Systemarchitektur

OS/2 2.x ist ein reines 32-Bit-Betriebssystem, das für Intel-Prozessoren des Typs 80386 und höher geschrieben wurde. Neben fortschrittlicher Nutzung der neusten Hardware-Umgebungen sind es vor allem seine besonderen Fähigkeiten, die es auszeichnen: preemptive Multitasking, lineare Speicherverwaltung und die DOS-Kompatibilität.

Diese und einige andere Eigenschaften von OS/2 sollen in diesem Kapitel aus technischer Sicht betrachtet werden. Hierbei muß ich mich wegen des großen Umfangs des Systems auf das Wesentliche beschränken und kann nur grobe Einblicke geben. In Anhang II finden Sie Hinweise auf weiterführende Literatur.

Das Kapitel für Benutzer, die bereits über Grundlagen im Bereich von Betriebssystemen verfügen.

5.1 Der Aufbau des Betriebssystems

5.1.1 Das Schichtenmodell

OS/2 läßt sich in mehrere Bereiche aufteilen. Ganz anders als DOS ergibt die Darstellung ein *Schichtenmodell* mit feststehender *Blockstruktur* (siehe Abbildung 5.1). Während es unter DOS durchaus möglich war, die vom Betriebssystem vorgesehenen Systemaufrufe beim Programmieren zu umgehen und sogenannte direkte Hardware-Zugriffe durchzuführen, läßt OS/2 kein Ausbrechen aus dem Schichtenmodell zu. Die jeweiligen Schnittstellen müssen streng eingehalten werden. OS/2 behält in jedem Fall die Oberhand und beendet ein Programm sofort, wenn es andere als die erlaubten Funktionen nutzt.

Die wichtigste Komponente des Betriebssystems ist der *Kern* (Kernel) mit einer Größe von etwa 700 Kbyte. Er steuert die Spei-


OS2KRNL

cher- und Prozeßverwaltung, ist für die Prioritätenvergabe und Inter-Prozeß-Kommunikation verantwortlich und organisiert die Multitasking-Struktur. Ein grundlegender Teil der Programmierfunktionen (*Application Programming Interface, API*), die von einem Programm für die Zusammenarbeit mit dem Betriebssystem genutzt werden müssen, ist in ihm implementiert.

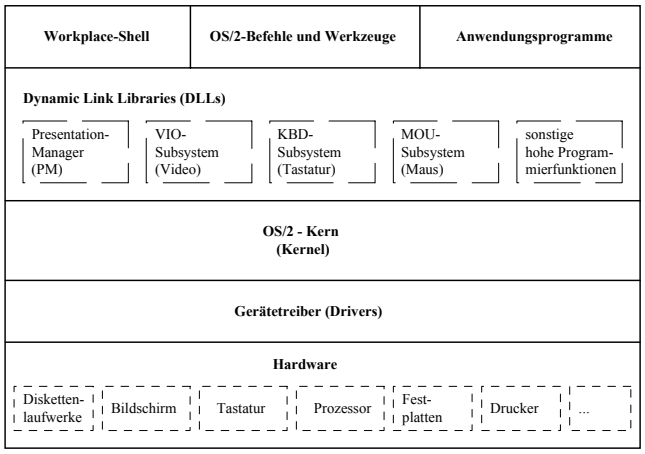

Abbildung 5.1
Der Aufbau von OS/2: Schichtenmodell

Mit der eigentlichen Hardware kommuniziert der Betriebssystemkern über *Gerätetreiber* (Drivers). Mit Hilfe dieser Treiber läßt sich das Betriebssystem an unterschiedliche Rechner-Hardware anpassen (*Portabilität*). Sämtliche Treiber werden vom Installationsprogramm automatisch in die Startdatei `CONFIG.SYS` (u.a. `BASEDEF`-Anweisungen) eingetragen und bei jedem Systemstart als erstes aktiviert.

Ähnlich seinem Schichtenmodell gibt es unter OS/2 verschiedene *Privileg-Ebenen* (siehe Abbildung 5.2), in denen unterschiedliche Funktionen zur Verfügung stehen. Ein direkter Zugriff auf eine Funktion einer weiter außen liegenden Ebene ist erlaubt, während der umgekehrte Weg unmöglich ist. Er läßt sich nur durch einen Betriebssystemaufruf realisieren. OS/2 wird dann die entsprechende Funktion kontrolliert durchführen und ihr Ergebnis über eine Schnittstelle dem aufrufenden Programm mitteilen. Anwendungsprogramme laufen üblicherweise in der äußersten Ebene ab.

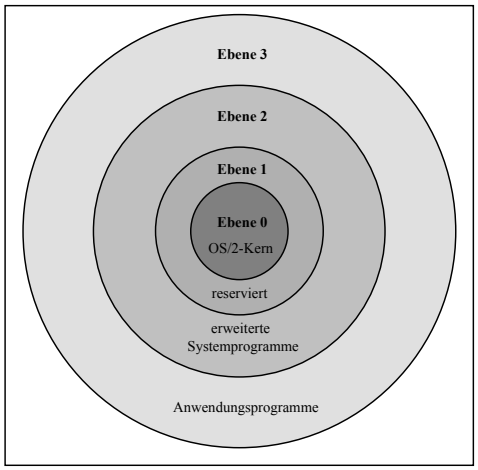

Abbildung 5.2
Privileg-Ebenen unter OS/2

Die technische Realisierung dieses Konzeptes erfolgt über die *vier Betriebsebenen*, die die Intel-Prozessoren ab dem 80286 zur Verfügung stellen. Auf der am weitesten innen liegenden Ebene 0 ist eine vollkommene Kontrolle des Rechners möglich, weshalb diese Ebene

allein dem OS/2-Kern zugänglich ist. Nach außen hin nehmen diese Rechte ab. So kann ein Programm in Ebene 3 beispielsweise bestimmte Maschinen-Instruktionen nicht mehr ausführen und nur einen begrenzten Bereich des Hauptspeichers ansprechen.

Ein weiterer Vorzug, der sowohl OS/2 als auch Windows 3.1 von DOS unterscheidet, ist die Verwendung von dynamischen Link-Bibliotheken (*Dynamic Link Libraries*, *DLL*). Diese sollen im folgenden Kapitel besprochen werden.

5.1.2 Dynamic Link Libraries (DLLs)

Externe Verweise verbinden mit Bibliotheksroutinen

Anders als der OS/2-Kern ist ein Großteil des Betriebssystems (genauso wie die Anwendungsprogramme, die für OS/2 geschrieben werden) in *Bibliotheksform* realisiert. Ein Programm enthält externe Verweise (Referenzen), die stellvertretend für die jeweilige Bibliothek stehen und zu ihr eine Verbindung aufbauen. Dies bedeutet, daß Teile des Programm-Codes auf der Festplatte abgelegt sind (Routinen-Sammlung) und von dort von verschiedenen Programmen geladen und eingebunden (gelinkt) werden können. Die Besonderheiten von dynamischen Link-Bibliotheken bestehen darin, daß Routinen entweder nur dann geladen werden, wenn sie gebraucht werden, und im Anschluß wieder frei gegeben werden können, oder mehreren Programmen gleichzeitig zur Verfügung stehen.

Neben dem Nachteil, daß das Auflösen der Referenzen (also dem Herstellen der Verbindung) zunächst einmal mehr Plattenzugriffe erfordert, bringt dieses Konzept jedoch folgende drei wesentlichen Vorteile mit sich:

- Kleinere Programme

  Dadurch, daß die Referenzen kleiner sind als die Module, auf die sie verweisen, sind die ausführbaren eigentlichen Programmdateien kleiner. Auf diese Weise wird Festplatten- bzw. Diskettenspeicherplatz gespart.

- Mehrfache Verwendung der gleichen Bibliothek

  Die Bibliotheksroutinen, die bereits durch einen Link in den Hauptspeicher geladen wurden, können von mehreren Programmen gleichzeitig benutzt werden, ohne daß sie erneut eingespeichert werden müßten (sogenannte *Reentrant-Prozeduren*). Der Code einer DLL befindet sich also jeweils immer nur einmal im Hauptspeicher.

Einfaches Update durch Patch

- Bei der Einbindung wird automatisch immer die neuste Version einer Routine verwendet. Erfolgt also ein Programm-Update, so müssen nur verschiedene Bibliothekenteile ersetzt werden (sogenannte *Patches*), während das eigentliche Programm unverändert erhalten bleibt.

Ein interessanter Aspekt im Zusammenhang mit DLLs ist, daß sich das gesamte Betriebssystem niemals komplett im Hauptspeicher befindet, sondern immer nur die gerade relevanten Teile und natürlich der OS/2-Kern. Für den Programmierer indes bleibt es vollkommen transparent, in welchem Modul sich die angesprochenen (API-) Funktionen befinden. Das Betriebssystem lädt nicht vorhandene Routinen bei Bedarf nach. Auf diese Weise bleibt das OS/2-Design für zukünftige Änderungen offen, während Anwendungsprogramme unverändert weiterverwendet werden können.

5.2 Die Multitasking-Struktur

5.2.1 Scheduling

Dispatcher und *Scheduler* steuern das Multitasking.

Seine Multitasking-Funktion erledigt OS/2 im *Zeitscheibenbetrieb* (siehe Kapitel 1.3) mit Hilfe spezieller Steuerungsprogramme, die Teile des Betriebssystem-Kerns sind. Eine zentrale Bedeutung kommt dem *Dispatcher* und dem *Scheduler* zu. Ersterer ist für die Betriebssteuerung zuständig und führt den Wechsel zwischen den laufberechtigten Prozessen durch (*Kontextwechsel*), so daß jeder von ihnen für eine kurze Zeitspanne den Zugriff auf den Prozessor erhält. Unterstützt wird er durch den *Scheduler*. Dieser ist für die Prioritätenberechnung zuständig und wählt den jeweils nächsten Prozeß aus, der durch den Dispatcher aktiviert werden soll.
Die Zuteilung der Rechenzeit erfolgt über drei Mechanismen:

Round-Robin-Verfahren

- *Round-Robin-Verfahren*:
  Jedem rechenwilligen Prozeß wird der gleiche Anteil an Prozessor-Zeit zugeordnet; in einer festen Reihenfolge erhalten alle Prozesse rundum immer wieder Zugriff auf den Prozessor.

Prioritätensteuerung

- *Prioritätengesteuert*:
  Die dynamische (oder wahlweise absolute) Prioritätenverwaltung kennt vier Prioritätenklassen, die sich wiederum in 32 Stufen unterteilen, womit eine feine Abstufung der parallel laufenden Prozesse ermöglicht wird.

Dynamische Prioritäten

Die dynamische Anpassung sorgt dafür, daß einem Vordergrundprozeß in jedem Fall eine höhere Priorität zugewiesen wird als denjenigen im Hintergrund. Außerdem ist die Prozessor-Zeitvergabe noch von weiteren Kriterien wie dem Nachrichtenaustausch, der Rechenwilligkeit oder der Ein- und Ausgabe-Funktion der Prozesse abhängig. Mit Hilfe der Maßnahme der *dynamischen* Verwaltung durch das Betriebssystem selbst wird die Systemauslastung optimiert und ein möglichst kurzes Antwortzeitverhalten erreicht.

Absolute Prioritäten

Setzen Sie dagegen mit `PRIORITY=ABSOLUTE` in der `CONFIG.SYS` eine *feste* Prioritätenvergabe fest, so wird das dynamische Angleichen durch OS/2 während des Programmablaufs inaktiviert. Programme behalten dann durchgehend diejenige Priorität, die ihnen beim Programmstart zuerkannt wurde.

Ereignissteuerung

↗ *Ereignisgesteuert*:

Dieses Prinzip trägt einer Eigenart moderner Programme Rechnung: die meisten der für eine graphische Benutzeroberfläche geschriebenen Programme sind in einem hohen Maße interaktiv und verbringen daher einen Großteil ihrer Rechenzeit damit, auf Eingaben (z.B. durch den Benutzer über die Tastatur) oder Ausgaben (z.B. durch das Betriebssystem beim Speichern einer Datei auf Festplatte) zu warten. Da während dieser Wartezeit aber bereits andere Programme den Prozessor nutzen könnten, *blockiert* der *Dispatcher* wartende Prozesse und aktiviert den nächsten durch den *Scheduler* vorgegebenen lauffähigen Prozeß.

OS/2-Treiber sind grundsätzlich Interrupt-gesteuert und lösen eine sogenannte Unterbrechung (den Interrupt) aus, sobald sie eine Aufgabe abgeschlossen haben. Dieses Ereignis wertet OS/2 aus und hebt die Blockade des auf dieses Ereignis wartenden Prozesses wieder auf. Der Prozeß kehrt daraufhin in die Reihe derjenigen Prozesse zurück, denen Rechenzeit durch den Scheduler zugeteilt wird und läuft normal weiter.

DOS-Programme lassen sich nicht durch Ereignisse steuern

Während diese dreigeteilte Berechnung bei "normalem" Betrieb für gleichmäßige Funktion und hohen Systemdurchsatz sorgt, zeigt das Prinzip jedoch bei alten DOS-Programmen seine Schwächen. Diese sind zum Beispiel nicht in der Lage, in einen *passiven Wartezustand*, in dem Sie durch die Ereignissteuerung von OS/2

blockiert würden, zu verfallen. Ganz im Gegenteil sogar springen solche Programme in eine eigene, aktive Warteschleife und belasten - wenn auch im Hintergrund - unnötig die CPU (*aktives Warten*).

5.2.2 Multitasking-Elemente

Bildschirmgruppen Prozesse und Threads

OS/2 teilt die laufenden Programme in *Bildschirmgruppen* (Screengroups oder Sessions) auf, die jeweils über einen virtuellen Bildschirm, eine virtuelle Tastatur und virtuelle Maus verfügen. Diese virtuellen Geräte existieren nur für die Bildschirmgruppe, der sie zugeordnet wurden, und werden durch das Betriebssystem mit der wirklichen Hardware verbunden, sobald sich die betreffende Bildschirmgruppe im Vordergrund befindet.

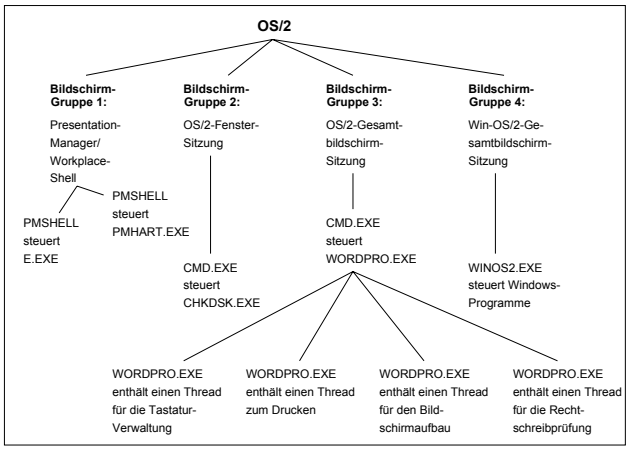

Abbildung 5.3 Multitasking-Ebenen unter OS/2

Eine solche Bildschirmgruppe kann nun mehrere *Prozesse* enthalten, die sich wiederum in *Threads* unterteilen lassen (siehe die Abbildungen 5.3 und 5.4). Letztere sind die kleinste Verarbeitungseinheit von OS/2, die man sich wie einen kleinen Prozessor vorstellen kann, der einen Teil des übergeordneten Prozesses abarbeitet. Ein Thread umfaßt im wesentlichen einen eigenen Registersatz, einen Befehlszähler und einen Stapelspeicher (Stack). Die meisten Prozesse beinhalten nur einen Thread, wenngleich Multithreating - also das Parallelisieren innerhalb eines Prozesses - unter OS/2 möglich ist und zu erheblichen Geschwindigkeits-Verbesserungen führt.

Jeder Prozeß verfügt über seine eigenen Einträge in das *Seitenverzeichnis* (page directory) des Systems. Die einzelnen Einträge ihrerseits verweisen jeweils auf eine *Seitentabelle* (page table), über die ein Speicherbereich von je 4 Mbyte adressiert werden kann. Da das Seitenverzeichnis insgesamt 1024 Seitentabellen-

Einträge enthalten kann, ergibt sich der *Gesamtspeicherbereich* zu: 1024 • 4 Mbyte = 4 Gbyte

Das Betriebssystem ist in der Lage, die Inhalte des Seitenverzeichnisses auszutauschen, so daß jedem einzelnen Prozeß ein völlig *separater Adreßraum* zur Verfügung steht. Dies gewährleistet einen optimalen Schutz und macht es einem Prozeß unmöglich, im Speicherbereich eines anderen Veränderungen vorzunehmen.

Auch das Betriebssystem selbst ist durch die Sicherheitsmechanismen des Protected-Mode vor mutwilliger Zerstörung durch andere Prozesse geschützt, so daß es theoretisch nicht vorkommen kann, daß ein Prozeß das Gesamtsystem zum Absturz bringt. In der Praxis jedoch kommt es ab und an vor, daß ein Programm oder die Kooperation mehrerer Programme den Rechner lahmlegen.

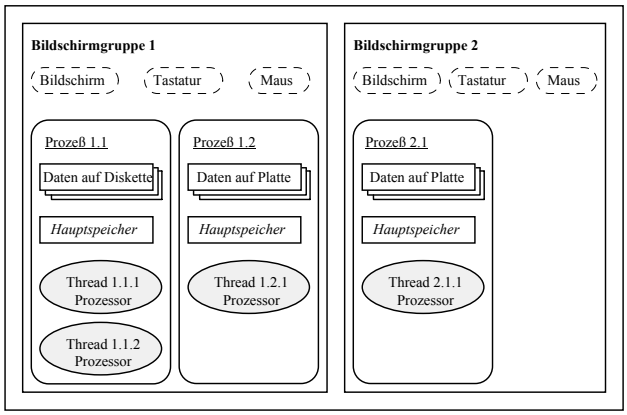

Abbildung 5.4
Die Multitasking-Organisation unter OS/2: Verschiedene Bildschirmgruppen

Multitasking geschieht unter OS/2 generell auf Thread-Ebene. Rechenzeit wird (daher der Vergleich mit einem kleinen Prozessor) nur auf dieser Ebene an Prozesse vergeben. Ressourcen wie Speicher, offene Dateien oder Nachrichtenmittel dagegen werden auf Prozeßebene, die Ein- und Ausgabegeräte schließlich auf Bildschirmgruppen-Ebene verteilt (siehe Abbildung 5.4).

Multitasking-Steuerung durch die CONFIG.SYS

In der `CONFIG.SYS` haben Sie mit den folgenden Befehlen die Möglichkeit, die Multitasking-Struktur von OS/2 zu steuern:

Details zur CONFIG.SYS in Kapitel 4.1

`THREADS` setzt die maximale Anzahl an Threads fest, die das Betriebssystem gleichzeitig verwalten kann.

`TIMESLICE` setzt die minimale und maximale Länge einer Zeitscheibe (in Millisekunden) fest, die einem Thread auf einmal zuerkannt werden kann.

`PRIORITY` steuert die Art der Prioritätenvergabe (siehe Kapitel 5.2.1).

MAXWAIT bestimmt die maximale Länge (in Sekunden), die ein Prozeß im Wartezustand (ohne Prozessor-Kontakt) verbringen darf, bevor seine Priorität von der dynamischen Anpassung hochgestuft wird. Voraussetzung für diese Anweisung ist PRIORITY=DYNAMIC.

5.3 Die Speicherverwaltung von 16- und 32-Bit-Programmen

Bereits in Kapitel 1 wurde angesprochen, daß OS/2 2.x als reines 32-Bit-System ein neues Speichermodell mitbringt, was sich gänzlich vom althergebrachten Segmentverfahren unterscheidet. Trotzdem ist es möglich, alte 16-Bit-Programme unter OS/2 2.x auszuführen.

Speicherverwaltung mit dem Segmentverfahren

Ein 16-Bit-Programm spricht den Speicher in Form von *Segmenten* mit einer maximalen Größe von 64 Kbyte an. Benötigt ein Programm mehr als diese 64 Kbyte, muß ein neues Segment angelegt und im Anschluß immer zwischen diesen Segmenten hin- und hergeschaltet werden. Dies bedeutet, daß permanent die Segmentregister des Prozessors umgeladen werden müssen, was sehr zeitaufwendig ist und einen hohen Programmieraufwand erfordert.

Das lineare Speichermodell von OS/2 2.x

Das *lineare Speichermodell* von OS/2 2.x dagegen stellt einen linearen Speicher zur Verfügung. Alle Segmentregister des Prozessors werden zunächst auf 0 gesetzt. Die einzelnen 32-Bit-Programme arbeiten dann nur noch mit dem Abstand der jeweiligen Speicheradresse zu 0 (*Offset*). Durch die fehlende Segmentverwaltung ergibt sich ein erheblicher Geschwindigkeitszuwachs.

Der *lineare Adreßraum* wird von der Speicherverwaltungseinheit (Memory Management Unit, MMU) des Prozessors in *Seiten* (pages) zu 4 Kbyte zerlegt, die an beliebiger Stelle in den Speicher eingeblendet werden können. Der Paging-Mechanismus moderner Intel-Prozessoren ermöglicht es damit, physikalisch beliebig liegende Seiten zu logisch zusammenhängenden Blöcken zu vereinigen (*virtueller Adreßraum*).

Abbildung 5.5 zeigt die Adreßräume im Überblick: Jeder der drei Prozesse verfügt über seinen eigenen *virtuellen Adreßraum*, der - so erscheint es dem Prozeß - vollkommen isoliert von den anderen und in sich zusammenhängend ist. Zur Verfügung gestellt

wird dieser *virtuelle Adreßraum* durch das lineare Speichermodell von OS/2. Es übernimmt die Verwaltung der einzelnen Seiten und bildet diese auf den wirklichen *physikalischen Adreßraum* ab. Die Abbildung zeigt außerdem einige Seiten, die auf die Festplatte ausgelagert wurden (*Swapping*).

Das neue Speichermodell schafft aber nicht nur Geschwindigkeits- und Programmiervorteile sondern ist auch für die Frage der Systemportabilität interessant. Der Verzicht auf Segmentierung macht das System nämlich in seiner Struktur Hardware-unabhängig und gestattet die spätere Portierung auf Systeme, die über keine Segmentierungs-Verfahren verfügen.

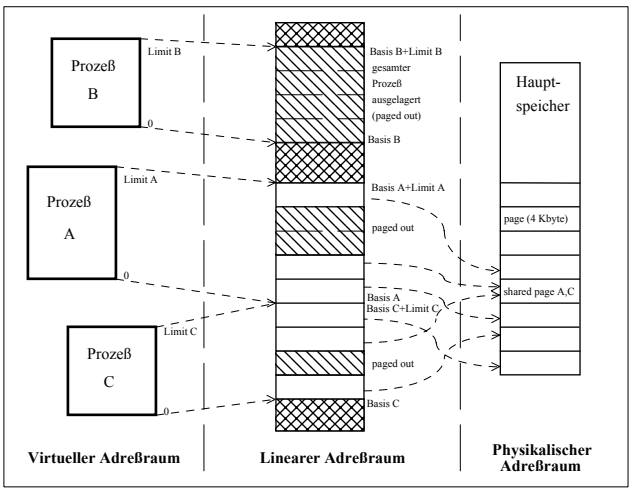

Abbildung 5.5
Adreßräume dreier Prozesse

5.4 Die DOS-Emulation

Um DOS- und Windows-Programme unverändert ablaufen lassen zu können, wird von OS/2 der *virtuelle 8086-Modus* des 80386 (und höher) genutzt. Es handelt sich hierbei um eine Betriebsart des Protected-Mode, bei der die Speicheradressierung genau wie im Real-Mode erfolgt, die aber über die Paging- und Schutzfunktionen des Protected-Mode verfügt.

OS/2 legt für jedes gestartete DOS-Programm eine eigene virtuelle DOS-Maschine (*VDM*) an, in der es eigenständig und isoliert von anderen Programmen ablaufen kann. Alle privilegierten Prozessorinstruktionen werden jedoch durch das Betriebssystem abgefangen und geeignet emuliert. Dies umfaßt auch direkte Ein- und Ausgabe-Instruktionen, für die OS/2 *virtuelle Gerätetreiber* zur Verfügung stellt (siehe Abbil-

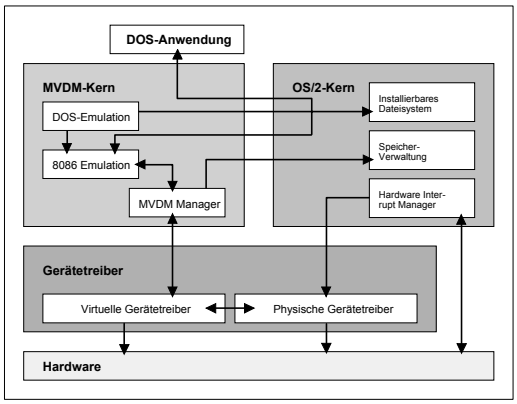

Abbildung 5.6
MVDM-System-Struktur

dung 5.6). Für das Programm selbst macht es keinen Unterschied, ob es unter OS/2 oder seinem Original-Betriebssystem läuft, weil alle Ergebnisse seiner Befehle von OS/2 entsprechend emuliert werden. Da zum gleichen Zeitpunkt mehrere DOS-Anwendungen parallel ausgeführt werden können, bezeichnet man die DOS-Kompatibilität von OS/2 2.x als *MVDM* (*Multiple Virtual DOS Machines*).

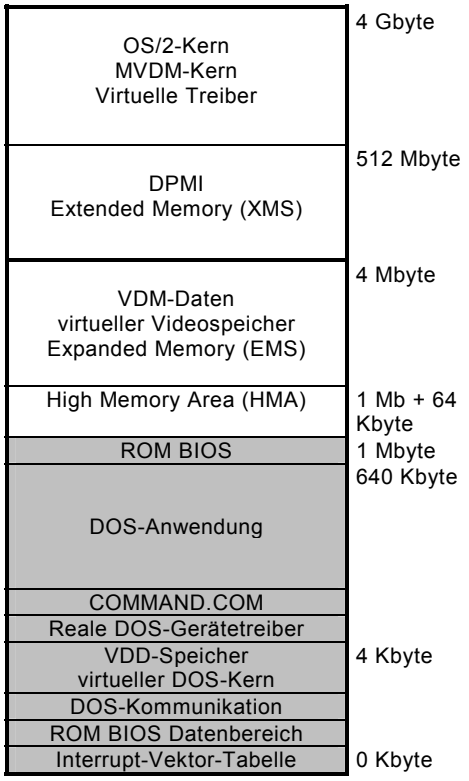

Abbildung 5.7
Speicheraufteilung einer VDM innerhalb von OS/2 2.0

Der *Paging-Mechanismus* von OS/2 macht es möglich, daß verschiedenen DOS-Umgebungen zur gleichen Zeit ein eigener virtueller Speicherbereich zur Verfügung gestellt werden kann. Da sich die 4 Kbyte-Seiten an beliebiger Stelle in den Adreßraum der VDM einblenden lassen, kann diese sowohl den konventionellen (bis 1 Mbyte) als auch erweiterten Speicher (EMS, XMS, DPMI) für ein DOS-Programm adressieren. Über die jeweilige Seitentabelle wird von OS/2 die reale 32-Bit-Adresse berechnet.

Abbildung 5.7 zeigt die Speicheraufteilung einer VDM unter OS/2 2.0. Genauso wie unter DOS wird der konventionelle Speicherbereich von Kommandointerpreter, Systemteilen der DOS-Umgebung sowie dem eigentlichen Programm benutzt. Der Bereich zwischen 1 Mbyte und den maximal möglichen 512 Mbyte steht den Daten des Programms, dem Videospeicher und erweitertem Speicher zur Verfügung. Alle für die Verwaltung der VDM nötigen Betriebssystemteile werden in den Systemadressraum oberhalb von 512 Mbyte geladen. Diese obere Grenze besteht aus Kompatibilitätsgründen zu OS/2 1.x und wird voraussichtlich in zukünftigen Versionen aufgehoben werden.