess übergeben, wenn andere Prozesse mit der Originalseite weiter arbeiten müssen oder sollen. Dies ist die Copy-on-Write-Technik. Um flexible zu bleiben werden nur relative Adressen im virtuellen Speicher benutzt, ein so genannter Position-Independent-Code. Erst bei mehr als 2GB physischem Speicher braucht man Address-Windowing-Extensions, dass auch Bank-Switching genannt wird.

5.2 Systemaufrufe zur Speicherverwaltung

Mit API-Funktionen kann man Seiten im Speicher verankern oder auch die Lesen/Schreiben/Ausführen-Bits einer Region verändern. Auch den Status von Regionen kann man mit diesen Funktionen abfragen. Auch das Verriegeln und Entriegeln ist möglich. Eine Datei kann so im Speicher abgebildet und geöffnet werden.

5.3 Implementierung der Speicherverwaltung

Die Speicherverwaltung kümmert sich fast ausschließlich um Prozesse. Ein VAD (Virtual Address Descriptor) wird immer angelegt, dies ist eine Tabelle, in der die Seiten eines Prozesses angelegt sind. So können auch verstreute Seiten verwaltet werden.

Wenn ein Prozess eine Seite anfordert wird diese von der Speicherverwaltung, in der Executive, in den VADs, gesucht und bereitgestellt. Die Verwaltung findet oder erzeugt die Seite, wenn das Erzeugen möglich ist. Durch Bits wird z.B. gekennzeichnet, ob eine Seite gültig ist oder ob sie in der Zwischenzeit verändert wurde. Seitenfehler werden in 5 Kategorien eingeteilt. Erstens: Die referenzierte Seite ist nicht belegt. Zweitens: Eine Schutzverletzung ist aufgetreten. Drittens: Auf eine geteilt genutzte Seite wurde geschrieben. Viertens: Der Stack muss wachsen Fünftens: Die referenzierte Seite ist belegt aber momentan nicht eingelagert. Die ersten Beiden Fehler bringen den Prozess zum abstürzen. Der dritte Fehler kommt nur bei lesbaren Seiten vor, dann kommt die Copy-on-Write-Technik. Beim vierten Fall muss einfach der Seitenrahmen vergrößert werden, jedoch mit einer Nullseite, um Sicher-

heitsvorschriften nicht zu verletzen. Beim Fall Fünf wird die Seite gesucht und eingelagert. Einmal pro Sekunde überprüft der Balance-Set-Manager, ob es genug freie Seiten gibt, um den Prozessen genug Platz zu geben. Bei zu wenig freien Seiten wird der Working-Set-Manager gestartet, der weniger gebrauchte Seiten entfernt. Dieser Working-Set-Manager läuft so lange bis genug Seiten frei sind und behandelt auch Speicherseiten, die zum Betriebssystemgehören. Es gibt 5 Listen auf den sich Seiten befinden können. Erstens: Zur Zeit benutze Seiten (Working Sets). Zweitens: Unveränderte Seiten (standby list) Drittens: Veränderte Seiten (modified list) Viertens: Freie Seiten (free list) Und Fünftens: Leere Seiten (zeroed pages list) Seiten, die sich auf dem Working Set, der modified- oder standby-list befinden gehören noch zu einem Prozess. Um dies zu überprüfen läuft alle 4 Sekunden der Swapper-Thread und sucht Prozesse, deren ganzen Threads im Leerlauf sind. Wenn er einen solchen Prozess findet schiebt er seine Seiten auf die modified- bzw. standby-list. Die beiden Threads Mapped-Page-Writer und Modified-Page-Writer schreiben veränderte Seiten auf die Platte und verschieben sie dann in die Unveränderte-Seiten-Liste. Der Zero-Page-Thread läuft die ganze Zeit auf niedrigster Priorität und löscht Seiten, die auf der Freien-Liste stehen. Woher jedoch Windows 2000 eine Seite nimmt, wenn eine Neue gebraucht wird, ist so komplex, das die Programmierer es selbst nicht mehr verstehen

6 Ein-/Ausgabe in Windows 2000

6.1 Grundlegende Konzepte

Der E/A-Manager ist eng verbunden mit dem PnP-Manager um so die Hardware zu finden und den Treiber zu laden. Danach wird ein Treiberobjekt erzeugt auf das alle Manager und Prozesse zugreifen können. Auch der Power-Manager muss eng mit dem E/A-Manager zusammenarbeiten. So können E/A-Geräte ohne Störungen im System ein- bzw. ausgeschaltet werden. Auch die Festplatte wird per Treiber vom E/A-Manager verwaltet. Der Manager holt auch die Blöcke für das System von der Festplatte. Threads haben immer die Möglichkeit synchron mit der Ein- und Ausgabe zusammen zu arbeiten.

6.2 API-Aufrufe für die Ein-/Ausgabe

Es gibt über 100 verschiedene APIs für die Ein- und Ausgabe. Die wichtigste API ist für das Grafiksystem, in ihr stehen Tausende von Win32-API-Aufrufen z.B. für Rollbalken und etc.. Zu diesen Aufrufen gehören auch das Malen und Zeichnen oder das Darstellen von Text (der von Windows selbstständig in Bitmaps umgewandelt wird).