

# **Betriebssysteme**

# Table of Contents

<u>Einführung in Betriebssysteme</u> .....	1
<b><u>Betriebssysteme</u></b> .....	<b>1</b>
<u>Inhalt</u> .....	1
<u>1 Was ist ein Betriebssystem</u> .....	1
<u>2 Prozesse</u> .....	1
<u>3 Speicherverwaltung</u> .....	1
<u>4 Peripherieverwaltung</u> .....	1
<u>5 Benutzerverwaltung</u> .....	1
<u>6 Netzwerkbetriebssysteme</u> .....	2
<u>7 PC-Betriebssysteme</u> .....	2
<u>Anhang</u> .....	2
<u>Einführung in Betriebssysteme</u> .....	3
<b><u>Was ist ein Betriebssystem</u></b> .....	<b>3</b>
<u>1.1 Grundlagen, Systemschnittstelle</u> .....	4
<u>1.2 Klassifizierung von Betriebssystemen</u> .....	5
<u>Klassifizierung nach der Betriebsart des Rechnersystems</u> .....	5
<u>Klassifizierung nach der Anzahl der gleichzeitig laufenden Programme:</u> .....	6
<u>Klassifizierung nach der Anzahl der gleichzeitig am Computer arbeitenden Benutzer:</u> .....	6
<u>Klassifizierung nach der Anzahl der verwalteten Prozessoren bzw. Rechner:</u> .....	6
<u>1.3 Architektur des Betriebssystems</u> .....	8
<u>1.4 Warum Multitasking mehr Rechnerleistung bringt</u> .....	9
<u>Copyright © FH München, FB 04, Prof. Jürgen Plate</u> .....	9
<u>Einführung in Betriebssysteme</u> .....	10
<b><u>2 Prozesse</u></b> .....	<b>10</b>
<u>2.1 Programme, Prozeduren, Prozesse und Instanzen</u> .....	10
<u>Programm:</u> .....	10
<u>Prozeß:</u> .....	11
<u>Instanz:</u> .....	11
<u>Prozeßmodell</u> .....	11
<u>2.2 Prozeßzustände</u> .....	12
<u>Zustandswechsel eines Prozesses:</u> .....	14
<u>2.3 Prozeßhierarchie und Prozeßrechte</u> .....	14
<u>2.4 Prozeß-Operationen des BS</u> .....	16
<u>2.5 Prozeß-Synchronisation</u> .....	17
<u>Beispiele für zeitkritische Abläufe</u> .....	22
<u>2.6 Prozeß-Kommunikation</u> .....	23
<u>2.7 Prozeß-Scheduling</u> .....	23
<u>Scheduling-Strategien</u> .....	24
<u>2.8 Beispiele</u> .....	25
<u>Copyright © FH München, FB 04, Prof. Jürgen Plate</u> .....	25
<u>Einführung in Betriebssysteme</u> .....	26
<b><u>3 Speicherverwaltung</u></b> .....	<b>26</b>
<u>3.1 Direkte Adressierung</u> .....	27
<u>Problem beim Mehrprogrammbetrieb:</u> .....	27
<u>3.2 Verschiebung (Relocation)</u> .....	28
<u>3.3 Overlay-Technik</u> .....	29
<u>3.4 Speicherbank-Umschaltung (bank-switching)</u> .....	29
<u>3.5 Virtuelle Speicherverwaltung</u> .....	31

# Table of Contents

<b>3 Speicherverwaltung</b>	
<u>Segmentierung</u> .....	31
<u>Seitenadressierung (Paging)</u> .....	33
<u>Seitenwechsel-(Segmentwechsel-)-Algorithmen</u> .....	34
<u>Einführung in Betriebssysteme</u> .....	35
<b>4 Peripherieverwaltung</b> .....	<b>35</b>
4.1 <u>Dateien</u> .....	36
4.2 <u>Verzeichnisse (Kataloge, Ordner, Directories, Folder)</u> .....	37
<u>System mit nur einem Verzeichnis: z. B. CP/M</u> .....	37
<u>Hierarchisches System: z. B. MS-DOS</u> .....	38
<u>Hierarchisches System: z. B. UNIX</u> .....	38
<u>Hierarchisches System: z. B. NTFS</u> .....	41
4.3 <u>Gemeinsam genutzte Dateien</u> .....	41
4.4 <u>Ein- und Ausgabegeräte</u> .....	42
<u>Gerätesteuerung</u> .....	43
4.5 <u>Gemeinsame Nutzung von Betriebsmitteln</u> .....	43
<u>Beispiel aus dem täglichen Leben:</u> .....	45
4.6 <u>Dienstprogramme des Dateisystems</u> .....	46
<u>Copyright © FH München, FB 04, Prof. Jürgen Plate</u> .....	46
<u>Einführung in Betriebssysteme</u> .....	47
<b>5 Benutzerverwaltung</b> .....	<b>47</b>
5.1 <u>Benutzer-Zugang</u> .....	47
<u>Arten der Autorisierung</u> .....	49
5.2 <u>Datei-Zugriffsschutz</u> .....	50
<u>Rechte und Attribute auf Dateien im Vergleich</u> .....	52
5.3 <u>Verteilte Dateisysteme (Beispiel NFS)</u> .....	52
5.4 <u>Remote-Zugang</u> .....	54
<u>Copyright © FH München, FB 04, Prof. Jürgen Plate</u> .....	54
<u>Einführung in Betriebssysteme</u> .....	55
<b>6 Netzwerkbetriebssysteme</b> .....	<b>55</b>
6.1 <u>Merkmale von Netzwerkbetriebssystemen</u> .....	56
6.2 <u>Remote Procedure Call (RPC)</u> .....	57
<u>Copyright © FH München, FB 04, Prof. Jürgen Plate</u> .....	58
<u>Einführung in Betriebssysteme</u> .....	59
<b>7 PC-Betriebssysteme</b> .....	<b>59</b>
7.1 <u>MS-DOS</u> .....	59
7.2 <u>Windows (Version 3.x)</u> .....	61
7.3 <u>Windows 95</u> .....	63
7.4 <u>Windows 98</u> .....	63
7.5 <u>Windows NT</u> .....	66
7.6 <u>OS/2 (Version 3.0 - Warp)</u> .....	67
7.7 <u>Unix (und Derivate)</u> .....	68
7.8 <u>Linux</u> .....	69
7.9 <u>OS/9</u> .....	69
<u>Copyright © FH München, FB 04, Prof. Jürgen Plate</u> .....	69
<u>Einführung in Betriebssysteme</u> .....	70

# Table of Contents

<b><u>Literatur</u></b> .....	<b>70</b>
<u>Copyright © FH München, FB 04, Prof. Jürgen Plate</u> .....	title

# Betriebssysteme

## Inhalt

### **1 Was ist ein Betriebssystem**

- 1.1 Grundlagen, Systemschnittstelle
- 1.2 Klassifizierung von Betriebssystemen
- 1.3 Architektur des Betriebssystems
- 1.4 Warum Multitasking mehr Rechnerleistung bringt

### **2 Prozesse**

- 2.1 Programme, Prozeduren, Prozesse und Instanzen
- 2.2 Prozeßzustände
- 2.3 Prozeßhierarchie
- 2.4 Prozeß-Operationen des BS
- 2.5 Prozeß-Synchronisation
- 2.6 Prozeß-Kommunikation
- 2.7 Prozeß-Scheduling
- 2.8 Beispiele

### **3 Speicherverwaltung**

- 3.1 Direkte Adressierung
- 3.2 Verschiebung (Relocation)
- 3.3 Overlay-Technik
- 3.4 Speicherbank-Umschaltung (bank-switching)
- 3.5 Virtuelle Speicherverwaltung

### **4 Peripherieverwaltung**

- 4.1 Dateien
- 4.2 Verzeichnisse (Kataloge, Ordner, Directories, Folder)
- 4.3 Gemeinsam genutzte Dateien
- 4.4 Ein- und Ausgabegeräte
- 4.5 Gemeinsame Nutzung von Betriebsmitteln
- 4.6 Dienstprogramme des Dateisystems

### **5 Benutzerverwaltung**

- 5.1 Benutzer-Zugang
- 5.2 Datei-Zugriffsschutz
- 5.3 Verteile Dateisysteme (Beispiel NFS)
- 5.4 Remote-Zugang

### **6 Netzwerkbetriebssysteme**

## **7 PC-Betriebssysteme**

[7.1 MS-DOS](#)

[7.2 Windows \(Version 3.x\)](#)

[7.3 Windows 95](#)

[7.4 Windows 98](#)

[7.5 Windows NT](#)

[7.6 OS/2 \(Version 3.0 - Warp\)](#)

[7.7 Unix \(und Derivate\)](#)

[7.8 Linux](#)

[7.9 OS/9](#)

## **Anhang**

[Literatur](#)

[Kleiner DOS-Kurs](#)

[Kleines BS-Quiz von Chip](#)

[Skript als PDF](#) (aus HTML konvertiert)

[Download](#) des gesamten Skripts

---

*Copyright © FH München, FB 04, Prof. Jürgen Plate*

*Letzte Aktualisierung: 15. Okt 2011*

## **Einführung in Betriebssysteme**



*von Prof. Jürgen Plate*

---

# Was ist ein Betriebssystem

Unter einem Betriebssystem (engl. operating system) versteht man Software, die zusammen mit dem Hardwareeigenschaften des Computers die Basis zum Betrieb bildet und insbesondere die Abarbeitung von Programmen steuert und überwacht.

Vereinfacht gesagt, macht das Betriebssystem die Benutzung des Computers erst möglich. Im folgenden werden Einordnungs- und Bewertungskriterien besprochen.

Einige Definitionen des Betriebssystembegriffes aus der Literatur:

## **DIN 44300:**

"Die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften dieser Rechenanlage die Basis der möglichen Betriebsarten des digitalen Rechensystems bilden und die insbesondere die Abwicklung von Programmen steuern und überwachen."

## **DUDEN Informatik:**

"Zusammenfassende Bezeichnung für alle Programme, die die Ausführung der Benutzerprogramme, die Verteilung der Betriebsmittel auf die einzelnen Benutzerprogramme und die Aufrechterhaltung der Betriebsart steuern und überwachen."

## **Wettstein, Architektur von Betriebssystemen:**

"...eine Sammlung von Programmen zu geregelter Verwaltung und Benutzung von Betriebsmitteln verschiedener Art."

## **Rembold, Einführung in die Informatik:**

"Der Zweck eines Betriebssystems liegt darin, Fähigkeiten zur Verfügung zu stellen, um eine Rechenanlage möglichst durch mehrere Anwender nutzen zu können. Diese Nutzung soll einfach, zuverlässig und wirtschaftlich sein."

## **Coy, Aufbau und Arbeitsweise von Rechenanlagen:**

"Betriebssysteme ermöglichen einen geordneten Ablauf der gestarteten Programme und nutzen dabei die vorhandenen Systemsoftware- und Gerätebetriebsmittel so, daß ein möglichst sparsamer und effizienter Programmdurchsatz erreicht wird."

## **Bic / Shaw, Betriebssysteme. Eine moderne Einführung:**

"Betriebssysteme haben zwei Hauptaufgaben: Sie stellen Dienste bereit, die die Aufgaben der Benutzer vereinfachen, und sie verwalten Betriebsmittel um wirkungsvollen Rechnerbetrieb sicherzustellen."

"Aus Benutzersicht erscheint das Betriebssystem als eine virtuelle Maschine mit der Menge ihrer Kommandos als 'Maschinen'-sprache."

## 1.1 Grundlagen, Systemschnittstelle

Bleiben wir bei DIN 44300. Danach umfaßt ein Betriebssystem:

Die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften dieser Rechenanlage die Basis der möglichen Betriebsarten des Rechensystems bilden und die insbesondere die Abwicklung von Programmen steuern und überwachen.

Ein Betriebssystem hat folgende grundlegende Aufgaben:

- Verbergen der Komplexität der Maschine vor dem Anwender (Abstraktion),
- Bereitstellen einer Benutzerschnittstelle ("Kommandointerpreter", "Shell")
- Bereitstellen einer normierten Programmierschnittstelle (API), ggf. auch Compiler, Linker, Editor
- Verwaltung der Ressourcen der Maschine

## Betriebssysteme

- ◆ Prozessor(en)
- ◆ Hauptspeicher
- ◆ Hintergrundspeicher (Platte, Band, etc.)
- ◆ Geräte (Terminal, Drucker, Plotter, etc.)
- ◆ Rechenzeit
- Verfolgung von Schutzstrategien bei dieser Ressourcenbereitstellung
- Koordination von Prozessen
- Abstraktion des Maschinebegriffes nach Coy:
  - ◆ Reale Maschine = Zentraleinheit + Geräte (Hardware)
  - ◆ Abstrakte Maschine = Reale Maschine + Betriebssystem
  - ◆ Benutzermaschine = Abstrakte Maschine + Anwendungsprogramm

Oft vermischen sich die Ebenen. So ist ein Teil des Betriebssystems moderner PCs, das Basissystem für die Ansteuerung der Hardware (BIOS = Basic Input/Output System) in einem Festwertspeicher (ROM, EPROM, etc.) fest auf dem Mainboard des PCs eingebaut.

## 1.2 Klassifizierung von Betriebssystemen

Die erste Computergeneration (ca. 1945-1955) besaß kein Betriebssystem.

- Programmierung direkt (Steckbrett, Lochstreifen, Lochkarte)
- Keine Programmiersprachen

Die zweite Generation (ca. 1955-1965) arbeitete mit Stapelverarbeitung. Ein Auftrag wird in geschlossener Form, bestehend aus Programm, Daten und Steueranweisungen, zusammengestellt. Die Resultate erhält der Benutzer erst nach Abschluß der Bearbeitung zurück (meist als Ausdruck). Der Ausdruck "Stapelverarbeitung" kommt einfach vom Lochkartenstapel her, den man eingelese hat. Typische Eigenschaften sind:

- Batch-Betrieb (Lochkarten)
- Einfache Job-Control-Sprachen
- Programmiersprachen (Assembler, Fortran, etc.)
- Magnetbänder als Zwischenspeicher

Die dritte Generation (ca. 1965-1980) beherrscht Dialogverarbeitung. Der Benutzer kommuniziert mit dem Computer über Tastatur und Bildschirm, mit deren Hilfe er Programme starten, verfolgen und beeinflussen kann.

- Multiprogramming = Mehrprogrammbetrieb = Mehrere Programme gleichzeitig im Speicher quasisimultane, zeitlich verschachtelte Bearbeitung auf der Auftragsebene.
- Hauptspeicheraufteilung für mehrere Programme
- Zeitliche Verschachtelung der Programme (z.B.: Prog. A wartet auf Ausgabe, Prog. B rechnet)  
--> Timesharingbetrieb mit Terminals
- SPOOLING (simultaneous peripheral operation on line) direktes Speichern von Rechenaufträgen auf der Platte, "Selbstbedienung" des BS
- MULTICS als UNIX-Vorgänger
- 1969 das erste UNIX

Die vierte Generation (ab ca. 1975) ist ein Dialogsystem, wie wir es heute kennen. Zunächst erfolgte der Dialog im Textmodus über Tastatur und Textbildschirm. Später wurden grafische Benutzeroberflächen entwickelt (GEM von Digital Research, Apple OS auf "Lisa" und "Macintosh", Windows von Microsoft und "X" unter UNIX).

## Betriebssysteme

- UNIX und C
- Multitasking als quasisimultane Ausführung weitgehend unabhängiger Programmabschnitte innerhalb eines Auftrags.
- Personal Computer (CP/M, MS-DOS, etc.)
- Netzwerkbetriebssysteme (Kommunikation mehrerer Computer)
- verteilte Betriebssysteme (mehrere Prozessoren = Multiprocessing) Mehrere Prozessoren bilden ein Computersystem --> Mehrere Programme werden von verschiedenen Prozessoren bearbeitet oder ein Programm von mehreren Prozessoren.

Bei bestimmten Betriebssystemen spielt auch die Verarbeitungszeit eine Rolle. Bei Realzeit-(Echtzeit-)Betriebssystemen für Steuerungs- und Regelungsaufgaben (sog. Prozeßrechner) spielt die Antwortzeit eine Rolle. Informationen werden hier zum Teil von elektronischen Meßwandlern (Sensoren) gewonnen. Das Programm reagiert auf äußere Ereignisse in angemessen kurzer Zeit (Maschinenregelung: 1 - 10 ms, Prozeßregelung: 10 - 100 ms, Prozeßsteuerung: 0,1 - 1 s). Es wird spezielle Hard- und Software benötigt.

Betriebssysteme lassen sich nach unterschiedlichen Kriterien klassifizieren:

### Klassifizierung nach der Betriebsart des Rechnersystems

- Stapelverarbeitungs-Betriebssysteme (batch processing) Frühe Betriebssysteme erlaubten nur den Stapelbetrieb (Lochkarten, etc.) und auch heutige Systeme besitzen vielfach die Möglichkeit, Programmabfolgen automatisch zu bearbeiten (z. B. Batch-Dateien bei DOS, Shell-Scripte bei UNIX, usw.)
- Dialogbetrieb-Betriebssysteme (interactive processing, dialog processing) Der/die Benutzer bedienen den Rechner im Dialog mittels Bildschirm, Tastatur, Maus, usw.). Die Bedienoberfläche kann textorientiert oder grafisch sein.
- Netzwerk-Betriebssysteme (network processing) Sie erlauben die Einbindung des Computers in ein Computernetz und so die Nutzung von Ressourcen anderer Computer. Dabei unterscheidet man zwischen Client-Server-Betrieb, bei dem Arbeitsplatzrechner auf einen Server zugreifen, und Peer-to-Peer-Netzen, bei denen jeder Rechner sowohl Serverdienste anbietet als auch als Client fungiert.
- Realzeit-Betriebssysteme (realtime-processing) Hier spielt, neben anderen Faktoren, die Verarbeitungszeit eine Rolle (s. oben).
- Universelle Betriebssysteme Diese Betriebssysteme erfüllen mehrere der oben aufgeführten Kriterien.

### Klassifizierung nach der Anzahl der gleichzeitig laufenden Programme:

In dieser Klassifikation kommt der Begriff "*Task*" vor. Alternativ kann der deutsche Begriff "*Prozeß*" Verwendung finden. Aus Anwendersicht kann an dieser Stelle auch der Begriff "*Aufgabe*" bzw. "*Auftrag*" verwendet werden.

- Einzelprogrammbetrieb (singletasking)  
Ein einziges Programm läuft jeweils zu einem bestimmten Zeitpunkt. Mehrere Programme werden nacheinander ausgeführt.
- Mehrprogrammbetrieb (multitasking)  
Mehrere Programme werden gleichzeitig (bei mehreren CPUs) oder zeitlich verschachtelt, also quasi-parallel, bearbeitet.

## Klassifizierung nach der Anzahl der gleichzeitig am Computer arbeitenden Benutzer:

- Einzelbenutzerbetrieb (singleuser mode)  
Der Computer steht nur einem einzigen Benutzer zur Verfügung.
- Mehrbenutzerbetrieb(multiuser mode)  
Mehrere Benutzer teilen sich die Computerleistung, sie sind über Terminals oder Netzwerkverbindungen mit dem Computer verbunden.

## Klassifizierung nach der Anzahl der verwalteten Prozessoren bzw. Rechner:

Es geht jedoch nicht darum, wieviel Prozessoren allgemein in einem Rechner verwendet werden, sondern wieviele *Universalprozessoren* für die Verarbeitung der Daten zur Verfügung stehen. Was ist damit gemeint? In einem modernen Rechner gibt es zumindest einen Hauptprozessor (CPU, Central Processing Unit). Ihn bezeichnen wir allgemein als den Prozessor. Aber auch der PC enthält unter Umständen weitere, etwas im Verborgenen wirkende Prozessoren, z. B. den Grafikprozessor, der spezielle Eigenschaften und auch einen speziellen Befehlssatz besitzt. Auch auf dem Controller für die SCSI-Schnittstelle sitzt oft ein eigener Prozessor und auch die Ein- und Ausgabe kann über eigene Prozessoren abgewickelt werden. Somit ergibt sich folgende Unterscheidung:

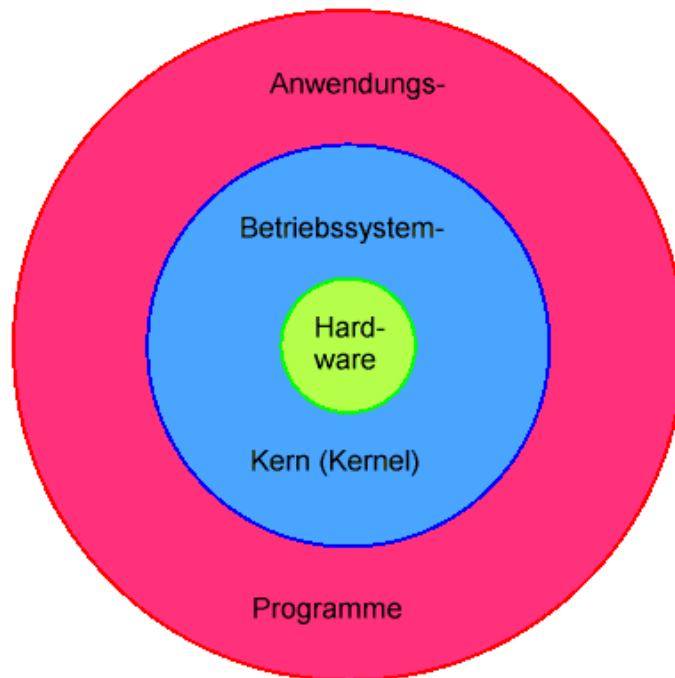
- Ein-Prozessor-Betriebssystem  
Die meisten Rechner, die auf der von-Neumann-Architektur aufgebaut sind, verfügen über nur einen Universalprozessor. Aus diesem Grund unterstützen auch die meisten Betriebssysteme für diesen Anwendungsbereich nur einen Prozessor.
- Mehr-Prozessor-Betriebssystem  
Für diese Klassifizierung der Betriebssysteme ist noch keine Aussage über die Kopplung der einzelnen Prozessoren getroffen worden. Auch gibt es keinen quantitativen Hinweis über die Anzahl der Prozessoren, nur, daß es mehr als ein Prozessor ist. Für die Realisierung der Betriebssysteme für die Mehrprozessorsysteme gibt es zwei Herangehensweisen:
  - ◆ Jedem Prozessor wird durch das Betriebssystem eine eigene Aufgabe zugeteilt. D.h., es können zu jedem Zeitpunkt nur soviel Aufgaben bearbeitet werden, wie Prozessoren zur Verfügung stehen. Es entstehen Koordinierungsprobleme, wenn die Anzahl der Aufgaben nicht gleich der Anzahl verfügbarer Prozessoren ist.
  - ◆ Jede Aufgabe kann prinzipiell jedem Prozessor zugeordnet werden, die Verteilung der Aufgaben zu den Prozessoren ist nicht an die Bedingung gebunden, daß die Anzahl der Aufgaben gleich der Anzahl Prozessoren ist. Sind mehr Aufgaben zu bearbeiten, als Prozessoren vorhanden sind, so bearbeitet ein Prozessor mehrere Ausgaben "quasi-parallel". Sind mehr Prozessoren als Aufgaben vorhanden, dann bearbeiten mehrere Prozessoren eine Aufgabe.Das Betriebssystem kann dabei seinerseits auch auf mehrere Prozessoren verteilt sein. Man spricht dann von "verteilten Betriebssystemen".

## 1.3 Architektur des Betriebssystems

Zur logischen Strukturierung wird das Betriebssystem normalerweise in mehrere Schichten oder Schalen eingeteilt. Die unterste Schale beinhaltet alle hardwareabhängigen Teile des Betriebssystems, insbesondere auch die Verarbeitung von Interrupts. Auf diese Weise ist es möglich, das BS leicht an unterschiedliche Rechnerausstattungen anzupassen. Die nächste Schicht enthält die grundlegenden Ein-/Ausgabe-Dienste für Plattenspeicher und Peripheriegeräte. Die

## Betriebssysteme

darauffolgende Schicht behandelt Kommunikations- und Netzwerkdienste, Dateien und Dateisysteme. Weitere Schichten können je nach Anforderung folgen. Ein Betriebssystem besitzt also drei oder mehr logische Schichten.



Jede Schicht bildet eine abstrakte (virtuelle) Maschine, die mit ihren benachbarten Schichten über wohldefinierte Schnittstellen kommuniziert. Sie kann Funktionen der nächstniedrigeren Schicht aufrufen und ihrerseits Funktionen für die nächsthöhere Schicht zur Verfügung stellen. Die Gesamtheit der von einer Schicht angebotenen Funktionen wird auch als "Dienste" dieser Schicht bezeichnet. Die Gesamtheit der Vorschriften, die bei der Nutzung der Dienste einzuhalten sind, wird als "Protokoll" bezeichnet.

Die unterste Schicht setzt immer direkt auf der Rechner-Hardware auf. Sie verwaltet die *realen Betriebsmittel* des Rechners und stellt stattdessen virtuelle Betriebsmittel bereit. Oft wird diese Schicht als "BIOS" (Basic I/O-System) bezeichnet. Alle weiteren Schichten sind von der Hardware unabhängig.

Durch jede Schicht wird eine zunehmende "Veredelung" der Hardware erreicht (z. B. wachsende Abstraktion, wachsende Benutzerfreundlichkeit).

Die frühen Computer (Großrechner, "Mittlere Datentechnik") zeichneten sich dadurch aus, dass Hardware und Betriebssystem vom gleichen Hersteller kamen und optimal aufeinander abgestimmt waren. Bei den heutigen PCs ist dies nur noch beim Macintosh von Apple der Fall. Bei Personal Computern auf Basis von Intel-Prozessoren kommen Hardware und Betriebssystem von unterschiedlichen Herstellern, auch wenn das Betriebssystem vielfach zusammen mit der Hardware ausgeliefert wird. So hat man die Wahl zwischen Betriebssystemen von Microsoft (Windows 98/ME, Windows 2000, Windows XP, usw.) oder freien UNIX-Implementierungen (Free BSD, Linux). Da Zusatz-Steckkarten und Peripheriegeräte (Drucker, Scanner, usw.) von den verschiedensten Herstellern kommen, liefern diese auch meist die Betriebssystem-Anpassung in Form von Treibern, die beim Laden des Betriebssystems ("Bootstrap") mit eingebunden werden.

Durch die Programmierschnittstelle (API, Applications Programmers Interface) der höheren Schichten wird auch vermieden, daß jeder Programmierer die grundlegenden Routinen für den Zugriff auf Ein-/Ausgabegeräte und Massenspeicher selbst programmieren muß. Das BS stellt also eine definierte Programmierschnittstelle zur Verfügung. Änderungen am BS oder der Hardware

## Betriebssysteme

wirken sich so nicht auf die Anwenderprogramme aus, die nach wie vor über die gleichen Betriebssystem-Aufrufe die Dienste des BS in Anspruch nehmen.

Bei Einzelbenutzer-Singletaskingsystemen können Anwenderprogramme die Schichtenstruktur durchbrechen und z. B. direkt auf einer bestimmten Hardwarekomponente aufsetzen. Bei Mehrbenutzer- oder Multitaskingsystemen ist dies nicht möglich, das hier der Schutz der einzelnen Ressourcen (CPU, Platte, Ein-/Ausgabe, usw.) für jedes Programm vor Beeinflussung durch andere Programme gewährleistet werden muß. Hier ist eine streng hierarchische Kommunikation nötig, die nur zwischen zwei benachbarten Schichten zulässig ist.

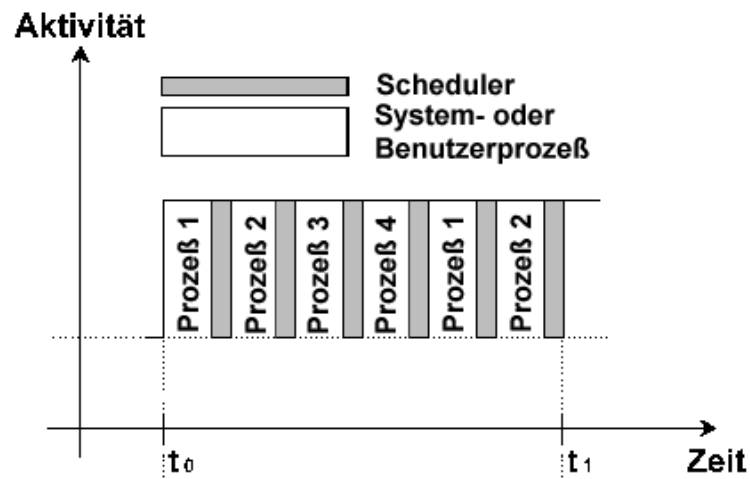
Bedenkt man, daß heutige Rechnersysteme sich selbst innerhalb einer Rechnerfamilie vielfältig in Speicherausstattung, Art und Umfang der angeschlossenen Geräte unterscheiden, so wird klar, daß die Erstellung monolithischer Programme für jede mögliche Rechnerkonstellation ein praktisch undurchführbares Unternehmen ist. Die Lösung dieses Problems heißt hier: *Modularisierung*.

Programme werden in Module zerlegt, die zueinander über definierte Schnittstellen in Beziehung stehen. Somit ist es möglich, innerhalb eines Programmes einen Modul durch einen anderen mit gleicher Schnittstelle zu ersetzen, um das Programm an eine andere Rechnerkonstellation anzupassen. Die Auswahl und Zusammenstellung der allgemeingültigen Module wird bestimmt durch die eingesetzte Hardware und die Art der Programme, die durch diese Module unterstützt werden sollen. Sie ist für viele Programme, die auf einem Rechner abgearbeitet werden sollen, gleich und unterscheiden sich wiederum etwas von Rechner zu Rechner.

Oft taucht in Zusammenhang mit Betriebssystemen auch der Begriff "Middleware" auf. Er bezeichnet zwischen den eigentlichen Anwendungen und der Betriebssystemebene angesiedelte System- und Netzwerk-Dienste (z.B. Datenbank, Kommunikation, Protokollierung, Sicherheit). Sie ist als Applikationsschicht eine Dienstleistungsschicht, die anstelle der Betriebssystemschnittstelle verwendet wird. Middleware-Systeme ermöglichen die Verteilung von Applikationen auf mehrere Rechner im Netzwerk. Die Verteilung ist objektorientiert: Server exportieren ihre Dienste als Klassenschnittstellen, Clients benutzen entfernten Methodenaufruf zum Zugriff auf die Dienste. Die Bindung kann statisch oder dynamisch erfolgen.

### **1.4 Warum Multitasking mehr Rechnerleistung bringt**

Man möchte annehmen, daß bei mehreren quasiparallel laufenden Programmen die reale Laufzeit für jedes einzelne Programm länger ist als wenn nur ein Programm auf dem Rechner laufen würde. Nehmen wir der Einfachheit halber an, daß jeder Prozeß dieselbe Laufzeit zugeteilt bekommt und auch jeder regelmäßig an die Reihe kommt.



Wenn man dieses Bild betrachtet, dauert freilich jedes Programm länger, denn die Zeit wird ja zwischen den Programmen geteilt. In der Praxis ist es jedoch so, daß Programme oftmals auf äußere Ereignisse warten müssen, z. B. bis eine Tasteneingabe getätigt wurde oder bis Daten von der Platte in den Speicher transportiert wurden. Wie Sie später sehen werden, kann ein Programm, das auf ein externes Ereignis wartet, in einen Wartezustand versetzt werden und inzwischen können andere Programme weiterarbeiten. Bei einem Multitasking-Betriebssystem kann der Rechner Datenbanken indizieren, Dokumente drucken, E-Mail absenden, usw. während Sie über den nächsten Zug gegen das Schachprogramm nachdenken.



[Zum vorhergehenden Abschnitt](#)



[Zum Inhaltsverzeichnis](#)



[Zum nächsten Abschnitt](#)

Copyright © FH München, FB 04, Prof. Jürgen Plate

## Einführung in Betriebssysteme



von Prof. Jürgen Plate

# 2 Prozesse

## 2.1 Programme, Prozeduren, Prozesse und Instanzen

### Programm:

Die Lösung einer Programmieraufgabe (=Algorithmus) wird in Form eines Programms realisiert. Teillösungen werden dabei als Prozeduren (Unterprogramme) formuliert, welche nach Beendigung ihrer Arbeit zum aufrufenden übergeordneten Programm zurückkehren. Damit die Leistungen des Betriebssystemkerns problemlos in Anwenderlösungen eingebunden werden können, sind sie ebenfalls als Prozeduren realisiert. Ein Programm (Prozedur, Unterprogramm) besteht aus:

- Befehlen (Codebereich, Textbereich)
- Programmdateien (Datenbereich)

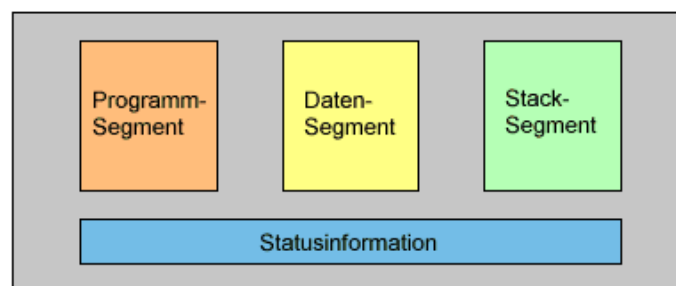
Beide Komponenten sind problemorientiert.

### Prozeß:

Wird ein Programm (Prozedur) unter der Kontrolle eines Betriebssystems (genauer gesagt unter der Kontrolle eines Betriebssystemkerns) ausgeführt, so wird dieser Ablauf als **Prozeß** (engl. Task) bezeichnet. Diese Betrachtungsweise macht es möglich, daß mehrere Programme gleichzeitig als Prozesse parallel auf einem sequentiell arbeitenden Rechnersystem (unabhängig von der realen Anzahl Prozessoren) ablaufen können. Als Sonderfall gilt die Ausführung mehrerer Prozesse auf einem Prozessor. Bei der Ausführung von Prozessen entstehen Daten, die durch den Betriebssystemkern verwaltet werden. Diese werden Statusinformationen genannt und sind systemabhängig, z. B. Registerinhalte.

Die Funktionalität des Betriebssystemkerns bezüglich der Verwaltung von Prozessen ist bei der Ausführung von  $n$  Prozessen auf einem Prozessor äquivalent der Verwaltung auf  $m > 1$  Prozessoren. Dabei kann das Verhältnis  $m : n$  sowohl statisch als auch dynamisch änderbar sein.

Als weitere Komponente wird beim Ablauf eines Programms ein Kellerspeicher (Stack) aufgebaut. Somit läßt sich ein Prozeß modellhaft folgendermaßen darstellen.



Alle vier Komponenten, die bei der Ausführung eines Programms (einer Prozedur) beteiligt sind, werden als Instanz zusammengefaßt.

## Instanz:

Eine Instanz umfaßt das Tupel (C, D, S, I):

- C: Codesegment** --> **problemorientiert**
- D: Datensegment** --> **problemorientiert**
- S: Stacksegment** --> **system-/problemorientiert**
- I: Statusinformation** --> **systemorientiert**

Die physische Anordnung dieser Komponenten im Arbeitsspeichers eines Rechners kann in unterschiedlichen Betriebssystemen verschieden sein.

Wir halten also fest:

## Prozeßmodell

- Ein Prozeß ist ein Programm während der Ausführung (Für uns gleichbedeutend mit "Task". Es gibt jedoch BS, bei denen ein Programm mehrere Prozesse startet, einen solchen Prozeß nennt man dann "Thread".)
- Es können sich mehrere Prozesse gleichzeitig im Speicher befinden, es ist jedoch immer nur ein Prozeß aktiv, d.h. er wird von der Hardware bearbeitet (außer es gibt mehrere CPUs) --> **parallel** falls die Zahl der Prozessoren größer oder gleich der Zahl der Prozesse ist, **quasiparallel** im anderen Fall.
- Ein Teil des BS, der Scheduler, wählt einen Prozeß aus, teilt ihm die CPU zu und läßt ihn eine gewisse Zeit rechnen.

Moderne Rechner können mehrere Dinge gleichzeitig ausführen. Unterstützt durch die Hardware lassen sich einzelne Aufgaben des BS parallelisieren. z. B. das Ausgeben einer Datei auf dem Drucker, während das Programm weiterläuft (auch im Einprogrammbetrieb!). Es gibt also in der Regel parallele Arbeit von CPU und E/A-Geräten.

Im Mehrprogrammbetrieb wird jedem Programm einen kurzen Zeitabschnitt (Zeitscheibe) lang die CPU zugeteilt, wodurch die Benutzer die Illusion erhalten, alle Programme würden gleichzeitig bearbeitet --> Pseudoparallelität. Damit lassen sich folgende Eigenschaften von Prozessen definieren:

- ◆ Jeder Prozeß besitzt seine eigene Prozeßumgebung (Instanz).
- ◆ Jeder Prozeß kann seinerseits andere Prozesse erzeugen und - mit Hilfe der BS-Kerns - mit anderen Prozessen kommunizieren.
- ◆ Prozesse können voneinander abhängen --> kooperierende Prozesse. Derartige Prozesse müssen sich untereinander synchronisieren.
- ◆ Prozessen kann eine Priorität zugeordnet werden, aus der sich die Reihenfolge ergibt, mit der die Prozesse der CPU zugeteilt werden.
- ◆ Die Speicherung der Prozeßzustände erfolgt in einer vom BS geführten Prozeßtabelle.

## 2.2 Prozeßzustände

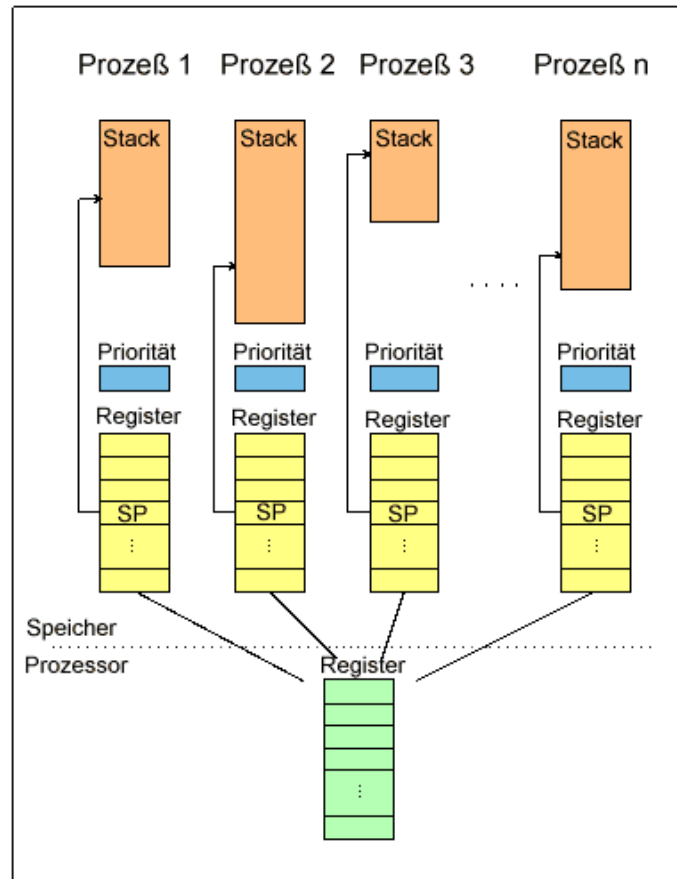
Während seiner Abarbeitung kann ein Prozeß verschiedene Zustände einnehmen:

- **aktiv (running):** Prozeß wird von der CPU bearbeitet
- **bereit (ready):** Prozeß kann die CPU benutzen, ist aber durch einen anderen Prozeß verdrängt worden
- **blockiert:** Prozeß wartet auf das Eintreten eines bestimmten Ereignisses (z. B. Drucker bereit, Benutzereingabe, etc.) Der Einfachheit halber wird hier ein Rechnersystem mit nur einer CPU angenommen, d. h. ein Prozeß ist aktiv, alle anderen sind bereit oder blockiert.
- Für bereite und blockierte Prozesse wird jeweils eine separate Warteliste geführt.

## Betriebssysteme

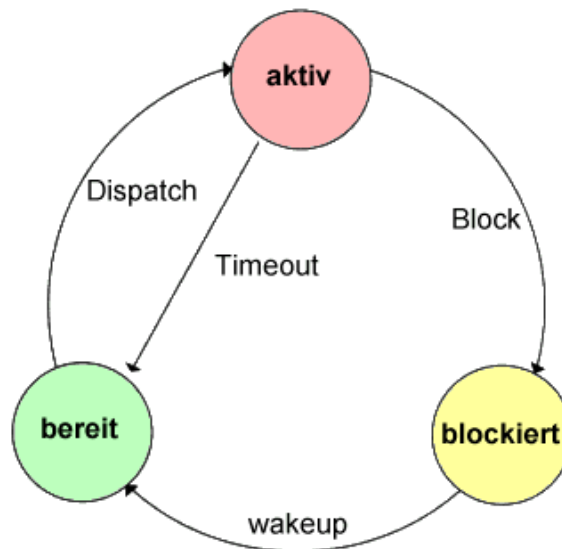
- Ein neu gestartetes Programm wird am Ende der Bereit-Liste eingetragen.
- Ein spezieller Teil des Betriebssystems, der Scheduler, teilt den Prozessen die CPU zu. Für die Zuteilung existieren unterschiedliche Algorithmen, die alle das Ziel haben, die CPU möglichst gerecht unter allen Prozessen aufzuteilen. Für die Steuerung der Zeitscheiben ist ein in regelmäßigen Zeitabständen auftretender Hardwareinterrupt notwendig --> Scheduler wird regelmäßig aufgerufen.

Die Situation in der Hardware stellt sich etwa folgendermaßen dar. Im Speicher liegen die einzelnen Instanzen der Prozesse. Jeweils ein Prozeß wird der CPU zugeteilt.



### Zustandswechsel eines Prozesses:

## Betriebssysteme



- Dispatch: bereit --> aktiv  
Zuteilung der CPU an einen Prozeß.
- Timerrunout: aktiv --> bereit  
Nach Ablauf einer Zeitscheibe wird dem Prozeß die CPU wieder entzogen.
- Block: aktiv --> blockiert  
Aktiver Prozeß hat eine E/A-Operation angefordert (oder sich selbst für eine bestimmte Zeit verdrängt), bevor seine Zeitscheibe abgelaufen war. Dies ist der einzige Zustandswechsel, den ein Prozeß selbst auslösen kann.
- Wakeup: blockiert --> bereit  
Das Ereignis, auf das der Prozeß gewartet hat ist eingetreten. Signal an den Prozeß

Zu jedem Prozeß legt das BS einen Prozeßsteuerblock (process control block = PCB) in der Prozeßtabelle ab, der alle notwendigen Informationen über einen Prozeß enthält, z. B.:

- PID (Process ID) - dies ist eine eindeutige ganze Zahl, über die der Prozeß im System identifiziert wird
- Prozeßzustand
- Verweise auf die dem Prozeß zugeteilten Speicherseiten
- Benutzerkennungen (die die Rechte des Prozesses bestimmen)
- Blockierursachen bei einem schlafenden Prozeß
- Identifikation empfangener, aber noch nicht bearbeiteter Signale
- etc.

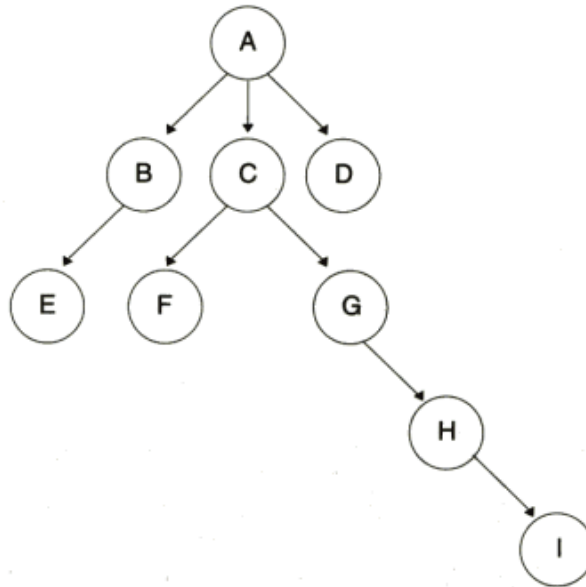
Zusätzlich zum PCB werden noch weitere, zum Prozeß gehörige Daten geführt, auf die er aber nur im Zustand "running" Zugriff hat. Dazu gehören:

- Eine Tabelle, in der die Reaktion des Prozesses auf jedes mögliche Signal festgelegt ist
- Verweis auf das zugeordnete Terminal
- Verweis auf das aktuelle Inhaltsverzeichnis
- Die Tabelle der Dateideskriptoren
- Eine Bitmaske, die die Zugriffsrechte von Dateien mitbestimmt, die vom jeweiligen Prozeß erzeugt werden
- etc.

## 2.3 Prozeßhierarchie und Prozeßrechte

Ein Prozeß kann einen neuen Prozeß starten (fork, spawn); ein solcher Prozeß heißt "Kindprozeß" oder "Sohnprozeß" (child process) und der Erzeuger-Prozeß wird "Elternprozeß" oder "Vaterprozeß" (parent process) genannt.

- Jeder Kindprozeß hat genau einen Elternprozeß
- Ein Elternprozeß kann mehrere Kindprozesse besitzen
- Eltern- und Kindprozeß können miteinander kommunizieren
- Wird ein Elternprozeß beendet, beenden sich normalerweise auch alle seine Kindprozesse



Die **Prozeßrechte** sollen anhand des Betriebssystems UNIX erläutert werden. Jeder Prozeß hat im Verlauf seiner Existenz zwei Benutzeridentifikationen:

Die "reale" Benutzer-ID bezeichnet den Benutzer, der für den Ablauf des Prozesses verantwortlich ist. Sie bleibt in der Regel konstant. Die "effektive" Benutzer-ID kann sich jedoch beliebig oft ändern und bestimmt jeweils die momentanen Rechte des Prozesses in Bezug auf Dateizugriffe und andere Mechanismen, die benutzerabhängigen Einschränkungen unterliegen.

Die Änderung der effektiven Benutzer-ID kann auf zweierlei Art erfolgen: Wenn der Prozeß ein Programm eines anderen Benutzers ausführen möchte, dann prüft das Betriebssystem zuerst seine Berechtigung dazu (über die entsprechenden Dateizugriffsrechte). Falls die Zugriffsrechte der Programmdatei außerdem das sogenannte "setuid-Bit" enthalten, dann wird die effektive Benutzer-ID des Prozesses mit der Eigentümerkennung des aufgerufenen Programms überschrieben, und der Prozeß hat demzufolge jetzt die Rechte des Eigentümers des Programms.

Die zweite Möglichkeit zur Veränderung der effektiven Benutzer-ID ist der explizite Aufruf der Systemfunktion `setuid()`. So ist es möglich, die effektive Benutzer-ID so zu verändern, daß sie den Wert der realen Benutzer-ID erhält oder den Wert der effektiven Benutzer-ID, die der Prozeß bei seiner Aktivierung von seinem Vater "erbt" hat.

## 2.4 Prozeß-Operationen des BS

Aus dem bisher gesagten kann man bestimmte Operationen ableiten, die das Betriebssystem zur Steuerung und Kontrolle von Prozessen ausführt. Die wichtigsten Prozeß-Operationen sind in der folgenden Tabelle zusammengefaßt.

## Betriebssysteme

Create Erzeugen eines Prozesses (z.B. Laden eines Programms)

- Vergabe einer PID, Anlegen eines PCB in der Prozeßtabelle
- Allokieren des benötigten Speichers
- Reservieren der benötigten Ressourcen
- Vergabe einer Priorität

Kill Löschen eines Prozesses

- Löschen aller Einträge aus den Systemtabellen
- Freigabe von Speicher und Ressourcen (z.B. Dateien schließen)
- Löschen aller Abkömmlinge (Kindp., Enkelp., usw.)

Suspend Suspendieren eines Prozesses

- Suspendierung normalerweise nur bei Systemüberlastung durch Prozesse höherer Priorität, Wiederaufnahme erfolgt sobald möglich.

Resume Wiederaufnehmen eines suspendierten Prozesses

- Suspendierung normalerweise nur bei Systemüberlastung durch Prozesse höherer Priorität, Wiederaufnahme erfolgt sobald möglich.

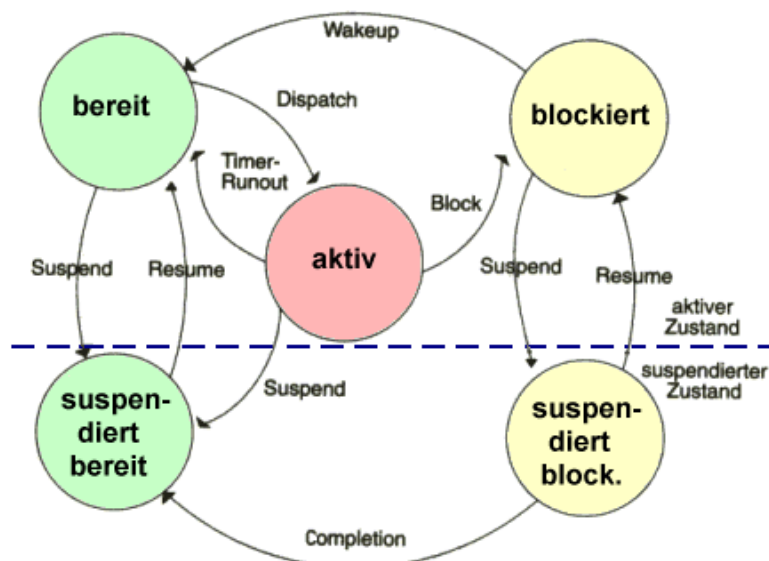
Block Blockieren eines Prozesses

Wakeup Aufwecken eines blockierten Prozesses

Dispatch CPU an einen Prozeß zuteilen

Change Priorität eines Prozesses ändern

In der Tabelle wird erstmals die Möglichkeit der "Suspendierung" eines Prozesses aufgeführt. Wird das System durch Prozesse höherer Priorität überlastet (kein Speicher mehr frei, keine Dateihandles mehr frei, Prozeßtabelle voll, etc.), müssen Prozesse aus dem Speicher entfernt und z. B. auf die Platte ausgelagert werden. Im einfachsten Fall reicht auch manchmal die Auslagerung eines Eintrags in der Prozeßtabelle in eine andere Tabelle. Sobald sich die Situation entspannt hat, wird der Prozeß wieder in den alten Stand zurückversetzt. Bei der Suspendierung wird man natürlich solche Prozesse zuerst auslagern, die sowieso auf ein Ereignis warten. Durch die Suspendierungsmöglichkeit erweitert sich das Diagramm der Prozeßzustände:



## 2.5 Prozeß-Synchronisation

In einigen Multitasking-Betriebssystemen teilen sich verschiedene Prozesse gemeinsame Betriebsmittel. Bei gleichzeitigem Zugriff zweier Prozesse auf diese Betriebsmittel kann es zu Inkonsistenzen der Daten kommen, die u. U. selten auftreten und sehr schwer aufzufassen sind. Kooperierende nebenläufige Prozesse müssen daher wegen der zwischen ihnen vorhandenen Abhängigkeiten miteinander synchronisiert (koordiniert) werden. Prinzipiell lassen sich zwei Klassen von Abhängigkeiten unterscheiden:

- Die Prozesse konkurrieren um die Nutzung gemeinsamer, exklusiv nutzbarer Betriebsmittel  
Beispiel: Zwei Prozesse greifen verändernd auf gemeinsame Daten zu. Der Zugriff zu den gemeinsamen Daten muß koordiniert werden, um eine Inkonsistenz der Daten zu vermeiden --> Sperrsynchrisation (gegenseitiger Ausschluß, mutual exclusion)
- Die Prozesse sind voneinander datenabhängig. Beispiel: Ein Prozeß erzeugt Daten, die von einem anderen Prozeß weiter bearbeitet werden sollen. Es muß eine bestimmte Abarbeitungsreihenfolge entsprechender Verarbeitungsschritte eingehalten werden --> Zustands- oder Ereignissynchronisation (z. B. Produzenten-Konsumenten-Synchronisation)

Ohne eine solche Sperrung entstehen Datenverluste, z. B. in folgenden Fall:

- Prozeß A und Prozeß B lesen ein Datenelement im Zustand X(0).
- A schreibt es mit dem Wert X(1) zurück,
- danach schreibt B seinen Wert X(2) zurück.

Damit bleibt die Änderung X(0) nach X(1) aus dem Prozeß A unberücksichtigt.

Oder auch bei folgendem Beispiel:

- Prozeß A informiert sich in der Spooler-Warteschlange über den nächsten freien Eintrag und notiert dessen Adresse, z. B. 7 in der Variablen `next_free_slot`. Danach wird ihm der Prozessor entzogen.
- Prozeß B findet die gleiche Adresse (7) und trägt sie in `next_free_slot` ein. Danach schreibt er an die Position 7 der Warteschlange den Namen der auszugebenden Datei und erhöht die Variable `next_free_slot` auf 8.
- Prozeß A erhält nun den Prozessor wieder zugeteilt und macht dort weiter, wo er unterbrochen wurde. Er findet in `next_free_slot` den Wert 7 und trägt den Namen der von ihm auszugebenden Datei in Position 7 der Spooler-Warteschlange ein und erhöht die Variable `next_free_slot` ohne zu bemerken, daß der Wert 8 dort schon eingetragen war.  
**Folge:** Prozeß A überschreibt den Dateinamen, den Prozeß B in der Spooler-Warteschlange auf Position 8 eingetragen hatte. Diese Datei wird vom Spooler "vergessen".

Solche Situationen heißen **zeitkritische Abläufe**. Programmabschnitte, die auf gemeinsam benutzte Daten zugreifen, heißen **kritische Abschnitte**.

Fehler in zeitkritischen Abläufen sind sehr schwer erkennbar, da sie nur sehr selten auftreten. Sie können nur durch wechselseitigen Ausschluß vermieden werden. Kritische Abschnitte treten überall auf, wo zwei oder mehr Prozesse auf einem Computer oder in einem Netz um mindestens eine Ressource konkurrieren, die sie schreibend benutzen wollen. Kritische Abschnitte können aber auch überall da auftreten, wo mehrere Prozesse um eine Ressource konkurrieren, die sie schreibend benutzen. Das trifft besonders auf Datenbankserver zu (z. B. Reservierungssysteme, zentrale Karteien, usw.).

*Vier Bedingungen für eine gute Lösung (nach Tanenbaum):*

## Betriebssysteme

1. Höchstens ein Prozeß darf sich in einem kritischen Abschnitt aufhalten. (Korrektheit)
2. Es dürfen keine Annahmen über Ausführungsgeschwindigkeit und Anzahl der Prozessoren gemacht werden.
3. Kein Prozeß, der sich in einem kritischen Abschnitt befindet, darf andere blockieren.
4. Kein Prozeß soll unendlich lange warten müssen, bis er in einen kritischen Bereich eintreten darf.

Die letzten beiden Punkte dienen der Stabilität, sie sollen Prozeßverklebungen verhindern.

### Beispiele für zeitkritische Abläufe

#### 1. Das Erzeuger-Verbraucher-Problem:

Der Erzeuger E stellt ein Produkt her und stellt es in einen begrenzten Pufferspeicher. Verbraucher V entnimmt dem Puffer ein Stück des Produktes, um es zu verbrauchen. Beides geschieht zu zufälligen Zeitpunkten. Der Puffer wird von beiden gemeinsam verwaltet. Solche Erzeuger-Verbraucher-Probleme treten beispielsweise bei Pipes auf (Ein Prozeß erzeugt Daten, der andere verarbeitet sie weiter.)

- Der Erzeuger muß zuerst prüfen, ob noch Platz im Puffer ist, bevor er ein Produkt ablegen kann.  
Er muß dann auch den Produktzähler `count` erhöhen. Ist der Puffer voll, muß er schlafen gehen.
- Der Verbraucher muß prüfen, ob der Puffer nicht leer ist, bevor er etwas entnimmt. Er muß dann `count` dekrementieren. Ist nichts im Puffer, muß er schlafen gehen.

Wenn der Erzeuger ein Produkt in den Puffer stellt, muß er den Verbraucher wecken. Analog muß der Verbraucher den Produzenten wecken, wenn er ein Produkt aus dem Puffer entnimmt. Tanenbaum verwendet dafür die Funktionen `SLEEP` und `WAKEUP`. Er zeigt eine Lösung für das Erzeuger-Verbraucher-Modell, die einen fatalen Fehler zuläßt:

```
#define N 100                /* Puffergröße */
int count = 0;              /* Tatsächlicher Pufferinhalt */

void producer (void)
{
    tinhalt item;
    while (1)                /* Endlosschleife */
    {
        produce_item (&item); /* Erzeuge 1 Stück */
        if (count == N) SLEEP(); /* Falls Puffer voll, schlafen */
        enter_item (item);     /* lege erzeugtes Stück in Puffer */
        count++;
        /* wenn Puffer vor dem Weiterzählen leer war, Verbraucher wecken */
        if (count == 1) WAKEUP(consumer);
    }
}

void consumer (void)
{
    tinhalt item;
    while (1)                /* Endlosschleife */
    {
        if (count == 0) SLEEP(); /* Falls Puffer leer, schlafen */
        remove_item (item);     /* entnehme dem Puffer ein Stück */
        count--;
        /* Pufferinhalt korrigieren */
        /* wenn Puffer vor Korrigieren voll war, Erzeuger wecken */
        if (count == N-1) WAKEUP(producer);
        consume_item (&item);  /* verbrauche 1 Stück */
    }
}
```

## Betriebssysteme

Die Funktionen sollen selbstständig laufende Programme repräsentieren, die beide zu beliebigen Zeitpunkt durch einen Scheduler-Eingriff unterbrochen oder wieder aktiviert werden können.

Wenn der Consumer schläft, weil der Puffer leer ist, muß man nicht davon ausgehen, daß der Puffer immer leer bleibt. Der Producer kann ja zwischendurch den Prozessor zugeteilt bekommen, etwas in den Puffer legen und den Consumer wieder wecken.

Umgekehrt schläft der Producer, wenn der Puffer voll ist. Während er schläft, kann der Consumer den Prozessor zugeteilt bekommen, etwas verbrauchen (so daß im Puffer wieder Platz wird) und den Producer wieder wecken.

Der Fehler tritt bei folgendem Szenario auf:

Der Puffer ist leer und der Verbraucher stellt das fest ( $\text{count} = 0$ ). Genau jetzt unterbricht der Scheduler den Verbraucher und startet den Produzenten. Dieser legt ein Produkt in den Puffer, setzt  $\text{count}$  auf 1 und startet WAKEUP. Wenn der Verbraucher vom Scheduler wieder die CPU zugeteilt bekommt, ist er noch wach. Der Weckruf verhallt ungehört, weil ja der Verbraucher nicht schläft. Der Verbraucher wertet die vorher festgestellte Tatsache " $\text{count} = 0$ " aus und geht schlafen. Es gibt für den Produzenten keinen Anlaß, nochmals zu wecken. Der Verbraucher wird nie mehr geweckt.

Ursache für diese Blockierung ist eine Prozeßunterbrechung im kritischen Abschnitt zwischen Erkennung der Bedingung, die zum Aufruf von SLEEP führt und dem SLEEP-Kommando selbst. Die gleiche Situation würde auftreten, wenn der Produzent zwischen der Erkenntnis, daß der Puffer voll ist ( $\text{free} = 0$ ) und dem Schlafengehen unterbrochen wird.

2. *Das Problem des schlafenden Friseurs:* Dieses Modell geht davon aus, daß beide nach dem Zeitscheibenprinzip nur abwechselnd handeln können.

Ein Friseur bedient, sobald er Zeit dafür hat, ankommende oder wartende Kunden. Der Warteraum ist beschränkt auf  $N$  Stühle.

- Der Friseur (= Prozessor) startet zu Arbeitsbeginn die Funktion `barbier()`. Wenn kein Kunde da ist, legt er sich schlafen.
- Kommt ein Kunde, prüft er die Bedingung  $\text{count} \geq N$ . Sind alle Stühle belegt, geht er wieder. Sonst bleibt er und erhöht  $\text{count}$  um eins. War  $\text{count}$  vorher 0, weckt er den schlafenden Friseur.
- Wenn der Friseur einen Kunden bedient hat, dekrementiert er  $\text{count}$ . Ist  $\text{count}$  dann 0 geworden, legt er sich wieder schlafen.

Die kritische Situation besteht darin, daß der Kunde zwischen der Dekrementierung von  $\text{count}$  und der Frage "Ist  $\text{count}$  gleich 0" ankommt und den Friseur wecken kann, obwohl der sich noch gar nicht schlafen gelegt hat.

### Lösungsversuche für das Problem der kritischen Abschnitte

1. Einfachste Lösung: Vor Eintritt in den kritischen Bereich alle Interrupts sperren und sie nach Verlassen des kritischen Bereichs wieder freigeben. Damit kann der Scheduler nicht während des kritischen Abschnitts den Prozeß unterbrechen.  
Nachteil: Kein verdrängendes Multitasking mehr. Der Anwenderprozeß kann den Scheduler blockieren (gewollt oder ungewollt durch einen Programmfehler).
2. **Verfahren mit gegenseitigem Ausschluß**  
Der Programmabschnitt, in dem ein Zugriff zu dem nur exklusiv nutzbaren Betriebsmittel (z. B. die gemeinsamen Daten) erfolgt, wird kritischer Abschnitt genannt. Es muß verhindert werden, daß sich zwei Prozesse gleichzeitig in ihren kritischen Abschnitten befinden.
  - ◆ Sperrvariable: Es wird eine logische Variable geführt, die mit 0 den Eintritt in den kritischen Bereich erlaubt und mit 1 sperrt. Der in den kritischen Bereich eintretende Prozeß muß dann vor seinem Eintritt prüfen, ob der Bereich frei ist, die Sperrvariable auf 1 setzen und nach Ende wieder freigeben.

## Betriebssysteme

Der Abschnitt von der Prüfung der Sperrvariablen bis zu ihrem Setzen ist selbst ein kritischer Abschnitt! Er ist zwar kürzer und damit die Konfliktwahrscheinlichkeit geringer, aber die Gefahr des Konfliktes ist nicht beseitigt! Es gibt bei vielen CPUs jedoch Befehle von der Form "teste und setze", bei denen der kritische Abschnitt innerhalb eines Maschinenbefehls "abgehandelt" wird. Damit sind folgende Lösungen möglich:

◆ "Aktives Warten":

```
bool v;
/* Warteschleife, bis Riegelvariable RV[i]= 0 ist */
do
    v = test_and_set(&RV[i]);
while (v == 0);
...
/* kritischer Abschnitt zur Datenmenge i */
...
RV[i] : = 1;
```

Die Ver- bzw. Entriegelung wird häufig mit den Funktionen LOCK und UNLOCK formuliert:

```
LOCK(RV[i]);
...
/* kritischer Abschnitt zur Datenmenge i */
...
UNLOCK(RV[i]);
```

Viele Computer, die für Mehrprozessorbetrieb konzipiert wurden, kennen den Maschinenbefehl "Test and Set Lock". Dabei wird ein Speicherwort aus einem Register gelesen und ein Wert ungleich Null hineingeschrieben. Für die TSL-Operation wird eine gemeinsame Variable namens "Flag" verwendet. Diese koordiniert den Zugriff. Beispiel:

```
enter_region: tsl register,flag kopiere flag ins Register und setze flag auf 1
               cmp register,#0   war flag Null?
               jnz enter_region  wenn nicht, Verriegelung gesetzt, Schleife

               ... kritischer Bereich ...

leave_region: mov flag,#0        flag auf 0 setzten
               ret               Rückkehr zum Aufrufer
```

Aktives Warten verbraucht CPU-Zeit durch Warteschleifen.

◆ "Striktes Alternieren":

Nur zwei Prozesse erlauben sich wechselweise den Eintritt in den kritischen Abschnitt mit einer logischen Sperrvariablen. Zum Beispiel dient die gemeinsame Sperrvariable turn zur Synchronisation zweier Prozesse. Es gilt:  $turn = i$  ( $i = 0,1$ )  $\rightarrow$  Prozeß  $P_i$  darf in den kritischen Bereich eintreten:

```
P0:
while (1)
{
    while (turn != 0) no_operation; /* warten */
    kritischer Bereich;
    turn = 1;
    unkritischer Bereich;
}
```

## Betriebssysteme

```
P1:
while (1)
{
while (turn != 1) no_operaion; /* warten */
kritischer Bereich;
turn = 0;
unkritischer Bereich;
}
```

Auch hier wird CPU-Zeit durch Warteschleifen verbraucht. Außerdem ist striktes Alternieren auch keine gute Lösung, wenn ein Prozeß wesentlich langsamer ist als der andere.

### 3. Semaphore

Bisher haben die Prozesse ihren Eintritt in den kritischen Abschnitt selbst gesteuert. Die folgenden Techniken erlauben es dem Betriebssystem, Prozessen den Zutritt zum kritischen Abschnitt zu verwehren. Im Zusammenhang mit Algol 68 entwickelte Dijkstra das Prinzip der Arbeit mit Semaphoren. Für jede zu schützende Datenmenge wird eine Variable (Semaphor) eingeführt (binäre Variable oder nicht-negativer Zähler).

Ein Semaphor (Sperrvariable, Steuervariable) signalisiert einen Zustand (Belegungszustand, Eintritt eines Ereignisses) und gibt in Abhängigkeit von diesem Zustand den weiteren Prozeßablauf frei oder versetzt den betreffenden Prozeß in den Wartezustand.

Beim Bemühen um unteilbare Ressourcen (z. B. den Prozessor) wird eine binäre Variable verwendet, bei N Teil-Ressourcen (z. B. Arbeitsspeicher-Segmente oder Plätze in einem Puffer) kommen Werte von 1 bis N vor. Die binären Semaphore werden auch "Mutexe" (von "Mutual exclusion") genannt, jene vom Typ Integer auch "Zähl-Semaphore".

Semaphore für den gegenseitigen Ausschluß sind dem jeweiligen exklusiv nutzbaren Betriebsmittel zugeordnet und verwalten eine Warteliste für dieses Betriebsmittel. Sie sind allen Prozessen zugänglich. Semaphore für die Ereignissynchronisation zweier voneinander datenabhängiger Prozesse sind diesen Prozessen direkt zugeordnet. Sie dienen zur Übergabe einer Meldung über das Ereignis zwischen den Prozessen.

Zur Manipulation und Abfrage von Semaphoren existieren zwei unteilbare, d. h. nicht unterbrechbare Operationen (Semaphor S):

#### ◆ P-Operation: Anfrage-Operation

```
if (s > 0)
s = s - 1; /* Prozeß kann weiterlaufen,
           Zugriff für andere Prozesse wird gesperrt */
else
/* der die P-Operation ausführende Prozeß wird "wartend";
   Eintrag des Prozesses in die vom Semaphor
   verwaltete Warteliste; */
```

#### ◆ V-Operation: Freigabe-Operation

```
if (Warteliste leer)
s = s + 1; /* Zugriff für andere, noch nicht wartende
           Prozesse wird freigegeben */
else
/* nächster Prozeß in Warteliste wird "bereit";
   Zugriff für wartenden Prozeß wird freigegeben */
```

Lösungsmöglichkeit für Erzeuger-Verbraucher-Synchronisation (Vorbereitung der Semaphore: start = 0; finish = 0):

## Produzent:

```

while (1)
{
  while (!buffer-full)
    write_into_buffer();
  signal(start); /* V-Operation */
  wait(finish); /* P-Operation */
}

```

## Konsument:

```

while (1)
{
  wait(start); /* P-Operation */
  while(!buffer-empty)
    read_from_buffer();
  signal(finish); /* V-Operation */
}

```

### 4. Ereigniszähler

Für einen Ereigniszähler E sind drei Operationen definiert:

- ◆ read (E) : Stelle Wert von E fest!
- ◆ advance (E) : Inkrementiere E (atomare Operation)
- ◆ await (E, v) : Warte bis E = v ist

Damit kann E nur wachsen, nicht kleiner werden! E sollte mit 0 initialisiert werden. Die hier gezeigte Produzent-Konsument Lösung verwendet die Operation read() nicht, andere Synchronisationsprobleme machen diese Operation dennoch notwendig. Hier arbeitet der Puffer als Ringpuffer.

Anmerkung: Mit % wird der Modulo-Operator gekennzeichnet (= Rest der ganzzahligen Division).

```

#define N 100 /* Puffergröße */

eventcounter inputcounter = 0, outputcounter = 0; /* Ereigniszaehler */
int inputsequence = 0, outputsequence = 0;

producer()
{
  tinhalt item;
  while (1)
  {
    produce(&item);
    inputsequence = inputsequence + 1;
    await(outputcounter, inputsequence-slots);
    buffer[(inputsequence - 1) % N] = item;
    advance(inputcounter);
  }
}

consumer()
{
  tinhalt item;

  while (1)
  {
    outputsequence = outputsequence + 1;
    await(inputcounter, outputsequence);
    item = buffer[(outputsequence - 1) % N];
    advance(outputcounter);
    consume(&item);
  }
}

```

Die Ereignis-Zähler werden nur erhöht, nie erniedrigt. Sie beginnen immer bei 0. In unserem Beispiel werden zwei Ereignis-Zähler verwendet. `inputcounter` zählt die Anzahl der produzierten Elemente seit dem Start. `outputcounter` zählt die Anzahl der konsumierten Elemente seit dem Start. Aus diesem Grunde muß immer gelten: `outputcounter <= inputcounter`. Sobald der Produzent ein neues Element erzeugt hat, prüft er mit Hilfe des `await`-Systemaufrufs, ob noch Platz im Puffer vorhanden ist. Zu Beginn ist `outputcounter = 0` und `(inputsequence - N)` negativ - somit wird der Produzent nicht blockiert. Falls es dem Produzenten gelingt N+1 Elemente zu erzeugen, bevor der

Konsument startet, muß er warten, bis `outputcounter = 1` ist. Dies tritt ein, wenn der Verbraucher ein Element konsumiert. Die Logik des Konsumenten ist noch einfacher. Bevor das m-te Element konsumiert werden kann, muß mit `await (inputcounter, n)` auf das Erzeugen des m-ten Elementes gewartet werden.

Auf die Probleme, die bei der Konkurrenz von Prozessen um exklusiv nutzbare Betriebsmittel auftreten, wird in [Kapitel 4.5](#) eingegangen.

## 2.6 Prozeß-Kommunikation

Bei Multitasking-Betriebssystemen spielt die Kommunikation bzw. Synchronisation zwischen den quasiparallel ablaufenden Prozessen eine herausragende Rolle. Die Gründe dafür sind vielfältig. Zum einen werden größere Softwaresysteme häufig als Systeme mit mehreren kooperierenden Prozessen gestaltet. Diese müssen normalerweise in ihren Abläufen synchronisiert werden. Ferner müssen häufig Daten von einem Prozeß zum anderen transferiert werden. Ein anderer Grund liegt im Problem der kritischen Abschnitte von Prozessen beim Zugriff auf nicht gemeinsam benutzbare Betriebsmittel. Auch hier sind Synchronisationsmethoden erforderlich, die den gegenseitigen Ausschluß gewährleisten (siehe oben: Semaphore). Einige Möglichkeiten der Prozeß-Kommunikation (Interprocess Communication (IPC)) sind:

- Kommunikation über **gemeinsame Speicherbereiche**  
Prozesse können gemeinsame Datenbereiche, Variablen etc. anlegen und gemeinsam nutzen.
- Kommunikation über **gemeinsame Dateien**  
Prozesse schreiben in Dateien, die von anderen Prozessen gelesen werden.
- Kommunikation über **Pipes**  
Dies sind unidirektionale Datenkanäle zwischen zwei Prozessen. Ein Prozeß schreibt Daten in den Kanal (Anfügen am Ende) und ein anderer Prozeß liest die Daten in der gleichen Reihenfolge wieder aus (Entnahme am Anfang). Realisierung im Speicher oder als Dateien. Lebensdauer in der Regel solange beide Prozesse existieren.
- Kommunikation über **Signale**  
Signale sind asynchron auftretende Ereignisse, die eine Unterbrechung bewirken (--> Software Interrupt). In der Regel zur Kommunikation zwischen BS und Benutzerprozeß.
  - ◆ Auslösung vom Benutzer (z.B. Tastendruck)
  - ◆ Auslösung durch Programmfehler (z.B. Division durch 0)
  - ◆ Auslösung durch andere Prozesse (z.B. Plattenzugriff durch BS-Dienstroutine, "Daten sind bereit")
  - ◆ ...
- Kommunikation über **Nachrichten** (Botschaften, Messages)  
Nachrichten werden vom BS verwaltet. Dieses stellt eine für die beteiligten Prozesse gemeinsam nutzbare Transportinstanz (z. B. "Mailbox") zur Verfügung. Auf diese greifen die Prozesse über bestimmte Transport-Funktionen des BS (Systemaufrufe) zu. Prozeß A sendet z. B. eine Botschaft an Prozeß B, indem er sie in der Mailbox ablegt (`send (message) ;`). Der Prozeß B holt die Nachricht dann von der Mailbox ab (`receive (message) ;`).
- Kommunikation über **Streams**  
Streams ermöglichen die Kommunikation über Rechnernetze. Logisch gesehen haben Streams dieselbe Aufgabe wie die lokalen Pipes.
- Kommunikation über **Prozedurfernaufrufe** (remote procedure call)  
Ein Prozeß ruft eine in einem anderen Prozeß angesiedelte Prozedur auf (also über seine Adreßgrenzen hinweg). Besonders für Client-Server-Beziehungen geeignet.

Selbst bei sehr einfachen Betriebssystemen ist eine IPC notwendig, da zumindest eine Kommunikation zwischen einem Prozeß und dem Scheduler möglich sein muß.

## 2.7 Prozeß-Scheduling

In Multitasking-Betriebssystemen ist ein spezieller Prozeß notwendig, der aus den bereiten Prozessen den nächsten aktiven Prozeß auswählt. Sobald mehr als ein Prozeß den Zustand "bereit" besitzt, muß der Scheduler des Betriebssystems entscheiden, welcher Prozeß die CPU erhält (wir gehen zur Vereinfachung von einem System mit nur einem Prozessor aus). Kriterien für einen guten Scheduler sind:

- **Gerechtigkeit:** Jeder Prozeß erhält einen "gerechten" CPU-Anteil
- **Effizienz:** Die CPU sollte immer zu 100% ausgelastet sein
- **Antwortzeit:** Minimale Antwortzeit für interaktive Benutzer
- **Verweilzeit:** Angemessen kurze Verweilzeit für Batch-Aufträge
- **Durchsatz:** Möglichst viele Aufträge/Zeitraum abarbeiten
- **Terminerverfüllung:** Bereitstellung bestimmter Ergebnisse zu festgelegten Zeitpunkten

Bei Multitasking-Betriebssystemen werden zwei Grundsysteme für das Scheduling unterschieden:

- **kooperatives Multitasking** (non preemptive)  
Der aktive Prozeß gibt von sich aus die CPU zu einem geeigneten Zeitpunkt frei. Es ist nur ein geringer Verwaltungsaufwand nötig. Es besteht jedoch die Gefahr, daß ein "unkooperativer" oder fehlerhafter Prozeß alle anderen Prozesse blockiert.
- **verdrängendes Multitasking** (preemptive) Der Scheduler kann einem Prozeß die CPU entziehen (z. B. ausgelöst durch einen Timer-Interrupt). Dadurch kann die Bearbeitung dringlicherer Aufgaben jederzeit begonnen werden (z. B. bei Echtzeit-BS). Ein fehlerhafter Prozeß kann das System nicht blockieren.  
Anstoß für den Prozeßwechsel durch Verdrängung:
  - ◆ **Zeitgesteuerte Strategien**  
Jeder Prozeß erhält die CPU für eine bestimmte Zeitspanne (Zeitscheibe). Danach wird die CPU dem nächsten Prozeß zugeteilt (Zeitscheibenverfahren, round robin).
  - ◆ **Ereignisgesteuerte Strategien**  
Ein Prozeßwechsel findet statt, wenn ein Ereignis (z. B. ein Hardwareinterrupt) einen anderen Prozeß benötigt. Hier werden allgemein den einzelnen Prozessen Prioritäten zugeordnet, die sich dynamisch ändern. Ein bestimmtes Ereignis verleiht "seinem" Prozeß eine höhere Priorität.

### Scheduling-Strategien

Bei kooperativem Multitasking oder ereignisgesteuerten Schedulingern wird die Zuteilungsstrategie über Prioritäten gesteuert, wobei man beim kooperativen System von relativem Vorrang spricht (der erst nach Freigabe der CPU durch den aktiven Prozeß wirksam wird) und beim ereignisgesteuerten System vom absoluten Vorrang (der sofort zum Prozeßwechsel führt).

Die Zeitscheibensteuerung kann als Sonderfall der Ereignissteuerung betrachtet werden, das Ereignis ist in diesem Fall der Ablauf der zugeteilten Zeitscheibe.

Einige Strategien, die in der Praxis verwendet werden, sind:

- Wer zuerst kommt, wird zuerst bedient (first come, first served):  
Verteilung der Prioritäten nach Ankunftszeit, ohne Vorrechte. Kommen zwei Prozesse genau gleichzeitig, wird eine zufällige Auswahl getroffen. Gute Systemauslastung, aber schlechtes Antwortzeitverhalten (lang laufende Prozesse behindern Kurzläufer). Einfach zu implementieren.
- Zeitscheibenverfahren (round robin):  
Jeder Prozeß erhält eine feste Zeitspanne (time slice) zugeordnet. Nach Ablauf dieser Zeitspanne wird er verdrängt und der nächste Prozeß erhält die CPU --> zyklische Zuteilung.

## Betriebssysteme

Alle Prozesse haben immer die gleiche Priorität. Die Zeitspanne kann konstant sein oder abhängig von der Prozessorbelastung variieren. Kurze Antwortzeiten bei kleinen Zeitscheiben, aber dann höhere Verluste durch die häufigen Prozeßwechsel.

- **Prioritätssteuerung:**  
Jedem bereiten Prozeß wird eine Priorität zugeordnet. Vergabe der CPU in absteigender Priorität. Ein Prozeß niedrigerer Priorität kann die CPU erst erhalten, wenn alle Prozesse höherer Priorität abgearbeitet sind. Ein bereit werdender Prozeß höherer Priorität verdrängt einen aktiven Prozeß niedrigerer Priorität. Alle Prozesse gleicher Priorität werden i.a. in jeweils einer eigenen Warteschlange geführt. Realisierung mehrerer unterschiedlicher Verfahren, z. Teil gemischt mit anderen Strategien, z. B.
  - ◆ **Reine Prioritätssteuerung:** Prozesse gleicher Priorität werden nach der Eingangsreihenfolge abgearbeitet (z. B. in Echtzeit-BS)
  - ◆ **Prioritätsgesteuerung mit unterlagertem Zeitscheibenverfahren:** Prozesse gleicher Priorität werden nach dem Zeitscheibenverfahren abgearbeitet
  - ◆ **Dynamische Prioritätsvergabe:** Die Priorität auf die CPU wartender Prozesse wird allmählich erhöht
  - ◆ **Mehrstufiges Herabsetzen (multilevel feedback):** Festlegung einer maximalen Rechenzeit für jede Prioritätsstufe. Hat ein Prozeß die max. Rechenzeit seiner Priorität verbraucht, bekommt er die nächstniedrigere Priorität, bis er die niedrigste Stufe erreicht hat.
- Es gibt noch zahlreiche weitere Scheduling-Strategien.

In Dialogsystemen wird normalerweise Round Robin verwendet, um den Benutzern akzeptable Antwortzeiten zu bieten. Bei Batchbetrieb und gemischten Systemen kommen oft Kombinationen der o. g. Strategien vor - z. B. getrenntes Scheduling für Dialog- und Batchbetrieb.

## 2.8 Beispiele

- **Windows bis Version 3.0**  
Jeder Prozeß erhält eine feste Zeitscheibe. Die Prozesse können untereinander nicht kommunizieren (außer auf dem Umweg über das Betriebssystem). Ereignisse (z. B. Mausbewegung) werden von Betriebssystem bearbeitet und den Prozessen gemeldet. Eigentlich nur die Vorstufe eines Multitasking-BS --> Prozeß-Swapper.
- **Windows ab Version 3.1**  
Kooperatives Multitasking. Ein Prozeß kann eine "öffentliche" Nachrichten senden, die dann von einem anderen Prozeß aufgenommen und beantwortet werden kann (Client-Server-Prinzip). Für Ereignisse (z. B. Mausbewegung) existieren jeweils einzelne Prozesse. Der BS-Kern stellt nur eine Schnittstelle für Systemaufrufe zur Verfügung.
- **Windows NT/2000/XP**  
Prinzipielle Arbeitsweise wie bei Windows ab 3.1. Jedoch überwacht der BS-Kern die Kommunikation der Prozesse und kann sie gegebenenfalls verhindern. Zur besseren Kooperation können einzelne Programme in mehrere Prozesse aufgeteilt werden, die wieder untereinander kommunizieren ("Multithreading", das englische Wort "Thread" bedeutet "Faden" - die Teilprozesse eines Programms sind sozusagen über ihre Kommunikationsverbindung "aufgefädelt").
- **Unix**  
Zeitscheibenverfahren mit Prioritätssteuerung. Diverse Möglichkeiten der IPC. Der BS-Kern verarbeitet alle Ereignisse und "weckt" den Prozeß auf, dem das Ereignis zugeordnet ist. Zeitscheiben sind je nach Bedarf unterschiedlich lang. Je nach Variante von Unix gibt es auch Multithreading. Zusätzlich ist ein Multiuserbetrieb möglich.

Realzeitbetriebssysteme arbeiten in der Regel mit Zeitscheibenverfahren, wobei zusätzliche Bedingungen hinzukommen. Ereignisse (in der Regel Hardware- oder Software-Interrupts) müssen

## Betriebssysteme

innerhalb einer bestimmten Sollzeit bearbeitet werden (Echtzeitbedingung). Daher findet sich hier häufig eine Aufteilung der Programme in einzelne Threads (bei Realzeitbetriebssystemen auch oft "Task" genannt).



[Zum vorhergehenden Abschnitt](#)



[Zum Inhaltsverzeichnis](#)



[Zum nächsten Abschnitt](#)

---

*Copyright © FH München, FB 04, Prof. Jürgen Plate*

*Letzte Aktualisierung: 12. Apr 2005*



## Einführung in Betriebssysteme

*von Prof. Jürgen Plate*

---

# 3 Speicherverwaltung

Der Arbeitsspeicher ist ein Betriebsmittel, ohne das kein Programm zur Ausführung kommen kann. Eine zentrale Aufgabe eines jeden BS ist deshalb die Speicherverwaltung. Sie organisiert die Zuweisung von Hauptspeicher (main memory) an Benutzeraufträge bzw. Tasks, und zwar sowohl Programmspeicher als auch Datenspeicher. Sie muß darüber Buch führen, welche Speicherbereiche gerade frei und welche belegt sind.

Vom Beginn der "Rechnergeschichte" bis heute kann das Phänomen beobachtet werden, daß der Bedarf an Arbeitsspeicher immer größer ist als der verfügbare physische Speicher. Fast immer ist der verfügbare Speicher zu klein, um ein großes Programm oder mehrere Programme gleichzeitig aufzunehmen. Im Lauf der verschiedenen Rechnergenerationen wurden eine Reihe von Lösungen für dieses Problem entwickelt.

Die moderneren Methoden gehen dabei von der Tatsache aus, daß zu einem beliebigen Zeitpunkt nur auf wenige Speicherplätze tatsächlich zugegriffen wird, während die anderen nur für einen späteren Zugriff (der u.U. nie erfolgt) bereitstehen. Die Grundidee ist also eine Erweiterung der Speicherkapazität unter Zuhilfenahme externer Massenspeicher, von denen bei Bedarf die benötigten Informationen in den Arbeitsspeicher geladen werden. In diesem Fall muß die Speicherverwaltung in der jeweiligen Situation festlegen, welche Programmabschnitte geladen werden sollen und welche auf dem Hintergrundspeicher bleiben sollen. Insbesondere beim Multiprogramming und beim Multitasking-Betrieb entstehen für die Speicherverwaltung vielfältige Aufgaben, denn es müssen sich mehrere unabhängige Programme im Hauptspeicher befinden, die u.U. häufig aus- und eingelagert werden müssen. Zu den Aufgaben der Speicherverwaltung gehört es auch, Mechanismen für den Schutz vor unbefugten Speicherzugriffen aus anderen Programmen bereitzustellen. Die folgende Auflistung von Speicherverwaltungsprinzipien beginnt mit den einfachsten - meist älteren und endet bei den heute üblichen Methoden des Memory-Management" in Universalrechnern.

## 3.1 Direkte Adressierung

Einfache universelle Betriebssysteme für den Einprogrammbetrieb verwalten den gesamten Speicher i. a. als einen Block, der aus einem Systembereich für das Betriebssystem und einem Benutzerbereich besteht. Das aktuelle Programm belegt einen zusammenhängenden Teil des Benutzerbereichs. Der restliche Speicher ist frei: "Zusammenhängende Einzelzuteilung" oder "single contiguous allocation".

- **Dedizierte Systeme** Da der Beginn des Benutzerbereichs in einem solchen System festliegt, können die Programme gleich so geschrieben, Übersetzt und gebunden werden, daß sie genau in diesen Speicheradressen ablauffähig sind. Die benötigten Werkzeuge werden dadurch sehr einfach. Dies war zu Beginn der Mikroprozessoranwendungen das übliche Verfahren und reicht auch heute noch für einfache Problemstellungen aus.
- **Universelle Systeme** Die wichtigsten Komponenten solcher einfachen Betriebssysteme sind ein sogenannter "Job-Monitor", der eine Job-Warteschlange verwaltet und ein "Lader", der das zu startende Benutzerprogramm ggf. zusammen mit (Teilen) einer Programmbibliothek aus einem Hintergrundspeicher lädt. Da alle Adressen der geladenen Programme (bzw. Programmteile) bereits im Objektcode als absolute Adressen vorliegen, nennt man einen solchen Lader auch "Absolutlader".

Es gilt:

- Beim **Einprogrammbetrieb** befindet sich nur ein einziger Prozeß im Speicher, er darf den gesamten Speicher benutzen. Diese Methode ist heute nicht mehr üblich. Selbst beim Einprogrammbetrieb teilen sich Betriebssystem und Anwenderprozeß den Speicher. Lediglich

Mikroprozessorsteuerungen bilden eine Ausnahme.

- Beim **Mehrprogrammbetrieb** werden mehrere Prozesse in den Speicher geladen und über den Scheduler der CPU zugeteilt.

### **Problem beim Mehrprogrammbetrieb:**

Ein Prozeß darf den Speicherbereich anderer Prozesse nie beeinflussen. Zu diesem Zweck sind Hard- und Softwaremaßnahmen nötig (Speicherschutz). Der Mehrprogrammbetrieb erhöht zudem die CPU-Auslastung. Wenn ein Prozeß auf das Ende einer E/A-Operation wartet, läuft die CPU im Einprogrammbetrieb leer. Im Mehrprogrammbetrieb können in dieser Zeit andere Prozesse rechnen. Der einfachste Weg ist die Aufteilung in feste Teile (Partitionen), die nicht notwendig gleich groß sein müssen. Zusätzlich zur Partitionierung sind Maßnahmen zur Relokation (Verschiebung) der Programme beim Laden und zum Speicherschutz notwendig (siehe unten).

Prozesse dürfen nur auf Bereiche innerhalb der Partition zugreifen (gilt für Code und Daten!), da sonst andere Prozesse unzulässig beeinflußt werden könnten. Die Speicherverwaltungseinheit muß also entsprechende Schutzfunktionen besitzen und Zugriffe auf "verbotene" Bereiche an den Betriebssystem-Kern melden. Eine Lösung besteht beispielsweise in zwei zusätzlichen Registern des Prozessors, Base- und Limit-Register. Das Baseregister enthält die Startadresse der dem aktiven Prozeß zugeordneten Partition und das Limitregister deren Länge. Alle Adressierungsvorgänge im Programm erfolgen relativ zum Base-Register. Eine Änderung der beiden Register erfolgt durch den Scheduler.

## **3.2 Verschiebung (Relocation)**

Bei der "Zusammenhängenden Einzelzuteilung" entsteht ein Adressierungsproblem für evtl. mitzuladende Bibliotheksroutinen wenn die Benutzerprogramme unterschiedlich lang sind, denn dann kommen diese Bibliotheksprogramme an unterschiedlichen Adressen zu liegen. Eine mögliche Lösung ist ein eigener fester Bereich für diese Routinen, was jedoch dazu führt, daß der Speicher zerstückelt wird und damit unnötig viel nicht genutzter Speicher entsteht.

Die flexiblere Lösung besteht darin, die Bibliotheksroutinen so aufzubereiten, daß die Festlegung der von ihnen benötigten Speicherplätze bis zum Ladezeitpunkt aufgeschoben werden kann. Man benötigt dazu einen "verschiebenden Lader" (Relocating Loader), der dann auch gleich zum Laden der Benutzerprogramme verwendet werden kann.

Die Aufbereitung der Programme besteht darin, die Adreßteile der Maschinenbefehle als absolute bzw. relative Adressen zu kennzeichnen. Somit besteht die Aufgabe des verschiebenden Laders darin, zu den relativen Adressangaben nur noch einen konstanten Offset zu addieren (die Lade-Startadresse des Programms) und diese nun absolute Adressangabe abzuspeichern; Voraussetzung ist natürlich, daß die Programme so geschrieben sind, als würden sie ab Adresse Null im Speicher zu liegen kommen. Es wird also zum Ladezeitpunkt jede relative Adresse in eine absolute umgeformt, wodurch der Ladvorgang mehr Zeit in Anspruch nimmt. Das geladene Programm kann nicht mehr verschoben werden, wenn es sich im Hauptspeicher befindet.

Eine weitere Steigerung der Flexibilität ist dann gegeben, wenn die Umsetzung in effektive Adressen nicht zum Ladezeitpunkt, sondern erst unmittelbar bei Ausführung des Befehls vorgenommen wird. Voraussetzung dafür ist ein Prozessor, der entsprechende Adressierungsarten der Befehle erlaubt. Dazu muß ein Adreßrechenwerk im Prozessor vorhanden sein, das entweder den Inhalt eines Basisadressregisters (geladen durch das Ladeprogramm; "basisregister-relative Adressierung") oder den Inhalt des Program-Counters addiert ("position independent code", "relocatable code" durch "PC-relative Adressierung"). Die Adressangaben bei den Befehlen werden also nicht mehr vom Lader modifiziert, der Ladevorgang läuft wesentlich schneller ab. Das Programm kann nach dem Laden

noch verschoben werden.

### 3.3 Overlay-Technik

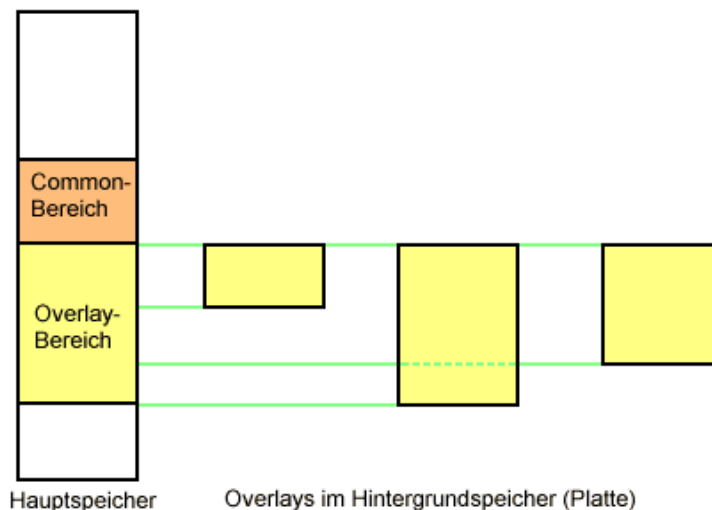
Um ein Programm ausführen zu können, das einschließlich seiner Daten nicht im Speicher unterzubringen ist, kann es in Teile zerlegt werden, die nicht ständig gleichzeitig im Speicher vorhanden sein müssen. Diejenigen Teile, die sich nicht im Arbeitsspeicher befinden, werden nachgeladen sobald sie benötigt werden. Weil die Entscheidung für das Nachladen dynamisch zur Laufzeit getroffen wird, spricht man hier von dynamischem Laden. Dabei muß zwangsläufig ein vorher im Speicher befindlicher Programmteil überschrieben (überlagert) werden, weshalb man diese Methode auch als "Overlay-Technik" bezeichnet.

Die Zerlegung eines Programms in Teile (Segmente), welche nicht gleichzeitig im Arbeitsspeicher sein müssen, muß der Programmierer vornehmen, weil es für den Compiler und den Linker keine praktikable Möglichkeit gibt, dies automatisch zu tun. Nur der Programmierer kann vorhersagen, in welcher Kombination verschiedene Programmteile zusammenarbeiten werden und sich daraufhin eine entsprechende Overlay-Struktur überlegen. Eine ungünstige Struktur wird zu unnötig vielen verlangsamen Ladevorgängen führen.

Eine Overlay-Struktur besteht aus

- einem speicherresidenten Programmteil (root program),
- aus mehreren Unterprogrammen, die sich gegenseitig überlagern,
- und aus gemeinsamen Daten (Common), die von Aufruf zu Aufruf erhalten bleiben sollen.

Der Speicheranteil, der für die Überlagerung von Programmabschnitten vorgesehen ist, wird als transienter Bereich bezeichnet.



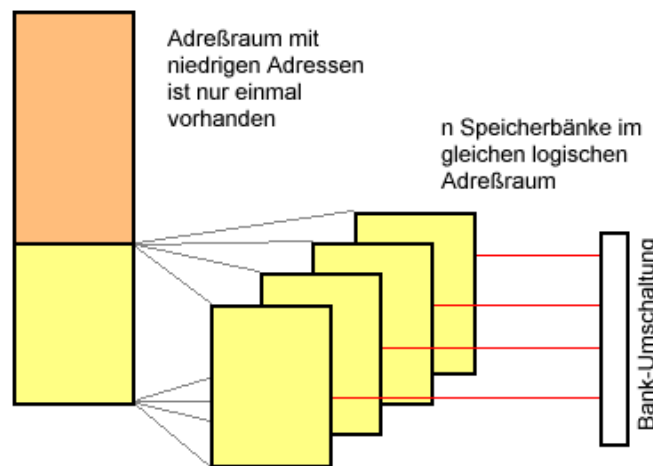
Eine ähnliche Technik, die ohne gemeinsames Hauptprogramm auskommt, bezeichnet man als "Chaining" (Verketteten). Eigenständige Programmmodule rufen sich gegenseitig auf und werden von diesen vollständig überlagert (typisch für BASIC).

Wenn aufgrund eines sehr ungünstigen Verhältnisses zwischen Programmgröße und Arbeitsspeichergröße ein mehrfach geschichtetes Überlagern nötig wird - innerhalb eines Überlagerungssegmentes wird wieder überlagert usw. - dann kann der Arbeitsaufwand für die Erstellung einer effizienten Overlay-Struktur erheblich werden!

### 3.4 Speicherbank-Umschaltung (bank-switching)

Bei häufigen Aufrufen von Overlay-Segmenten wird die Overlay-Technik langsam. Deshalb wird speziell bei kleineren dedizierten Systemen mit 8-Bit-Mikrocomputern (meist nur 64 KByte-Adreßraum!) das einfache Adreßraum-Erweiterungsverfahren der Speicherbank-Umschaltung häufig angewandt.

Dabei werden mehrere Speicherbänke von jeweils bis zu 64 KByte im Rechner implementiert. Zu jedem Zeitpunkt ist nur eine Speicherbank freigegeben und somit für Programm und Daten zugänglich. Die Umschaltung auf eine andere Speicherbank erfolgt über einen Speicherbank-Selektor (realisiert als Parallel-AusgabeBaustein), der z. B. mit Hilfe eines Ausgabe-Befehls angesprochen werden kann. Zu Kommunikationszwecken kann ein Speicherbereich allen Speicherbänken gemeinsam zugeordnet werden (Common).



In Multitasking-Systemen können so z. B. die Task-Adreßräume unabhängig voneinander verwaltet werden, wenn jeder Task eine eigene Speicherbank zugewiesen wird. Dadurch wird aber zugleich die Zahl der möglichen Tasks durch die vorhandenen Speicherbänke begrenzt. Zusätzlicher Vorteil bei ungünstigen Umgebungsbedingungen: auf diese Weise wird ein rotierender magnetischer Massenspeicher vermieden!

### 3.5 Virtuelle Speicherverwaltung

Man spricht von virtueller Speichertechnik, wenn der Speicheradreßraum, den die Befehle des Prozessors referieren, getrennt ist von realen Adreßraum des Arbeitsspeichers, in dem sich das Programm bei der Abarbeitung befindet.

Der Adreßraum, auf den sich die Programmbefehle beziehen, ist der logische Adreßraum. Der Adreßraum des realen Arbeitsspeichers ist der physische Adreßraum. --> Programme können unabhängig vom physischen Adreßraum geschrieben werden.

Der logische Adreßraum beschreibt einen gedachten, nicht real vorhandenen Arbeitsspeicher, den man als virtuellen Speicher bezeichnet.

- logischer Adreßraum = virtueller Adreßraum
- logische Adresse = virtuelle Adresse

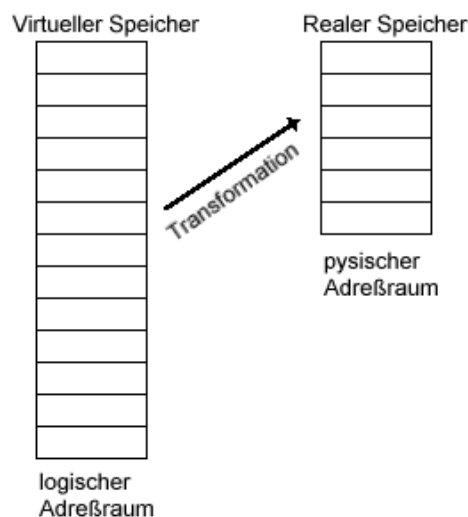
## Betriebssysteme

Normalerweise ist der logische Adreßraum größer als der physische Adreßraum (meist sogar sehr viel größer). Der virtuelle Speicher wird auf der Platte abgebildet. Zur Ausführung muß ein Programm in den Arbeitsspeicher geladen werden. Wegen der sequentiellen Abarbeitung werden innerhalb eines bestimmten Zeitintervalls nicht alle Teile des Programms und analog auch nicht der gesamte Datenbereich des Programms benötigt.

Es reicht also aus, nur die jeweils benötigten Programm- und Datenbereichs-Teile (= "working set") im Arbeitsspeicher zu halten. Die Programme und Daten werden in einzelne Teilabschnitte zerlegt, die dann nur bei Bedarf (wenn sie von der CPU benötigt werden) in den Speicher geladen werden.

- Programme und Datenbereiche sind in ihrer Länge nicht durch die reale Größe des Arbeitsspeichers begrenzt
- Es können gleichzeitig mehrere Programme ausgeführt werden, deren Gesamtlänge die Arbeitsspeicher-Größe überschreitet.

Zur Abarbeitung der einzelnen Befehle des Programms ist eine Transformation (Abbildung) der logischen Adressen in physische Adressen erforderlich.



Die logische Adresse in den Befehlen bleibt auch nach dem Laden in den Arbeitsspeicher unverändert erhalten. Die Transformation erfolgt erst bei der Befehlsausführung --> dynamische Adreßtransformation (DAT). Liegt eine Adresse in einem Programm- oder Datenabschnitt, der sich nicht im Arbeitsspeicher befindet, wird der entsprechende Abschnitt nachgeladen.

Zur Realisierung der virtuellen Speicherverwaltung muß eine Kombination aus Hard- und Software eingesetzt werden. Das Nachladen der Programm- und Datenabschnitte besorgt ein speziell konzipiertes Betriebssystem. Grundvoraussetzung für die Realisierung ist die Fähigkeit der CPU, einen laufenden Befehl zu unterbrechen (beim Nachladen) und nach erfolgtem Nachladen den begonnenen Befehl erneut aufzusetzen und dann vollständig auszuführen. Die virtuelle Speichertechnik ist für den Anwender vollkommen transparent. Sowohl das Nachladen von Programm- und Datenabschnitten als auch die Adreßumsetzung geschieht automatisch und braucht bei der Anwendungsprogrammierung nicht berücksichtigt zu werden. Die Aufteilung des Speichers kann erfolgen als:

- Segmentierung
- Seitenadressierung (Paging)

## Segmentierung

Der logische Adreßraum wird in Abschnitte variabler Größe gemäß den logischen Einheiten des Programms (Unterprogramme, Datenbereiche, etc.) unterteilt. Die Abschnitte nennt man Segmente. Die minimale/maximale Segmentgröße hängt vom jeweiligen System ab (typisch: 256 Byte bis 64 KByte).

Die logische Adresse besteht aus 2 Teilen:

- Segment-Nummer (höchstwertiger Teil)
- Wortadresse (Offset, Displacement; niederwertiger Teil; Adresse relativ zum Segmentanfang).

Für die in den Arbeitsspeicher geladenen Segmente ist in einer Segmenttabelle die jeweilige reale Anfangsadresse des Segments festgehalten (Basisadresse des Segments). Im Multiprogramming-Betrieb werden für die verschiedenen "Jobs" meist jeweils eigene Tabellen bereitgestellt damit mit den Segment-Nummern verschiedener Jobs keine Verwechslung möglich ist. Somit muß vor dem Segmenttabellenzugriff noch der Inhalt eines jobspezifischen Segmenttabellenregisters zur Segmentnummer addiert werden. Bei jedem Umschalten der Jobs (z. B. wegen "Verdrängen" einer Task am Zeitscheibenende) muß deshalb auch das Segmenttabellenregister umgeladen werden! Im allgemeinen enthält die Segmenttabelle auch Angaben über die Größe der Segmente (Angabe der letzten physischen Adresse des Segments oder der Segmentgröße direkt). Dadurch können fehlerhafte Zugriffe, die aus dem Segment herausführen, erkannt und verhindert werden.

Weiterhin sind in der Segmenttabelle jedem Segment noch Zustands- und Zugriffsinformationen zugeordnet (Erkennen nicht geladener Segmente, Verhinderung unberechtigter Zugriffe). Das Segment ist die kleinste Austauschereinheit. Falls kein Speicherplatz zum Nachladen mehr frei ist, muß ein vorhandenes, derzeit nicht benötigtes Segment entfernt werden ("demand segment swapping").

Probleme (gleichzeitig Nachteile der Segmentierung):

- Speichersplitterung ("externe Fraktionierung"): Zwischen den im Arbeitsspeicher befindlichen Segmenten sind nicht belegte Lücken vorhanden, die entstehen, wenn ein Segment gegen ein kleineres Segment ausgetauscht bzw. entfernt wird.
- Es kann Fälle geben, bei denen der verfügbare zusammenhängende Speicherraum nicht groß genug ist, um ein neu zu ladendes Segment aufzunehmen, obwohl insgesamt genügend freier Speicherplatz vorhanden ist.
- Neuordnung der Arbeitsspeicher-Belegung durch das Betriebssystem wird notwendig (Zusammenschieben der Segmente).
- Umständlicher Austauschalgorithmus: Das zugehörige Systemprogramm belegt selbst viel Speicherplatz, zusätzlicher Zeitbedarf für die Ausführung.

## Seitenadressierung (Paging)

Logischer und physischer Adreßraum werden in Abschnitte gleicher Länge eingeteilt, die man Seiten (Pages) nennt (typische Seitengrößen: 0,5 KByte ... 4 KByte). Eine Seite stellt die Austauschereinheit dar. Das Laden bzw. Nachladen einer Seite erfolgt bei Bedarf, d.h. wenn eine in dieser Seite liegende Adresse referiert wird; "demand paging". Falls noch physische Seiten frei sind, wird einer neu zu ladenden logischen Seite die nächste freie physische Seite zugewiesen. Sind alle physischen bereits besetzt, muß eine logische Seite ausgelagert werden. --> Seitenwechsel. Verantwortlich hierfür ist das Betriebssystem-Programm "Seiten-Supervisor". Ein Programm kann seitenweise gestückelt im Arbeitsspeicher stehen, also ohne Rücksicht auf irgendwelche logischen Grenzen an Seitengrenzen geteilt. Die logische Adresse wird zerlegt in

- logische Seitennummer (Seitenadresse, höherwertiger Teil)

## Betriebssysteme

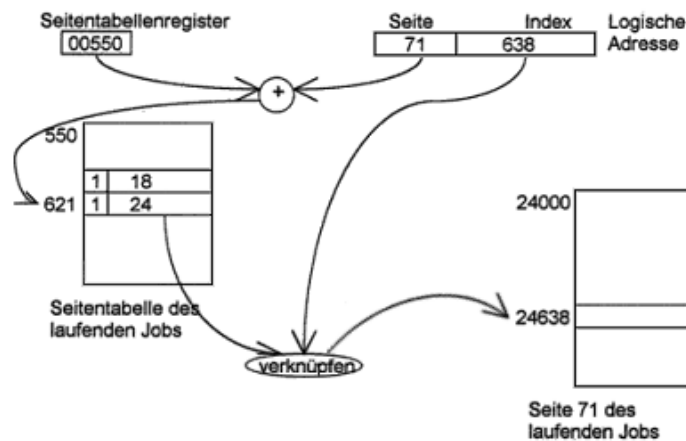
- Wortadresse (Zeilenadresse, niederwertiger Teil).

Die Wortadresse ist die relative Adresse zum Seitenanfang. Sie kann unverändert in physische Adresse übernommen werden. Die Physische Adresse besteht auch aus zwei Teilen:

- physische Seitennummer (Seitenadresse, höherwert. Teil),
- Wortadresse (niederwertiger Teil), die unverändert aus der logischen Adresse übernommen wird.

Wegen der festen Seitengröße liegt auch die Grenze zwischen Wortadresse und Seitennummer immer an der gleichen Stelle. Die physische Seitennummer muß mittels der Adreßtransformation aus der logischen Seitennummer ermittelt werden.

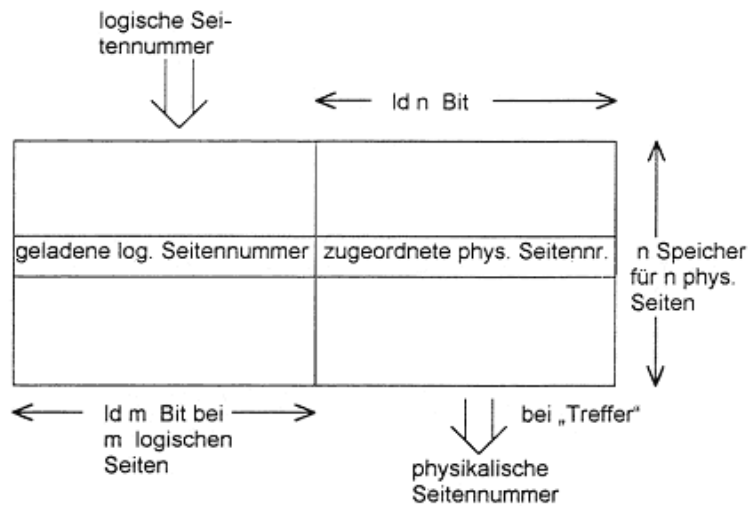
Die Adreßtransformation findet erst zum Zeitpunkt der Ausführung eines Befehls statt, man spricht deshalb von einer dynamischen Adreßumsetzung. Sie geschieht i. a. unter Verwendung einer Adreßumsetzungstabelle (Adreßumsetzungsspeicher, translation buffer), die für die im Arbeitsspeicher befindlichen Seiten die Zuordnungspaare (log. Seitennummer, phys. Seitennummer) enthält.



Besonders vorteilhaft für diesen Zweck sind sogenannte Assoziativspeicher (= CAM, Content Addressable Memory), deren Zugriff nicht über eine Adresse, sondern über Zelleninhalte erfolgt. Es wird ein Suchwort (Schlüssel) anstatt der Adresse angelegt und als Ergebnis erhält man eine Trefferanzeige, u. U. auch keinen Treffer. Der Schlüssel ist in diesem Fall die logische Seitennummer, 'Treffer' bedeutet, es existiert ein entsprechender Eintrag in der Tabelle, im speziellen Fall der Adreßumsetzung: die gesuchte logische Seite ist geladen, die eigentliche Information des Tabelleneintrags, die physische Seitennummer, wird ausgelesen.

Meldet der Assoziativspeicher keinen Treffer, (d. h. die Seite befindet sich nicht im Arbeitsspeicher --> "Page Fault") so wird das Laden dieser Seite eingeleitet (Seitenwechsel und Eintrag in die Tabelle!).

## Betriebssysteme



Vorteile des Paging-Verfahrens (gegenüber Segmentierung):

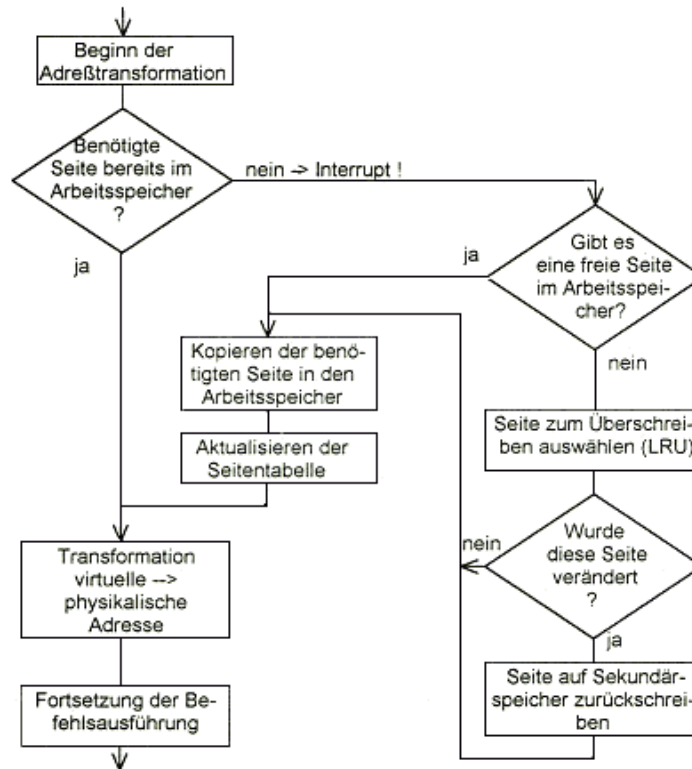
- keine Speicherzersplitterung: jede freie physische Seite kann jede logische Seite aufnehmen.
- keine Suche nach einem "passenden Loch" im Arbeitsspeicher für eine neu zu ladende Seite erforderlich. Macht die Software für die Durchführung des Seitenwechsels einfach.
- bedeutend weniger Zeitaufwand für den Transfer zwischen Sekundärspeicher und Arbeitsspeicher.
- im Arbeitsspeicher können sich i. a. gleichzeitig mehr aktive Programme befinden. (Arbeitsspeicher ist nicht durch selten oder gar nicht benötigte Programmabschnitte belegt).

Fazit: Paging ist für die Realisierung eines virtuellen Speicher-Systems geeigneter als die Segmentierung.

## Seitenwechsel-(Segmentwechsel-)-Algorithmen

Kennzeichen der virtuellen Speichertechnik ist das Laden bzw. Nachladen von Programmabschnitten (Seiten bzw. Segmente) bei Bedarf während der Programmausführung. Grundvoraussetzung ihrer Realisierung ist deshalb die Fähigkeit der CPU, einen laufenden Befehl zu unterbrechen (bei erforderlichem Nachladen einer Seite bzw. Segments) und nach erfolgtem Nachladen den begonnenen Befehl erneut aufzusetzen und dann ganz auszuführen. Für die Auswahl derjenigen Seite (bzw. Segment) die bei einem Seitenwechsel aus dem Arbeitsspeicher entfernt werden soll, gibt es verschiedene Strategien.

- Bei der verbreitetsten Strategie wird diejenige Seite (bzw. Segment), zu der am längsten nicht mehr zugegriffen wurde, ausgewählt: "Least Recently Used" (LRU). Hier wird davon ausgegangen, daß die Seite, die am längsten nicht benutzt wurde, aller Wahrscheinlichkeit nach auch in Zukunft nicht benötigt wird.
- Seiten bzw. Segmente, in die nicht geschrieben wurde, deren Inhalt also nicht verändert wurde, brauchen nicht in den Sekundärspeicher zurückgespeichert werden, sondern können sofort durch die neue Seite überschrieben werden (Kennzeichnung durch "dirty bit" in der Seitentabelle). Daraus ergibt sich eine Verkürzung der Seitenwechselzeit!



[Zum vorhergehenden Abschnitt](#)



[Zum Inhaltsverzeichnis](#)



[Zum nächsten Abschnitt](#)

Copyright © FH München, FB 04, Prof. Jürgen Plate

## Einführung in Betriebssysteme



von Prof. Jürgen Plate

# 4 Peripherieverwaltung

Die Peripheriegeräte gehören zusammen mit dem Prozessor und dem Hauptspeicher zu den elementaren Hardwarekomponenten einer Rechenanlage. Entsprechend ihrer Funktion können sie in drei Gruppen unterteilt werden:

- Die erste Gruppe setzt sich aus den Geräten zusammen, die zur Speicherung großer Datenmengen innerhalb eines Systems verwendet werden. Dazu gehören Festplatten-, CD-ROM-, DVD-, Disketten-, Kassetten- und Magnetbandeinheiten (Sekundär- und Tertiärspeicher der Speicher-Hierarchie). Die Datenübertragung von bzw. auf magnetische Datenträger wird hardwaremäßig wie die eigentliche Ein-/Ausgabe abgewickelt.
- Die zweite Gruppe umfaßt die eigentlichen Ein-/Ausgabe-Geräte wie Drucker, Tastatur, Bildschirmgeräte, Maus, Plotter, (Standard-Peripherie). Diese Geräte bilden die Benutzerschnittstelle eines Rechners.
- Die dritte Gruppe bilden die Geräte der sogenannten Prozeßperipherie, die vorwiegend im technischwissenschaftlichen Einsatzbereich von Rechnern (Prozeßrechner) zu finden sind. Für diese Anwendungen gibt es nur wenige Standard-Treiber des Betriebssystems (z. B. zur Vernetzung der Prozeßperipherie mit dem Rechner über Standard-Feldbusse, Laborbusse etc.).

## 4.1 Dateien

Zur Erinnerung: Dateien speichern eine Menge von gleichartigen Objekten (z. B. Bytes) unter einem Namen. Das BS stellt Operationen zum Bearbeiten von Dateien zur Verfügung:

- Erzeugen und Löschen von Dateien
- Lesen aus und Schreiben in Dateien

Alle Betriebssysteme zielen auf Geräteunabhängigkeit ab, d. h. ein Programm muß sich nicht (oder zumindest nur wenig) darum kümmern, ob eine Datei sich auf Platte, Magnetband oder einem anderen Massenspeicher befindet. Weiterhin sollen Ein- und Ausgabe auf einem Peripheriegerät prinzipiell genauso erfolgen, wie die Lese- und Schreiboperationen auf einer Datei. Auch Ressourcen auf einem Server sollen zugreifbar sein wie lokale Ressourcen. Bei vielen BS ist es möglich, auf einen beliebigen Datensatz innerhalb einer Datei zuzugreifen (wahlfreier Zugriff). Bei anderen Systemen ist nur sequentielles Lesen möglich.

Bei einigen BS wird zwischen verschiedenen Speichergeräten unterschieden (z. B. MS-DOS), der Benutzer muß genau spezifizieren, auf welchem Gerät (Laufwerk) sich die gewünschte Datei befindet. Bei anderen BS (z. B. UNIX) kann jedes Gerät in das Dateisystem eingebunden werden, das sich um die, auf diesem Gerät befindlichen, Dateien erweitert.

Ein Sektor ist auch die kleinste Informationsmenge, die von der Platte gelesen oder darauf geschrieben werden kann. In der Regel nimmt eine Datei mehrere Sektoren (typische Größen: 512 Byte - 8 KByte) in Anspruch. Das BS sorgt dabei:

- für die Zuordnung Spurnummer & Sektornummer --> Datei
- für die "Buchführung", in welcher Reihenfolge die Sektoren innerhalb der Datei aufeinanderfolgen
- bei Erzeugen oder Erweitern einer Datei für die Allokierung freier Sektoren auf der Platte
- beim Löschen einer Datei für die Freigabe der verwendeten Sektoren

Für die "Buchführung" über freie und belegte Sektoren gibt es verschiedene Methoden. Am gebräuchlichsten sind:

- Führen einer Freiliste (file allocation table = FAT) über alle Sektoren.
- Verbinden der Sektoren in Form linearer Listen. Dabei gibt es eine spezielle Liste, in der sich alle freien Sektoren befinden.

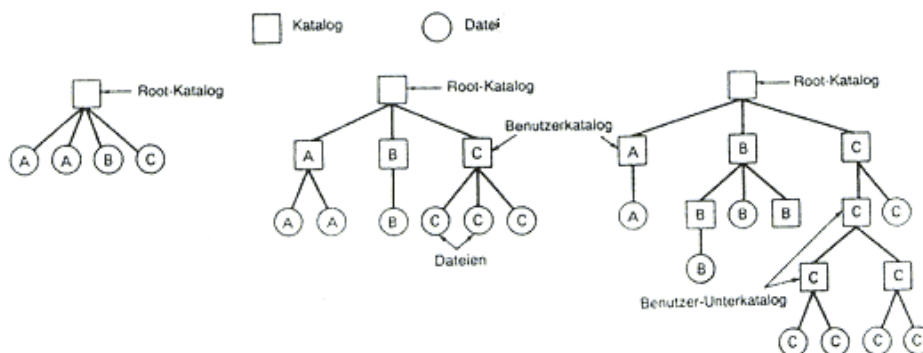
Zusätzlich werden Informationen über die Datei selbst benötigt, z. B.:

- Dateiname
- Dateityp
- Zugriffsrechte
- Datum von Erstellung/letzter Änderung/letztem Lesezugriff
- Eigentümer
- Größe
- durch die Datei belegte Sektoren
- ...

## 4.2 Verzeichnisse (Kataloge, Ordner, Directories, Folder)

Bei frühen BS bestand das Dateisystem lediglich aus einem Verzeichnis, das alle oben aufgeführten Informationen zu jeder der Dateien in einer Liste zusammengefasst hat. Über dieses Verzeichnis kann dann gezielt auf eine bestimmte Datei zugegriffen werden.

Ein erster Schritt der Verbesserung war die Unterteilung in Benutzerbereiche auf der Platte (z. B. bei CP/M) oder ein Verzeichnis für jeden Benutzer. Für Benutzer mit vielen Dateien war diese Lösung jedoch immer noch unbefriedigend, da eine logische Strukturierung der Dateien unmöglich war. Erst eine allgemeine Hierarchie von Katalogen (z. B. als Baum) bot die gewünschte Strukturierungsmöglichkeit (UNIX, MS-DOS).



Ist ein Dateisystem als Baum organisiert, benötigt man eine Methode, den Dateinamen zu spezifizieren:

- absoluter Pfadname:  
Ausgehend vom Wurzelverzeichnis wird ein Pfadname vergeben, in dem alle Verzeichnisse aufgeführt sind, die zur Datei führen. Der absolute Pfadname ist immer eindeutig.
- relativer Pfadname:  
Er wird in Verbindung mit dem Konzept des Arbeitsverzeichnis (working directory) oder aktuellen Verzeichnisses verwendet. Der Benutzer kann jederzeit ein beliebiges Verzeichnis zum aktuellen Verzeichnis erklären. In diesem Fall werden alle nicht absolut angegebenen

## Betriebssysteme

Pfadangaben relativ zu diesem Verzeichnis angeben.

Zum Beispiel:

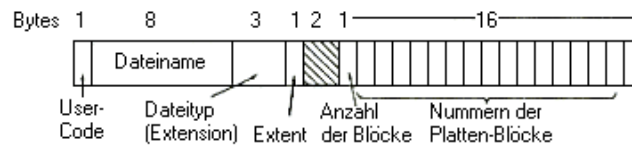
absoluter Pfadname: /meier/projekt/daten/dat1

working directory: /meier/projet

relativer Pfadname: daten/dat1

### System mit nur einem Verzeichnis: z. B. CP/M

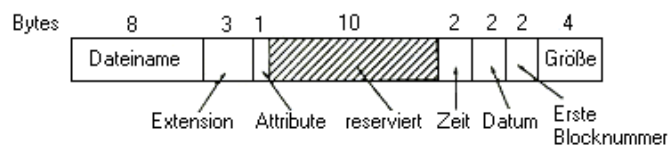
Das Verzeichnis besteht aus einer Tabelle mit Einträgen fester Länge:



Die Datei ist in Blöcken auf der Platte gespeichert, wobei der letzte Block nicht vollständig gefüllt sein muß. Das Feld "Extend" wird benötigt, wenn die Länge der Datei 16 Blöcke überschreitet. In diesem Fall wird ein zweiter Eintrag angelegt, und "Extend" hochgezählt. Das Verzeichnis hat eine feste Größe.

### Hierarchisches System: z. B. MS-DOS

Hier enthält das Verzeichnis nur die erste Blocknummer der Datei, die als Index für die FAT (File Allocation Table) dient und so die Bestimmung der weiteren Blocknummern gestattet. Das Root-Verzeichnis hat eine feste Größe; die Unterverzeichnisse sind Dateien, die beliebige viele Einträge aufnehmen können.



Bevor man die Festplatte benutzen kann, muß man die Festplatte partitionieren und ein Dateisystem einrichten, damit das Betriebssystem darauf Daten ablegen kann. Das FAT-Dateisystem besteht im wesentlichen aus zwei Teilen:

- Dem Inhaltsverzeichnis: Hier werden die Namen der gespeicherten Dateien sowie deren Position auf der Festplatte mit dem eigentlichen physischen Bereich auf der Festplatte verknüpft.
- Der Datenbereich: Hier sind unsortiert und durcheinander die Datenblöcke aller Dateien gespeichert, die auf der Festplatte abgelegt wurden.

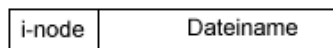
Die FAT führt Buch über die freien und belegten Plattenblöcke.

Mit der Entwicklung von Windows 95 sah sich auch Microsoft genötigt, das betagte FAT-Dateisystem durch einen leistungsfähigeren Nachfolger zu ersetzen. Das Ergebnis heißt VFAT oder FAT32 (Virtual-FAT). VFAT unterstützt bis zu 255 Zeichen lange Dateinamen. Sogar unter DOS behalten

diese ihre Gültigkeit in Form einer Ersatzdarstellung mit 8+3 Zeichen. Ferner besitzt die VFAT eine Pseudo-Unterscheidung zwischen Gross- und Kleinschreibung, konvertiert aber alle im 8+3-Format vorliegenden Dateinamen in große Buchstaben um.

### Hierarchisches System: z. B. UNIX

Bei UNIX ist die Verzeichnisstruktur sehr einfach. Jeder Eintrag besteht nur aus dem Dateinamen und einer Nummer. Der Dateiname war anfangs auf 14 Zeichen begrenzt, kann aber heute bei den meisten Systemen sehr viel länger sein. Die Nummer verweist auf den sogenannten i-node, der alle weiteren Informationen zur Datei enthält. Die Verzeichnisse sind Dateien. Alle i-nodes haben einen festen Platz auf der Platte.



Zusätzlich zu den Dateiverweisen muß bei den hierarchischen Dateisystemen in Unterverzeichnissen (außer Root) noch mindestens ein Verweis auf das übergeordnete Verzeichnis enthalten sein, da sonst ein "Navigieren" in den Verzeichnissen nicht möglich ist (in der Regel ".."). In der Regel existiert noch ein weiterer Verweis auf das Verzeichnis selbst (in der Regel ".").

Im Gegensatz zu CP/M oder MS-DOS, bei dem die einzelnen Plattenlaufwerke durch einen Buchstaben, gefolgt von einem Doppelpunkt, gekennzeichnet werden, sind bei UNIX die Platten in das Dateisystem eingebunden und können an ein beliebiges Unterverzeichnis gekoppelt werden.

### Hierarchisches System: z. B. NTFS

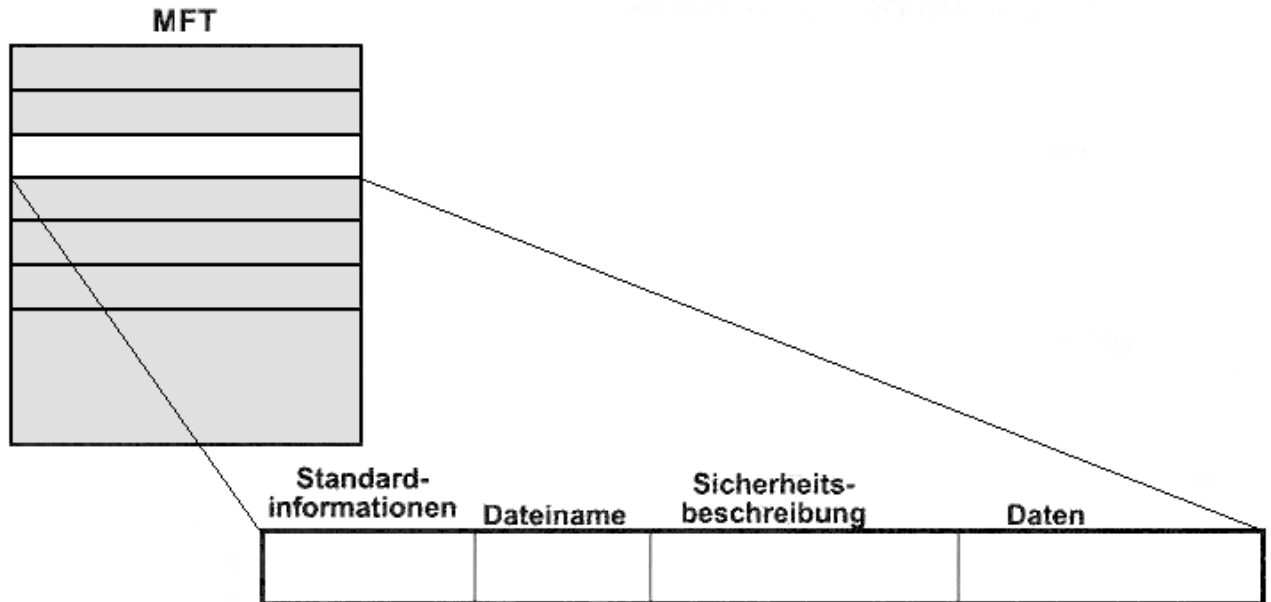
"NTFS" ist eine Abkürzung für "New Technology File System". Es handelt sich um ein Dateisystem mit neuerer Technologie, das mit WINDOWS NT eingeführt wurde. Dieses Dateisystem unterstützt lange Dateinamen, erweiterte Dateiattribute und sehr große Speicherkapazitäten (bis zu  $2^{64}$  Byte, also ca. 17 Milliarden GByte). Dateinamen werden im Unicode, einem 16 Bit-Zeichensatz, gespeichert das heisst gute Austauschbarkeit zwischen Systemen. NTFS löst Probleme, die mit dem FAT-Dateisystem (DOS/Windows) bestanden Dateinamen können wie bei VFAT bis zu 255 Zeichen lang sein und es gibt raktisch keine Beschränkung im Hinblick auf die Größe einer Datei, eines Verzeichnisses oder einer Festplatte. Außerdem ist die Clustergröße auf großen Platten frei wählbar und es entfällt der FAT-typische Verschnitt bei großen Festplatten. Alle Dateien und Verzeichnisse sind in den Zugriffskontrollmechanismus von Windows NT eingebunden. Die Zugriffsoperationen (Lesen, Schreiben, Löschen ) können separat für beliebige Benutzer oder Gruppen erlaubt bzw. verboten werden. Damit haben nur bestimmte Personen Zugriff auf vorher vom Systemadministrator festgelegte Dateien. Bei der Entwicklung von Windows NT und NTFS wurde besonders auf die Sicherheit gegen Datenverlust geachtet. NTFS arbeitet bei Metadaten (z. B. Verzeichnisse) transaktionsorientiert. So werden unvollständige Änderungen nach einem Systemausfall entweder vervollständig oder rückgängig gemacht.

Die wichtigste Komponente von NTFS ist die MFT (Master File Tabelle). Jede Datei wird durch einen Eintrag in der MFT repräsentiert, denn auch die MFT ist eine Datei. Die ersten 16 Einträge dieser Tabelle sind reserviert.

- Der erste Eintrag beschreibt MFT selbst,
- der zweite Eintrag ist ein MFT-Spiegeleintrag,
- der dritte Eintrag verweist auf eine Logdatei, die für die Wiederherstellung von Dateien benötigt wird.

## Betriebssysteme

Ab dem 17. Eintrag beziehen sich alle Einträge auf die einzelnen Dateien und Verzeichnisse des Datenträgers. Jedem Dateieintrag wird eine bestimmte Menge an Speicherplatz zur Verfügung gestellt, in den die Attribute der Datei geschrieben werden. Kleine Dateien bzw. Verzeichniseinträge können vollständig vom MFT-Eintrag aufgenommen werden.



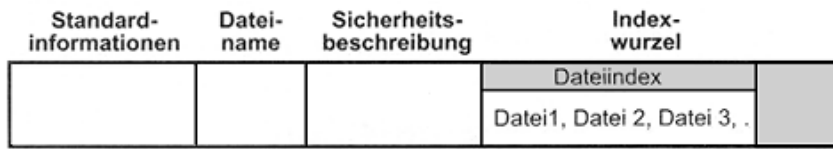
Größere Verzeichniseinträge werden in Form von B-Trees organisiert, deren Einträge Zeiger auf externe Cluster enthalten, welche die Verzeichniseinträge aufnehmen. Jeder Datei bzw. jedem Verzeichnis ist ein Satz von Attributen (Dateiname, Daten, Sicherheits-ID) zugeordnet.

### Einige Typen von NTFS Dateiattributen

Standardinformation	Zeitpunkt der letzten Speicherung usw.
Attributliste	Liste aller weiteren Attribute (nur für grosse Datei)
Dateiname	langer Dateiname bis zu 255 Unicode-Zeichen, kurzer Dateiname 8+3 Zeichen (MS-DOS), zusätzliche Namen als weitere Dateinamensattribute möglich (Hard Links von POSIX gefordert)
Sicherheitsdeskriptor	Informationen über Eigentümer der Datei, über Benutzer die auf diese Datei Zugriff haben usw.

## Betriebssysteme

Daten	eigentlichen Daten der Datei
Index-Wurzel	bei der Implementierung von Verzeichnissen notwendig
Index-Zuordnung	bei der Implementierung von Verzeichnissen notwendig
Datenträgerinfo	wird nur bei der Systemdatei des Datenträgers verwendet und enthält u.a. die Version und den Namen des Datenträgers
Bitmuster	eine Karte der in der MFT oder im Verzeichnis belegten Einträge



### Aufbau eines MFT Eintrages

	FAT	NTFS
Dateiname	8+3 ASCII-Zeichen, durch einen Punkt getrennt	255 Unicode-Zeichen; mehrere Punkte sind als Trennzeichen zulässig
Dateigröße	232 Bytes	264 Bytes
Partition	232 Bytes	264 Bytes
max. Länge des Suchweges	64	Unbegrenzt
Attribute	einige Bitflags	Alles, inklusive der Daten, wird als Dateiattribut behandelt
Verzeichnisse	unsortiert	B-Tree
Konzept	einfach	Schnell, mit Datenwiederherstellung / Sicherheit
eingebaute Sicherheit	Nein	Ja

NTFS unterstützt Hot-Fixing: Bei fehlerhaften Sektoren werden die Daten in einen anderen Sektor verschoben und der defekte Sektor wird in die Karte der fehlerhaften Sektoren eingetragen die allen Anwendungen mit Schreib- und Leserechten zugänglich ist.

Jede E/A-Operation, die eine Datei auf einem NTFS-Datenträger verändert, wird als Transaktion betrachtet und kann als geschlossene Einheit verwaltet werden. In der Protokolldatei werden alle Infos zum Wiederholen bzw. Rückgängigmachen der Transaktion gespeichert --> hoher Aufwand für die Protokollierung.

### 4.3 Gemeinsam genutzte Dateien

Wenn mehrere Benutzer die gleiche Datei verwenden wollen, kann es nützlich sein, wenn jeder Benutzer die Datei aus einem seiner Kataloge ansprechen (ohne den absoluten Pfad zu verwenden) und u. U. auch einen eigenen Namen für die Datei vergeben kann (bei Namenskonflikt mit eigenen Dateien).

Es wird dann ein Verweis auf die Datei installiert, ein Link. Durch die Einführung von Links dürfen die durch die Datei belegten Plattenblöcke nicht im Katalog gespeichert werden, da sonst alle Katalogeinträge, die auf die Datei verweisen, bei jeder Änderung an der Datei aktualisiert werden müssten. Die Lösung kann auf zwei Arten erfolgen:

- Die Verzeichnisse verweisen auf die Datei-Datenstruktur, die gesondert gespeichert ist (i-nodes bei UNIX)
- Es wird für ein Link eine spezielle Datei angelegt, die den absoluten Pfadnamen auf die gelinkte Datei enthält (symbolic linking).

**Probleme:** Die erste Lösung kann in Mehrbenutzersystemen zu Inkonsistenzen führen. Angenommen Benutzer A ist Eigentümer einer Datei und gestattet Benutzer B, ein Link auf diese Datei zu setzen. Zu einem späteren Zeitpunkt wird der Benutzer A aus dem System gelöscht (und damit auch seine Dateien). Nun gibt es über das Link von B eine Datei, zu der es keinen Eigentümer mehr gibt.

Die zweite Lösung vermeidet solche Inkonsistenzen, erfordert beim Dateizugriff jedoch einen höheren Verwaltungsaufwand (aus der Link-Datei den Pfad lesen und erst damit die Datei eröffnen). Ist die Datei nicht mehr vorhanden, erfolgt eine Fehlermeldung.

### 4.4 Ein- und Ausgabegeräte

E/A-Geräte werden in der Regel in das Dateisystem eingebunden und aus Benutzersicht wie Dateien behandelt (die u. U. von "normalen" Dateien abweichende Eigenschaften haben).

- Auf reine Ausgabegeräte kann nur geschrieben werden
- Von reinen Eingabegeräten kann nur gelesen werden
- Bei kombinierten E/A-Geräten kann gelesen/geschrieben werden

E/A-Geräte können grob in zwei Kategorien eingeteilt werden:

- blockorientierte Geräte (block devices), z. B. Platte, Band Es verarbeitet Daten nur in Form von Datenblöcken fester Länge (Blockgrößen zwischen 128 Byte und 8 KByte).
- zeichenorientierte Geräte (character devices), z. B. Terminal Es verarbeitet einen Datenstrom aus einzelnen Bytes.

Einige Geräte lassen sich nicht definiert einordnen (z. B. Zeitgeber = Timer). Zur Ansteuerung der Geräte ist systemnahe und z. T. hardwareabhängige Software des BS notwendig. In der Regel erhält jedes Gerät einen fest vorgegebenen Namen, unter dem es dann wie eine Datei angesprochen werden kann. Je nach Typ können Geräte gemeinsam von allen Prozessen verwendet werden (z. B. Platte)

oder sie sind exklusiv für einen Prozeß reserviert (z. B. Plotter). Die Software zur Ansteuerung von Geräten läßt sich in zwei Ebenen strukturieren:

- **Gerätetreiber**  
BS-Programmteil für den geräteabhängigen Code. Hier sind alle Prozeduren vereinigt, die in irgend einer Weise hardware-spezifisch sind. Siehe unten: Gerätesteuerung.
- **Geräteunabhängige Software des Betriebssystems**  
In dieser Schicht wird die Schnittstelle zum Dateisystem realisiert. Sie enthält eine einheitliche Schnittstelle zum Treiber, den Gerätenamen, Pufferung der Daten, Schutz der Geräte, Vergabe exklusiver Geräte und Fehlerbehandlung.

### Gerätesteuerung

Die Peripheriegerätesteuerung erfolgt auf der Hardware-Ebene über den Austausch von Statussignalen und Daten zwischen den angeschlossenen Geräten und dem Prozessor. Die Programme zur Steuerung von Peripheriegeräten werden Treiber (driver, handler) genannt. Die Komplexität der Steuerungsaufgaben ist weitgehend vom Geräteverhalten abhängig. Alle drei Methoden des Signal- und Datenaustausches können zum Einsatz kommen (programmgesteuert, interruptgesteuert, DMA).

- **Programmgesteuerte Ein-/Ausgabe**  
Im einfachsten Fall sind alle Grundfunktionen der Ein-/Ausgabe vollständig in die Treiberprogramme integriert. Bei der Durchführung einer Datenübertragung oder einer Steuerfunktion muß der Prozessor dann bis zu deren Beendigung explizit warten, bevor er die Bearbeitung weiterer Programmschritte aufnehmen kann.
- **Unterbrechungsgesteuerte Ein-/Ausgabe**  
Damit der Prozessor nicht durch unnötige Wartezeiten und Statusabfragen zeitlich belastet wird, erfolgt die Gerätesteuerung i. a. unter Verwendung von Unterbrechnungen. Beispiel:
  - ◆ Tastendruck am Terminal erzeugt Unterbrechung;
  - ◆ das entsprechende Interrupt-Service-Programm liest das Zeichen vom Empfangsport der seriellen Schnittstelle und speichert es in einem Zwischenpuffer (Puffer für eine Zeile).
  - ◆ Falls das Betriebssystem auf die Eingabe eines Kommandos wartet (ein entspr. Parameter wurde von höheren BS-Schichten an den Treiber übergeben), wird gleich geprüft, ob das Zeichen CR (Carriage Return = Eingabeende) eingetroffen ist (in einem Textverarbeitungsprogramm muß diese Sonderbehandlung des CR ausgeschaltet sein!).
  - ◆ Falls das CR erkannt wird, veranlaßt das Interrupt-Service-Programm eine Meldung (z. B. per Mailbox) an ein hierarchisch höher liegendes Auswerteprogramm, das den Inhalt des eingegebenen Textes analysiert, um - wieder eine Ebene höher - z. B. die gewünschte Betriebssystem-Funktion zu starten.

Die Unterbrechungssignale verschiedener Geräte sollten unterschiedliche Prioritäten haben, um die dringlichen Aufgaben vorrangig bearbeiten zu können.

- **DMA-Betrieb**  
Um den Prozessor von ständig zu wiederholenden Formen der Datenübertragung (großer Datenmengen) zu entlasten, wurde das Verfahren des direkten Speicherzugriffs (direct memory access, DMA) entwickelt. Die Datenübertragung zwischen dem Hauptspeicher und einem Peripheriegerät wird vollständig von der DMA-Steuerung durchgeführt, und die Transferrate wird durch die Geschwindigkeitsanforderungen des beteiligten Geräts oder (seltener) durch die Länge der Speicherzyklen bzw. die Übertragungskapazität des Busses bestimmt.

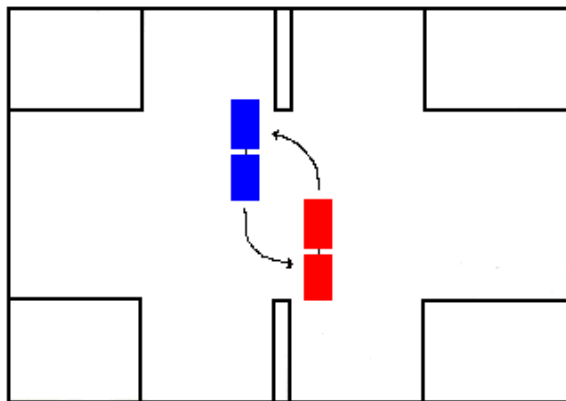
## 4.5 Gemeinsame Nutzung von Betriebsmitteln

Manche Betriebsmittel dürfen nicht gleichzeitig von mehreren Prozessen genutzt werden. Klassisches Beispiel ist hier der Drucker. Die Ausgaben mehrerer Prozesse würden vermischt. Abhilfe bietet hier die Nutzung des Druckers durch einen einzigen Prozeß, den Druckerspooler. Alle Prozesse übergeben ihre Ausgaben an diesen Prozeß, der die Druckaufträge in einer Warteschlange speichert und nacheinander abarbeitet.

Beim Wettbewerb mehrerer Prozesse um Betriebsmittel (Ressourcen), zu denen ja primär Speicher, Dateien und die (als Dateien eingebundenen)Geräte gehören, kann es zu sogenannten **Verklemmungen (dead locks)** kommen.

Eine Verklemmung tritt bei Anforderungen von Ressourcen durch mehrere Prozesse dann auf, wenn ohne drastische Aktionen des BS all diese Anforderungen niemals erfüllt werden können. Folge: Die Prozesse blockieren sich gegenseitig.

### Beispiel aus dem täglichen Leben:



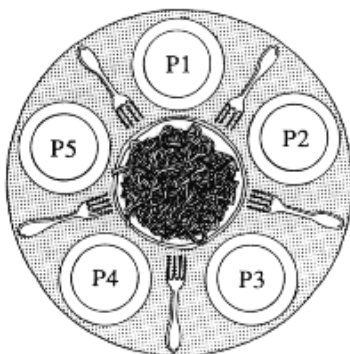
Zwei LKW an einer  
Straßenkreuzung  
"verklemmt"!

Betriebsmittel: "Straße"

Task: "Transportieren  
von Gütern"

Das Problem wurde schon vor vielen Jahren behoben. Die Abbiegevorschriften wurden so geändert, daß die Autos **voreinander** abbiegen dürfen.

### Die speisenden Philosophen



Dieses Beispiel stammt von Dijkstra und ist also Demonstrationsmodell für das Entstehen von Deadlocks gedacht. Es wird seither auch immer als Testproblem für neue Prozeß-Synchronisations-Algorithmen verwendet. Bei Tanenbaum essen die Philosophen Spaghetti und zwar immer mit zwei Gabeln.

Programm für einen Philosophen mit zwei Gabeln:

```
#include "prototypes.h"
#define N 5

void philosoph (int i)
```

## Betriebssysteme

```
{
while (1)                # Endlosschleife
{
  denke();
  NimmGabel(links);
  NimmGabel(rechts);
  esse();
  LegeGabel(links);
  LegeGabel(rechts);
}
}
```

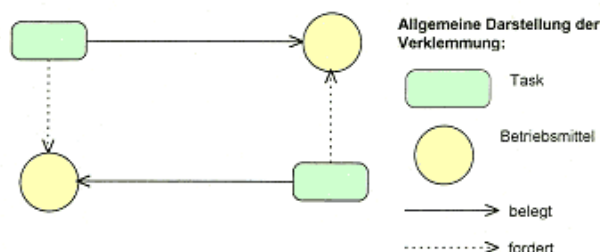
Unterstellt man, daß die Philosophen pseudoparallel, also in Zeitscheiben arbeiten, ist das Modell gleichzeitig auch als Modell für die Beschreibung kritischer Abschnitte (zeitkritischer Situationen) geeignet. Die Funktion `NimmGabel(Seite)`, müßte logischerweise zuerst prüfen, ob die Gabel frei ist, und erst dann die Gabel aufnehmen. Der zeitkritische Charakter der Funktion `NimmGabel(Seite)` besteht darin, daß zwischen der Prüfung "Gabel frei" und der Ausführung `NimmGabel(Seite)` ein zeitkritischer Abschnitt besteht.

### Beispiel: System mit Drucker und Magnetband, 2 Prozesse

- Prozeß A fordert den Drucker an, Prozeß B die Bandstation. Beiden Anforderungen wird entsprochen.
- Nun fordert A die Bandstation an, ohne den Drucker freizugeben. Prozeß B verlangt dagegen den Drucker an ohne die Bandstation freizugeben.
- Folge: gegenseitige Blockierung

Setzt man für "Drucker" und "Bandstation" zwei Datensätze einer Datenbankdatei, so führt auch diese Situation zur Verklemmung. Nahezu jede Situation, in der Prozesse Ressourcen **exklusiv** anfordern, kann zur Verklemmung führen. Jedes exklusive Betriebsmittel kann immer nur von einem Prozeß angefordert werden. Solange das Betriebsmittel nicht frei ist, verbleibt der Prozeß im Zustand "blockiert". Nicht jede Anforderung von exklusiven Ressourcen führt zur Verklemmung. Für das Auftreten einer Verklemmung müssen folgende Bedingungen erfüllt sein:

- **Exklusive Nutzung:** Das BM wird entweder von genau einem Prozeß verwendet oder es ist frei (z. B. Druckerspöler).
- **Wartebedingung:** Prozesse belegen verfügbare BM, während sie auf zusätzliche BM warten.
- **Nichtentziehbarkeit:** Einem Prozeß können zugeordnete BM nicht zwangsweise entzogen werden; er muß sie explizit freigeben.
- **Geschlossene Kette:** Es existiert eine geschlossene Kette von 2 bis n Prozessen. Jeder von ihnen wartet auf ein BM, das durch den nächsten Prozeß in der Kette gehalten wird.



Im allgemeinen werden vier **Strategien zur Behandlung von Verklemmungen** verwendet:

- **Ignorieren des Problems**

Die Frage ist, wie oft eine Verklemmung auftritt. Bei durchschnittlich einer Verklemmung pro Monat kann das Problem durchaus ignoriert werden. Interaktiv arbeitende Benutzer werden

Beispiel aus dem täglichen Leben:

sowieso bald die Geduld verlieren und den Prozeß abbrechen. Bei Batch-Systemen wird die Verklemmung bei der täglichen oder wöchentlichen Systemwartung entdeckt.

- **Entdecken und Beheben von Verklemmungen**

Das BS hält Anforderungen und Freigaben von BM fest und stellt sie in Form eines BM-Grafen dar. Bei jeder Anforderung/Freigabe wird der Graf auf Zyklen untersucht.

Billigere Methode: Zyklisch prüfen, ob ein Prozeß längere Zeit (einige Stunden) blockiert ist und diesen dann entfernen. Nachteil: möglicherweise inkonsistente Daten.

- **Verhindern von Verklemmungen durch Negation einer der 4 Bedingungen**

- ◆ Beispiel Drucker: Es wird ein Prozeß eingeführt, der als einziger Prozeß den Drucker exklusiv "besitzt".
- ◆ Durchbrechen der Wartebedingung, indem ein Prozeß die vorher angeforderten BM freigibt. Nur wenn die Anforderung erfolgreich war, erhält er sie zurück.
- ◆ Es gibt noch eine Reihe weiterer Methoden.

- **Vermeiden der Verklemmungen durch sorgfältige BM-Vergabe**

Durch geeignete BM-Zuweisung kann eine Verklemmung vermieden werden, wenn einige Informationen im voraus verfügbar sind.

### Banker-Algorithmus: Eine Methode zur Vermeidung von Verklemmungen

Wie eine Bank niemals soviel Geld bereithält, daß alle Kreditrahmen der Kunden voll ausgeschöpft werden können, kann auch das BS die Ressourcen so verwalten, daß immer mindestens ein Prozeß voll befriedigt werden kann. Dazu muß bei jedem Prozeß ein Maximalwert für die anzufordernden BM existieren.

Beispiel: Der Rechner hat 10 Magnetbandstationen

Zustand 1 Prozess bel. Max.	Zustand 2 Prozess bel. Max.	Zustand 3 Prozess bel. Max.																																				
<table style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10%;">A</td><td style="width: 10%;">0</td><td style="width: 10%;">6</td></tr> <tr><td>B</td><td>0</td><td>5</td></tr> <tr><td>C</td><td>0</td><td>4</td></tr> <tr><td>D</td><td>0</td><td>7</td></tr> </table>	A	0	6	B	0	5	C	0	4	D	0	7	<table style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10%;">A</td><td style="width: 10%;">1</td><td style="width: 10%;">6</td></tr> <tr><td>B</td><td>1</td><td>5</td></tr> <tr><td>C</td><td>2</td><td>4</td></tr> <tr><td>D</td><td>4</td><td>7</td></tr> </table>	A	1	6	B	1	5	C	2	4	D	4	7	<table style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10%;">A</td><td style="width: 10%;">1</td><td style="width: 10%;">6</td></tr> <tr><td>B</td><td>2</td><td>5</td></tr> <tr><td>C</td><td>2</td><td>4</td></tr> <tr><td>D</td><td>4</td><td>7</td></tr> </table>	A	1	6	B	2	5	C	2	4	D	4	7
A	0	6																																				
B	0	5																																				
C	0	4																																				
D	0	7																																				
A	1	6																																				
B	1	5																																				
C	2	4																																				
D	4	7																																				
A	1	6																																				
B	2	5																																				
C	2	4																																				
D	4	7																																				
sicher 10 frei	sicher 2 frei	unsicher 1 frei																																				

Ein Zustand ist dann sicher, wenn das BS mindestens bei einem Prozeß seine Maximalforderung erfüllen kann (die anderen müssen u. U. warten).

Zustand 1 ist sicher, da jeder Prozeß befriedigt werden kann.

Zustand 2 ist sicher, da Prozeß C befriedigt werden kann.

Zustand 3 ist unsicher, da keiner der der Prozesse voll befriedigt werden kann.

Das Schema kann auf beliebig viele BM erweitert werden.

## 4.6 Dienstprogramme des Dateisystems

Auf der obersten BS-Ebene treten die Benutzer über eine Menge von Kommandos (= Programme) mit dem Rechner in Dialog. Die Benutzerschnittstelle muß dazu eigentlich nur eine Eingabe des Benutzers entgegennehmen (z. B. den Namen eines Programms) und diese Eingabe interpretieren --> Kommandointerpreter oder Shell. Auch bei BS mit geringem Befehlsumfang existieren üblicherweise Kommandos:

Beispiel aus dem täglichen Leben:

## Betriebssysteme

- Zum Beginnen und Beenden einer Sitzung
- Ausführen von Programmen
- Abrufen von Informationen über das System
- Bearbeiten von Dateien und Katalogen

Diese Kommandos können entweder Teil des Kommandointerpreters sein (schnelle Ausführung) oder durch getrennte Programme ausgeführt werden, die auf der Platte gespeichert sind. Dabei stützt sich der Kommandointerpreter wieder auf elementare Operationen des BS.



Zum vorhergehenden Abschnitt



Zum Inhaltsverzeichnis



Zum nächsten Abschnitt

---

*Copyright © FH München, FB 04, Prof. Jürgen Plate*

## Einführung in Betriebssysteme



*von Prof. Jürgen Plate*

---

# 5 Benutzerverwaltung

## 5.1 Benutzer-Zugang

In Mehrbenutzersystemen ist es üblich, jedem Einzelbenutzer eine fest eingegrenzte Arbeitsumgebung zu bieten, die ihn von anderen Benutzern abschirmt:

- Prozesse eines Benutzers können gleichzeitig laufende Prozesse anderer Benutzer nicht beeinflussen (Speicherschutz, etc.)
- Auf Dateien eines Benutzers können andere Benutzer nur mit dessen ausdrücklicher Erlaubnis zugreifen
- Für gemeinsam genutzte Ressourcen und Dateien können definierte Zugriffsrechte vergeben werden
- Die Nutzungsdauer (Rechenzeit), der Zeitraum der Nutzung (z. B. nur von 8 ... 17 Uhr) oder die maximal zu beanspruchende Plattenkapazität können festgelegt werden
- Zu Beginn der Arbeit am Rechner (login) muß sich der Benutzer identifizieren (Benutzerauthentizität)

Bei besonders kritischer Rechnerumgebung kann es sogar wünschenswert erscheinen, daß sich die Benutzer beim Aufruf bestimmter Programme nochmals identifizieren müssen.

Die o. g. Schutzmaßnahmen sind ins BS integriert (z. B. Prozeß-Schutz oder Dateischutz). Die Benutzeridentifikation wird durch ein eigenes Login-Programm vollzogen.

Spezielle Geräte erfordern besondere Schutzmaßnahmen; so ist es inzwischen üblich, bei Benutzern, die sich über ein Modem (also per Telefon) anmelden, diesen nicht sofort Zugang zum Rechner zu gewähren, sondern den Anschluß des Benutzers vom Rechner aus zurückzurufen. Zusätzliche Aufgaben der Benutzerverwaltung betreffen Abrechnungsinformationen:

- die Zeit, die der Benutzer angemeldet ist
- die verbrauchte Rechenzeit
- die Verweilzeit von Prozessen im Rechner
- Zugriff und Nutzung der Ressourcen (Platte, Drucker, etc.)

Derartige Informationen werden in bestimmten Dateien des Betriebssystems protokolliert (sogenannte 'Logfiles'), deren Information statistisch ausgewertet werden kann.

### Arten der Autorisierung

- Knowledge: Passwort, PIN Dies ist der derzeit häufigste Fall. Das Verfahren ist einfach zu implementieren, Passwörter und PINs können geändert werden. Sie lassen sich jedoch auch unbemerkt und einfach kopieren. Sichere Passwörter müssen lang sein und sollten oft geändert werden. Sie sind daher für die Benutzer schwer zu handhaben. Passwörter/PINs können vergessen werden.
- Challenge: Es wird eine Frage gestellt, deren Antwort überprüft wird und die nur die autorisierte Person wissen sollte. Das Frage-Antwort-Protokoll kann auch auf kryptographischen Verfahren basieren (Public Key Kryptosysteme). Auch eine Liste von Einmal-TANs gehört in diese Rubrik.
- Token: Schlüssel, Smartcard, Secure ID Card, usw. Mit 'Token' wird als etwas 'greifbares' bezeichnet. Tokens sind einfach in der Anwendung. Sie lassen sich zudem mit anderen Sicherungssystemen/Datenerfassungssystemen z.B. Zeiterfassung oder Paßwort kombinieren. Unbemerkte Kopien sind schwierig bis unmöglich. Ein Nachteil ist der Kostenaufwand durch zusätzliche Hardware. Karten oder Schlüssel können vergessen oder verloren werden. Die unbefugte Benutzung verlorener Tokens ist möglich.

## Betriebssysteme

- Biometrie: Fingerabdruck, Gesichtserkennung, Iris-Scanner, Schriftynamik, Tippverhalten, DNA-Analyse, usw. Auch Biometrieverfahren sind einfach in der Anwendung. Biometrische Merkmale können nicht vergessen werden. Derzeit (2002) sind die angebotenen Verfahren jedoch noch nicht für allgemeine Anwendung geeignet. Einfache Sensoren lassen sich relativ leicht austricksen, z.B. durch Kopie eines Fingerabdrucks oder Fotos. Der Kostenaufwand durch zusätzliche Hardware ist noch recht hoch. Biometrische Informationen lassen sich schwer oder gar nicht ändern.

Das Paßwort ist der kritische Punkt in jeder Benutzerauthentisierung. Man hat deshalb die Sicherheit von Paßwortprüfungen ständig verbessert.

1. Paßwörter werden einwegverschlüsselt gespeichert. Sie sind nicht entschlüsselbar. Die Prüfung erfolgt durch Einwegverschlüsselung der Tastatureingabe und Vergleich der beiden einwegverschlüsselten Bitmuster.
2. Paßwörter müssen eine definierte Mindest- und Maximallänge haben.
3. In einer Liste festgelegte Wörter oder Wortbestandteile dürfen nicht verwendet werden, um leicht zu erratende "weiche" Paßwörter wie "System", "Test" usw. zu verhindern.
4. Paßwörter müssen nach Ablauf einer definierten Frist geändert werden.
5. Paßwörter dürfen nicht oder erst nach n-maliger Änderung wieder benutzt werden.

Trotzdem ist das Paßwort in Verruf geraten, weil seine Sicherheit ausschließlich vom sorgfältigen Umgang des Benutzers abhängt. Befragungen von Computerbenutzern haben immer wieder ergeben, daß sie das Paßwort nur allzu sorglos behandeln.

Eine vernünftige Paßwortpolitik trägt in sehr wesentlichem Maße zur Sicherheit bei. Dies setzt voraus, daß die Benutzer die Bedeutung von Paßwörtern und deren Verwendung respektieren. Es gibt jedoch von Unternehmen zu Unternehmen und von Mitarbeiter zu Mitarbeiter große Unterschiede im Grad der Paßwortdisziplin. Die wenigsten Mitarbeiter würden auch nur davon träumen, die ID-Karte ihrer Firma irgendwo liegenzulassen, wohingegen dieselbe Person weit größere Nachlässigkeit im Zusammenhang mit der Wahl oder Weitergabe von Paßwörtern zeigt. Nutzlose Paßwörter, wie z. B. der Vorname des Benutzers oder gleich die Benutzer-ID, kommen auch häufig vor.

Damit die Paßwörter effektiv angewendet werden, ist es notwendig, eine Reihe von Richtlinien in Form einer entsprechenden Paßwortpolitik festzulegen. Die Paßwortpolitik des lokalen Netzes sollte Richtlinien für folgende Punkte enthalten:

- Mindestlänge des Paßworts (Mindestanzahl von Zeichen)
- Bestandteile eines Paßworts (Buchstaben, numerische Zeichen und Sonderzeichen)
- Maximaler Gültigkeitszeitraum eines Paßworts und Regeln für die obligatorische Erneuerung sowie Begrenzungen für die Wiederverwendung von Paßwörtern.
- Protokollierung der Benutzeraktionen nach einer vorgegebenen Anzahl mißglückter Login-Versuche.
- Regeln für die Übertragung von Paßwörtern an andere.

Im Gegensatz zu einem physischen Gegenstand, wie z. B. ein Schlüssel, dessen Verlust man bemerkt, kann ein Paßwort entschlüsselt werden, ohne daß sich der Benutzer dessen unmittelbar bewußt wird. Dagegen hilft nur die zeitliche Begrenzung des Gültigkeitszeitraums von Paßwörtern. Die Regel sollte streng gehandhabt werden, d. h. Sperrung von Benutzer-IDs, die es unterlassen, ihr Paßwort zu ändern. Müssen Paßwörter zu häufig aktualisiert werden, tendieren die Benutzer dazu, Paßwörter aufzuschreiben. Bei 30-tägiger Paßwortlebensdauer neigen viele Benutzer dazu, den Monatsnamen als Paßwort zu verwenden. Benutzer-ID und Paßwort dürfen nie von mehreren Benutzern gleichzeitig verwendet werden. Paßwörter müssen streng vertraulich und persönlich sein. Ein Benutzer darf unter keinen Umständen sein Paßwort an andere weitergeben.

Die Paßwortpolitik des Netzes sollte aus den Benutzungsrichtlinien deutlich hervorgehen und konsequent gehandhabt werden. Eine starke Authentisierung, wie sie zunehmend gefordert wird, kann

jedoch nicht mehr auf dem Paßwort basieren. Sie kann beispielsweise auf dem "Challenge-Response"-Prinzip beruhen. Voraussetzung hierfür ist neben der Software ein sogenannter Token, ein spezieller Taschenrechner im Kreditkartenformat mit Tastatur, Verschlüsselungsprozessor und persönlichem Schlüssel (z.B. die SecureID-Card). Zunächst muß sich der Benutzer des Tokens durch Eingabe seiner PIN als berechtigter Besitzer ausweisen. Wenn der Benutzer sich am Computersystem anmelden möchte, generiert dieser eine Zufallszahl (Challenge), die am Bildschirm erscheint. Der Benutzer tippt die Challenge in seinen Token ein, der durch Verschlüsselung die Antwort (Response) ermittelt. Diese wiederum wird als Paßwort in die Computertastatur eingegeben und vom Computersystem verifiziert. Manche Tokens, wie z.B. SecureID, generieren die Zufallszahl zeitabhängig selbst, so daß eine Eingabe entfällt. Challenge-Response hat folgende Vorteile:

1. Ein Paßwort besteht immer nur für eine Sitzung. Ein Hacker, der das Paßwort abfangen würde, könnte damit nichts anfangen, weil bei der nächsten Anmeldung ein neues Paßwort generiert wird.
2. Es wird das Prinzip "Besitz und Wissen" angewendet. Ein Benutzer muß über einen Token verfügen und die richtige PIN kennen. Ein verlorener Token ist für den Finder solange wertlos, solange die PIN unbekannt bleibt.

Die Verwendung von SmartCards als Token anstelle des speziellen Rechners macht die starke Authentisierung benutzerfreundlicher und sehr sicher. Das Verfahren ist ähnlich, läuft aber im wesentlichen automatisch ab. Die einzige Aktion des Benutzers besteht aus seiner Authentisierung gegenüber der SmartCard durch Eingabe einer PIN oder durch ein biometrisches Merkmal. Danach fordert der Host die SmartCard durch Übermittlung einer Random-Zahl heraus (Challenge). Die SmartCard errechnet die Antwort und sendet sie an den Host zurück, der sie verifiziert (Response). Nachdem die Anwendung von SmartCards nicht mehr so teuer ist, hat sie nur noch Vorteile gegenüber den auf Taschenrechnern basierenden Tokens.

### **Möglichkeiten der Benutzerverwaltung im Netzwerk**

Die einfachste Möglichkeit stellt die Arbeitsgruppe dar, wie sie beispielsweise bei Windows 95/98/ME oder MacOS zu finden ist. Es gibt keine zentrale Benutzerverwaltung. Der Verwaltungsaufwand ist gering, aber nur für wenige Benutzer und Rechner geeignet ('Wohngemeinschafts-Netz'). Bei größeren Systemen greift man zum 'Domänen-Service', wie er beim Windows NT/2000 Server (mit Domain Controller) oder bei NIS, NIS+ (Network Information Services, zuerst bei UNIX verwendet), NDS (Network Directory Services von Novell) und LDAP (Lightweight Directory Access Protocol, eine vereinfachte und für das TCP/IP-Protokoll angepaßte Version des X.500 Protokolls). Die Benutzerverwaltung erfolgt zentral für alle Rechner. Das System ist übersichtlich, leichter zu warten und bietet hohe Sicherheit im Betrieb und Kontrolle. Nachteil: Benutzung ist von zentralen Komponenten abhängig.

Die zentrale Benutzerverwaltung kann man in zwei Gruppen einteilen:

- Flache Benutzerverwaltung, bei der alle Benutzer/Gruppen/Rechner, Services und Shares (Freigaben) in einem einzigen Verzeichnis liegen (z.B.: Windows NT 4.0/2000, Unix, Netware 3.X). Sie ist geeignet für bis zu 100 Rechner und eine homogene Benutzergruppe.
- Strukturierte (baumartige) Benutzerverwaltung, wie bei Netware 4 bis 6, NDS on Unix, NDS on NT, NDS on SAA, Windows 2000 mit AD, LDAP. Sie ist geeignet für beliebig viele Rechner und eine beliebige Benutzergruppe.

## **5.2 Datei-Zugriffsschutz**

Der Dateieintrag im Verzeichnis wird um einen Schutzeintrag erweitert, das die Zugriffsrechte festlegt. Diese Rechte können je nach System feiner oder gröber gestaffelt sein. Am einfachsten ist die

## Betriebssysteme

Vergabe von Lese- oder Schreibrecht für alle Benutzer. Die Zugriffsrechte selbst lassen sich weiter unterteilen und auch für verschiedene Benutzer unterschiedlich gestalten - bis hin zur Zuteilung dedizierter Rechte an einen oder mehrere bestimmten Benutzer.

Einen Mittelweg beschreitet UNIX, das hier als einfaches Beispiel dienen soll. Für eine Datei gibt es unter UNIX drei Zugriffsarten:

- R (read): Lesen aus der Datei
- W (write): Schreiben in die Datei (Ändern/Verkürzen/Erweitern)
- X (execute): Ausführen der Datei (bei Programmen)

Da es durchaus möglich sein sollte, anderen Benutzern das Lesen (nicht aber das Schreiben) einer Datei zu gestatten; es u. U. auch Dateien mit wichtigen Informationen gibt, die man vor eigenen Fehlern schützen will, gibt es drei Gruppen der Zugriffsberechtigung:

- Zugriffsrechte für den Datei-Eigentümer
- Zugriffsrechte für die Arbeitsgruppe des Eigentümers
- Zugriffsrechte für alle anderen

Es ergeben sich also neun Zugriffsrechte:

S	I	S	I	S	I	R	W	X	R	W	X	R	W	X
Spezial			Benutzer			Gruppe			Andere					

Die im Bild links angegebenen weiteren drei Zugriffsrechte dienen besonderen Aufgaben. Ein Benutzer kann durchaus mehreren Arbeitsgruppen angehören. Durch die Zuordnung von Gerädateien zu bestimmten Gruppen, kann man die Verwendung der Geräte auf eine bestimmte Gruppe von Programmen beschränken. Beispiele:

Der Zugriff auf die Druckerschnittstellen wird nur einem User gewährt. In diesem Fall stehen die Drucker nur den Druckerprozeß zur Verfügung und sind für andere Prozesse gesperrt.

Die seriellen Schnittstellen, an denen Modems angeschlossen sind, werden der Gruppe "Modem" zugeordnet. Um Benutzern den Zugriff auf die Modems zu gestatten, muß der Systemverwalter nur diese Benutzer der Gruppe "Modem" zuordnen.

Bei einigen Betriebssystemen werden die Zugriffsrechte noch verfeinert, so läßt sich beispielsweise die Schreiberlaubnis noch weiter unterteilen:

- beliebiges Schreiben auf die Datei
- Ändern von bestimmten Teilen der Datei
- Anhängen von Daten an die Datei
- Löschen der Datei

### Rechte und Attribute auf Dateien im Vergleich

Attribut/Recht	Windows NT/2000	Unix	Netware
Lesen	R	r	R
Schreiben	W	w	W
Löschen	D	w	E
Ausführen	X	x	X
Rechte setzen	P	w (Verzeichnis)	A

## Betriebssysteme

Besitzer ändern	O	w (Verzeichnis)	A
Attribute ändern	W	w (Verzeichnis)	M
Super-Rechte		root-User	S

### Rechte und Attribute auf Verzeichnisse

Attribut/Recht	Windows NT/2000	Unix	Netware
Schreiben	W	w	W
Löschen	D	w	E
Verzeichnis anzeigen	X	x	F
Rechte setzen	P	w	A
Besitzer ändern	O	w	A
Attribute ändern	W	w	M
Super-Rechte	û	root-User	S
Dateien anlegen	C	w	C

*Anmerkungen:* Unter Windows NT/2000 können mit jeder Freigabe (Share) allgemeine rechte für Benutzer und Gruppen eingestellt werden. Dies ist eine Art von "Ersatz" für die fehlende Vererbung. Das "x" unter Unix kennzeichnet ausführbare Dateien. Die Nameserweiterung ist bei UNIX vollkommen egal. Das "X" unter Windows NT/2000 und Netware ist ein 'eXecute only'-Attribut. Unter Unix gibt es noch die Attribute "l" für symbolische Links (Zeiger auf eine andere Datei, 'Verknüpfung'), "s" beim Eigentümer für SUID (Ausführbare Dateien behalten die User-ID des Dateieigentümers, denn normalerweise laufen alle Programme unter der User-ID des Aufrufenden.) und "s" bei der Gruppe für SGID (Bei Dateien wie SUID, jedoch gruppenbezogen, bei Verzeichnissen wird die Gruppenzugehörigkeit neuer Dateien automatisch auf die des Verzeichnisses gesetzt).

Die Dateifreigabe erfolgt mit den jeweiligen Kommandos des Betriebssystems. Bei UNIX ist dies meist das Shell-Kommando 'chmod' (Change Modus), mit dem für Dateien, Verzeichnisse oder ganze Verzeichnisbäume die Zugriffsrechte gesetzt werden können. Die Zuordnung der Daten und Verzeichnisse zu bestimmten Benutzern und Gruppen erfolgt über die Kommandos 'chown' (Change Owner) und 'chgrp' (Change Group), die aber nur vom Superuser 'root' verwendet werden können.

Bei Windows 95/98 können Sie Ordner und Drucker freigeben, indem Sie mit der rechten Maustaste darauf klicken und Freigabe auswählen. Dabei ist es möglich, nur Lese- oder Lese- und Schreibzugriff zu gestatten und dafür jeweils Paßwörter einsetzen. Die Option Freigabe auf Benutzerebene kann nur in Verbindung mit einem Windows NT- oder Windows 2000-Server verwendet werden. Die Freigabe der Ordner und Drucker erfolgt ebenfalls durch unter 'Freigabe' im jeweiligen Kontextmenü (rechte Maustaste). Bei NTFS-Laufwerken Windows NT/2000) können Sie detaillierte Rechte für das Verzeichnis, alle Unterverzeichnisse oder auch einzelne Dateien vergeben. Im Kontextmenü unter Sicherheit-Berechtigungen legen Sie fest, welche Benutzer welche Aktionen ausführen dürfen. Sind Windows 95/98-Rechner im Netzwerk aktiv, dürfen Sie für die Freigabennamen nicht mehr als zwölf Zeichen verwenden. Ansonsten erscheint die Freigabe auf diesen Rechnern nicht. Windows NT/2000 versucht zunächst, Sie mit dem Benutzernamen und Kennwort anzumelden, das Sie bei der Windows-Anmeldung verwendet haben. Ist damit keine Anmeldung möglich, können Sie Kenn- und Paßwort neu eingeben. Datei- und Druckerfreigaben sollten immer mit Paßwort geschützt werden.

Viele Betriebssysteme kennen weitere Einschränkungen, z. B.:

- Account Restrictions  
Den Benutzern kann eine Reihe von Einschränkungen auferlegt werden, unter anderem die Höchstanzahl gleichzeitiger Logins, eine Mindestlänge für Paßwörter und eine Geltungsdauer

für Paßwörter (z. B.: UNIX, Novell).

- Time Restrictions

Der Login im Netz kann auf bestimmte Zeiten begrenzt werden, z. B. nur innerhalb der normalen Arbeitszeit. Die Zeitbeschränkung kann sowohl auf Default- als auch auf individueller Ebene angegeben werden (z.B.: Novell).

- Station Restrictions

Außer der Einschränkung der Anzahl sämtlicher Logins für die einzelne Benutzer-ID ist es auch möglich, das Login auf eine bestimmte Rechner beschränken. Die Station kann durch die MAC- oder IP-Adresse identifiziert werden.

### 5.3 Verteile Dateisysteme (Beispiel NFS)

Das Network File System (NFS) ist ein verteiltes Dateisystem, das transparenten Zugriff auf die Festplatten eines fernen Rechners bietet. Für den Benutzer unterscheiden sich ferne über NFS eingehängte Dateisysteme nicht von lokalen Dateisystemen. NFS stellt eine Client/Server-Beziehung zwischen den beteiligten Rechnern her. Der NFS-Server stellt Dateisysteme im Netz zur Verfügung, die ein NFS-Client mit Hilfe des mount-Protokolls in sein lokales Dateisystem einhängen kann. Auch wenn das NFS-Protokoll seinen Ursprung in der Unix-Welt hat, ist es auf Portabilität zwischen verschiedenen Rechnerarchitekturen, Betriebssystemen und Netzwerkarchitekturen ausgelegt. Als offener Standard für die Nutzung zentraler Dateisysteme ist NFS in heterogenen Netzen oft eine bessere Lösung als proprietäre Protokolle. NFS betrachtet immer nur Komponenten eines Pfades, nie den kompletten Pfad auf einmal. Der Grund ist, daß verschiedene Betriebssysteme verschiedene Darstellungsformen für Pfade haben. Die Auflösung von Pfadnamen in die Komponenten erfolgt clientseitig.

Ziel der Entwicklung von NFS war ein zustandsloses Protokoll. Der Server braucht keine Informationen über die Zustände seiner Clients zu speichern. Zustandslose Server haben im Fehlerfall einen entscheidenden Vorteil gegenüber zustandsbehafteten Servern, da nach einem Neustart keine Zustände wiederhergestellt werden müssen. Ein Client braucht seine Anfrage nur solange wiederholen, bis der Server antwortet. Wenn der Server kurzzeitig nicht erreichbar ist, kann die Clientanfrage, sobald der Server wieder läuft, bearbeitet werden. Bei zustands-behafteten Protokollen hätte der Client dagegen nach dem Serverneustart dessen Zustand wieder aufbauen müssen (z.B. neue Anmeldung) oder die Operation abbrechen müssen. Dies ermöglicht sehr einfache Serverimplementierungen ohne komplizierte Fehlerbehandlung. Einige Datei- und Verzeichnisoperationen, z.B. das exklusive Öffnen von Dateien (locking), sind jedoch zustandsbehaftet. Deshalb werden diese nicht im NFS-Server, sondern als separate Dienste (z.B. der Network Lock Manager für das Sperren von Dateien) implementiert.

Zum Nutzen entfernter Dateisysteme müssen diese zuerst lokal bekanntgemacht werden. Dies geschieht mit Hilfe des Mount-Protokolls. Man spricht vom Mounten oder Einhängen der entfernten Dateisysteme. Das Mount-Protokoll ist strenggenommen nicht Teil des NFS-Protokolls, wird aber für dessen korrekte Funktionsweise benötigt. Der Client kann eine Anfrage nach einer Serverfreigabe an den Mount-Server stellen. Dieser liefert, falls diese Freigabe existiert und der Client das Zugriffsrecht besitzt, ein Handle zurück. Mit Hilfe dieses Starhandles kann der Client das entfernte Dateisystem per NFS durchsuchen. Der Mount-Server ist ein zustandsbehafteter Server. Er verwaltet eine Liste aller Clients mit gemounteten Dateisystemen. Diese Liste ist jedoch für die korrekte Funktionsweise nicht nötig, sie wird nur für Nachrichten an die Clients genutzt (z.B. vor dem Herunterfahren des Servers).

### 5.4 Remote-Zugang

Der Zugang zum Netz über Wählleitungen (analoges Telefon, ISDN, xDSL) erfolgt normalerweise

## Betriebssysteme

über einen oder mehrere Remote Access Server (RAS), in Einzelfällen auch über einen Rechner mit angeschlossenem Modem, ISDN-Karte oder xDSL-Anschluß. Deren Aufgabe ist es, ankommende digitale oder analoge Anrufe entgegenzunehmen, eine Benutzerauthorisierung durchzuführen und, falls diese erfolgreich war, die Verbindung des anrufenden Rechners mit dem internen Datennetz freizugeben. Der ferne Rechner verhält sich dann so, als ob er direkt am Datennetz angeschlossen wäre. Als Übertragungsprotokoll wird in der Regel *PPP* (Point to Point Protokoll, erlaubt IP- und IPX-Verbindungen), *SLIP* (Serial Line Internet Protokoll, veraltet, nur für IP-Verbindungen) und ggf. *ARAv2* (Apple Remote Access Version 2) angeboten.

Ein spezieller Terminalserver-Modus gestattet es, sich mit einem normalen Terminalprogramm (z.B. Hyperterminal, Kermit, usw.) auf dem Access Server anzumelden und von dort aus Telnetverbindungen aufzubauen. IP Adressen werden normalerweise aus einem *IP Adresspool* vergeben. Oft werden auch *virtuelle Verbindungen* unterstützt. Diese erlauben den physikalischen Abbau von Verbindungen, wenn gerade keine Daten übertragen werden, ohne daß die logische Verbindung verloren geht. Die Verbindung wird automatisch mit den gleichen Parametern wie vorher wieder aufgebaut, wenn Daten wieder übertragen werden müssen. Für die standardisierte Authentifizierung am Modem- und am Internetzugang setzt sich zunehmend das RADIUS-Protokoll (Remote Authentication and Dial-In User Service) durch. Seine Client-Proxy-Server-Architektur erlaubt die flexible Positionierung an Netzzugangspunkten und wird von fast allen Herstellern von Modemservern unterstützt. In Kombination mit DHCP und PPP ist die Aufgabe der Konfiguration der anwählenden Endsysteme in automatisierter Weise gelöst. Der Radius-Server ist ein zentraler Authentifizierungs-Server, an den sich alle RAS-Server wenden. Auf diese Weise lassen sich unabhängig von der Netz-Infrastruktur alle Remote-User zentral verwalten und Benutzerprofile mit Zugangsrestriktionen definieren, aber auch zusätzliche Sicherheitsverfahren vorsehen. Beispielsweise kann festgelegt werden, dass der Nutzer nur nach einem Rückruf durch den Einwahlknoten an eine zuvor vereinbarte Rufnummer Zugriff auf das Unternehmensnetzwerk bekommen darf. Diese Informationen übergibt der Radius-Server an den RA-Server, der das weitere Login entsprechend koordiniert. Der Vorteil dieses Verfahrens liegt in den einmalig generierten Zugangsdaten der Nutzer, die auch in verteilten Netzwerken jederzeit aktuell verfügbar sind und mit einfachen administrativen Eingriffen an zentraler Stelle definiert und verändert werden können. Darüber hinaus ist die innerbetriebliche Abrechnung der Nutzung des Systems durch ein entsprechendes Accounting möglich.

Das Radius-Protokoll setzt auf UDP auf. Die Struktur eines Radius-Pakets ist ausgesprochen einfach. Es besteht aus fünf grundlegenden Elementen: einem Radius-Code, einem Identifier, einer Angabe zur Paketlänge, einem Authenticator und gegebenenfalls aus einer Reihe von Attributen. Der Radius-Code beschreibt die Aufgabe des Datenpakets. Die Codes 1, 2 und 3 verwalten den reinen Access vom Request bis zur Bestätigung oder Abweisung. Die Codes 4 und 5 dienen dem Accounting. Der Identifier ist acht Bit lang und dient der Zuordnung von Anfragen und Antworten. Das sicherheitstechnisch wichtigste Feld eines Radius-Rahmens ist der Authenticator, der eine Länge von 16 Oktetts beziehungsweise vier 32-Bit-Worten hat. Dabei wird zwischen dem Request Authenticator und dem Response Authenticator unterschieden. Inhalt des Request Authenticators ist eine Zufallszahl, die das gesamte Feld ausfüllt. Die Länge dieser Zufallszahl gewährleistet mit einer sehr hohen Wahrscheinlichkeit die Einmaligkeit dieses Wertes. Damit bietet das System einen gewissen Schutz vor Hackerattacken. Mit dem Versand des Request Authenticators werden die Zugangsdaten des Nutzers, der sich im gesicherten Netzwerk anmelden möchte, als Attribute übergeben. Der Radius-Server wird diese Anfrage entweder mit einer Access-Accept-, Access-Reject- oder Access-Challenge-Nachricht beantworten, die ihrerseits mit einem 16 Oktett langen Response Authenticator versehen ist. Dieser ist ein MD5-Hash-Fingerprint setzt sich zusammen aus dem empfangenen Radius-Paket einschließlich der Attribute sowie den geheimen Zugangsdaten, die auf dem Server abgelegt sind, zusammensetzt. Die Attribute eines Radius-Pakets beinhalten alle wichtigen Informationen, die zwischen dem RAS und dem Radius-Server ausgetauscht werden müssen. Aufgrund dieser Werte wird dann der Benutzer vom RAS akzeptiert oder zurückgewiesen.

Der Remote angemeldete Benutzer kann nach der Authentisierung auf alle Serverdienste innerhalb des Firmennetzes zugreifen. Problematisch wird es, wenn auch ein administrativer Zugriff auf die Server erfolgen soll. Bei Servern auf UNIX- oder Linux-Basis ist der Zugang über Telnet möglich. Der Administrator hat dann ein Terminal-Fenster mit textbasiertem Dialog auf seinem lokalen Rechner und kann per Kommandozeilen alle Aufgaben erledigt. Da beim Telnet-Protokoll alle Informationen im Klartext übertragen werden, ist es auf jeden Fall ratsam, statt Telnet die 'Secure Shell' (SSH) zu verwenden, bei der alle Datendialoge verschlüsselt erfolgen. Ist der Rechner des Administrators mit einer grafischen Oberfläche (X Window System) ausgestattet, können die Arbeiten des Administrators auch grafisch erfolgen.

Bei Windows-Servern ist die Fernadministration schon schwieriger. Bei Windows NT ist man auf Fernwartungs-Programme wie "Pcanywhere" von Symantec oder das kostenlose Hackertool "BackOrifice" (<http://www.cultdeadcow.com/tools/bo.html>) angewiesen. Die Installation von Windows 2000 ist in bezug auf Kommandozeilen-Utilities für die Remote-Administration etwas mangelhaft. Unter Windows 2000 wurde für die Fernadministration von Servern ein Remoteverwaltungstool entwickelt, der in den Terminal-Server Diensten integriert wurde. Bei Windows 2000 wurden zum Unterschied zu Windows NT die Terminalserverdienste schon integriert. Der Terminaldienst kann in zwei Betriebsmodi installiert werden, im "Remoteverwaltungsmodus" oder im "Anwendungsmodus".

- Der "Remoteverwaltungsmodus" ist ausschließlich für die Fernadministration der Server gedacht. Darum sind hierfür auch keine extra Lizenzen erforderlich. Die Anzahl der möglichen Verbindungen zum Server sind begrenzt.
- Der "Anwendungsmodus" entspricht dem ehemaligen Terminal-Server weshalb für diesen Modus auch entsprechende Client-Lizenzen erforderlich sind. Hier können dann aber auch 32-Bit-Anwendungen über diesen Modus ausgeführt werden.

Auch der Telnetserver kann zur Fernadministration benutzt werden. Unter Windows NT gab es nur einen Telnet-Client und der Telnet-Server mußte aus dem Resource-Kit oder anderen Tools genommen werden. Windows 2000 liefert nun auch einen Terminal-Server mit, welcher automatisch mit installiert wird, aber noch als Dienst aktiviert werden muß. Der Telnetserver bietet aber nur eine Befehlszeile, wogegen beim Terminalserver auch die grafischen Administrationstools benutzen werden können.



[Zum vorhergehenden Abschnitt](#)



[Zum Inhaltsverzeichnis](#)



[Zum nächsten Abschnitt](#)

---

Copyright © FH München, FB 04, Prof. Jürgen Plate

## Einführung in Betriebssysteme



von Prof. Jürgen Plate

---

# 6 Netzwerkbetriebssysteme

Die ursprüngliche und wohl immer noch wichtigste Aufgabe von Netzwerkbetriebssystemen ist die Nutzung von Ressourcen wie Drucker, Plotter, Festplatten, CD-ROMs usw. sowie eine sichere und zentrale Speicherung der Daten. Die Kommunikation der Nutzer im internen Netz (Intranet), aber auch in weltweiten Datennetzen (z. B. E-Mail, Internetzugang, WWW, Groupware usw.), gehört heute ebenso zu den Standardaufgaben, wie die Integration von Internet-Protokollen und -Diensten in die PC-Betriebssysteme Novell Netware und Windows zeigen. Auch die Anbindung an Großrechner sowie die Integration von spezifischen Applikationsservern sind wesentliche Aufgaben. An Bedeutung gewinnt insbesondere aus Kostengründen immer mehr die zentrale und einfache Konfiguration und Administration der Rechner im Netz.

In den 90er Jahren war verstärkt die Tendenz festzustellen, daß Systeme der mittleren Datentechnik (Minicomputer), in zunehmenden Maße die Mainframes verdrängten. Insbesondere UNIX- und Linux-Systeme haben hier viel Land gewonnen. Aus der ursprünglich Host-zentrierten DV fand eine Verlagerungen der EDV auf dezentrale Systeme in den Abteilungen statt. Merkmale sind u. a.:

- Verarbeitung am Arbeitsplatz
- verteilte Verarbeitung
- Verarbeitung auf Abteilungsebene
- Gleichrangige Netzwerkverarbeitung
- Individuelle Datenverarbeitung

Zu den Ursachen hierfür zählten die großen Fortschritte in der Entwicklungen lokaler Netze, z. B.:

- Festplatten-Verwaltung in Servern
- Multi-User-Software
- Datenverarbeitung in Netzen, Hosts usw.
- Fehlertoleranz
- Datenschutz und -sicherheit

Es setzte sich durch, daß Mainframes, Minis und PC-basierte LANs nicht gegeneinander antraten, sondern sich zunehmend ergänzten. Im Mittelpunkt sollte die Anwendung stehen, das heißt die zu erledigenden Aufgaben.

## 6.1 Merkmale von Netzwerkbetriebssystemen

Es lassen sind zwei Typen von Servern für Netzwerkbetriebssysteme unterscheiden:

- dedizierte Server (z. B. Novell): der Server ist nicht gleichzeitig als Arbeitsstation einsetzbar.
- nicht-dedizierte Server (z. B. UNIX, Windows 2000): der Server ist gleichzeitig auch als Arbeitsstation nutzbar (Client-Server-Prinzip).

Wichtige Merkmale von Netzwerkbetriebssystemen sind:

- Architektur des Betriebssystems
  - ◆ Server-Plattform als Kernstück des Betriebssystems, das alle Netzfunktionen bereitstellt, darunter Dateisysteme und Festplatten, Memory-Management, Prozeß- und Taskscheduling, File-, Print- und Backupdienste, File- und Recordlocking usw.
  - ◆ Redirection-Software auf den Clients, um den Zugriff auf Netzlaufwerke und -ressourcen möglichst transparent zu erlauben
  - ◆ unterstützte Netzwerkdienste

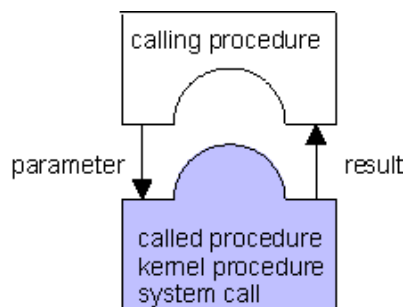
## Betriebssysteme

- ◆ Kommunikationssoftware
- ◆ Unterstützung verschiedener Client- und Server-Plattformen (z. B. Linux mit NFS für die UNIX-Umgebung und "Samba" für Windows-Clients)
- Leistungsfähigkeit und Zuverlässigkeit
  - ◆ Datendurchsatz
  - ◆ Verkabelung des Netzes
  - ◆ Netz-Komponenten (Repeater, Switches, Router)
  - ◆ kritische Anwendungen
  - ◆ Abhängigkeit des Betreibers vom Funktionieren des Netzes
- Sicherheit
  - ◆ Accounting- und Passwort-Sicherheit
  - ◆ Datei- und Directory-Sicherheit
  - ◆ Internetwork-Sicherheit
  - ◆ Fileserver-Sicherheit
- Standards
  - ◆ genormte Standards (ISO, IEEE, DIN, ANSI usw.)
  - ◆ Industriestandards
  - ◆ Anwendungsstandards, z.B. für serverbasierende Applikationen, clientbasierende Applikationen, verteilte Applikationen,
  - ◆ Protokollstandards, z.B. Medien-Protokolle (Ethernet, ATM, FDDI usw.), Transport-Protokolle (IP, IPX usw.), Client-Server-Protokolle
  - ◆ Standards der Interprozeß-Kommunikation, z.B. Sockets, TLI, Corba usw.

Die meisten Netzwerksysteme arbeiten nach dem **Client-Server-Prinzip**. Abhängig von der Arbeitsweise im Netzwerk sind zu unterscheiden LAN und Single-User-Anwendungen, wo über das Netz meist nur ein Dateisystem zur Verfügung gestellt wird oder LAN und netzwerkfähige Software, mit Ausnutzen von File-Sharing sowie File- und Record-Locking (mehrere Nutzer arbeiten mit gemeinsamen Daten).

## 6.2 Remote Procedure Call (RPC)

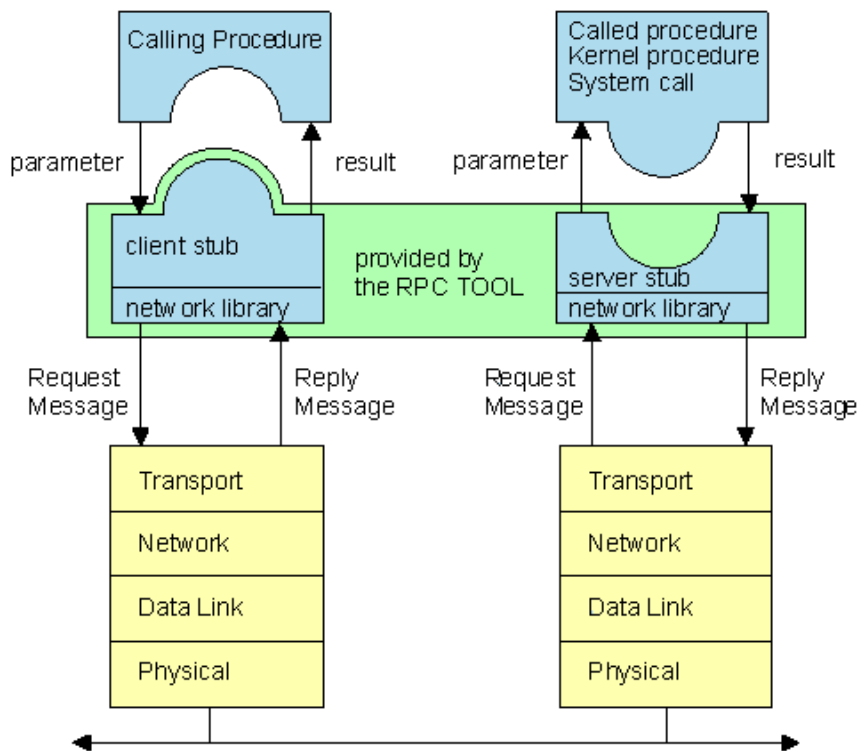
Das Verfahren der Remote Procedure Calls wurde Anfang der 80er Jahre vom Sun Microsystems für ihr Network File System (NFS) entwickelt. Es ist derzeit das wesentliche Element in Netzwerkbetriebssystemen, um Serverdienste für Clients zur Verfügung zu stellen. Ein lokaler Prozeduraufruf kann folgendermaßen skizziert werden:



Eine Prozedur oder Funktion wird mit den entsprechenden Parametern aufgerufen, und kehrt nach erledigter Arbeit mit einem Resultat zurück. Für Dienste des Betriebssystems werden i.d.R. sog. System Calls, also Aufrufe von Prozeduren des Systems genutzt.

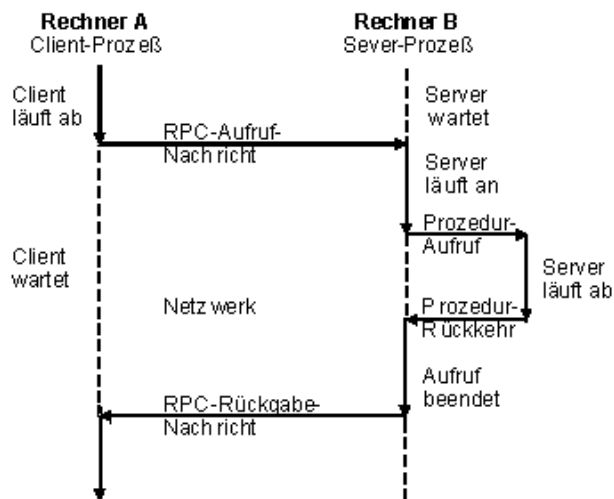
Nach diesem Schema arbeiten auch RPCs:

## Betriebssysteme



Eine Anwendung ruft einen Netzwerkdienst auf wie eine lokale Prozedur mit Übergabe von Parametern auf. Der Aufruf wird von der RPC-Library in ein RPC-Paket verpackt und über das Netz an der Server gesendet, der den Dienst ausführt und das Resultat liefert. Dieses wiederum wird an die Anwendung zurückgegeben.

Damit kann der zeitliche Ablauf eines RPC so skizziert werden:



[◀ Zum vorhergehenden Abschnitt](#)

[⬆ Zum Inhaltsverzeichnis](#)

[▶ Zum nächsten Abschnitt](#)

Copyright © FH München, FB 04, Prof. Jürgen Plate



Betriebssysteme

# Einführung in Betriebssysteme

*von Prof. Jürgen Plate*

---

# 7 PC-Betriebssysteme

## 7.1 MS-DOS

MS-DOS (Microsoft Disk Operating System) ist aus heutiger Sicht veraltet; berücksichtigt man aber die Hardwaregegebenheiten der ersten PCs, war es damals durchaus sinnvoll konzipiert. Die wichtigsten Eigenschaften in Stichpunkten:

- für 16-Bit-Prozessoren (8086 und höher)
- für Speicherbereich bis 1 MByte (real mode)
- kommandozeilenorientiert
- Singleuser-Singletasking-Betriebssystem
- hierarchisches Dateisystem
- spezielle Hardware über Treiber einbindbar
- Systemaufrufe nicht reentrant
- Geräte nur rudimentär ins Dateisystem eingebunden
- sehr einfache Kommandosprache

MS-DOS besteht aus mehreren Teilen (= Module), die in einem reservierten Bereich auf der Festplatte oder Diskette liegen und von Rechner beim Systemstart in den Hauptspeicher geladen und aktiviert werden.

- **IO.SYS (IBMBIO.SYS)**  
Dieser Teil wird auch BIOS (Basic Input/Output-System) genannt. Er enthält alle Teile von DOS, die sich mit den physikalischen Geräten befassen. Jedesmal, wenn DOS auf Tastatur, Drucker, Platte oder Bildschirm zugreift wird dieses Modul aktiv. Neben den Geräteroutinen enthält IO.SYS auch die Routinen zur Initialisierung von DOS.
- **MSDOS.SYS (IBMDOS.SYS)**  
Dieses Modul enthält die geräteunabhängigen Systemroutinen und die Platten- und Dateiverwaltung. Auf diese Routinen greifen auch die Anwenderprogramme zu. MSDOS.SYS arbeitet eng mit IO.SYS zusammen. Bei jedem Gerätezugriff werden die Routinen von IO.SYS aufgerufen. Beim Start dieses Teils von MS-DOS wird auch die Datei CONFIG.SYS ausgewertet und dort verzeichnete Gerätetreiber geladen.
- **COMMAND.COM**  
Dies ist der sogenannte Kommandointerpreter, der die Eingaben des Benutzers entgegennimmt und dann das gewünschte Kommando ausführt. Einfache und häufig benötigte Befehle sind direkt im Kommandointerpreter eingebaut. Der Kommandointerpreter verarbeitet auch die Batch-Dateien.

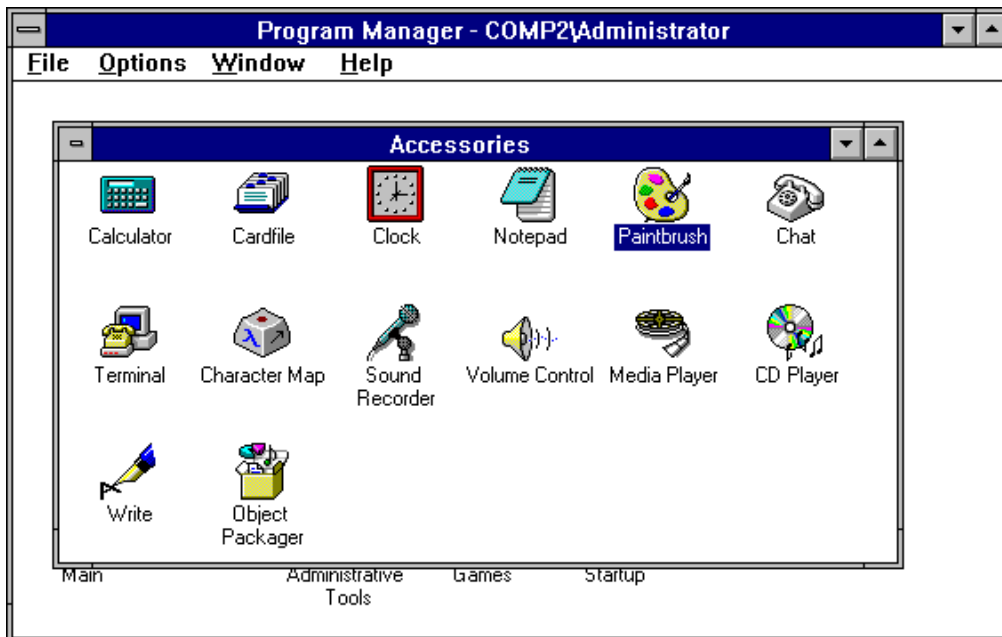
DOS-basierte Applikationen durchbrechen oft die o. g. Schalenstruktur (teilweise, weil das BS die gewünschten Dienste nicht zu leisten imstande ist). Vielfach wird direkt auf die Hardware des PC (bis hinunter zur Registerebene) zugegriffen. Daher kommt es manchmal auch zu Inkompatibilitäten. Andererseits kann man eben unter DOS wirklich "alles" mit der Hardware machen, z. B. das Einbinden von Interrupt-Serviceroutinen in eigene Programme oder Gerätetreiber.

## 7.2 Windows (Version 3.x)



Im Gegensatz zu Windows 3.x hatten die Versionen 1 und 2 eine reine Textoberfläche

Mit Microsoft Windows wurde erstmals eine Betriebssystem-Erweiterung mit grafischer Benutzeroberfläche für Intel-PCs auf den Markt gebracht. In Ihren Eigenschaften entspricht die Version 3 in etwa dem Apple-Macintosh-System (Motorola-Prozessoren). Windows setzt auf DOS auf.



Die Eigenschaften in Stichpunkten:

- 32-Bit-Adressierung im protected mode (386/486/Pentium)
- kooperatives Multitasking
- grafische Benutzeroberfläche
  - ◆ Steuerung über Maus (Pointer Device)
  - ◆ Desktop als Systemumgebung
  - ◆ Fenster als Arbeitsfläche
  - ◆ Pulldownmenüs und Rollbalken
  - ◆ Sinnbilder (Icons) für Programme und Daten
  - ◆ Dialogboxen, Knöpfe (Buttons), Schieberegler,..
  - ◆ Schnittstelle für Programme (GDI=Graphics Display Interf.)
- einheitliche Gerätetreiber, auf welche die Programme aufsetzen
- Erweiterung der Systemaufrufe (API=Application Program Interf.)

## Betriebssysteme

Windows-Programme setzen also auf GDI und API auf. Der Vorteil liegt darin, daß alle Programme geräteunabhängig programmiert werden können und auf den Treibern für Bildschirm, Drucker, CD-ROM, etc. aufsetzen- für Anwendungsentwickler ein Vorteil. Für diese hat Microsoft auch Richtlinien für die Oberfläche und Menügestaltung herausgegeben (CUA=Common User Access), die jedoch nicht so konsequent eingehalten werden, wie dies bei Macintosh-Software der Fall ist (wo die Bedienung der Programme wirklich total einheitlich gestaltet ist).

Um Programme möglichst klein zu halten und mehrere Programme auf gemeinsam verwendete Unterprogramme zugreifen zu lassen, sind sogenannte "Dynamic Link Libraries" (DLLs) eingeführt worden. Diese Codebibliotheken werden je nach Bedarf nachgeladen.

Erste Schritte in das objektorientierte Arbeiten sind mit "Drag-and-Drop" eingeführt worden. Wenn ein Datei-Icon (z. B. ein Dokument) auf das Symbol seines erzeugenden Programms (z. B. ein Texteditor) gezogen wird, startet dies Programm automatisch zur Bearbeitung dieser Datei gestartet. Beim Macintosh genügt hier das "Anklicken" des Dokuments mit der Maus.

Zum Datenaustausch der Programme untereinander dient zum einen das Clipboard (Zwischenablage), das die Windows-Formate für Text, Grafik, etc. unterstützt, zum anderen OLE (Object Linking and Embedding). OLE wird nicht von allen Applikationen unterstützt. Es wird hier ein Verweis auf das gewünschte Objekt (Daten + Programm zu deren Erzeugung) eingetragen. Wird dann z. B. in einem Text ein Bild referiert, startet automatisch die Applikation zur Bearbeitung.

Noch eine Stufe weiter geht DDE (Dynamic Data Exchange). Hier werden Änderungen in den Originaldaten automatisch in alle Dateien übertragen, in denen auf das Objekt bezug genommen wird.

Das kooperative Multitasking läßt zwar das gleichzeitige Arbeiten mit mehreren Programmen zu, jedoch ist nur immer eine Applikation aktiv. Lediglich einfache Dienste wie z. B. serielle Datenübertragung oder Druckausgabe laufen im Hintergrund weiter. Zeitkritische Anwendungen - wobei es sich dabei auch nur um so einfache Dinge wie Datenübertragung oder Meßwerterfassung handeln kann - können schon Probleme bringen. Abhilfe kann manchmal eine Änderung der Priorität bringen. Nicht "kooperierende" Anwendungen (z. B. Absturz eines DOS-Programms im sogenannten DOS-Fenster) können das gesamte System lahmlegen.

Mit der Erweiterung "Windows für Workgroups 3.11" ist auch eine grundlegende Netzwerkunterstützung gegeben. Es handelt sich um ein Peer-to-Peer-Netz, bei dem von jedem Rechner aus Ressourcen eines anderen Rechners im Netz genutzt werden können.

## 7.3 Windows 95

Mit der Entwicklung von Microsofts Windows 95 wurde der richtige Schritt in Richtung Stabilität und Anwenderfreundlichkeit getan, Windows 95 ist das erste 32-bit Microsoft Windows Betriebssystem für den Mainstream-PC. Die alten Windows-Versionen hatten den Nachteil, daß sie nur auf MS-DOS Basis liefen, es sich also nicht um ein Betriebssystem, sondern mehr um eine grafische Benutzeroberfläche handelte, die mit den DOS Speicherbegrenzungen zu kämpfen hatte.

Windows 95 hingegen ist ein vollwertiges Betriebssystem, dem zwar MS-DOS 7.0 beiliegt, das aber genausogut auch ohne DOS arbeiten kann. Es besitzt eine Oberfläche, die nicht mehr programmsondern objektbezogen arbeitet, unterstützt preemptives Multitasking, beinhaltet ein verbessertes Dateisystem, das lange Dateinamen erlaubt und verfügt über Plug & Play Funktionalität, die ein schnelleres und bequemerer Wechseln von Hardwarekomponenten erlaubt. Windows 95 kann zwar auch auf einem PC mit 386SX CPU und 4 MByte Hauptspeicher installiert werden, ein flüssiges Arbeiten wird jedoch erst erreicht mit einem PC mit Pentium CPU, 16 MByte RAM und mindestens einer 1 GByte Festplatte, mehr Hauptspeicher kann die Verarbeitungsgeschwindigkeit erheblich erhöhen, insbesondere beim Einsatz umfangreicher Office-Pakete sind 32 MByte oder mehr

## Betriebssysteme

empfehlenswert. Um die Soundfunktionen von Windows 95 zu nutzen, benötigt man eine Soundkarte im Rechner.

Windows 95 unterstützt vollständige 32-bit Protected-Mode-Versionen von TCP/IP, IPX/SPX und NetBEUI, damit ist der Client Zugriff auf einen NOVELL NetWare/IntraNetWare oder Windows NT Server möglich. Unterstützung mehrerer simultaner Kommunikationsprotokolle erlauben Netzwerkverbindungen unterschiedlichen Typs gleichzeitig aufrecht zu erhalten. Damit ist das parallele Arbeiten in einem NetWare-Netzwerk über IPX und einem Unix-Netzwerk über TCP/IP möglich. Die eingebetteten Netzwerk-Clients für NetWare, Windows NT Server und Microsoft-Exchange-Server sorgen dafür, daß sich der Anwender nur einmal anmelden muss. Durch ein DFÜ-Netzwerk (RAS, Remote Access Services) können Benutzer von unterwegs schnell und einfach mit häufig genutzten Netzwerken verbunden werden.

In den bisherigen Windows-Versionen wurden die Programmeinstellungen und Windows Hardwaretreiber in den sogenannten INI Dateien im Windows Verzeichnis verwaltet. Windows 95 verwaltet nun zu den INI Dateien (um mit 16-bit Applikationen kompatibel zu bleiben) eine eigene Datenbank (Registry), in der Windows 95 und 32-bit Programme Informationen ablegen. Sie ermöglicht anwenderspezifische Einstellungen, wie etwa individuelle Einstellungen des Desktops oder des Netzwerkzugriffes und enthält hardwarespezifische Einstellungen des PCs. Sie bietet System-Richtlinien, mit denen der Systemverwalter die Konfiguration überwachen und Benutzereinstellungen festlegen kann. Die automatische Hardwareerkennung erleichtert das Aufrüsten und Erweitern des PCs, insbesondere bei modernen Komponenten nach dem Plug'n'Play (PnP) Standard. Sobald eine neue Komponente ins System integriert wird, erkennt dies Windows 95 und schlägt die Installation des Treibers vor. Die Unterstützung des dynamischen Anschlusses von PC-Cards (den ehemaligen PCMCIA-Karten) erlaubt es Notebookbesitzern, z.B. die Netzwerk-PC-Card im laufenden Zustand (Online) in den PC-Card-Slot zu stecken und direkt aufs Netzwerk zuzugreifen.

Die Eigenschaften in kürze:

- kein DOS mehr als "Unterlage", jedoch DOS 7.0 als "DOS-Fenster"
- preemptive Multitasking
- 32-bit-Operationen
- Treiber laufen im protected mode (Speicher oberhalb 1 MByte)
- besserer Speicherschutz der Applikationen
- Netzwerkintegration (IPX/SPX von Novell, TCP/IP, NetBEUI)
- Multimedia-Schnittstelle (Video und Sound)
- dynamische Cacheverwaltung (weniger Plattenzugriffe)
- Ordner (beliebige Verschachtelungsebenen von Applikationen)
- Links und Verknüpfungen (Daten <--> Programm)
- Dateimanager verbessert/erweitert (Explorer)
- 'Plug & Play': automatische Erkennung und Installation von Hardwarekomponenten
- Autokonfiguration (kein Setup mehr bei Hardwareänderung)
- Verbesserte Bedienoberfläche Statt des Program-Managers jetzt eine 'Taskbar' zum Programmstart, Icons für Programme/Verknüpfungen direkt auf dem Desktop, kontextbezogene Popup-Menüs über rechte Maustaste.
- lange Dateinamen
- Konfigurationsdatenbank
- Benutzerprofile

## 7.4 Windows 98

Windows 98 beinhaltet zahllose Detailverbesserungen sowie eine neue Benutzeroberfläche, die den Umgang mit einem Windows-98-PCs deutlich erleichtern soll. Neue Assistenten und Dienstprogramme sorgen dafür, daß die Systeme zuverlässiger laufen und einfacher zu verwalten sind. Wesentliche Neuerungen sind:

- **FAT32**  
FAT32 als Standarddateisystem. FAT32 ist eine verbesserte Version des Dateisystems FAT, mit dem Festplatten mit mehr als 2 GByte Kapazität als ein einziges Laufwerk formatiert werden können. FAT32 wurde bisher nur von der Windows 95 OEM-Version unterstützt.
- **Verbessertes Power-Management**  
Unterstützung für das Advanced Configuration and Power Interface (ACPI). ACPI ist eine von Intel, Microsoft und Toshiba vorgeschlagene offene Industriespezifikation, in der Hardware-Schnittstellen definiert werden, die ein standardisiertes Power-Management durch das Betriebssystem für alle Komponenten eines PC-Systems ermöglichen.
- **Assistent zur Datenträgeroptimierung**  
Er steigert mit Hilfe der Defragmentierung die Ladegeschwindigkeit der Anwendungen und den Zugriff auf die Dateien, die am häufigsten benutzt werden. Dazu legt der Assistent eine Protokolldatei an, in der aufgezeichnet ist, welche Programme am häufigsten verwendet werden. Nachdem diese Datei angelegt wurde, kann der "Assistent zur Datenträgeroptimierung" die Dateien, die mit diesen häufig ausgeführten Programmen verknüpft sind, hintereinander auf der Festplatte ablegen. Durch diese kontinuierliche Anordnung werden Anwendungen wesentlich schneller ausgeführt.
- **Windows-System-Update**  
Es handelt sich um einen neuen Web-basierten Dienst (in Form eines ActiveX-Bedienelements), der das System durchsucht und feststellt, welche Hardware und Software installiert ist. Anschliessend vergleicht er diese Informationen mit einer Back-End-Datenbank und ermittelt, ob neuere Treiber oder Systemdateien zur Verfügung stehen. Ist das der Fall, kann der Dienst die neuen Treiber automatisch installieren. Der Benutzer kann diesen Vorgang vollständig konfigurieren.
- **Dienstprogramm zur Systemdateiprüfung**  
Dieses Dienstprogramm stellt auf einfache Weise sicher, daß die Systemdateien von Windows 98 nicht verändert oder beschädigt wurden. Es bietet ausserdem eine einfache Methode für die Wiederherstellung der Originalversionen von veränderten oder fehlenden Systemdateien.
- **Win32 Driver Model**  
Dies ist ein völlig neues Modell für Treiber, die sowohl unter Windows 98 als auch Windows 2000 funktionieren. Das WDM ermöglicht für einige verbreitete Gerätetypen (z.B. USB und IEEE 1394) den Einsatz eines einzigen Treibers für beide Betriebssysteme. Das WDM wurde implementiert, indem mit Hilfe eines speziellen virtuellen Gerätetreibers (NTKERN.VXD) bestimmte NT-Kerndienste zu Windows 98 hinzugefügt wurden. Auf diese Weise ist Windows 98 in der Lage, alte Treiber ohne Einschränkung zu unterstützen und gleichzeitig den Einsatz neuer WDM-Treiber zu ermöglichen.
- **integrierte Internet Benutzeroberfläche**  
Hiermit wird der Internet-Zugriff zum festen Bestandteil der Benutzeroberfläche (weshalb dieses Feature auch bereits die Gerichte beschäftigte). Der Anwender muss nicht mehr die Bedienung mehrerer Umgebungen erlernen. Mit diesem universellen Programm können lokale, Netzwerk-, Intranet- und Internet-Daten auf die gleiche Weise angesehen werden.

## 7.5 Windows NT

Windows NT (New Technology) ist ein Server-basiertes Multitasking-Betriebssystem, also mit

## Betriebssysteme

Netzwerkunterstützung. Da auch Windows 95 oder IBMs OS/2 den Netzbetrieb unterstützen, ist die Zielgruppe recht indifferent. Unbestreitbarer Vorteil ist jedoch, daß es portable konzipiert ist und so auf verschiedenen Prozessoren implementierbar ist (Intel, Mips, DEC Alpha, etc.), was außer Unix bei keinem anderen der vorgestellten BS der Fall ist. Applikationen können somit Quelltextkompatibel erstellt werden. Interessant ist bei NT derzeit nur die Server-Anwendung. NT-3.x-Arbeitsstationen können etwa genausoviel wie Windows-95-Rechner. Die über das "normale" Windows hinausgehenden Eigenschaften sind:

- preemptive multitasking
- Aufteilung von Programmen in threads
- für Mehrprozessorsysteme geeignet
- virtuelle Speicherverwaltung (je Prozeß bis zu 4 GByte)
- Benutzerverwaltung, Zugriffsrechte, Authentifizierung
- Unterstützung großer Platten (bis 17'000'000 GByte)
- Unterstützung verschiedener Dateisysteme (FAT von DOS, HPFS von OS/2, NTFS von NT)
- Dateisystem NTFS (NT File System)
  - ◆ Zugriffsrechte
  - ◆ Fehlertoleranz (Wiederherstellung nach Systemabsturz)
  - ◆ Links, transaktionsorientierte Zugriffe
  - ◆ lange Dateinamen, Unterscheidung Groß-/Kleinschreibung
- dynamisch ladbare Gerätetreiber
- Netzwerkunterstützung (auch für Novell, TCP/IP, OS/2)

Das System zeigt sich gut strukturiert. Die Basis wird von drei Schichten gebildet:

- Hardware Abstraction Layer (HAL)
- Kern
- NT-Executive (Systemdienste)

Darauf setzen Subsysteme auf, die verschiedene Betriebssystem-Emulationen bereitstellen. Für DOS, OS/2 oder Windows-Anwendungen wird jeweils ein eigenes, abgeschottetes 32-Bit-Subsystem angelegt. Fehlerhafte Programme oder Verletzung von Zugriffsrechten beeinflußt die Arbeit des Servers nicht. Alle Netzzugriffe werden vom "Advanced Server" behandelt, einer übergeordneten Verwaltungsinstanz.

Konfigurationsdaten werden nicht in einzelnen Dateien, sondern in einer einzigen Konfigurationsdatenbank gespeichert. Das System und die Treiber greifen auf diese Datenbank zurück. Beim Bootvorgang überprüft das System die Funktion aller Komponenten.

Der Nachfolger **Windows NT 4.x** ist das leistungsstarke 32-bit-Betriebssystem von Microsoft. Es ist speziell optimiert für den Netzbetrieb in Verbindung mit dem Client-Server-Konzept. Microsoft vertreibt Windows NT in zwei Versionen, den Windows NT Server und die Windows NT Workstation. Beide Versionen werden auf getrennten CD-ROMs ausgeliefert. Die Windows-NT-Server-CD-ROM ist für den Aufbau eines kompletten NT-Netzwerk-Servers bestimmt, die Windows NT-Workstation-CD-ROM kann dazu genutzt werden, Arbeitsplatz-PCs mit dem Windows-NT-Workstationsystem auszustatten. Windows NT 4.0 stellt sich in der Oberfläche von Windows 95 dar, hat jedoch einen völlig anderen inneren Aufbau.

In Windows NT 4.0 ist der Microsoft Internet Explorer (Web-Browser) und bei Windows NT Server auch der Internet Information Server (IIS, Web-Server) integriert. Der Internet Information Server bietet die Errichtung eines eigenen Web-Servers mit den Diensten WWW, Gopher und FTP. Dieser Dienst integriert sich nahtlos in die BackOffice-Strategie von Microsoft. Windows NT 4.0 enthält einen DNS-Server (Domain Name Service) der in der alten Windows NT 3.51 Version oft vermisst wurde. Mit dem Remote Program Load (RPL) ist es möglich, diskless Workstations (ohne

## Betriebssysteme

Festplatte) unter Windows95 vom Windows NT-Server zu booten. Datenbankverbindungen über ODBC können über Internet Server realisiert werden. Wichtige Highlights von Windows NT 4.0 sind neben dem kostenlosen Internet Explorer und dem Exchange Client die Unterstützung einer grossen Anzahl neuer Treiber, höhere Grafikleistung durch die Verlagerung von Teilen des GDI in den Kernel, Einrichtung von Hardware Profiles und einer DirectX Schnittstelle um Spiele auf Windows NT lauffähig zu machen.

Windows NT unterstützt sowohl preemptives Multitasking und Multithreading wie auch Multiprocessing (Verteilen von Programmteilen auf mehrere CPUs). Es ist skalierbar auf bis zu 32 CPUs (Windows NT Workstation kann max. 2 CPUs ansprechen, für Windows NT Server gibt es je nach Prozessoranzahl unterschiedliche Lizenzen). Pro System werden 4 GByte RAM unterstützt, jeder Anwendung kann bis zu 2 GByte virtueller Arbeitsspeicher zugewiesen werden. Datenspeicher werden vom System bis zu 402 Mio. Terabyte unterstützt. Microsoft gibt als Systemvoraussetzung mind. eine 486 CPU, 16 MByte RAM und eine 500 MByte Festplatte an. Es hat sich jedoch gezeigt, dass ein Arbeiten mit akzeptabler Performance erst ab einer Pentium 100 CPU, 64 MByte RAM und einer 1 GByte Platte möglich ist. Windows NT unterstützt die Prozessorplattformen Intel x86/Pentium, und DIGITAL Alpha AXP/21x64. Die Software für beide Plattformen ist auf der CD enthalten und dort in verschiedenen Verzeichnissen untergebracht. Windows NT kann Anwendungen, die für IBMs Presentation Manager (bis Version 1.3) sowie POSIX 1003.1 geschrieben wurden, ausführen.

Am weitesten verbreitet ist Windows NT auf der Intel-Architektur. Windows NT auf Alpha-Architektur wird meist dann eingesetzt, wenn man auf sehr hohe CPU- oder I/O-Leistung Wert legt. Die Alpha-Version wird jedoch nicht mehr weiterentwickelt.

Im Lieferumfang ist die Unterstützung der Protokolle TCP/IP, NetBEUI, IPX/SPX, DLC und AppleTalk enthalten. Windows NT enthält Telnet und FTP-Clients sowie in der Server-Version einen FTP-Server Dienst. Das Dynamic Host Configuration Protocol (DHCP) ermöglicht die dynamische Einrichtung und Verwaltung von TCP/IP Adressen. Windows Internet Naming Service (WINS) ordnet den TCP/IP Adressen Namen zu. Für den Administrator und Benutzer wird es dadurch leichter in einem TCP/IP Netzwerk zu arbeiten. Es ist ebenfalls leicht möglich Windows NT in ein NetWare Netzwerk zu integrieren. Der Client Service für NetWare (CSNW) ermöglicht den Zugriff auf die Datei- und Druck-Services eines NetWare 3.x Servers. Der Gateway Service für NetWare (GSNW) bietet Arbeitsstationen im Windows NT Server-Netzwerk den Zugriff auf NetWare Server. Weiterhin werden Migrationstools angeboten, welche die Benutzerinformationen und Verzeichnisse übernehmen. Es ist somit möglich, mit einem Windows NT Server die komplette Benutzerverwaltung von NetWare zu übernehmen oder zu steuern.

Windows NT unterstützt Fernzugriffe mit den Protokollen NetBEUI, IPX/SPX und TCP/IP; dies kann über ISDN, X25 oder analoge Telefonleitungen realisiert werden. Als Client erlaubt Windows NT Unix über PPP/SLIP, NetWare, LanRovers, Windows 3.x, Windows95 sowie LAN Manager. Über diesen Remote Access Service (RAS) sind bis zu 256 gleichzeitige Verbindungen erlaubt. Mit RaRAS (Routing and Remote Access Service) bietet Microsoft einen Software-basierten Multiprotokollrouter für Windows NT Server 4.0. RaRAS unterstützt Routing von TCP/IP und IPX. Als Routing-Protokolle werden RIP und OSPF unterstützt, sowie statisches Routing. Bei der Authentisierung über PAP/CHAP greift RaRAS auf die Windows NT Domain-User-Authentisierung zurück. Unterstützt werden zudem RADIUS-Clients. Zentraler Bestandteil von RaRAS ist der Routing Table Manager. Hier werden die Routing-Tabellen verwaltet. Für Konfiguration und Management steht eine grafische Oberfläche zur Verfügung.

Windows NT ist zwar ein Multitasking-, aber im Gegensatz zu Unix kein Multiuser-Betriebssystem. Dieses Defizites hatte sich der amerikanische Softwarehersteller Citrix Systems angenommen. Bereits 1992 schlossen Citrix und Microsoft ein Abkommen über eine strategische Partnerschaft zur Entwicklung des Multiuser NT WinFrame. WinFrame ist die Grundlage für "Application Publishing",

einem neuen Weg für moderne Client/Server-Architekturen im PC Umfeld. Die Grundidee von Application Publishing ist nicht neu. Das Prinzip erinnert in weiten Teilen an Unix basierende Netze mit X-Window Terminals. Bei Application Publishing laufen die Anwendungen nicht auf dem einzelnen Arbeitsplatz-PC, sondern auf dem Server. Der Arbeitsplatzrechner bekommt nur die Fensterdarstellung über das Netz zugespielt, braucht also weder installierte Anwendungen noch hohe Rechenleistung oder Speicherausbau.

**Microsoft Windows 2000**, bisher unter der Bezeichnung Windows NT 5.0 bekannt, wird um etliche neue Eigenschaften und Funktionen erweitert. Dazu gehören die Bereiche Administrierbarkeit, Skalierbarkeit und Erweiterbarkeit sowie Storage und Hardware Management. Microsoft wird Windows 2000 wie NT in drei Versionen anbieten: Windows 2000 Professional entspricht der Windows NT Workstation, Windows 2000 Server dem NT Server und die NT Enterprise Edition wird als Windows 2000 Advanced Server weitergeführt.

Microsoft Windows 2000 implementiert Active Directory als zentrale Plattform, die den Zugriff und Management auf Netzwerk- und Systemressourcen vereinfacht. Weitere Features sind ein zentralisiertes Konfigurationsmanagement und die konfigurierbare und erweiterbare Microsoft Management Console (MMC).

Windows 2000 unterstützt max. 4 GByte physischen Speicher, bei 64-bit CPUs (Digital Alpha, Intel Merced) können 32 GByte adressiert werden. Mit dem Microsoft Cluster Server können zwei Server im Verbund arbeiten. Dabei überwachen sich die Geräte gegenseitig um bei einem Ausfall eines Servers ohne Unterbrechung den Betrieb aufrecht zu halten. Während dem normalen Betrieb können die Server die Arbeitslast untereinander aufteilen, um eine höhere Produktivität zu erreichen.

Das Dateisystem NTFS implementiert nun auch eine Quotierung, mit der den Benutzern der maximal zur Verfügung stehende Plattenplatz festgelegt werden kann. Die NTFS-Erweiterung EFS (Encryption File System) ermöglicht die Verschlüsselung sensibler Daten auf Datei- oder Directoryebene. Wegen der Exportbeschränkungen für Kryptoverfahren handelt es sich aber um sogenannte "schwache Verschlüsselung".

Plug-and-Play hält nun auch bei Windows 2000 Einzug. Dies ermöglicht dann auch den problemlosen Betrieb von PC-Cards in mobilen Rechnern. Zusätzlich soll durch Erweiterung des Windows Driver Models (WDM) erreicht werden, dass in Windows 98 und Windows 2000 identische Treibersoftware zum Einsatz kommen kann.

## 7.6 OS/2 (Version 3.0 - Warp)

OS/2 war in seiner ersten Version (ebenso wie Windows NT) sehr "speicherhungrig" - und das zu einer Zeit, wo PCs mit 8 MByte Hauptspeicher eher die Ausnahme waren. Inzwischen sind die Anforderungen geringer (4 MByte) und gleichzeitig die Rechner besser ausgestattet. Es ist ein eigenständiges Produkt, das nicht auf DOS oder Windows aufbaut. Im Gegensatz zu Windows 95 kommt es mit einem 'Bonuspack', der nahezu alles für einfache Anwendungen enthält. Das 'Works-Paket' enthält Textverarbeitung, Kalkulation, Datenbank, Business-Grafik, Terminplaner, usw. Ein Terminalprogramm für die Datenkommunikation und Fax-Software sind ebenfalls dabei. Die Eigenschaften in Kurzform:

- preemptive Multitasking
- DOS und Windows laufen unter OS/2 (nicht im Paket enthalten)
- 32-bit-Operationen
- shared Libraries
- grafische Benutzeroberfläche (WPS)
- Netzwerkintegration (TCP/IP, LAN Server)

## Betriebssysteme

- Ordner (beliebige Verschachtelungsebenen von Applikationen)
- Links und Verknüpfungen (Daten <--> Programm)
- 'Plug & Play' (ähnlich Windows 95)
- Work Place Shell (WPS) mit mächtiger Scriptsprache
- Klickstartleiste (entspr. Taskbar von Windows 95) mit Untergruppen
- lange Dateinamen, effizientes Dateisystem (HPFS)
- Konfigurationsdatenbank
- Treiber für "exotische" Hardware fehlen manchmal

Durch die Unterstützung von DOS und Windows zeigt sich OS/2 weniger gut strukturiert. Neu gegenüber den anderen Systemen sind sogenannte 'Sessions' - OS/2-Module, die jeweils über eigene virtuelle Ressourcen verfügen (Tastatur, Maus, Bildschirm, Drucker, etc.). Auf die gleiche Weise werden DOS- und Windows-Applikationen abgeschottet; diese laufen in sogenannten 'Virtual DOS Machines' (VDMs), so daß sie andere Applikationen nicht beeinflussen können.

Die Verbreitung von OS/2 krankte von Anfang an daran, daß IBM das Produkt mal pushte und mal unter den Teppich kehrte. Ein zweiter Grund für die mangelnde Akzeptanz lag auch in der neuen und von Windows abweichenden Benutzeroberfläche und darin, daß nicht alle Windows-Software auf OS/2 lief.

## 7.7 Unix (und Derivate)

Unix ist eines der ältesten Betriebssysteme (gerade 25 Jahre alt geworden). Es ist in vielen Eigenschaften beispielgebend für andere Systeme gewesen (Auch DOS, Windows und OS/2 haben bei UNIX "abgeschaut"). Inzwischen ist es für nahezu jede Hardwareplattform verfügbar. Eigentlich muss hier von einer Betriebssystemfamilie gesprochen werden, denn praktisch jeder Workstationhersteller liefert sein eigenes Unix aus, das sich zumindest in der Benutzerschnittstelle sehr unterscheidet. Es gibt hier allerdings eine Tendenz, die Vielfalt an Oberflächen zu überwinden, da einzelne Hersteller angefangen haben, ihr System auf Fremdarchitekturen zu portieren. Die Unix-Implementationen lassen sich in zwei Standards zusammenfassen: Berkeley Unix (BSD) sowie AT&T's System V Release 4 (SVR4). Letzteres ist momentan dabei, den Vorrang zu gewinnen. Neu entstehende Unix Versionen folgen diesem Standard. Im allgemeinen gilt: ist ein Programm für einen der beiden Standards geschrieben, so lässt es sich ohne allzugrosse Probleme auf ein anderes System des gleichen Standards portieren. Auch bei den verwendeten Benutzeroberflächen (GUI - Graphical User Interface) gibt es unterschiedliche Standards. Die neueren folgen aber alle der X11 Definition. Seit einigen Jahren klar auf dem Vormarsch ist die - ebenfalls auf X11 basierende - MOTIF Definition. Mehr und mehr Unix Implementationen bedienen sich dieser Oberfläche, während Konkurrenten wie OPENLOOK eher rückläufig sind.

Die wichtigsten Eigenschaften in Kürze:

- UNIX ist ein portables, einfach aufgebautes Betriebssystem
  - ◆ Multitasking-BS (Multiprocessing-BS)
  - ◆ Multiuser-BS (Mehrbenutzer-BS)
  - ◆ dialogorientiert
- UNIX ist ein Werkzeugkasten
  - ◆ viele hundert Dienstprogramme
  - ◆ flexibel: kleine Tools sind schnell erstellt
- UNIX ist geeignet für Mikrocomputer der Oberklasse, Mini-Computer, Großrechner
- UNIX ist schrecklich
  - ◆ Befehlsname sind kryptisch (ls, pwd, cat, awk, grep, ..)
  - ◆ Motto 1: "Keine Nachricht ist eine gute Nachricht!"
  - ◆ Motto 2: "Wer will schon schlechte Nachrichten?"

## Betriebssysteme

- mit grafischer Oberfläche (X Window) bedienbar wie Windows
- UNIX ist in Schichten strukturiert
  - ◆ Shell (Kommandointerpreter) mit mächtiger Scriptsprache
  - ◆ Kern
  - ◆ Treiber
- Aufgaben des Kerns
  - ◆ Prozeß-Scheduling
  - ◆ Prozeß-Umschaltung
  - ◆ Prozeß-Kommunikation
  - ◆ Dateisystem verwalten
  - ◆ Ein-/Ausgabesteuerung
  - ◆ Gerätsteuerung (device driver)
  - ◆ Zugangskontrolle und Abrechnung
  - ◆ alle Systemdienste für Programmier-Schnittstellen
- Das Dateisystem ist hierarchisch strukturiert
  - ◆ lange Dateinamen
  - ◆ Normale Dateien (normal files)
  - ◆ Verzeichnisse (directories)
  - ◆ Spezialdateien (special files) = Geräteschnittstelle
  - ◆ Named Pipes
  - ◆ Links
  - ◆ Jede Datei besitzt 12 voneinander unabhängige Schutzbits

Netzwerkbetrieb über TCP/IP ist bei UNIX von Anfang an möglich gewesen. Die meisten Netzwerkdienste anderer Systeme basieren auf den UNIX-Diensten. Insbesondere die 'Internet-Connectivity', die jetzt in aller Munde ist, basiert auf UNIX. So ist es z. B. auch möglich, Platten anderer Rechner über das Netz einzubinden.

Mehr als 90% des Codes ist in C programmiert --> portabel. Für unzählige Applikationen ist auch Quellcode erhältlich (für eigene Anpassungen).

## 7.8 Linux

Linux ist ein frei verfügbares Multitasking- und Multiuser-Betriebssystem auf UNIX-Basis für Systeme mit Intel-Prozessoren. Erfunden wurde Linux von Linus Torvalds und weiterentwickelt von einer Vielzahl von Entwicklern in aller Welt. Linux wurde von Anfang an unter die GPL, der General Public License gestellt. Es kann frei und kostenlos verteilt, eingesetzt und erweitert werden. Entwickler haben so Einblick in sämtliche Quellcodes und können dadurch sehr einfach neue Funktionen integrieren bzw. Programmierfehler schnell finden und eliminieren. Treiber für neue Adapter (SCSI Controller, Grafikkarten etc.) können dadurch sehr schnell integriert werden. Inzwischen hat Linux mit vergleichbaren UNIX-Implementierungen gleichgezogen - oft ist es sogar robuster und stabiler als kommerzielle Produkte.

Linux kann auf zwei verschiedene Arten bezogen werden: Alle benötigten Teile können kostenlos aus dem Internet geladen werden. Einfacher ist der Einsatz einer sogenannten Distribution, diese werden von verschiedenen Firmen angeboten und enthalten neben einer Vielzahl von Anwendungen ein Installationsprogramm, welches die Installation von Linux wesentlich vereinfacht. Zu empfehlen sind die Distributionen von RedHat und S.u.S.E.

Linux wird mittlerweile von mehreren Millionen Anwendern weltweit erfolgreich eingesetzt. Die Benutzergruppen reichen von privaten Anwendern über Schulungsfirmen, Universitäten, Forschungszentren bis hin zu kommerziellen Anwendern und Firmen, die in Linux eine echte Alternative zu anderen Betriebssystemen sehen.

## 7.9 OS/9

OS/9 ist ein Echtzeitbetriebssystem, das schon vor 15 Jahren auf dem Motorola-Prozessor 6809 implementiert und später auf die 68000er-Prozessorfamilie portiert wurde. Um Echtzeitbedingungen zu genügen, unterscheidet sich sein Aufbau von anderen Betriebssystemen.

- alle Programme liegen als sogenannte 'Module' vor
- alle Module haben die gleiche Struktur und sind frei im Speicher verschieblich
- OS/9 kann zwischen RAM
- und ROM-Bereich unterscheiden
- beim Systemstart werden Module im ROM selbständig gefunden
- es werden möglichst viele Module im Speicher gehalten; deshalb können OS/9-Systeme ohne Massenspeicher arbeiten
- Das Betriebssystem besteht aus vier Teilen:
- Kernel (nur 26 KByte) mit preemptive Multitasking, Basisfunktionen für E/A, Ressource-, Prozeß und Benutzerverwaltung
- Filemanager für Random-Block-Geräte (Platte), Sequential-Character-Geräte, Pipes u. a.
- Treiber für E/A-Geräte
- Deskriptoren für BS, Netzwerk und Geräte
- Anpassung des Kerns an unterschiedliche Hardware über Initialisierungsmodule
- Zeitscheibenverfahren (10 ms) mit Prioritätenvergabe
- Prioritäten werden dynamisch vergeben (Prozeß bekommt z. B. höhere Priorität bei Auftreten eines für ihn bestimmten Interrupts (schnelle Reaktion!))
- Zugriff auf die Hardware nur über den Kernel

Prozeß <---> BS-Kern <---> Dateimanager <---> Treiber

- Shared Libraries (ähnlich DLL bei Windows), die über sogenannte Traphandler angesprochen werden
- Prozeßkommunikation über Signale, Events (=Semaphore) und Daten-Module
- hierarchisches Dateisystem
- Auf Wunsch graphische Benutzeroberfläche (X-Window, MGR Window Manager, OS-9-Windows)



[Zum vorhergehenden Abschnitt](#)



[Zum Inhaltsverzeichnis](#)



[Zum nächsten Abschnitt](#)

---

Copyright © FH München, FB 04, Prof. Jürgen Plate

## Einführung in Betriebssysteme



von Prof. Jürgen Plate

---

# Literatur

Andrew S. Tanenbaum:  
**"Moderne Betriebssysteme"**  
Verlag Prentice Hall

Bic/Shaw:  
**"Betriebssysteme"**  
Hanser Verlag

Peter Monadjemi:  
**"Windows 98"**  
Verlag Markt & Technik

Dapper/Dietrich/Klöppel:  
**"Windows NT 4.0 (2 Bände)"**  
Hanser Verlag

Michael Kofler:  
**"Linux"**  
Verlag Addison Wesley

William Boswell:  
**"Windows 2000"**  
Verlag Markt & Technik

Levine, Levine und Young:  
**"Windows XP"**  
Verlag Markt & Technik

Handbücher der diversen Betriebssystem-Hersteller.



[Zum vorhergehenden Abschnitt](#)



[Zum Inhaltsverzeichnis](#)



[Zum nächsten Abschnitt](#)

---

Copyright © FH München, FB 04, Prof. Jürgen Plate