

# Betriebssysteme

## Theorie der Betriebssysteme

Prof. Dr. Ingolf Brunner

# Gliederung

- 1 Einführung
- 2 Prozesse und Threads
- 3 Speicherverwaltung
- 4 Dateisysteme
- 5 Eingabe und Ausgabe
- 6 Deadlocks
- 7 Virtualisierung und die Cloud
- 8 Multiprozessorsysteme
- 9 IT-Sicherheit
- 10 Fallstudie 1: Linux
- 11 Fallstudie 2: Windows
- 12 Entwurf von Betriebssystemen

# Literatur

- Tanenbaum, A. S.; Bos, H.: Moderne Betriebssysteme. Pearson Studium 2016.
- Glatz, E.: Betriebssysteme. dpunkt Verlag.
- Stallings, W.: Betriebssysteme, Pearson Studium.
- Ehses, E.; Köhler, L.; Riemer, P.; Stenzel, H.; Victor, F.: Betriebssysteme – Ein Lehrbuch mit Übungen zur Systemprogrammierung in UNIX/Linux. Pearson Studium.
- Stallings, W.: Betriebssysteme – Prinzipien und Umsetzung, Pearson Studium.

# 1 Einführung

## ***1 Einführung***

- 2 Prozesse und Threads
- 3 Speicherverwaltung
- 4 Dateisysteme
- 5 Eingabe und Ausgabe
- 6 Deadlocks
- 7 Virtualisierung und die Cloud
- 8 Multiprozessorsysteme
- 9 IT-Sicherheit
- 10 Fallstudie 1: Linux
- 11 Fallstudie 2: Windows
- 12 Entwurf von Betriebssystemen

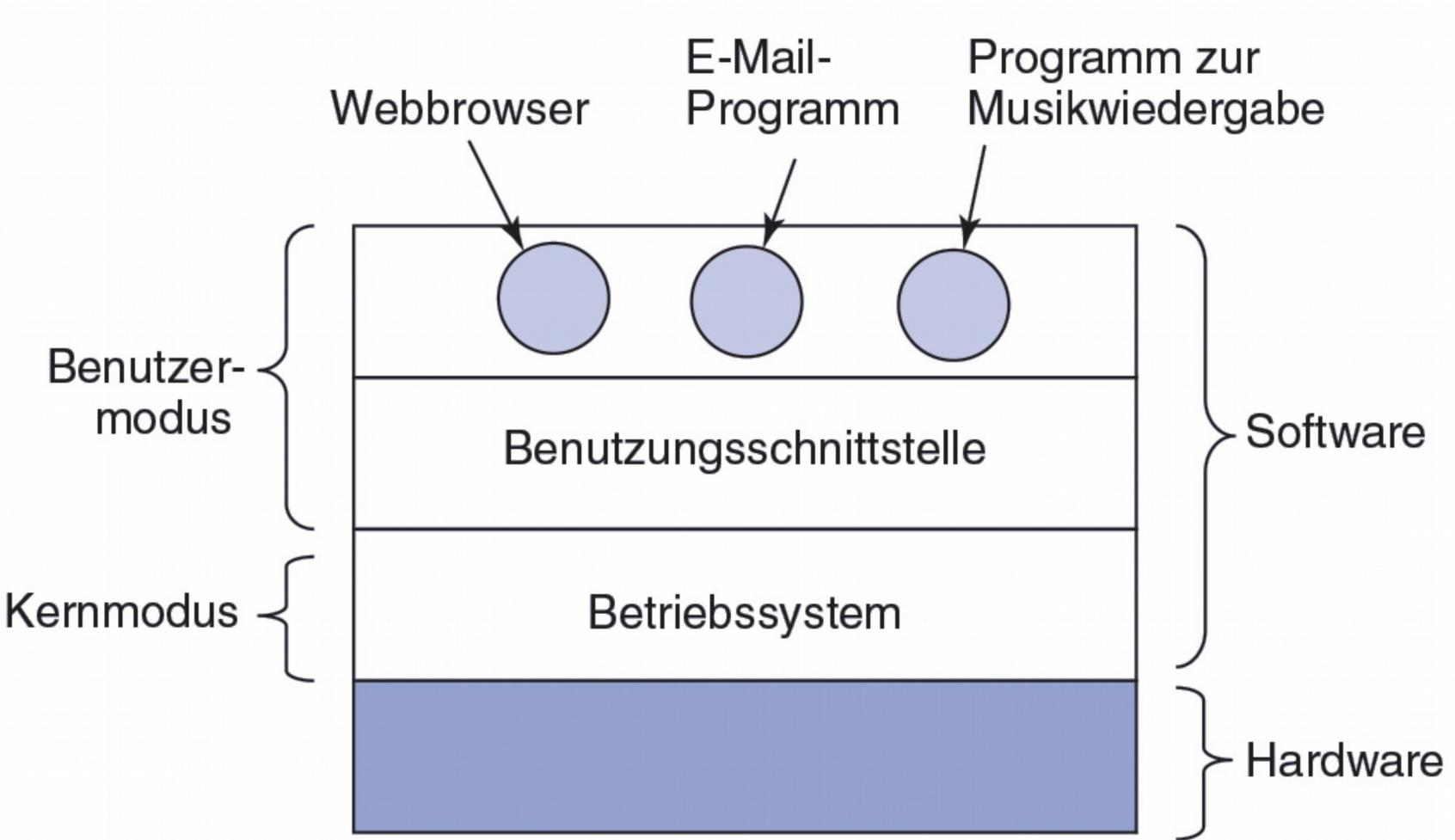
# 1 Einführung

- 1.1 Was ist ein Betriebssystem?
- 1.2 Geschichte der Betriebssysteme
- 1.3 Überblick über die Computerhardware
- 1.4 Die Betriebssystemfamilie
- 1.5 Betriebssystemkonzepte
- 1.6 Systemaufrufe
- 1.7 Betriebssystemstrukturen
- 1.8 Die Welt aus der Sicht von C

# Komponenten eines modernen Computersystems (1)

- Ein oder mehrere Prozessoren
- Hauptspeicher
- Disks
- Drucker
- Tastatur
- Maus
- Monitor
- Netzwerkschnittstellen
- I/O Geräte

# Komponenten eines modernen Computersystems (2)

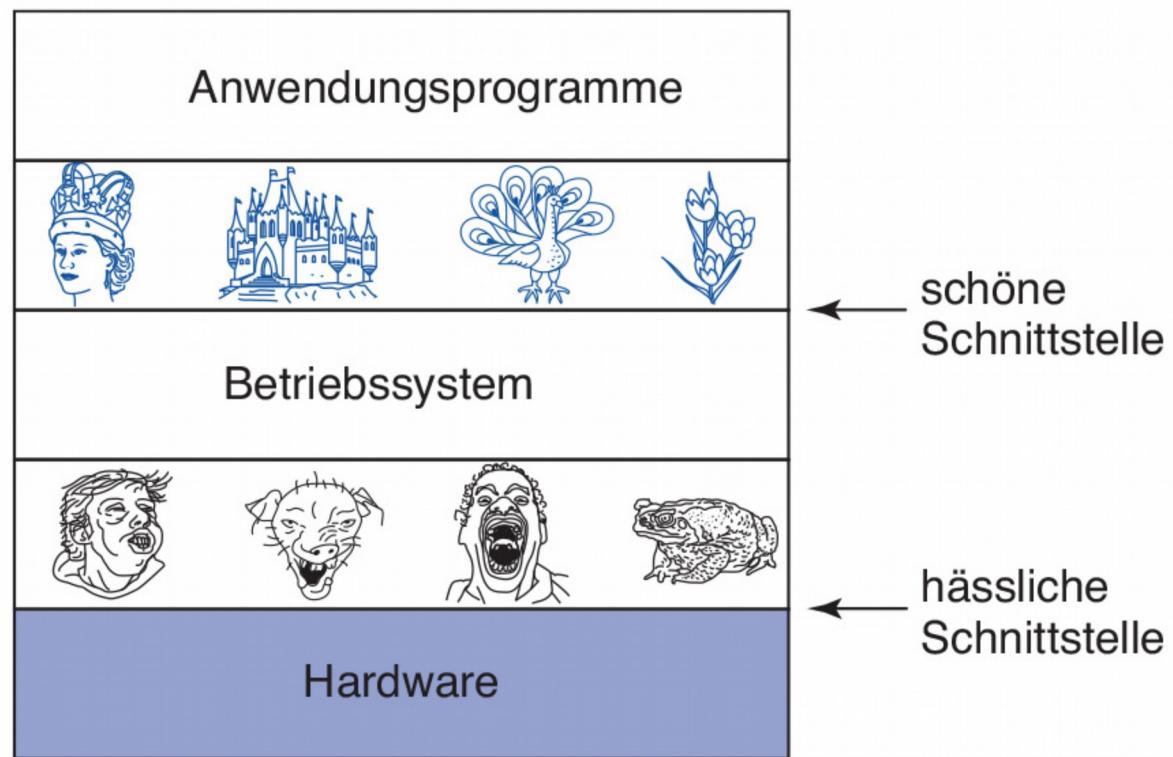


**Abbildung 1.1:** Die Einordnung des Betriebssystems.

# Was ist ein Betriebssystem?

- 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.

# Das Betriebssystem als erweiterte Maschine



**Abbildung 1.2:** Betriebssysteme verwandeln die hässliche Hardware in wunderschöne Abstraktionen.

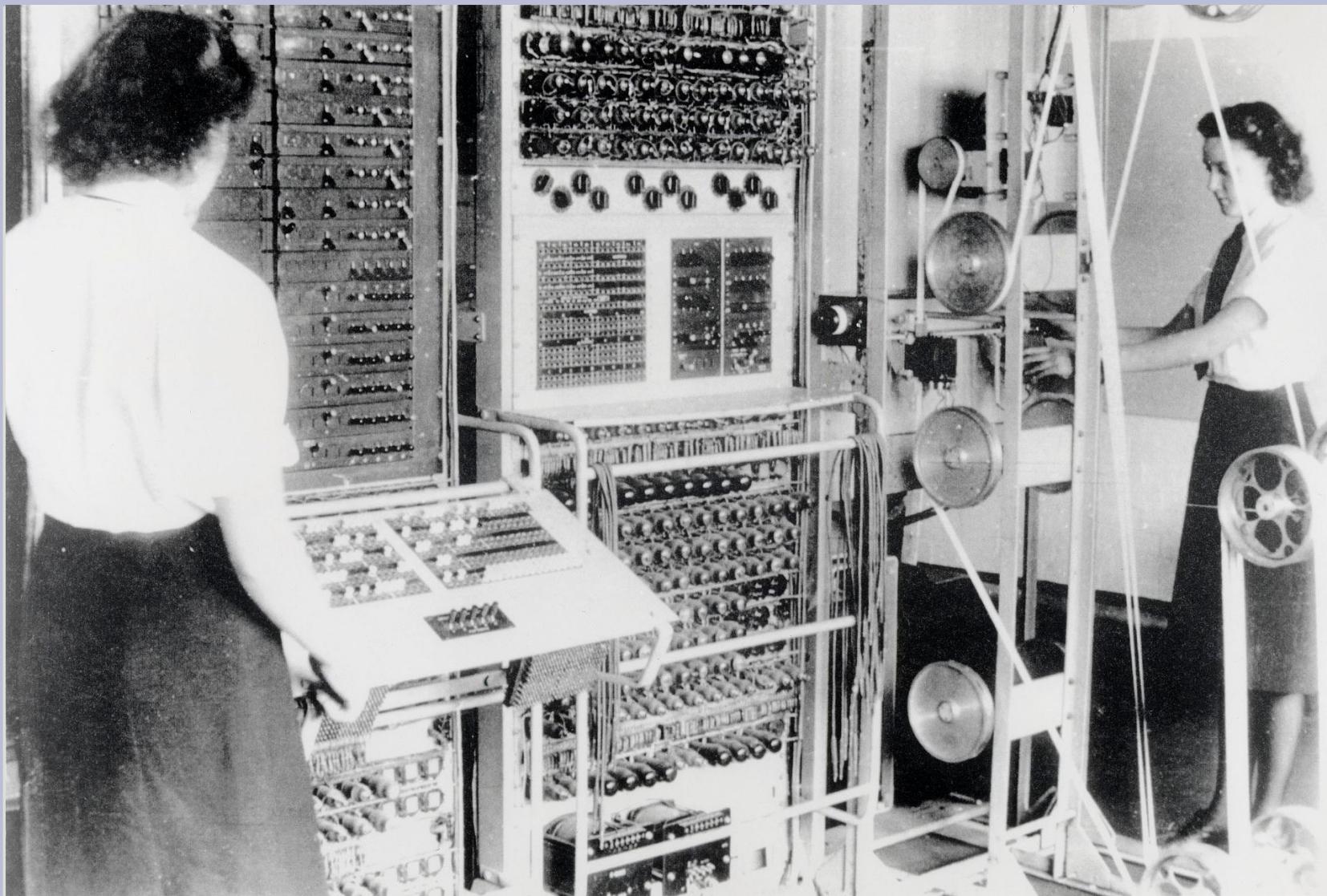
# Das Betriebssystem als Ressourcenverwalter

- Top down View
  - Stellt den Anwendungsprogrammen Abstraktionen der Hardware bereit
- Bottom up View
  - Verwaltet die Teile eines komplexen Systems
- Alternative Sicht
  - Stellt die geordnete, kontrollierte Zuweisung von Ressourcen bereit

# Geschichte der Betriebssysteme

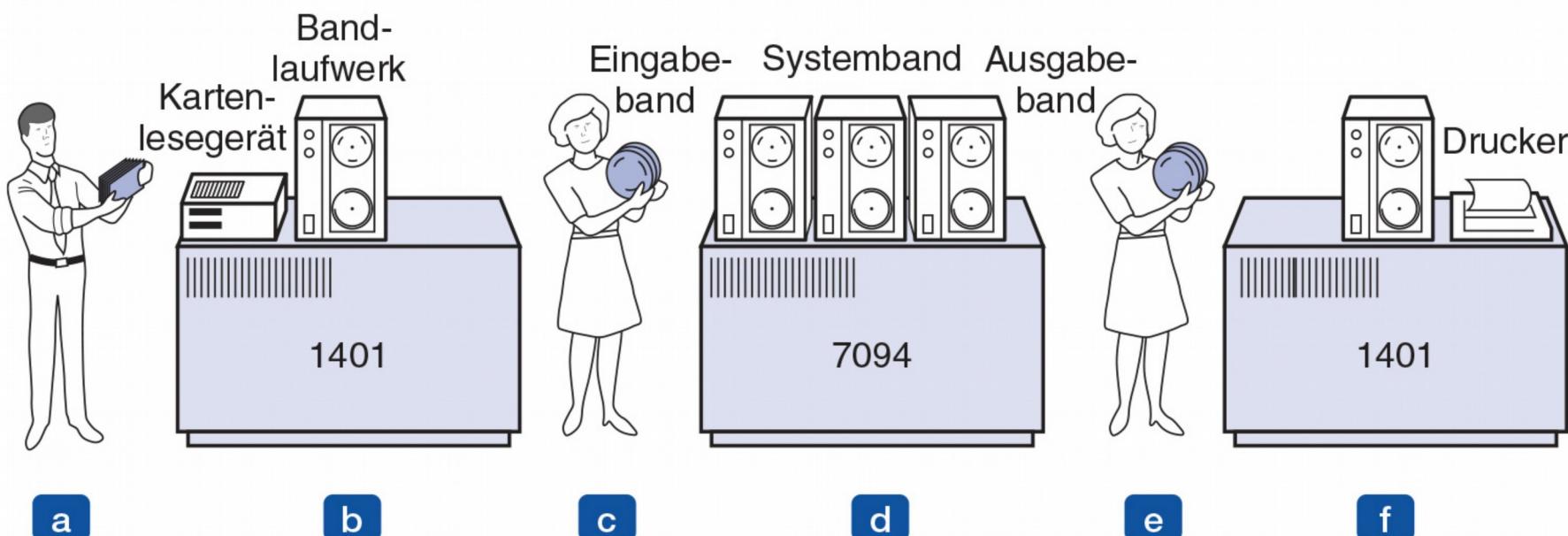
- Die erste Generation (1945 – 1955) – auf der Basis von Elektronenröhren
- 1.2.2 Die zweite Generation (1955 – 1965) – Transistoren und Stapelverarbeitungssysteme
- 1.2.3 Die dritte Generation (1965 – 1980) – integrierte Schaltkreise und Multiprogrammierung
- 1.2.4 Die vierte Generation (1980 – heute) – der PC
- 1.2.5 Die fünfte Generation (1990 – heute) – mobile Computer

# Colossus



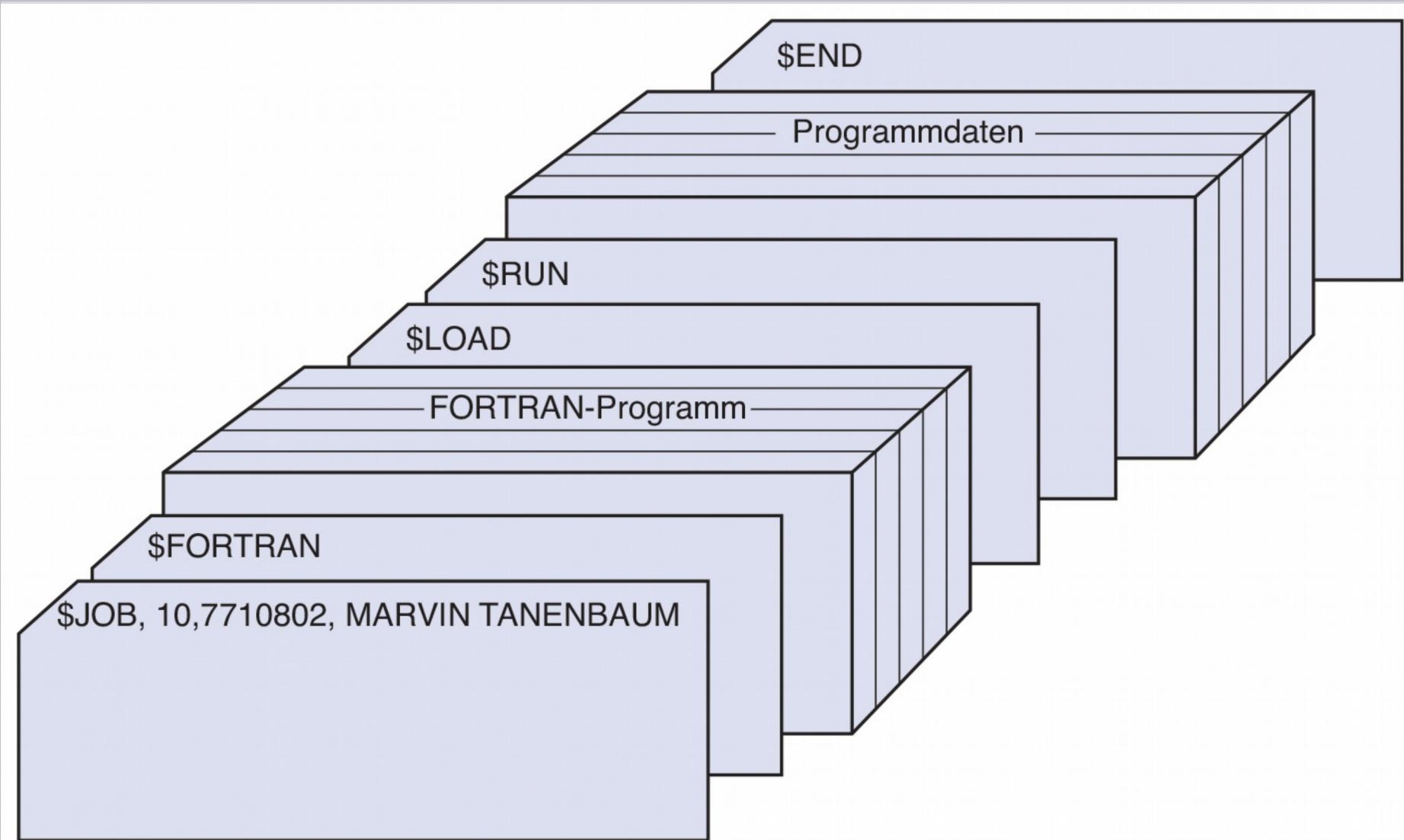
Public domain, via Wikimedia Commons

# Transistoren und Stapelverarbeitungssysteme (1)



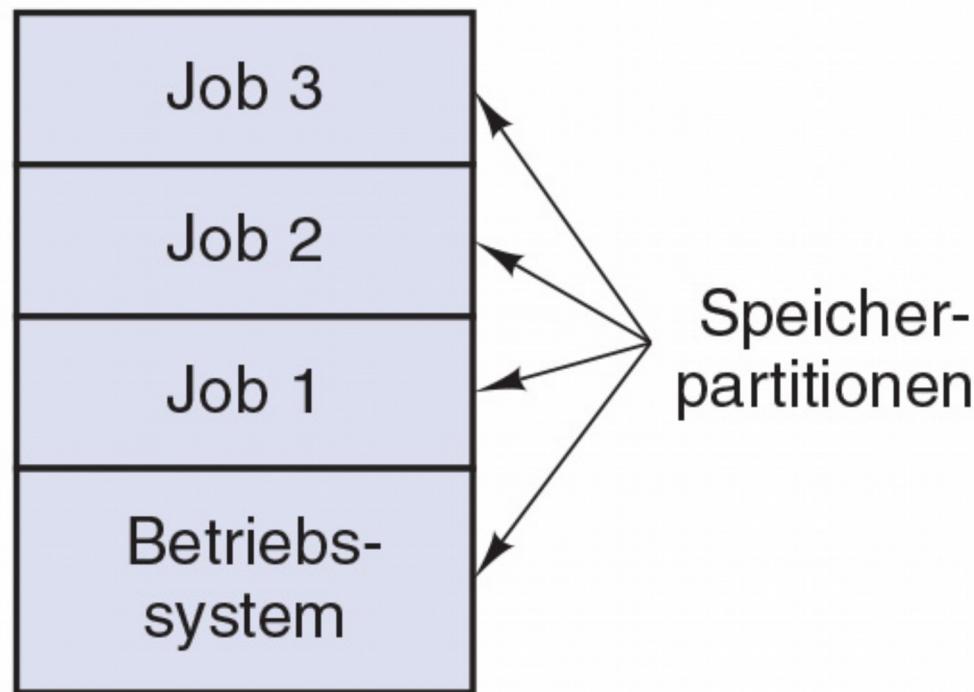
**Abbildung 1.3:** Ein frühes Stapelverarbeitungssystem. (a) Die Programmierer bringen die Stapel zur 1401. (b) Die 1401 liest den Stapel von Jobs auf ein Band. (c) Ein Operator trägt das Eingabeband zur 7094. (d) Die 7094 führt die Berechnung durch. (e) Ein Operator trägt das Ausgabeband zur 1401. (f) Die 1401 druckt die Ausgabe.

# Transistoren und Stapelverarbeitungssysteme (2)



**Abbildung 1.4:** Struktur eines typischen FMS-Jobs.

# Integrierte Schaltkreise und Multiprogrammierung



**Abbildung 1.5:** Ein Multiprogrammiersystem mit drei Jobs im Speicher

# Der PC



**1981: IBM 5150 PC**

- 64kB RAM
- 4,77 MHz
- MS-DOS

# Mobile Geräte

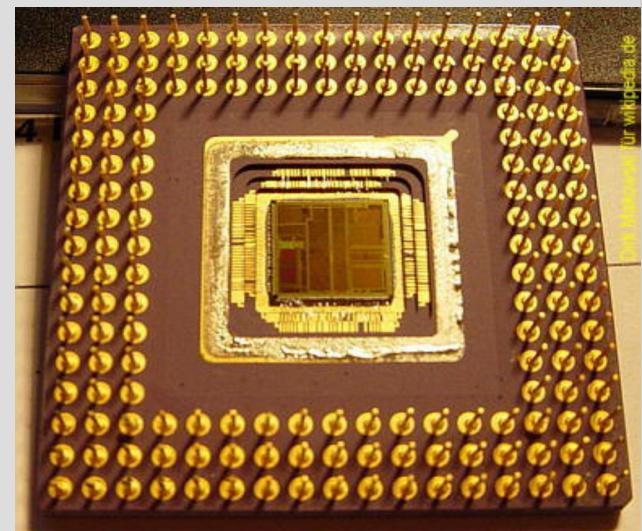
- Mobile Geräte:
  - Personal Digital Assistants (PDAs)
  - Mobiltelefone (z. B. Smartphones)
  - Tablet-Computer
- Im erweiterten Sinn:
  - Notebooks und Subnotebooks
- Außerdem:
  - Walk- und Discmans, MP3-Player
  - Taschenfernseher (tragbare Fernsehgeräte)
  - E-Book-Lesegeräte u.a. tragbare Ausgabegeräte für elektronische Medien.
  - GPS-Geräte u.a. tragbare Schnittstellengeräte der Satellitenkommunikation.

# Prozessoren (1)

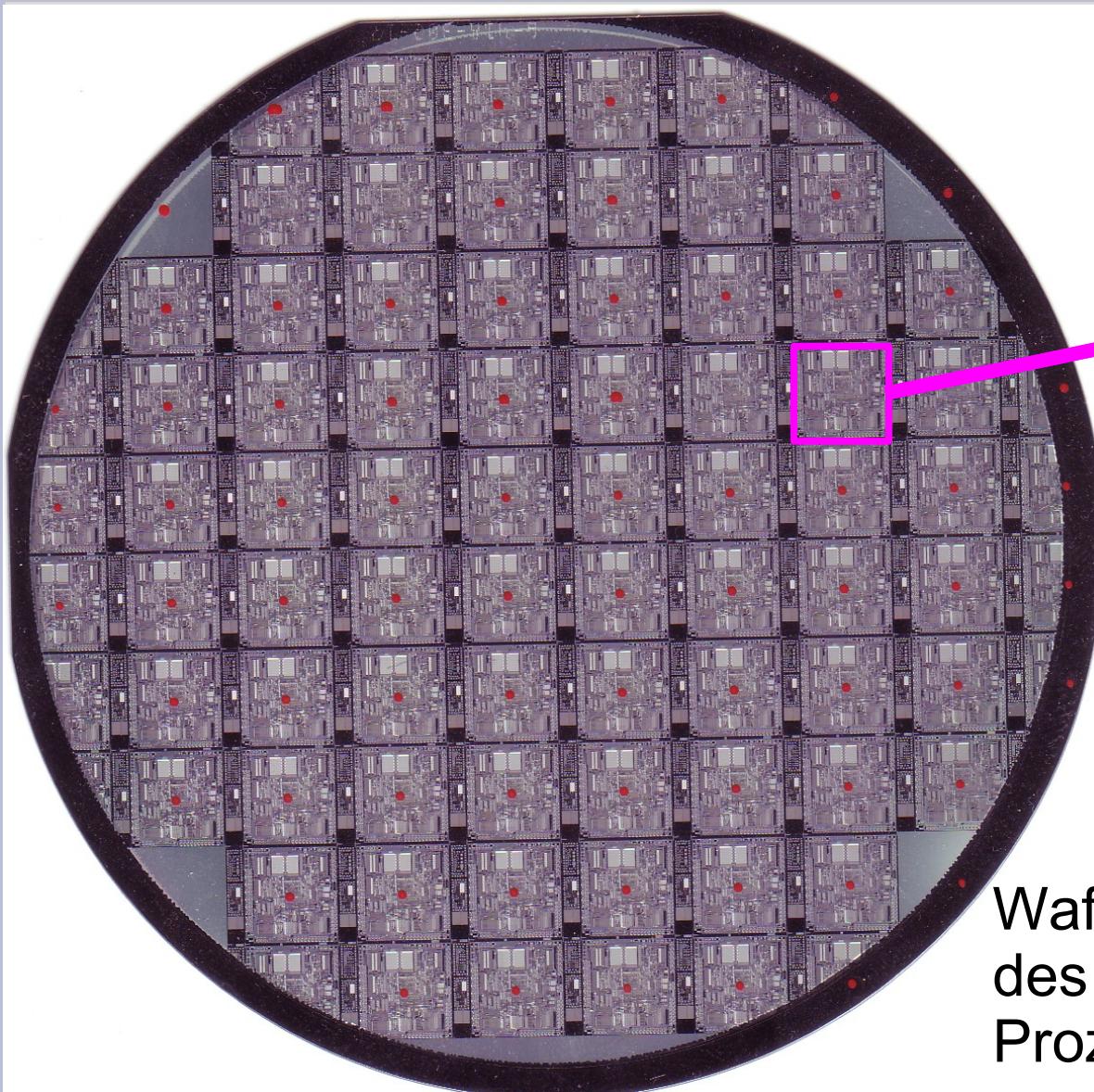
- Central Processing Unit
  - Strukturen 65 nm bis 22 nm klein (Lichtwellenlänge: ca. 400 nm bis 700 nm)
  - Anzahl Transistoren: bis 5 Milliarden 62-Core Xeon Phi
  - Hauptprobleme
    - Design: „Speed Path“  Signallaufzeit
    - Betrieb:

Abwärme	Pentium IV:	100 W / cm <sup>2</sup>
Hüllrohr	Kernbrennstab:	65 W / cm <sup>2</sup>
Elektrische	Kochplatte:	10 W / cm <sup>2</sup>

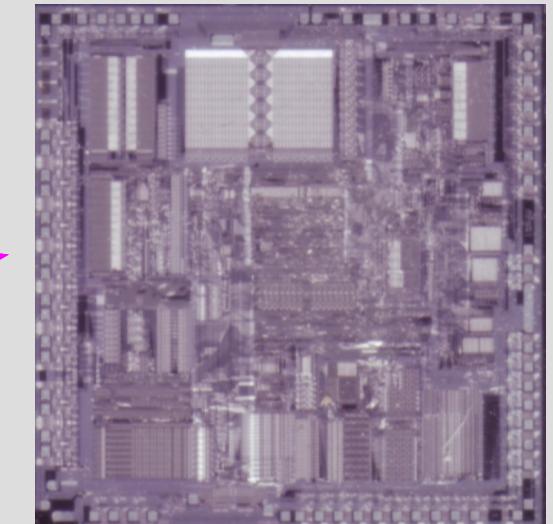
AMD 486 DX2 66



# Prozessoren (1)

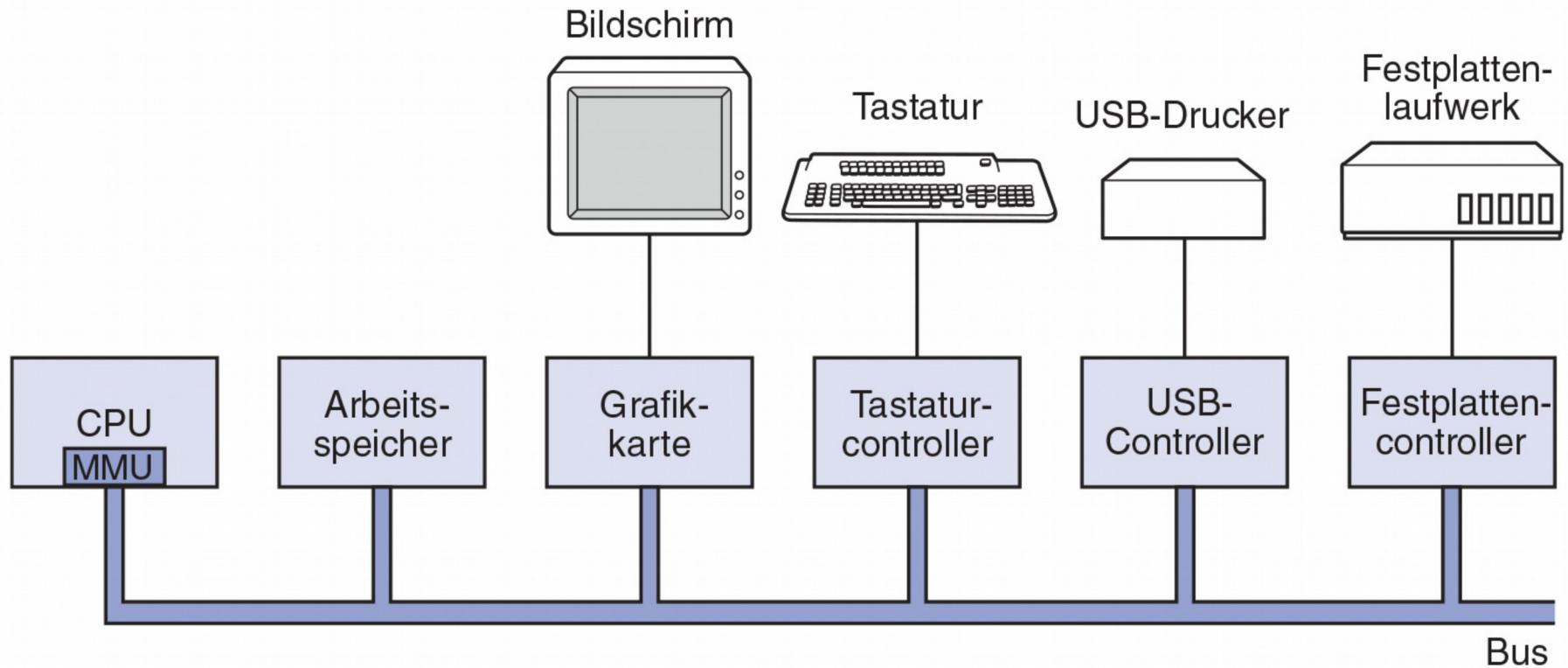


Wafer mit Nachbauten  
des Intel 80286  
Prozessors



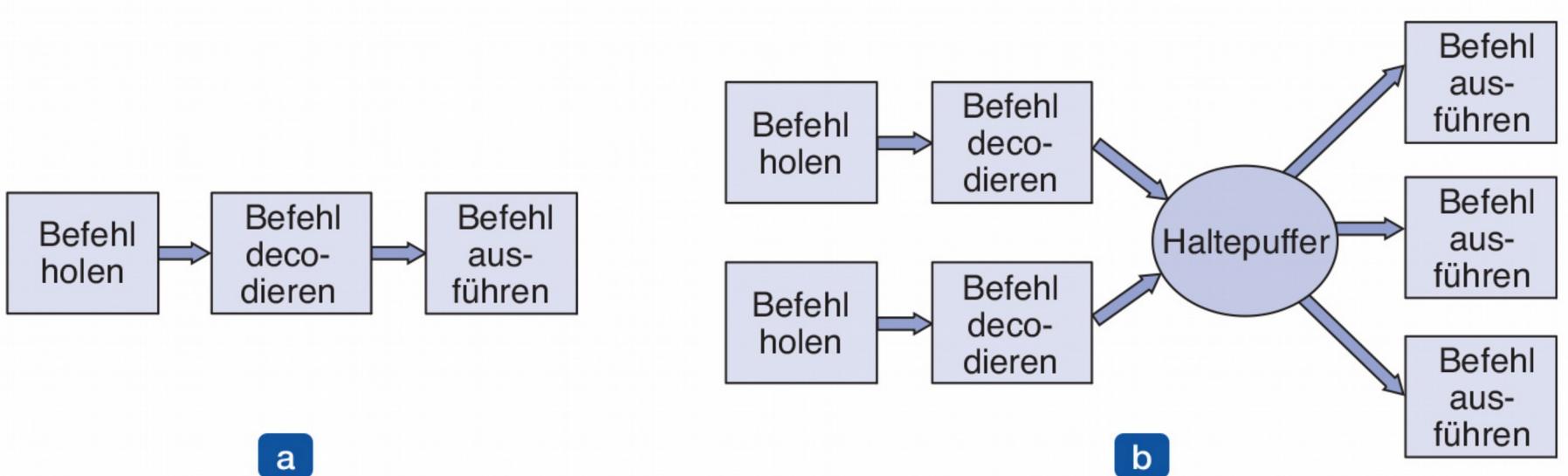
einzelner Prozessor  
(nach dem Heraus-  
trennen Die genannt)

# Prozessoren (2)



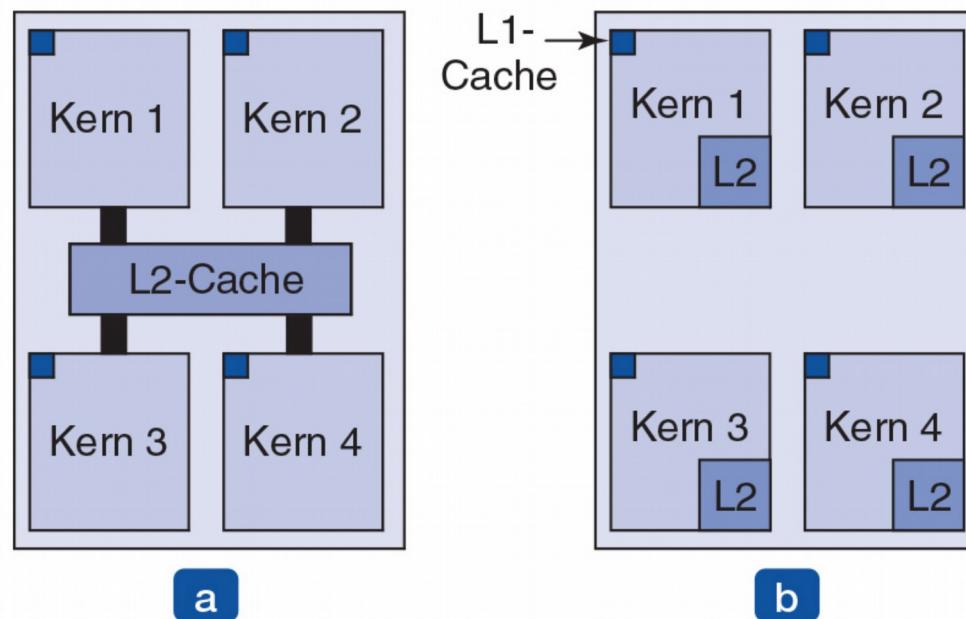
**Abbildung 1.6:** Einige Komponenten eines einfachen PCs.

# Prozessoren (3)



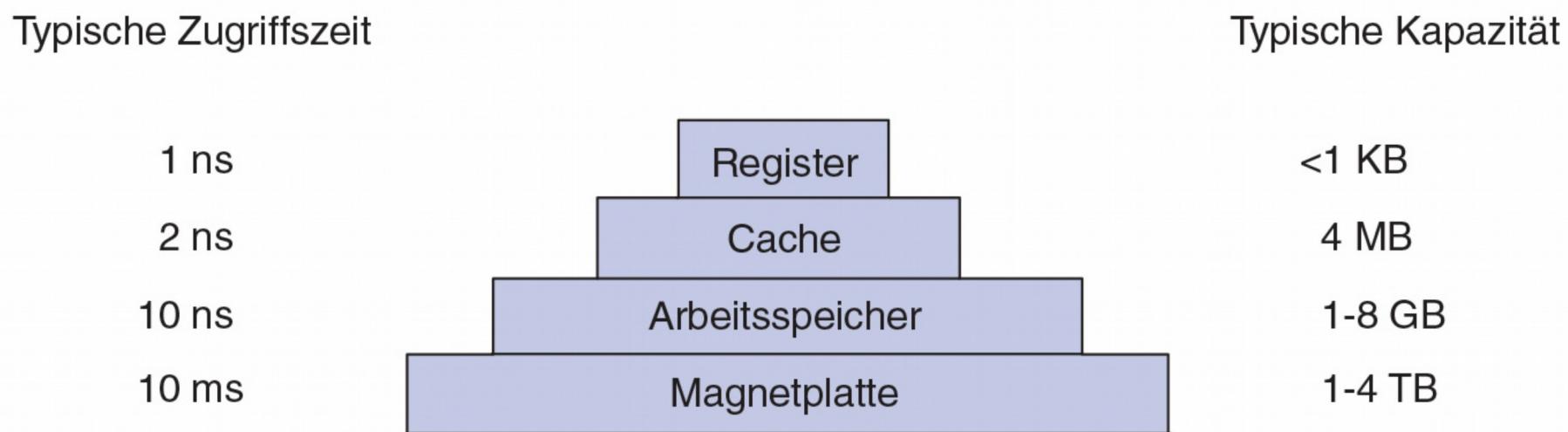
**Abbildung 1.7:** (a) Eine dreistufige Pipeline; (b) eine superskalare CPU.

# Arbeitsspeicher (1)



**Abbildung 1.8:** (a) Ein Vier-Kern-Chip mit einem gemeinsam benutzten L2-Cache; (b) ein Vier-Kern-Chip mit separaten L2-Caches.

# Arbeitsspeicher (2)

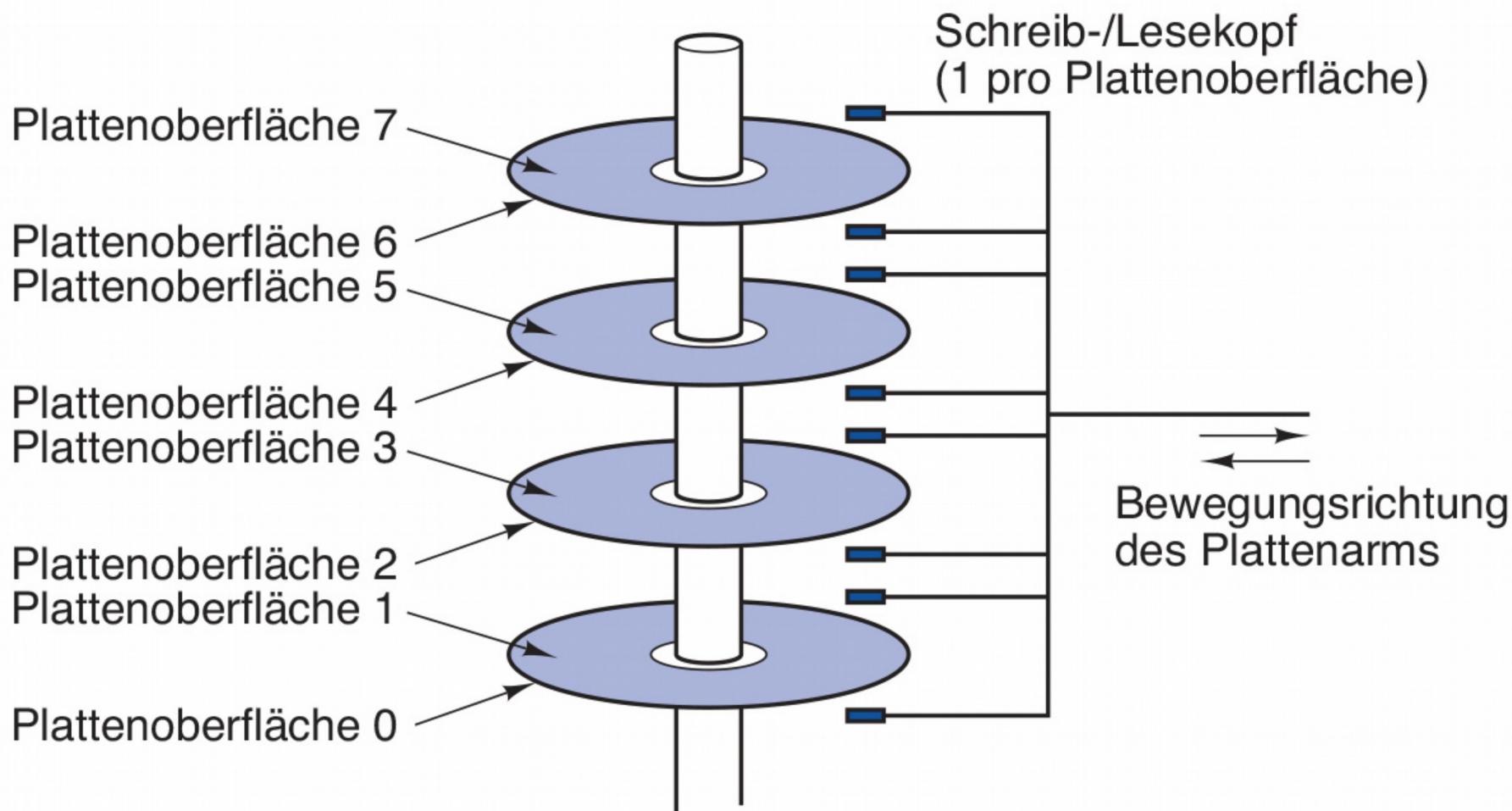


**Abbildung 1.9:** Eine typische Speicherhierarchie. Die Werte sind sehr grobe Annäherungen.

# Arbeitsspeicher (3)

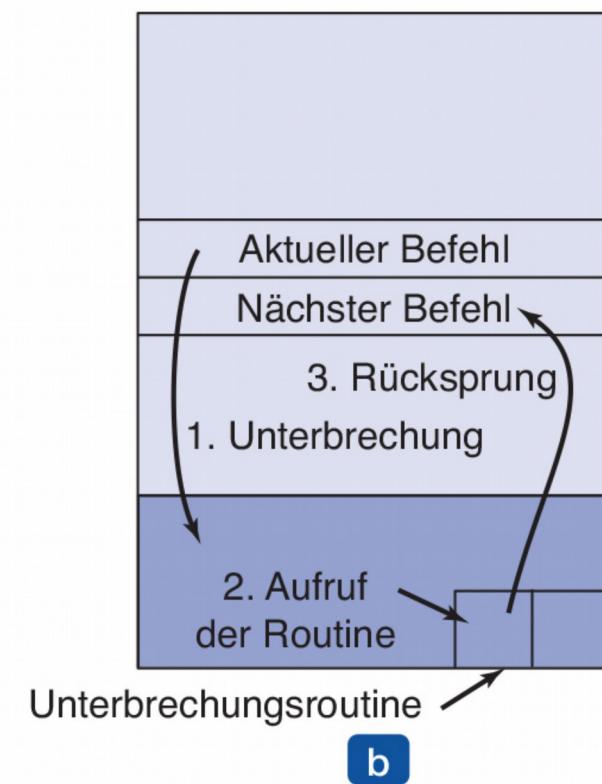
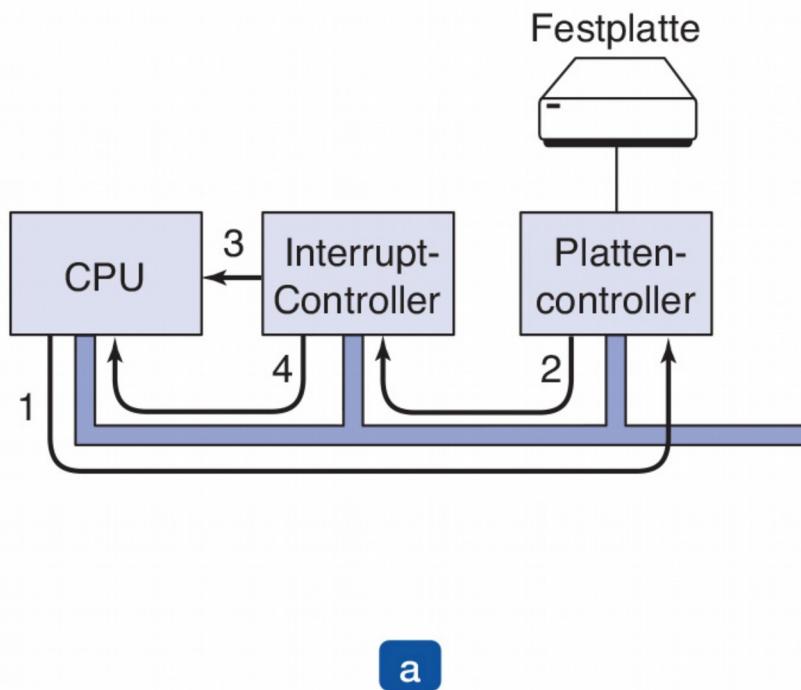
- Probleme mit Caches:
  - Wann soll man ein neues Objekt in den Cache einfügen
  - In welche Cache Line soll man das neue Objekt einfügen
  - Welches Objekt soll man aus dem Cache entfernen wenn es notwendig ist
  - Wo soll man eine neu entferntes Objekt in den Hauptspeicher einfügen

# Festplatten



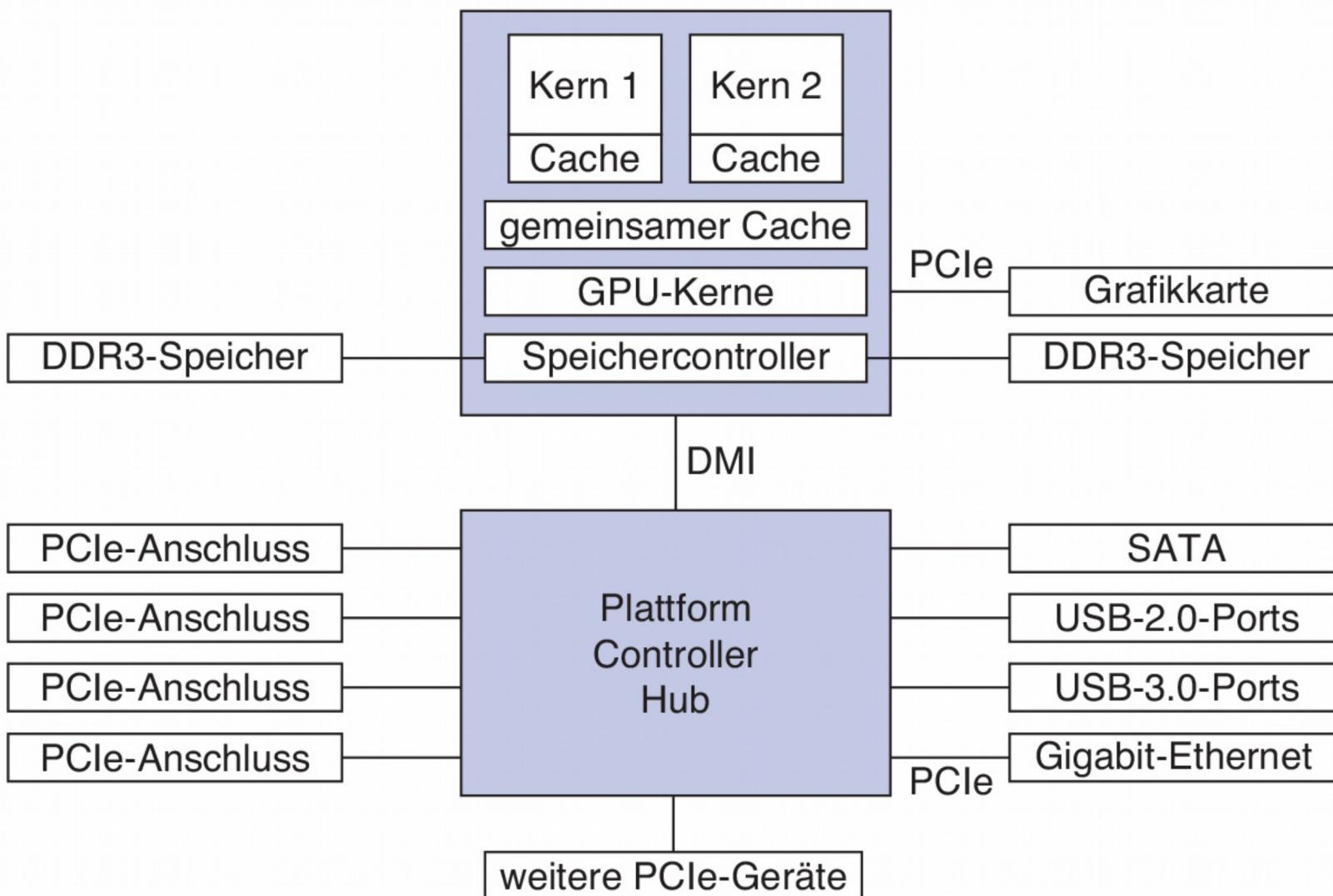
**Abbildung 1.10:** Struktur einer Festplatte.

# Ein-/Ausgabegeräte



**Abbildung 1.11:** (a) Die Schritte beim Starten eines Ein-/Ausgabegeräts und Erzeugen einer Unterbrechung. (b) Interruptbehandlung besteht aus dem Auffangen des Interruptsignals, dem Ausführen der Unterbrechungsroutine und dem Rücksprung zum Benutzerprogramm.

# Bussysteme



**Abbildung 1.12:** Der Aufbau eines ausgebauten x86-Systems.

# Hochfahren des Computers

Basic Input Output System (BIOS) / Unified Extensible Firmware Interface (EFI oder UEFI)

- POST (Power On Self Test)
- Konfiguration von Rechnerkomponenten (Plug & Play)
- Laden des OS
- Stellt OS Konfigurationsdaten zur Verfügung
- Grundlegende Programmschnittstellen zur Hardware

# Die Betriebssystemfamilie

- Betriebssysteme für Großrechner
- Betriebssysteme für Server
- Betriebssysteme Für Multiprozessorsysteme
- Betriebssysteme für PCs
- Betriebssysteme für Handheld-Computer
- Betriebssysteme für eingebettete Systeme
- Betriebssysteme für Sensorknoten
- Echtzeitbetriebssysteme
- Betriebssysteme für Smartcards

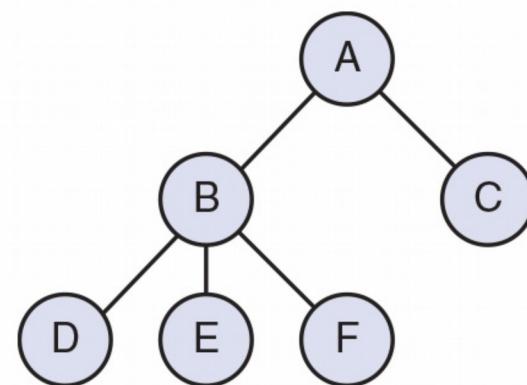
# Betriebssystemkonzepte

- Prozesse
- Adressräume
- Dateien
- Ein- und Ausgabe
- Datenschutz und Datensicherheit
- Die Shell
- Die Ontogenese rekapituliert die Phylogenie

# Prozesse (1)

- Schlüsselkonzept in allem Betriebssystemen
- Definition: ein ausgeführtes Programm
- Ein Prozess ist mit einem Adressraum verbunden
- Er ist ebenso mit einer Menge von Ressourcen verbunden
- Einen Prozess kann man sich als Container vorstellen:
  - Dieser hält alle Informationen vor, welche zur Ausführung des Programms notwendig sind

# Prozesse (2)



**Abbildung 1.13:** Ein Prozessbaum. *A* hat zwei Kindprozesse *B* und *C* erzeugt. Prozess *B* hat wiederum die Prozesse *D*, *E* und *F* erzeugt.

# Dateien (1)

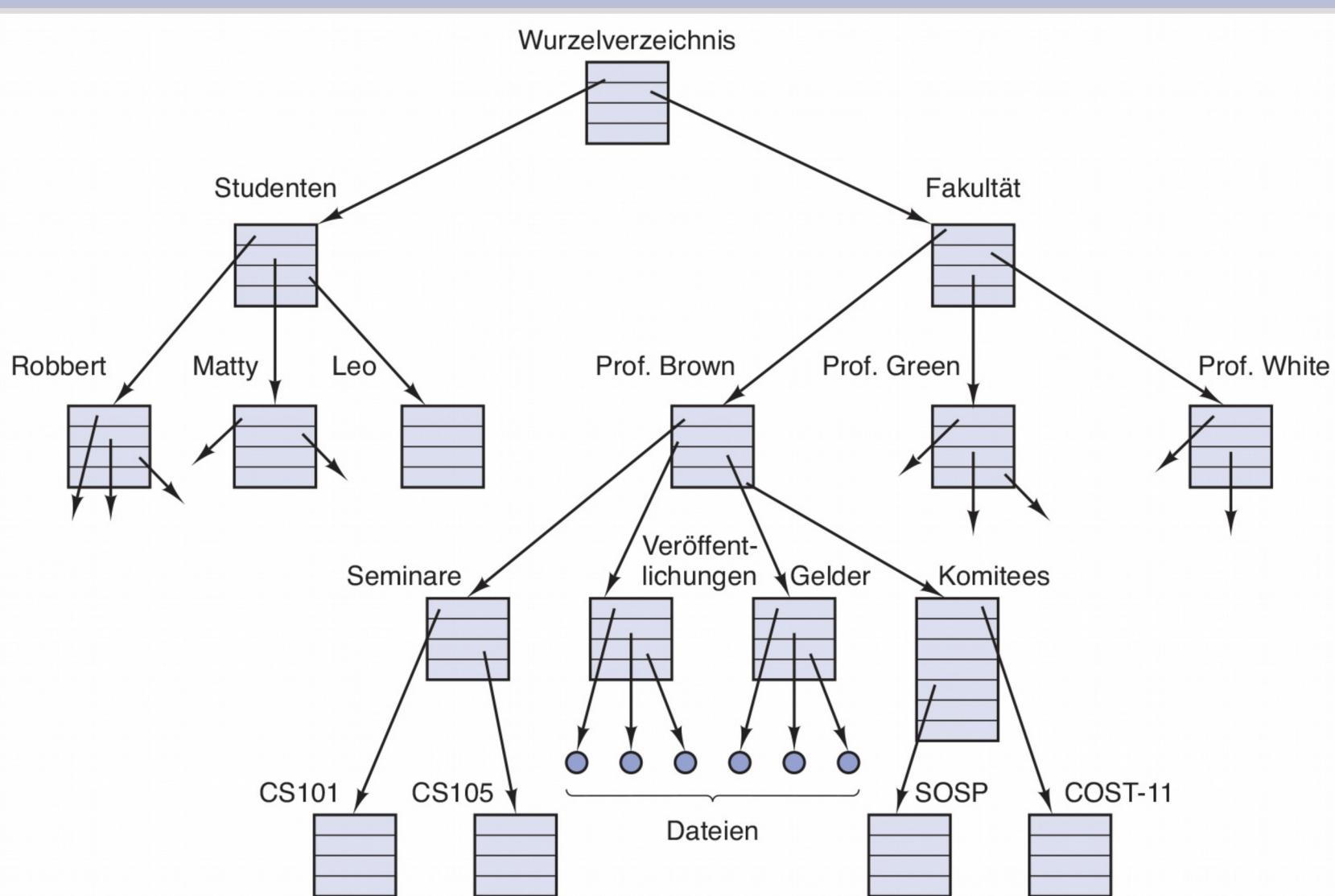
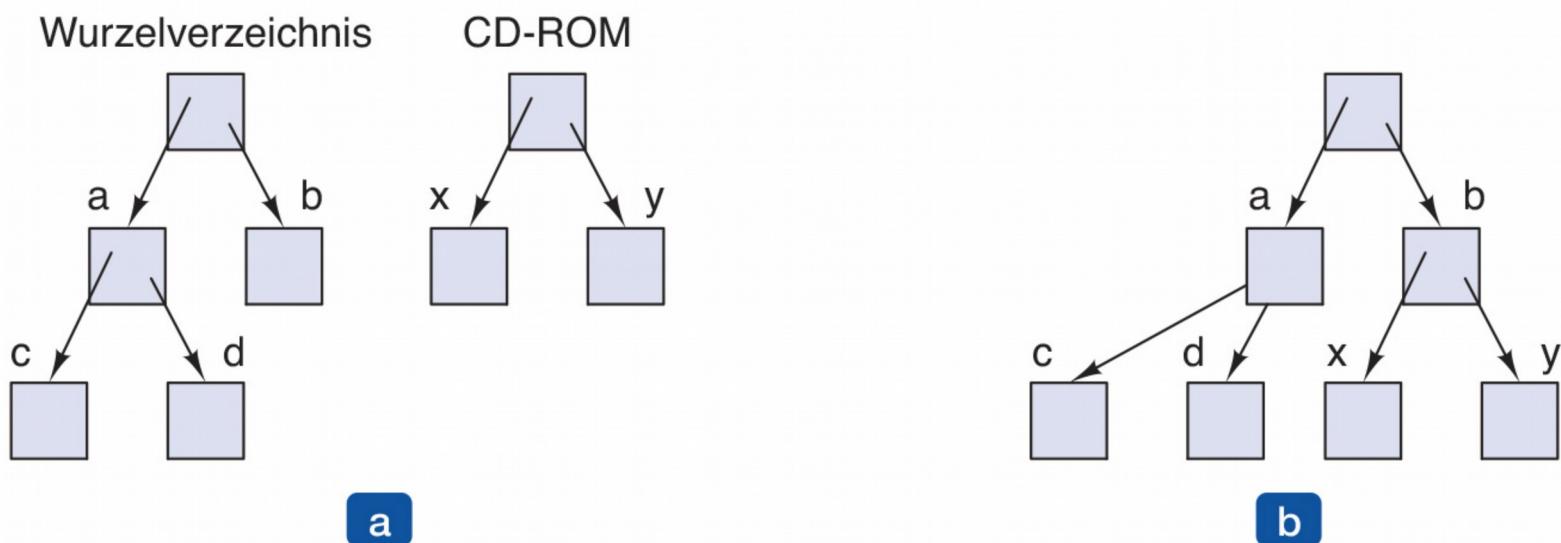


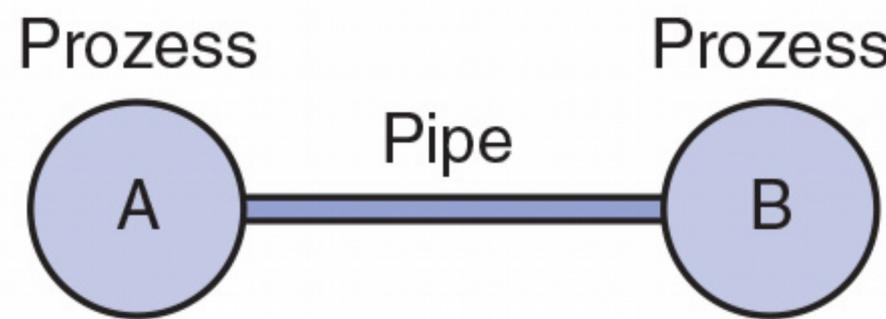
Abbildung 1.14: Das Dateisystem einer Fakultät in einer Universität.

# Dateien (2)



**Abbildung 1.15:** (a) Vor Ausführung des `mount`-Befehls konnte auf die Dateien der CD-ROM nicht zugegriffen werden. (b) Nach dem `mount`-Befehl sind sie Teil der Dateihierarchie.

# Dateien (3)



**Abbildung 1.16:** Zwei Prozesse, die durch eine Pipe verbunden sind.

# Die Ontogenese rekapituliert die Phylogenie

- Jede neue „Spezies“ von Computer
  - Durchläuft die gleiche Entwicklung wie die „Vorfahren“
- Konsequenzen der Unbeständigkeit
  - Der Text schaut häufig auf „veraltete“ Konzepte
  - Änderungen in der Technologie können diese zurückbringen
- Das passiert mit großem Speicher, Schutzhardware, Disks, Virtuellem Speicher

# Systemaufrufe (1)

- Systemaufrufe zur Prozessverwaltung
- Systemaufrufe zur Dateiverwaltung
- Systemaufrufe zur Verzeichnisverwaltung
- Sonstige Systemaufrufe
- Die Win32-Programmierschnittstelle (API) unter Windows 95

# Systemaufrufe (2)

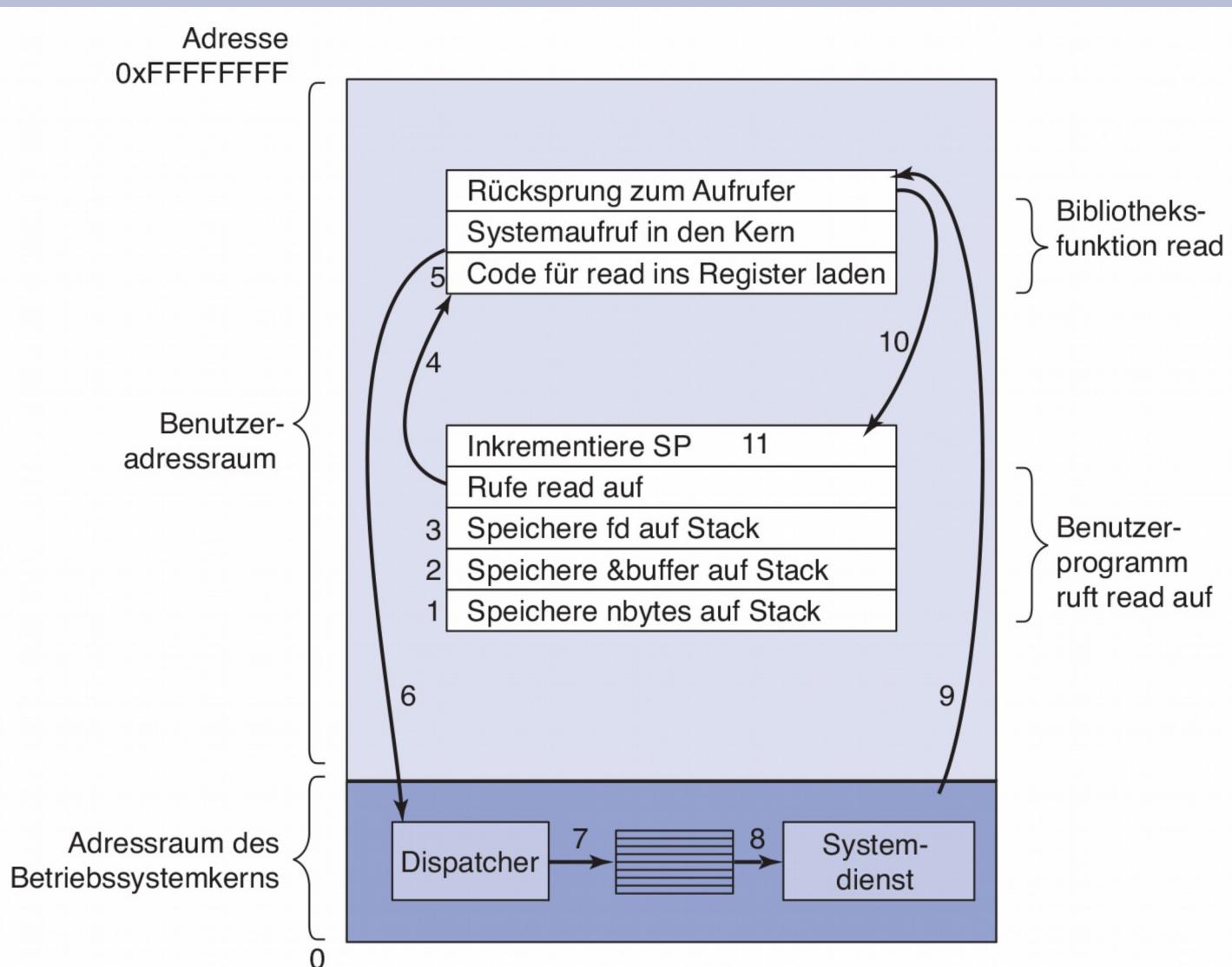


Abbildung 1.17: Die elf Schritte für den Systemaufruf `read(fd, buffer, nbytes)`.

# Systemaufrufe (3)

## Prozessverwaltung

Aufruf	Beschreibung
pid = fork( )	Erzeugen eines neuen Kindprozesses
pid = waitpid(pid, &statloc, options)	Warten auf Beendigung eines Kindprozesses
s = execve(name, argv, environp)	Speicherabbild eines Prozesses ersetzen
exit(status)	Prozess beenden und Status zurückliefern

## Dateiverwaltung

Aufruf	Beschreibung
fd = open(file, how, ...)	Datei zum Lesen, Schreiben oder für beides öffnen
s = close(fd)	Offene Datei schließen
n = read(fd, buffer, nbytes)	Daten aus Datei in Puffer lesen
n = write(fd, buffer, nbytes)	Daten vom Puffer in Datei schreiben
position = lseek(fd, offset, whence)	Dateipositionszeiger bewegen
s = stat(name, &buf)	Status einer Datei ermitteln

# Systemaufrufe (4)

## Verzeichnis- und Dateisystemverwaltung

Aufruf	Beschreibung
s = mkdir(name, mode)	Erzeugen eines neuen Verzeichnisses
s = rmdir(name)	Löschen eines leeren Verzeichnisses
s = link(name1, name2)	Erzeugen eines neuen Eintrags name2, der auf name1 zeigt
s = unlink(name)	Verzeichniseintrag löschen
s = mount(special, name, flag)	Dateisystem einhängen
s = umount(special)	Eingehängtes Dateisystem entfernen

## Sonstige Systemaufrufe

Aufruf	Beschreibung
s = chdir(dirname)	Wechseln des Arbeitsverzeichnisses
s = chmod(name, mode)	Ändern der Dateirechte
s = kill(pid, signal)	Signal an einen Prozess senden
seconds = time(&seconds)	Abgelaufene Zeit seit dem 1. Januar 1970 erfragen

**Abbildung 1.18:** Einige der wichtigsten POSIX-Systemaufrufe. Der Rückgabewert ist  $-1$ , wenn ein Fehler aufgetreten ist. Die angegebenen Rückgabewerte bedeuten Folgendes:  $pid$  ist eine Prozess-ID,  $fd$  ist ein Dateideskriptor,  $n$  ist eine Anzahl von Zeichen,  $position$  ist die Position innerhalb einer Datei,  $seconds$  ist die abgelaufene Zeit. Die Parameter werden im Text erklärt.

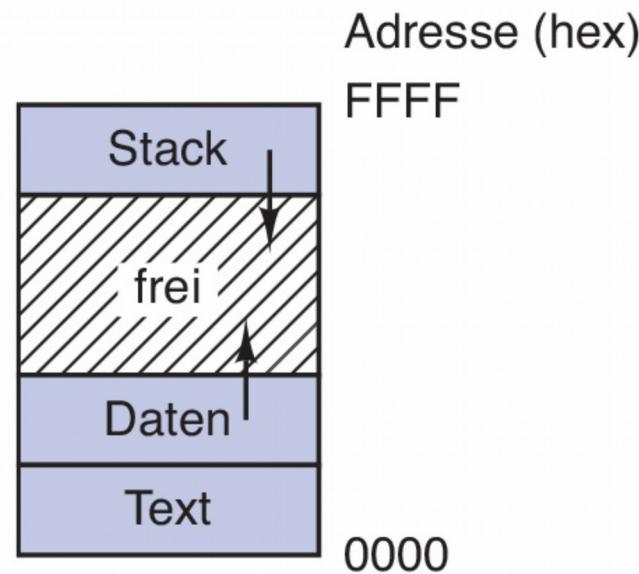
# Systemaufrufe (5)

```
#define TRUE 1
while (TRUE) {
    type_prompt( );
    read_command(command, parameters);
    /* Endlosschleife */
    /* Prompt ausgeben */
    /* Befehl einlesen */

    if (fork( ) != 0) {
        /* Code des Elternprozesses */
        waitpid(-1, &status, 0);
        /* Kindprozess erzeugen */
        /* auf Beendigung von Kindprozess warten */
    } else {
        /* Code des Kindprozesses */
        execve(command, parameters, 0);
        /* Befehl ausführen */
    }
}
```

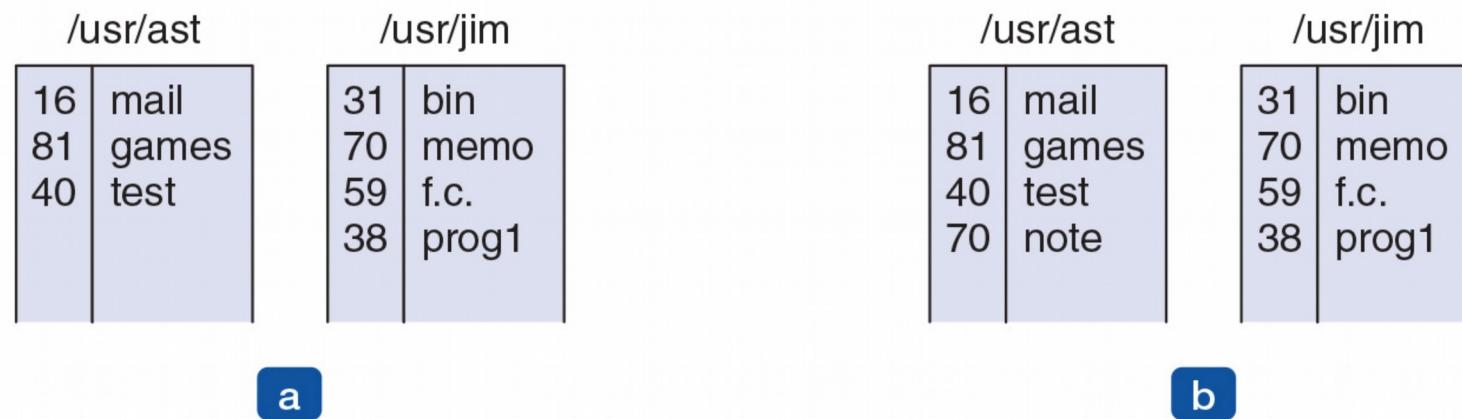
**Abbildung 1.19:** Eine kleine Version einer Shell. In diesem Buch wird *TRUE* durchgängig als Variable mit Inhalt 1 definiert.

# Systemaufrufe (6)



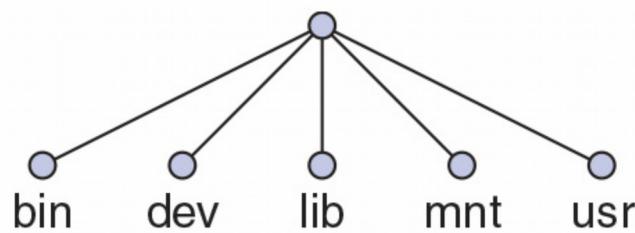
**Abbildung 1.20:** Prozesse besitzen drei Segmente: das Text-, das Daten- und das Stacksegment.

# Systemaufrufe (7)

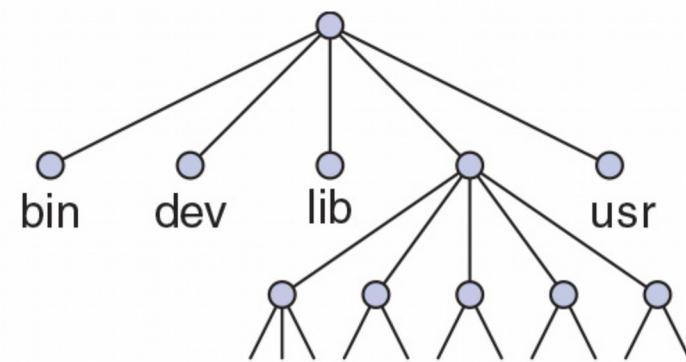


**Abbildung 1.21:** (a) Zwei Verzeichnisse, bevor */usr/jim/memo* in das Verzeichnis *ast* verlinkt wurde; (b) dieselben Verzeichnisse nach dem Aufruf von `link`.

# Systemaufrufe (8)



a



b

**Abbildung 1.22:** Dateisystem vor dem Aufruf von `mount`; (b) Dateisystem nach dem `mount`-Aufruf.

# Win32 API (1)

UNIX	Win32	Beschreibung
fork	CreateProcess	Erzeugen eines neuen Prozesses
waitpid	WaitForSingleObject	Warten auf das Ende eines Prozesses
execve	(nicht vorhanden)	CreateProcess = fork + execve
exit	ExitProcess	Ausführung beenden
open	CreateFile	Erzeugen einer Datei oder Öffnen einer existierenden Datei
close	CloseHandle	Datei schließen
read	ReadFile	Daten aus einer Datei lesen
write	WriteFile	Daten in eine Datei schreiben
lseek	SetFilePointer	Dateizeiger bewegen
stat	GetFileAttributesEx	Dateiattribute erfragen
mkdir	CreateDirectory	Erzeugen eines neuen Verzeichnisses

# Win32 API (2)

rmdir	RemoveDirectory	Löschen eines leeren Verzeichnisses
link	(nicht vorhanden)	Win32 unterstützt keine Links
unlink	DeleteFile	Löschen einer existierenden Datei
mount	(nicht vorhanden)	Win32 unterstützt kein Einhängen
umount	(nicht vorhanden)	Win32 unterstützt kein Einhängen, somit gibt es kein umount
chdir	SetCurrentDirectory	Ändern des aktuellen Arbeitsverzeichnisses
chmod	(nicht vorhanden)	Win32 unterstützt Security nicht (NT schon)
kill	(nicht vorhanden)	Win32 unterstützt keine Signale
time	GetLocalTime	Aktuelle Zeit erfragen

**Abbildung 1.23:** Die Win32-API-Aufrufe, die in etwa mit den UNIX-Systemaufrufen aus ► *Abbildung 1.18* übereinstimmen. Es soll nicht unerwähnt bleiben, dass Windows noch sehr viele weitere Systemaufrufe besitzt, die aber kein Pendant in UNIX haben.

# Betriebssystemstrukturen

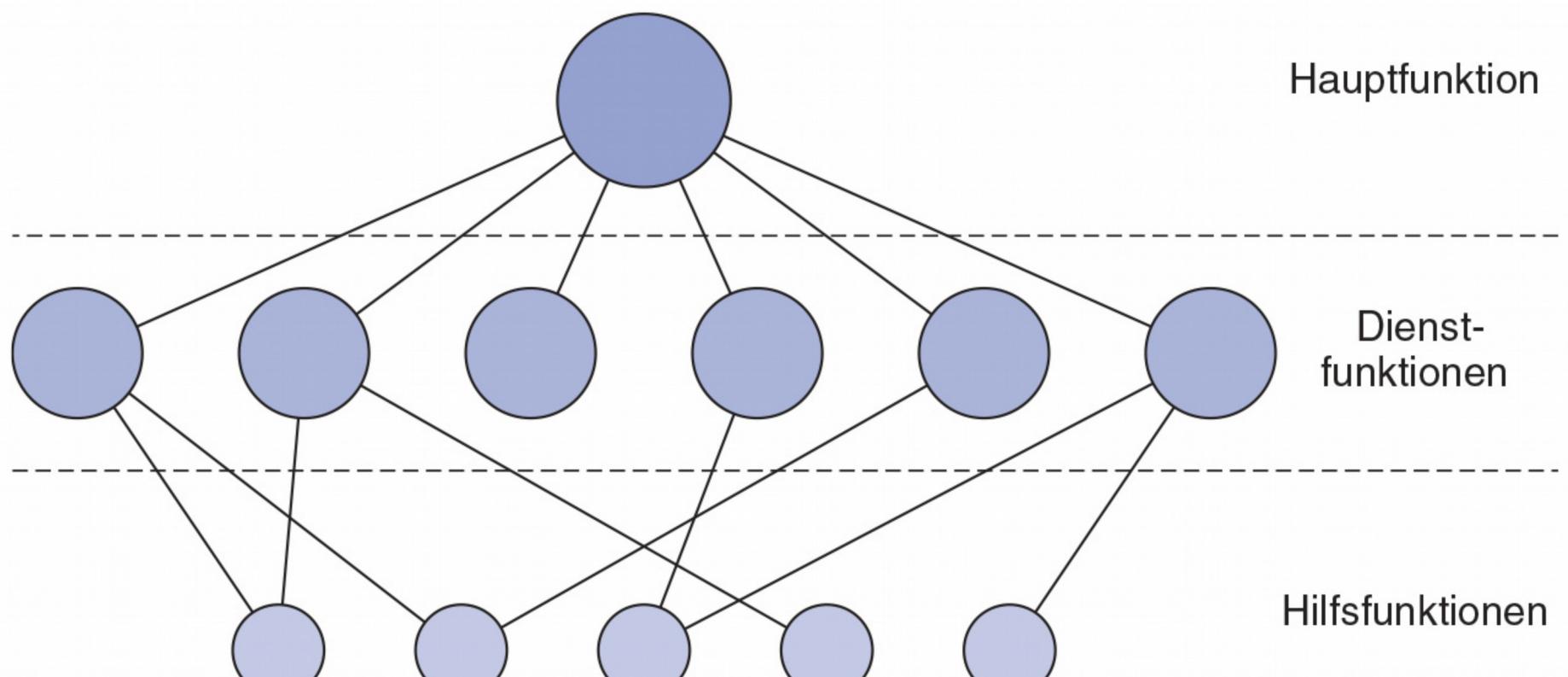
- Monolithische Systeme
- Geschichtete Systeme
- Mikrokerne
- Das Client-Server-Modell
- Virtuelle Maschinen
- Exokerne

# Monolithische Systeme (1)

## Die Grundstruktur von Betriebssystemen

- Eine Hauptfunktion ruft die angeforderte Dienstfunktion auf
- Eine Menge von Dienstfunktionen führt die Systemaufrufe aus.
- Eine Menge von Hilfsfunktionen unterstützt die Dienstfunktionen.

# Monolithische Systeme (2)



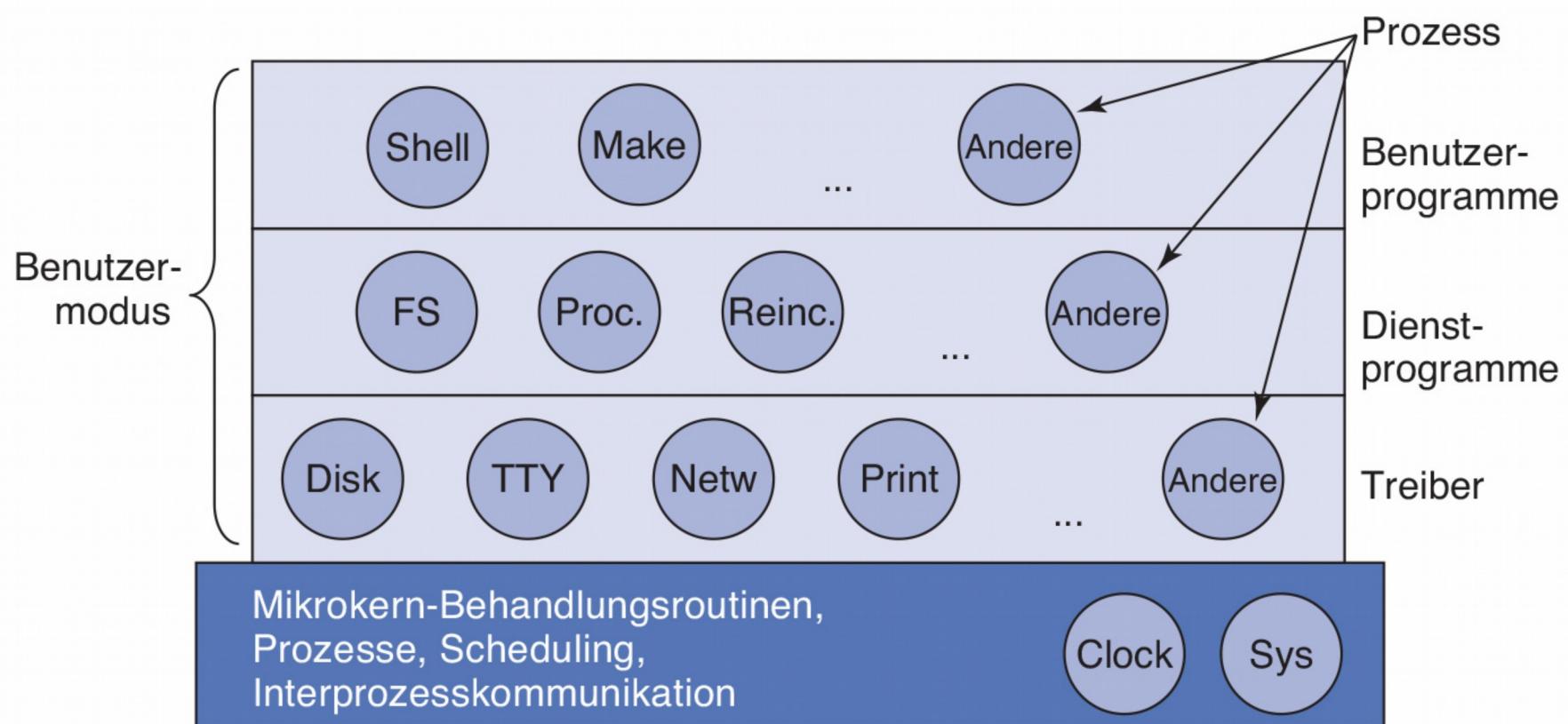
**Abbildung 1.24:** Ein einfaches Strukturmodell für ein monolithisches System.

# Geschichtete Systeme

Layer	Funktion
5	Der Anwender
4	Anwenderprogramme
3	Ein-/Ausgabeverwaltung
2	Anwender-Prozess Kommunikation
1	Speicher- und Trommelmanagement
0	Prozessor Zuweisung und Multiprogrammierung

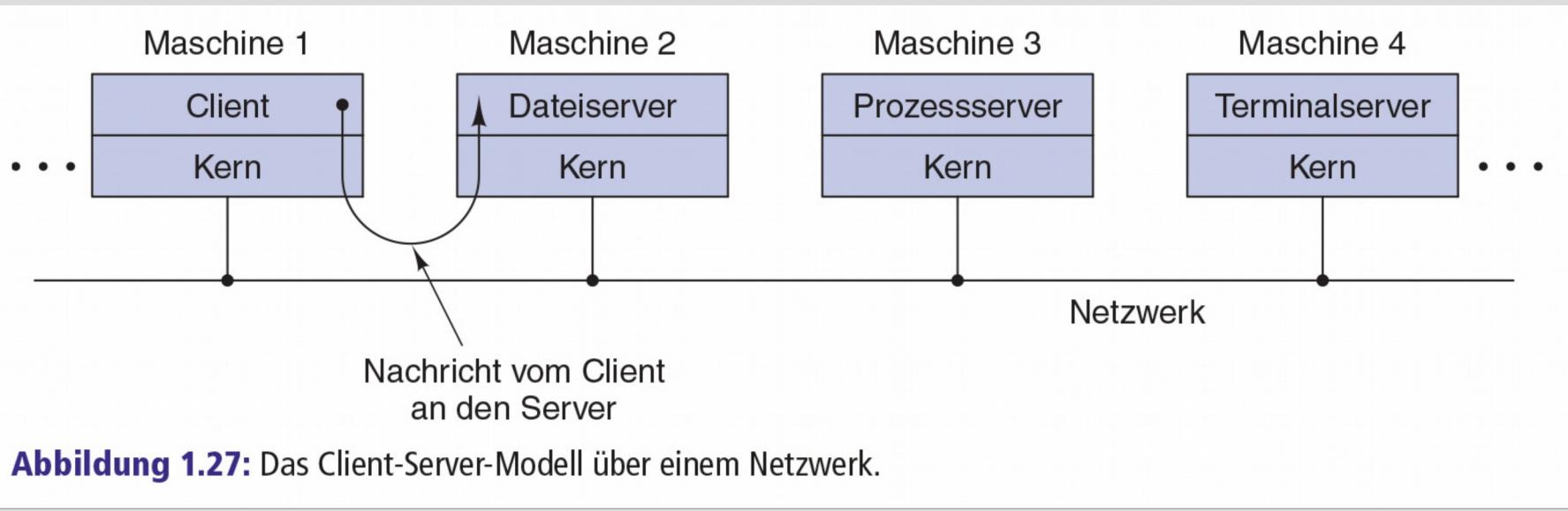
Struktur des THE Betriebssystems

# Mikrokernel



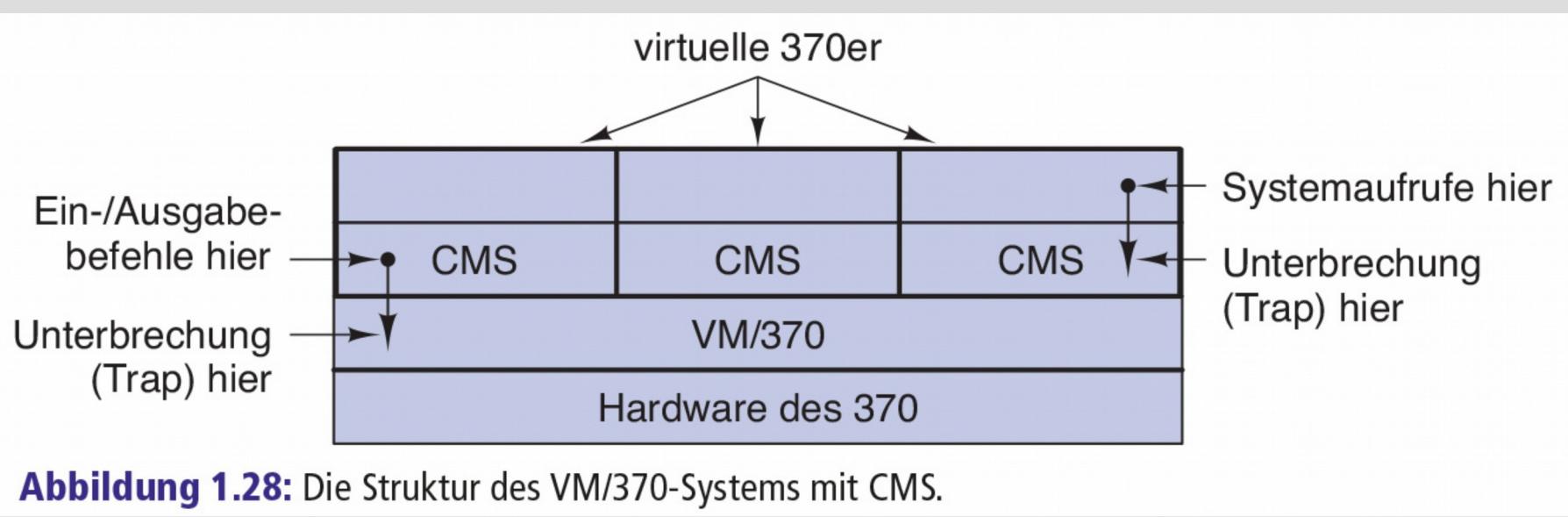
**Abbildung 1.26:** Vereinfachte Struktur des MINIX-Systems.

# Client-Server-Modell



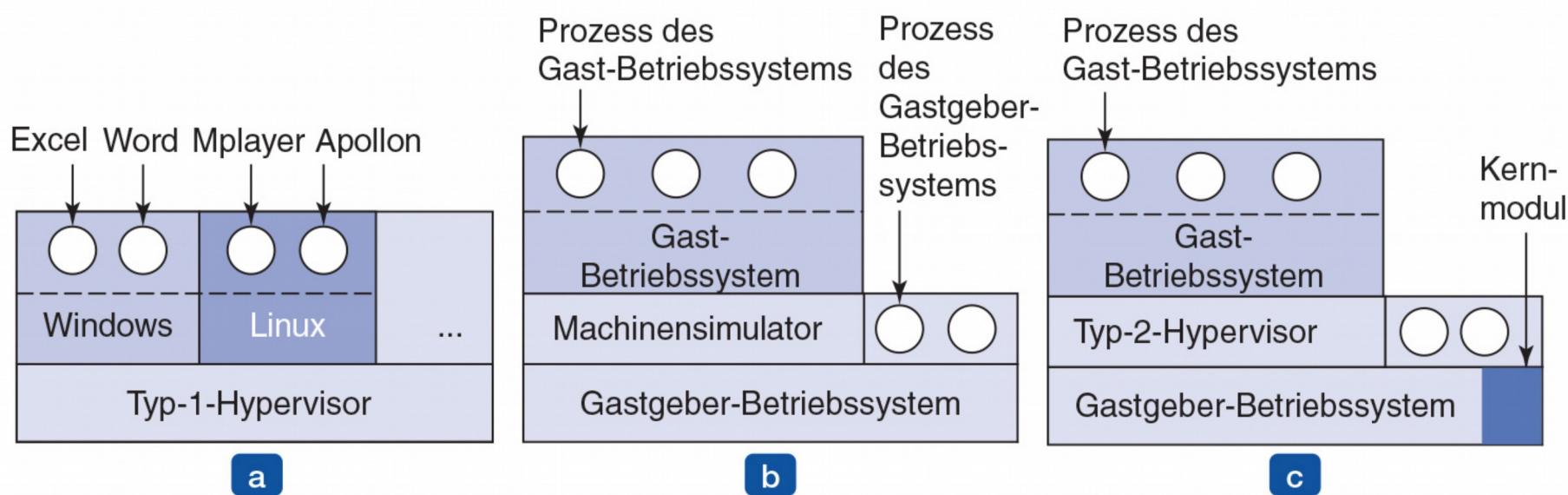
**Abbildung 1.27:** Das Client-Server-Modell über einem Netzwerk.

# Virutelle Maschinen



**Abbildung 1.28:** Die Struktur des VM/370-Systems mit CMS.

# Virutelle Maschinen



**Abbildung 1.29:** (a) Typ-1-Hypervisor; (b) reiner Typ-2-Hypervisor; (c) Typ-2-Hypervisor in der Praxis.

# Die Welt aus der Sicht von C

- Die Programmiersprache C
- Header-Dateien
- Große Programmierprojekte
- Das Laufzeitmodell

# Große Programmierprojekte

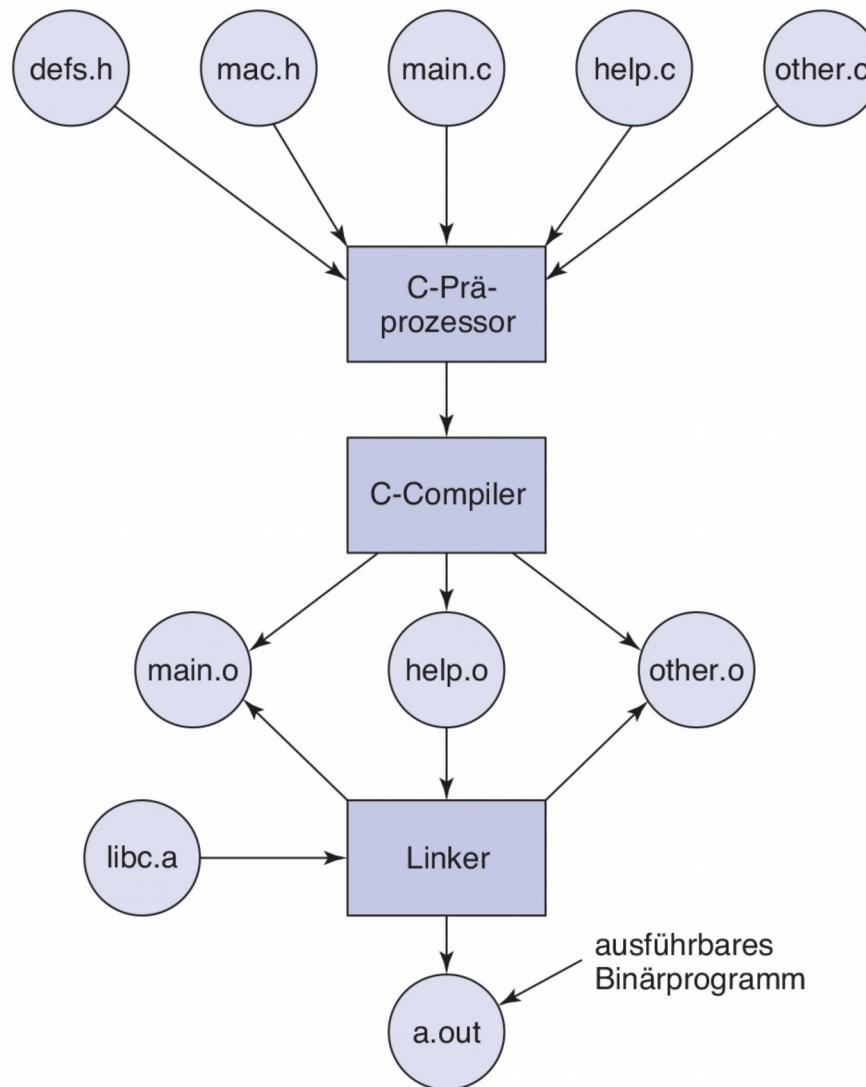


Abbildung 1.30: Der Übersetzungsprozess von C- und Header-Dateien, um eine ausführbare Datei zu erstellen.

# Metrische Präfixe

Exp.	Ausgeschrieben	Präfix	Exp.	Ausgeschrieben	Präfix
$10^{-3}$	0,001	Milli	$10^3$	1.000	Kilo
$10^{-6}$	0,000 001	Mikro	$10^6$	1.000.000	Mega
$10^{-9}$	0,000 000 001	Nano	$10^9$	1.000.000.000	Giga
$10^{-12}$	0,000 000 000 001	Pico	$10^{12}$	1.000.000.000.000	Tera
$10^{-15}$	0,000 000 000 000 001	Femto	$10^{15}$	1.000.000.000.000.000	Peta
$10^{-18}$	0,000 000 000 000 000 001	Atto	$10^{18}$	1.000.000.000.000.000.000	Exa
$10^{-21}$	0,000 000 000 000 000 000 001	Zepto	$10^{21}$	1.000.000.000.000.000.000.000	Zetta
$10^{-24}$	0,000 000 000 000 000 000 000 000 001	Yokto	$10^{24}$	1.000.000.000.000.000.000.000.000.000	Yotta

**Abbildung 1.31:** Die wichtigsten metrischen Präfixe.