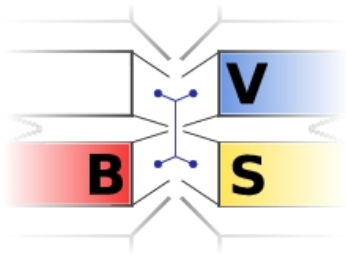
Grundlagen Betriebssysteme und Rechnernetze

Prof. Bettina Schnor

Betriebssysteme und Verteilte Systeme
Institut für Informatik und Computational Science



Wintersemester 2025/26

Voraussetzung/Empfehlung:

- „Maschinenmodelle“ (Ulrike Lucke)
- C-Kenntnisse aus „Grundlagen der Programmierung“ (Henning Bordihn)
- „Algorithmen und Datenstrukturen“ (Henning Bordihn)

$\Longrightarrow$ GBR ist ab dem 3. Semester empfohlen

Lehrangebot

- 3 SWS Vorlesung + 1 SWS Übung:
  1. Woche: 2x VL (Die 10-12 h, Fr 12-14 h)
  2. Woche: 1x VL + 1x Übung
- Freitags abwechselnd Übung (12-14 h) oder **Tutorium** (fakultativ) nach der VL (14 -16 h):
  Im Tutorium können Fragen zu den Hausaufgaben gestellt werden.
- **Erstes Tutorium:** Wiederholung C und Einführung in CoFee, Freitag 17.10.2025, 14:00 Uhr - 16:00 Uhr, Raum 02.70.0.11

Termine/Ablauf:

- **Dienstag 14.10.2025:**
  Erste Vorlesung, 10-12 h, Raum 02.25.F0.01
- **Freitag 17.10.2025:**
  Zweite Vorlesung, 12-14 h, Raum 02.70.0.11
- **Freitag 17.10.2025:**
  Tutorium, 14-16 h, Raum 02.70.0.11
  Max Schrötter gibt eine Einführung in **gitlab** und das Feedback-Tool **CoFee** für die Bearbeitung der Programmieraufgaben.
  GGf. Laptop mitbringen!
- **Dienstag 21.10.2025:** Dritte Vorlesung
- **Freitag 24.10.2025:** Erste Übung, 12-14 h, Ausgabe von Blatt 1
- ...

- Die Folien-PDFs finden Sie auf der Webseite der Lehrveranstaltung passwort-geschützt.
  (Folien von vorherigen Semestern sind ebenfalls einsehbar!)
- 7 bewertete Hausaufgaben:
  - Alle Blätter enthalten 50 % Programmieraufgaben und 50 % theoretische Aufgaben.
  - in 2er- oder 3er-Gruppen bearbeiten
- **Wichtig:** Melden Sie sich umgehend beim zugehörigen Moodle-Kurs an, damit wir alle im Übungsbetrieb aktiven Studenten per email erreichen.

Ansprechpartner:
Prof. Dr. Bettina Schnor, schnor@cs.uni-potsdam.de

Übungsbetrieb:
Max Schrötter, max.schroetter@cs.uni-potsdam.de

Tutoren: Andreas Niemann, Philipp Ungrund

Prüfung

Zulassung: Mit 50 % der Hausaufgabenpunkte wird man zur Klausur zugelassen.

Die Note ergibt sich aus der Klausurnote.

Klausurtermin:

TODO

Anmeldung: Für die Teilnahme an der Klausur ist die Anmeldung zur Modulprüfung in PULS erforderlich. Über den Anmeldungsbeginn werden Sie in der Übung und per E-Mail informiert.

Teil I: Grundlagen Betriebssysteme

1. Einführung
   - 1.1 Definition eines Betriebssystems
   - 1.2 Aufgaben eines Betriebssystems
   - 1.3 Geschichte der Betriebssysteme
   - 1.4 Architekturbeispiele: Windows 2000, Microkernel
2. Prozeßverwaltung
   - 2.1 Einführung
   - 2.2 Leichtgewichtsprozesse/Threads
   - 2.3 Scheduling
   - 2.4 Interprozeßkommunikation
3. Synchronisation nebenläufiger Prozesse
   - 3.1 Synchronisationshardware
   - 3.2 Busy Waiting
   - 3.3 Semaphor
   - 3.4 Monitor
   - 3.5 Nachrichtensystem

Teil II: Grundlagen Rechnernetze

1. **Einführung, Terminologie**
2. **Das ISO-Referenzmodell OSI**
3. Die physikalische Schicht
4. **Die Sicherungsschicht**
5. **Die Vermittlungsschicht (Beispielprotokoll IP)**
6. **Die Transportschicht (Beispielprotokolle TCP und UDP)**
7. Die Sitzungsschicht
8. Die Darstellungsschicht
9. Die Anwendungsschicht

Literatur - Betriebssysteme

1. William Stallings: *Operating Systems – Internals and Design Principles*, 9. Auflage, Pearson, 2018
2. Andrew S. Tanenbaum, Herbert Bos: *Modern Operating Systems*, 1136 Seiten, Pearson Studium, Addison Wesley, 4. Auflage, 2016.
3. Marshall Kirk McKusick, George V. Neville-Neil: *The Design and Implementation of the FreeBSD Operating System*, Addison-Wesley, 2. Auflage, 2014.
4. Wolfgang Mauerer: *Professional Linux Kernel Architecture*, John Wiley & Sons (Wrox), 2008
5. Robert Love: *Linux Kernel Development*, Addison Wesley,Third Edition, 2010
6. Mark E. Russinovich, David A. Solomon, Alex Ionescu: *Windows Internals Part 1: System architecture, processes, threads, memory management and more*, 7th Edition, Microsoft Press, 2017

Literatur - Rechnernetze

1. James F. Kurose, Keith W. Ross: *Computer Networking - A Top-Down Approach*, Pearson, Eigth Edition, 2020.
2. Andrew S. Tanenbaum, David J. Wetherall: *Computernetzwerke*, Pearson Studium, 5. Auflage, 2012 (in der Bibliothek)
   Computer Networks, Pearson, 6th Edition, Taschenbuch 2021
3. Douglas E. Comer: *TCP/IP - Studienausgabe: Konzepte, Protokolle, Architekturen*, Verlag mitp, 2011
4. Kevin R. Fall und W. Richard Stevens: *TCP/IP Illustrated Volume 1: The Protocols*, Addison-Wesley, 2011.

"The problems that were first encountered in operating systems are now present in many other software products. Ideas that were pioneered in operating systems are now commonly used in those products."

David Lorge Parnas, Software-Engineering-Pionier,
Frühjahrstreffen der GI Fachgruppe Betriebssysteme, 29.6.12

Neben grundlegend neuen Konzepten gibt es immer wieder auch *Abfallprodukte* der Betriebssystementwicklung:

- Experimentelle Betriebssystem Plan 9
  Plan 9 from Bell Labs ist ein Betriebssystem, entwickelt in den späten 1980er Jahren von den Bell Laboratories, die zuvor bereits Unix entwickelt hatten. Es ist benannt nach dem Film Plan 9 from Outer Space (1959) von Ed Wood, in dem die Außerirdischen Tote wieder zum Leben erwecken.
  ...
  UTF-8 (8-Bit Universal Character Set Transformation Format) ist die am weitesten verbreitete Kodierung für Unicode-Zeichen. Die Kodierung wurde im September 1992 von Ken Thompson und Rob Pike bei Arbeiten an dem Plan 9 Betriebssystem festgelegt.

  Quelle: Wikipedia

- Experimentelle verteilte Betriebssystem Amoeba:
  Guido von Rossum: 1986-1991 im Amoeba-Projekt tätig

"Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office (a government-run research lab in Amsterdam) would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus)."

Definition 1: Das Betriebssystem wird durch die Programme eines digitalen Rechensystems gebildet, die zusammen mit den Eigenschaften der Rechenanlage die Basis der möglichen Betriebsarten des digitalen Rechensystems bilden und insbesondere die Abwicklung von Programmen steuern und überwachen.
(DIN 433 00)

Betriebsarten:

1. Einprogramm–/Mehrprogrammbetrieb
   (single/multi programming)
2. Stapelbetrieb / Dialogbetrieb
   (batch/interactive mode)

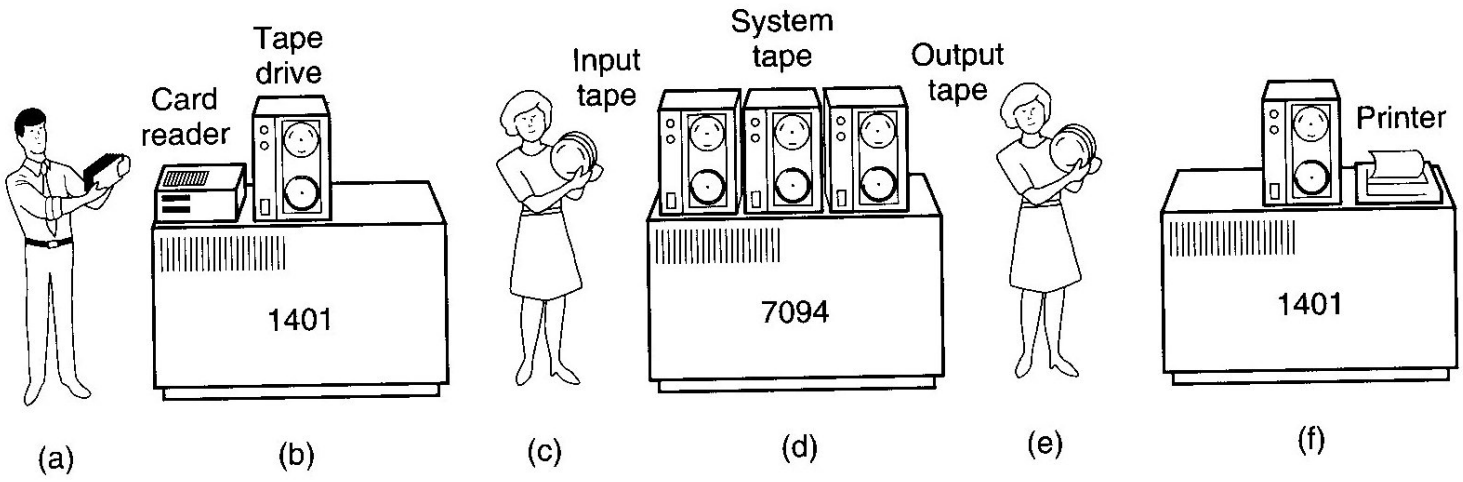


Figure 1-2. An early batch system. (a) Programmers bring cards to 1401. (b) 1401 reads batch of jobs onto tape. (c) Operator carries input tape to 7094. (d) 7094 does computing. (e) Operator carries output tape to 1401. (f) 1401 prints output.

(entnommen Tanenbaum)

Definition 2: Ein Betriebssystem ist eine virtuelle Maschine, die dem Anwender eine einfache (dateiorientierte) Schnittstelle zur Hardware zur Verfügung stellt und eine Programmierung dieser Hardware auf hohem logischen Niveau ermöglicht.

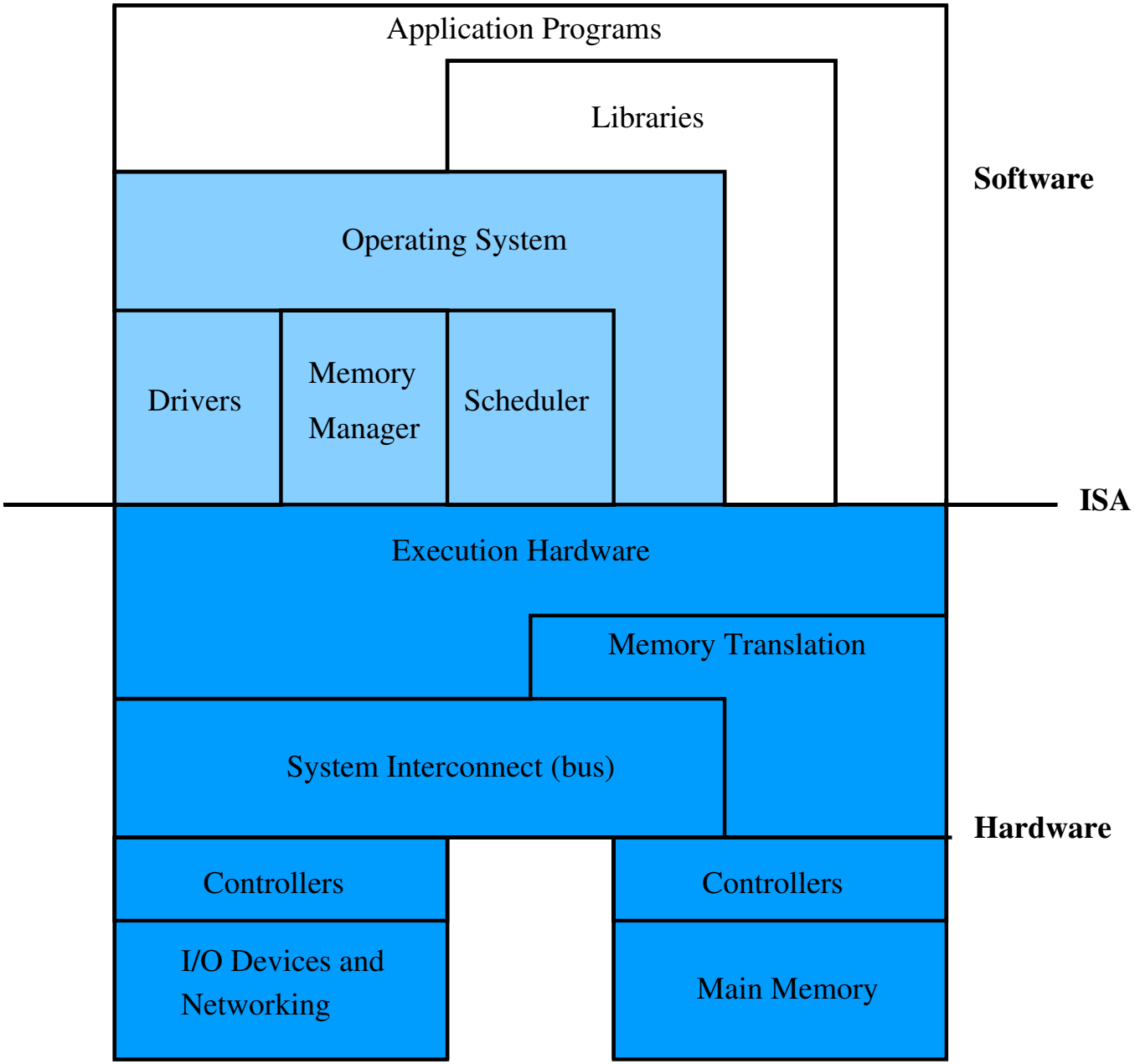
Schichtenmodell eines Rechensystems:

Compiler, Editor, Spiele, ...
Kommandointerpreter (Shell), Graphische Oberfläche
Betriebssystem
Maschinenprogrammebene
Mikroprogrammebene
physikalisches Gerät

Instruction Set Architecture (ISA): Schnittstelle zwischen Hard- und Software (deutsch: Befehlssatzarchitektur)

Eine Implementierung der ISA ist Hardware, die der Architekturabstraktion gehorcht, d.h. die ISA zur Verfügung stellt.

$\Longrightarrow$ ermöglicht die Portabilität eines Betriebbssystems



adaptiert nach Smith/Nair: *Virtual Machines, 2005*

Betriebssystem als Ressourcenverwalter

$\hat{=}$ Sicht der BS-Entwickler

Definition 3: Die Ressourcen (Betriebsmittel) eines Betriebssystems sind alle Hard- und Softwarekomponenten, die für die Programmausführung relevant sind.
Betriebsmittel: Prozessor, Hauptspeicher, Ein-/Ausgabegeräte, Daten, ausführbarer Code, ...

Definition 4: Ein Betriebssystem bezeichnet alle Softwarekomponenten eines Rechensystems, die die Ausführung der Benutzerprogramme, die Verteilung der Ressourcen auf die Benutzerprogramme und die Aufrechterhaltung der Betriebsart steuern und überwachen.

1. Prozeßverwaltung

Definition 5: Ein Prozeß (process, task) ist ein in Ausführung befindliches Programm bestehend aus

- einer Folge von Maschinenbefehlen, die durch das ausgeführte Programm festgelegt sind (program code, text section),
- dem Inhalt des Stapel-Speichers (stack), auf dem temporäre Variablen und Parameter für Funktionsaufrufe verwaltet werden,
- dem Inhalt des Speichers, in dem die globale Daten des Prozesses gehalten werden (data section) und
- Verwaltungsinformationen, die den aktuellen Bearbeitungszustand (Kontext) beschreiben.
  Der Programmkontext wird u.a. durch den Programmzähler und die Registerinhalte des Prozessors beschrieben (internal state).

Relation auf der Menge der Prozesse:

Definition 6: Die Laufzeit eines Prozesses P wird durch ein Tupel $(a_1, a_2) \in \mathrm{R}^2$ beschrieben, wobei $a_1 :=$ die Startzeit von P und $a_2 :=$ die Endezeit von P ist.

Definition 7: A und B seien Prozesse mit Laufzeit (a_1, a_2) bzw. (b_1, b_2).

a) A läuft vor B (i.Z. $A \to B$): $\iff \quad a_2 < b_1$.

b) A und B heißen sequentielle Prozesse $:\iff A \to B \quad \vee B \to A$

c) A und B heißen nebenläufig (engl. concurrent) $:\iff$ $[a_1, a_2] \cap [b_1, b_2] \neq \emptyset$.

d) A und B heißen parallel $:\iff$ Es existiert ein Zeitintervall, während dessen A und B gleichzeitig rechnen.

Bem.:

1. "$\rightarrow$" irreflexive Halbordnung

   $A \rightarrow B \quad \Rightarrow \quad B \not\rightarrow A$ asymmetrisch

   $A \rightarrow B \quad \wedge \quad B \rightarrow C \quad \Rightarrow \quad A \rightarrow C$ transitiv

   $A \not\rightarrow A$ irreflexiv

2. Einprozessorsystem $\Rightarrow$ sequentielle Prozesse und nebenläufige Prozesse möglich

Aufgaben:

- Erzeugen und Löschen von Prozessen (`Fork`)
- Prozessorzuteilung (Scheduling)
- Prozeßkommunikation (Pipe, Sockets, Signale, ...)
- Synchronisation nebenläufiger Prozesse, die gemeinsame Daten benutzen

Definition 8:

1. Mittels Systemaufrufen (englisch system calls) wird Anwendungsprozessen die Funktionalität des Betriebssystems zur Verfügung gestellt. Dabei wird die Kontrolle vom Anwendungsprogramm an den Betriebssystemkern übergeben.
2. libc ist die Standardbibliothek der Programmiersprache C. Verbreitete Implementierung auf UNIX-Systemen: glibc vom GNU-Projekt

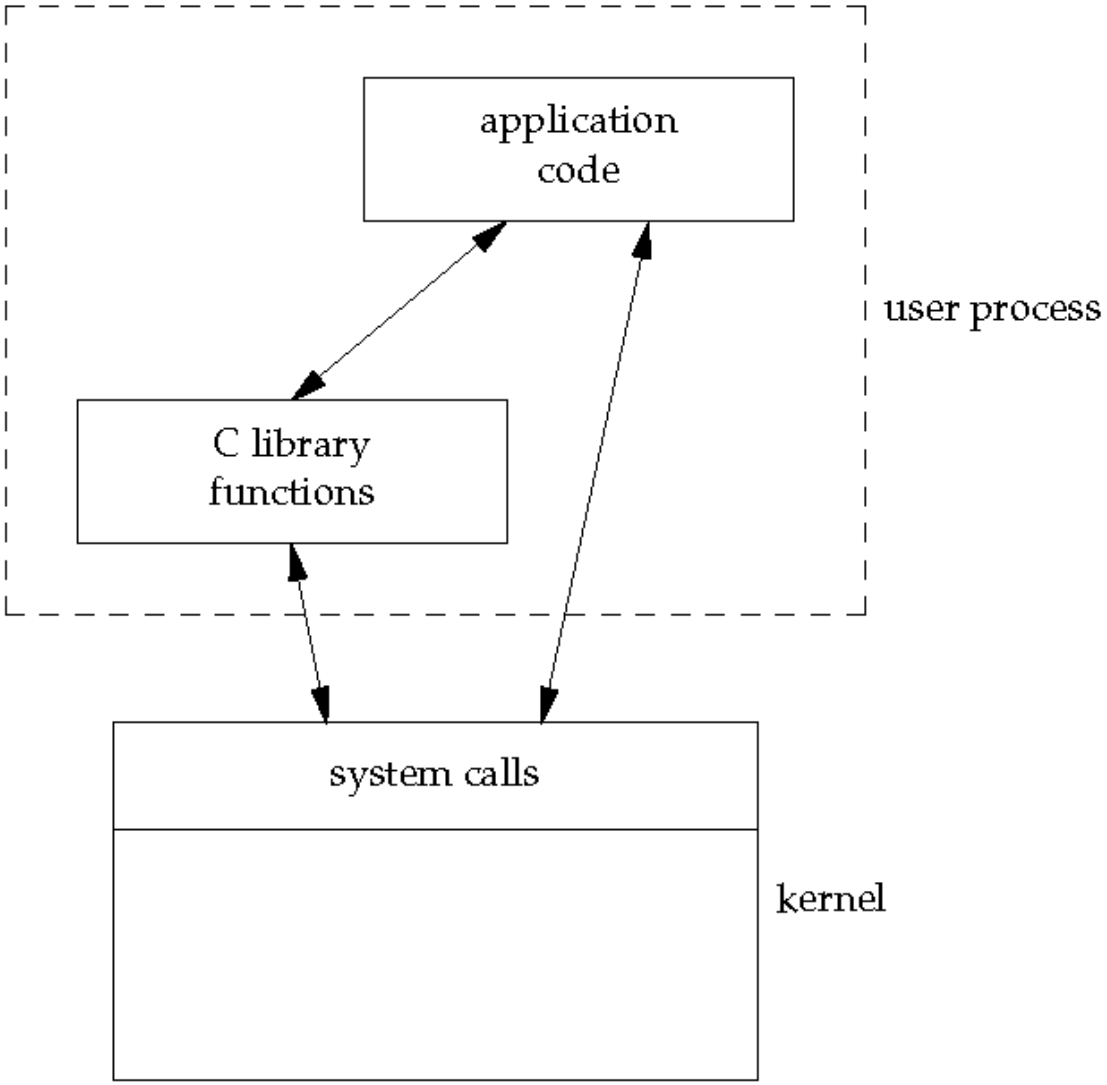
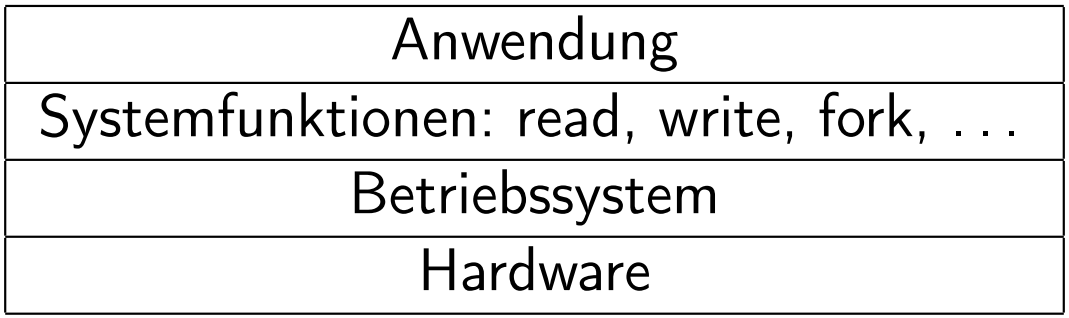


Figure 1.12 Difference between C library functions and system calls

Quelle: Stevens, Rago: *Advanced Programming in the UNIX Environment*

User-Mode und Kernel-Mode



Damit nebenläufige Prozesse sich nicht gegenseitig (zer-)stören, unterscheiden moderne Betriebssysteme zwei Zustände:

Eine Benutzeranwendung läuft i.a. im unpriviligierten User-Mode, der nur eingeschränkten Zugriff auf die Hardware erlaubt.
Alle Betriebssystemkomponenten laufen im priviligierten Kernel-Mode, der uneingeschränkten Zugriff auf die Hardware erlaubt.

$\Longrightarrow$ Spezielle Speicherbereiche dürfen nur im Kernel-Mode gelesen werden.

Wie wird diese Unterscheidung realisiert?

Die Trennung von User- und Kernel-Mode benötigt Hardwareunterstützung im Prozessorregister:

1. **Programmzähler:** enthält die Adresse des aktuellen Befehls
2. **Instruction Register:** aktuell auszuführender Befehl
3. **Program Status Word:** Menge von Registern, die Statusinformationen enthalten: User/Kernel Mode Bit, Interrupt enabled/disabled Bit, ...

2. Speicherverwaltung

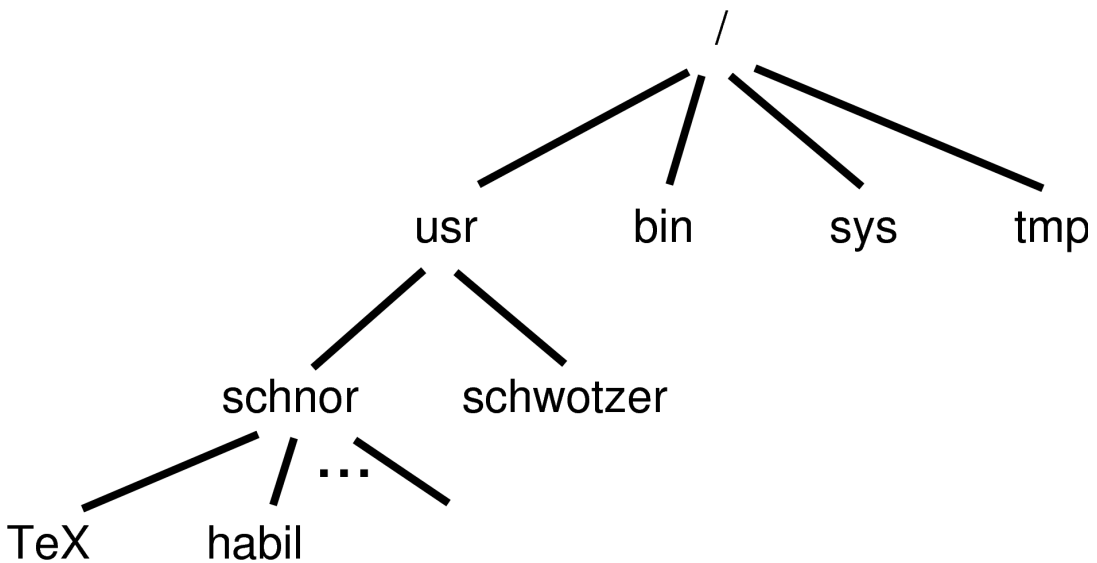
- Zuteilung des verfügbaren physikalischen Speichers an Prozesse
- Einbeziehen des Hintergrundspeichers (Platte) durch virtuelle Adressierung

Techniken:
Swapping := Ein-/Auslagern von Prozessen
Demand Paging, z.B. UNIX, Windows NT, nicht MS-DOS

3. Verwaltung des Dateisystems

- logische Sicht auf Speichereinheiten (Dateien)
  (Benutzer arbeitet mit Dateinamen. Wie und wo Dateien gespeichert werden, ist ihm egal.)
- Systemaufrufe für Dateioperationen
  (Erzeugen, Löschen, Öffnen, Lesen, Schreiben)
- Strukturierung mittels „Verzeichnisse" (Directory) $\Longrightarrow$
  hierarchisches Dateisystem:

Beispiel UNIX:



```
> ls /usr
bin/    i686-w64-mingw32/  lib/    local/       sbin/   src/
games/  include/           lib32/  lost+found/  share/  x86_
```

Etwas Historie: Linux Filesystem Hierarchy - Auszug

`/usr` usually contains by far the largest share of data on a system. Hence, this is one of the most important directories in the system as it contains all the user binaries, their documentation, libraries, header files, etc. X and its supporting libraries can be found here. User programs like telnet, ftp, etc.... are also placed here.
In the original Unix implementations, /usr was where the home directories of the users were placed (that is to say, /usr/someone was then the directory now known as /home/someone). In current Unices, /usr is where user-land programs and data (as opposed to 'system land' programs and data) are. The name hasn't changed, but it's meaning has narrowed and lengthened from "everything user related" to "user usable programs and data". As such, some people may now refer to this directory as meaning 'User System Resources' and not 'user' as was originally intended.

- Schutz von Dateien und Verzeichnissen vor unberechtigtem Zugriff.
  in MS-DOS: $\emptyset$

  in UNIX: 9–Bit für Zugriffsrechte (Read, Write, eXecute)
  Datei/-verzeichnis gehört einem Eigentümer (Owner) und einer Benutzergruppe (Gruppe der Mitarbeiter, Gruppe der Studenten, projektspezifische Gruppe, . . .)

owner	group	others	
`rwx`	`r-`	`r-`	Übung

  Ändern von Gruppenrechten:
  `> chmod g+w Übung`

4. Geräteverwaltung

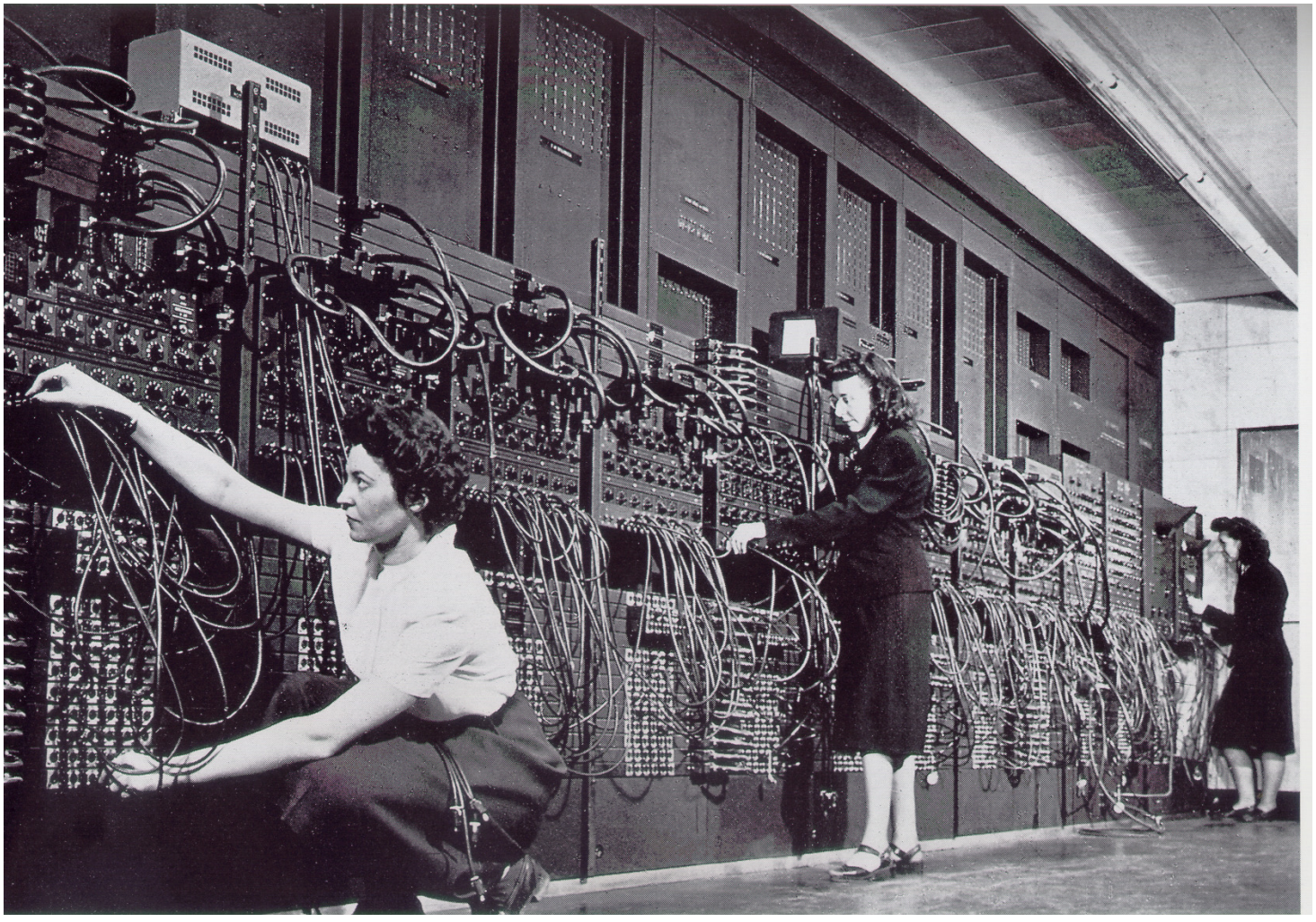
- Auswahl und Bereitstellung von E/A-Geräten
- Anpassung an physikalische Eigenschaften der Geräte
- Überwachung der Datenübertragung

5. Monitoring, Accounting, Auditing

- Systemstatistiken:
  aktuelle Auslastung (`top, ps`),
  aktueller freier Speicher (`vmstat`),
  Netzverkehr (`netstat`)
  ...
- optionale Systemstatistiken:
  Laufzeit von Prozessen mitprotokollieren
  Speicherbedarf von Prozessen mitprotokollieren
  eingeloggte Benutzer mitprotokollieren
  ...

1. Generation 1945 - 1955: Röhren
Programmierung in Maschinensprache
kein Betriebssystem

1946 J. P. Eckert und J. W. Mauchly entwerfen und bauen den ersten Elektronenrechner:
Electronic Numerical Integrator and Computer (Eniac).
Er hatte 18 000 Elektronenröhren und 1 500 Relais
und war aufgrund dieser Komplexität sehr unzuverlässig.



6 Frauen (Kathleen McNulty, Frances Bilas, Elizabeth Snyder, Ruth Lichterman, Marlyn Wescoff und Jean Bartik) sind die ENIAC-Programmierer

2. Generation 1955 - 1965: Transistor

Lochkarten

Betriebssystem für Stapelbetrieb

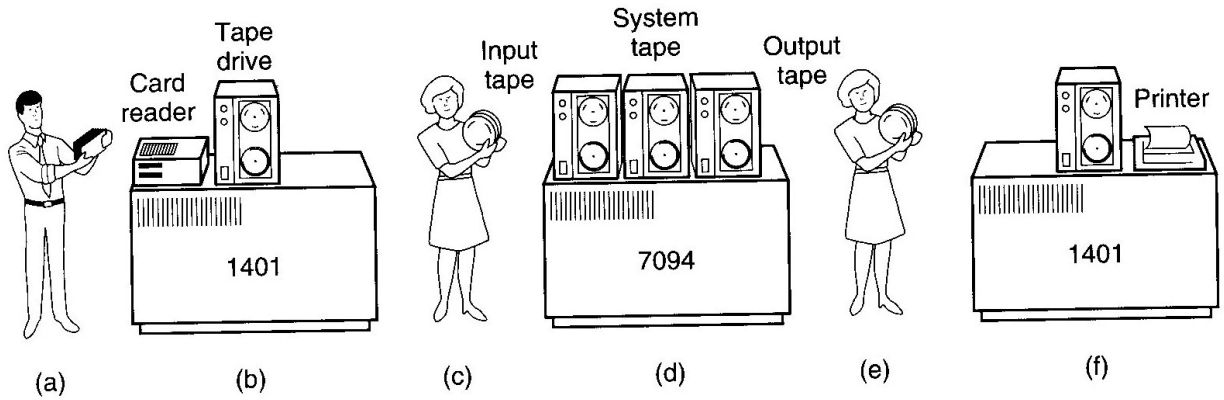


Figure 1-2. An early batch system. (a) Programmers bring cards to 1401. (b) 1401 reads batch of jobs onto tape. (c) Operator carries input tape to 7094. (d) 7094 does computing. (e) Operator carries output tape to 1401. (f) 1401 prints output.

(entnommen Tanenbaum)

3. Generation 1965 - 1980: Entwicklung integrierter Schaltkreise

Rechnerfamilie IBM 360 mit OS/360

neu: Multiprogramming, um I/O-Wartezeiten auszunutzen

- CTSS (Compatible Time Sharing System), MIT
  experimentelles Betriebssystem (Anfang der 60er)
- ca. 1965: MULTICS (MULTiplexed Information and Computing System)[1]
  MIT, Bell Labs (AT&T), General Electrics,
  Entwicklungsflopp!
- 1961: DEC bringt PDP-Serie auf den Markt
  4K Hauptspeicher, für ca. 120 000 $ extrem günstig für damalige Verhältnisse
- 1969: Ken Thompson (Bell Labs) entwickelt UNICS (UNIplexed Information and Computing System) auf PDP-7 ("Eunuch")
  $\Longrightarrow$ UNIX
  1. UNIX-Version in Assembler,
  Portierung auf PDP-11/20, -11/45, -11/70

[1] http://www.multicians.org/

- Um den Portierungsaufwand zu minimieren entwirft Thompson die Sprache B,
- Dennis Ritchie entwirft C $\Longrightarrow$ C-Implementierung von UNIX von Thompson/Ritchie,
- AT&T gibt UNIX gegen eine geringe Lizenzgebühr an Universitäten frei (Sourcen!),
- 1984 AT&T wird umstrukturiert und steigt in das Computergeschäft ein: kommerzielles Produkt System III
  System V, Release 2,3,4,
- zweite Entwicklungslinie an der University of California, Berkeley: 1 BSD (Berkeley Software Distribution) $\Longrightarrow$
  4.4 BSD
  neu: virtuelle Speicherverwaltung, Paging, Internetprotokolle, Hilfsprogramme (vi, csh, ...),
- UNIX-Stammbaum von Eric Levenez:
  `http://levenez.com/unix/`

4. Generation 1980 - 1990: LSI/VLSI Entwicklung $\Longrightarrow$
leistungsfähige Einbenutzersysteme: PC/MS-DOS,
Workstation/UNIX
IBM OS/2, Linux (POSIX-basiert, 1991)

Wichtige UNIX-Konzepte

- Repräsentiere alle Ressourcen als Dateien!
- hierarchisches Dateisystem mit der Wurzel '/' : /bin/....
- 9-Bit Rechteuniversum:

owner			group			others		
r	w	x	r	w	x	r	w	x

- Mr. Allmächtig: ROOT

```
-rw-------  1 root  wheel  3088 23 Sep  2007 /etc/master.passwd
```

- UNIX verknüpft jedes laufende Programm („Prozeß“) mit einem Standardein- und Standardausgabe-Kanal, sowie Standard-Error.
  Umlenken von Ein-/Ausgabe in der Kommandozeile mittels: $>$, $\gg$, $<$ und $|$
- IPC (Inter Process Communication) mittels Signale, Pipes, Sockets
- Prozesserzeugung mittels `fork()`: Vererben aller I/O-Kanäle

In the 1980s, during the so-called “Unix wars” ...

Standardisierungsversuch in den 80er Jahren:

- IEEE Standard 1003.1 (1988): POSIX (Portable Operating System Interface based on UNIX)
- Definition von Bibliotheksfunktionen z.B. open, read, fork, ... Schnittmenge aus System V und 4.3 BSD

POSIX Standards:

1003.0	Open-system architecture
1003.1	Posix application interface (system calls and library functions)
1003.2	Shell and command utilities
1003.3	Testing and verification methods
1003.4	Real-time extensions to Posix
1003.5	Ada language bindings
1003.6	System security extensions
1003.7	System administration
1003.8	Transparent file access
1003.9	FORTRAN language bindings
1003.10	Supercomputing profile
1003.11	Transaction processing
1003.12	Protocol-independent network access
1003.13	Application environment profiles
1003.14	Multiprocessing application environment profiles
1003.15	Batch services

In the 1980s, during the so-called "Unix wars" ...

1988 bedeutende Computerhersteller (Komplement zu AT&T) gründen OSF (Open Software Foundation)[2] z.B. IBM, DEC, HP, ...

- Betriebssystem OSF/1
- Entwicklung 1995 eingestellt
- 1996 ist OSF in der Open Group aufgegangen.

Literatur: W. Richard Stevens and Stephen A. Sago: *Advanced Programming in the UNIX Environment*, Addison-Wesley, 2. Auflage, 2008.

[2] OSF/Motif zur Entwicklung grafischer Benutzerschnittstellen unter POSIX-konformen Systemen

Historie von MS-DOS

1975-1980: Erster PC/Microcomputer: Altair von MITS, 8-Bit Intel 8080 CPU, 256 Bytes Hauptspeicher.

- Auf fast allen Microcomputern läuft das Betriebssystem CP/M (Control Program for Microcomputers) von Digital Research. Bill Gates schreibt einen BASIC Interpreter für den Altair und gründet Microsoft, um ihn kommerziell zu vertreiben. Weiterhin vertreibt Microsoft AT&T UNIX.

1980: IBM beauftragt einen seiner Manager, Philip Estridge, einen PC einzukaufen inklusive Betriebssystem.

- Unter den beiden 8080-Nachfolgern, den 8-bit 8088 und den 16-bit 8086, wählt Estridge den billigeren 8088. Der IBM PC hat 64 KByte Speicher und keine Festplatte.
  Estridge kontaktiert Gates wegen einer Lizenz seines BASIC Interpreters. UNIX kommt als Betriebssystem nicht in Frage, da es allein für das OS 100 KByte Speicher und eine Festplatte benötigt. Gates empfiehlt Digital Research. Doch Digital Research kann Lieferzeit nicht garantieren.

  Gates kennt eine Firma Seattle Computer Products, die ein CP/M-ähnliches Betriebssystem, 86-DOS, geschrieben haben, um ihre Speicherbänke zu testen. Gates kauft 86-DOS und stellt den Autor, Tim Paterson ein.

1981 IBM PC wird mit MS-DOS 1.0 ausgeliefert: 4000 Zeilen Assembler, MS-DOS benötigt 12 KByte Speicher, auf Intel 8088 Einbenutzerbetriebssystem, flaches Dateisystem, Entwicklungszeit 1/2 Jahr.

1983 IBM bringt ersten PC mit Festplatte heraus, den PC/XT, mit MS-DOS 2.0: komplett neu, 20 000 Zeilen Assembler, 4 Entwickler, neu: hierachische Dateistruktur und Systemaufrufe `open, close, read, write` a la UNIX, Druckabwicklung. $\Longrightarrow$ PC wird ernsthaftes Arbeitsgerät.

1984 MS-DOS 3.0 für IBMs PC/AT (auf Intel 80286): 40 000 Zeilen Assembler, 30 Entwickler
Der PC/AT unterstützt bis zu 16 MByte Speicher, kennt User und Kernel Mode und Zugriffsschutz ähnlich Multics. Dies wird vom MS-DOS 3.0 nicht ausgenutzt.

11/1984 MS-DOS 3.1 unterstützt Netzkommunikation, funktioniert aber nur für *"well-behaved programs."*

1986 MS-DOS 3.2 (3 1/2 Zoll Diskette), *"full of bugs"*

1988 MS-DOS 3.3 für IBMs PS/2 (auf Intel 80386)
Alternativ bringen IBM und Microsoft ein neues Betriebssystem OS/2 auf den Markt. Obwohl OS/2 technisch MS-DOS überlegen ist (volle Speichernutzung, Multiprogramming, protected Mode) wird es am Markt nicht akzeptiert.

1990 MS-DOS 4.0 unterstützt Platten bis 2 GByte.
Windows 3.0

1991 Microsoft lässt OS/2 fallen. IBM schließt Vertrag mit Apple. Jahre später wird IBM das Open Source Betriebssystem Linux unterstützen.

1995 Windows95 on top of MS-DOS7.0 besitzt virtuelle Speicherverwaltung, Multiprogramming, benutzt aber noch das traditionelle DOS Dateisystem

1998 Windows98

2000 Windows Me (Millennium Edition)

Neuentwicklung:

1992 Windows NT, Windows 2000, Windows XP, ..., seit 2007 Windows Vista (NT 6.0), Windows 7 in Oktober 2009, Windows 8 Oktober 2012 (Unterstützung für ARM-Architektur), ... Windows 11 in Oktober 2021.

The Windows family tree

1992 erhielt Bill Gates die National Medal of Technology *"For his early vision of universal computing at home and in the office."*

5. Generation 1990 - : Verteilte Systeme durch Vernetzung

Definition 9: Ein Netzbetriebssystem ist ein Betriebssystem, das um einen Kommunikationsmechanismus erweitert wurde, so daß Kommunikation und Kooperation zwischen Betriebssystemen möglich ist und damit der Zugriff auf entfernte Ressourcen.

Beispiele: UNIX, Windows NT, Linux

Linux (http://www.kernel.org/)

- Linus Torvalds, Undergraduate Student an der Uni Helsinki, schreibt einen Terminal Emulator unter Minix („Lern-“Betriebssystem von Andrew S. Tanenbaum) für besseren Netzzugang zur Uni.

- Daraus wird sein Disk driver / File System Project.
- 1991 stellt Torvalds seinen ersten Linux-Kern für 386er per ftp zu Verfügung.
- März 1994 LINUX 1.0

+ Open Source Software: GNU General Public License (GPL), http://www.kernel.org/

Vorteile:

+ stabil
  „*Cost is one of the last things people mention when talking about Linux's benefits. Reliability is one of the first.*“
  Byte Magazin
+ schnell
+ sicherer

Linux im Backbone-Einsatz

- Im Internet:
  - Internet-Server (www, mail, dns etc.): Anno 2017 knapp 55% der Webserver Linux-basiert[3], ANNO 2022 80 %
  - Wikipedia-Server (www, MySQL, etc.)
  - Deutsche Bahn: Fahrplan-Auskunft, interne Server-Landschaft[4]
- zu Hause und Unterwegs:
  - Router: Fritz!Box[5]
  - Smartphone: Android (Linux-Kernel), 80% Marktanteil Anno 2017[6]
  - Waschmaschine, Mikrowelle[7] und Fernseher[8]

[3] http://w3techs.com/technologies/overview/operating_system/all

[4] http://www.tecchannel.de/server/linux/402505/die_bahn_faehrt_auf_linux_ab/

[5] http://www.wehavemorefun.de/fritzbox/Main_Page

[6] https://www.idc.com/promo/smartphone-market-share/os

[7] http://www.linux-community.de/Internal/Nachrichten/Android-schon-bald-in-Waschmaschinen-und-Mikrowellen-Geraeten

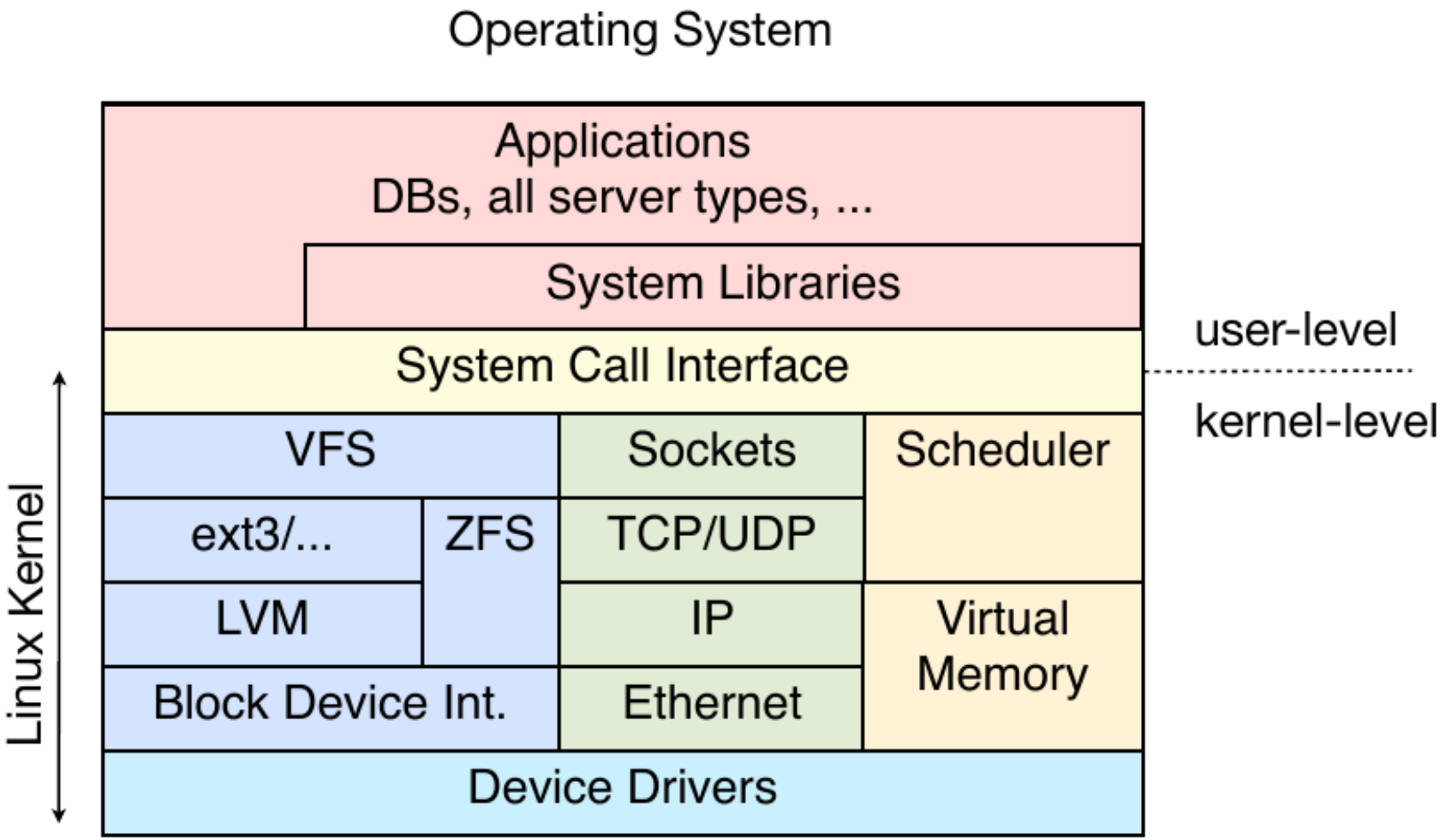
[8] http://www.linux-magazin.de/Heft-Abo/Ausgaben/2010/06/Frei-Zeit-im-Pantoffelkino

UNIX/Linux-Einsatz im IFI

- Linux auf allen Serversystemen (Fileserver, DNS, Webserver)
- Linux auf den Rechenknoten des Turing-Cluster
- Forschungsprojekte dank Linux bzw. BSD UNIX
- Erstsemesterausbildung

Monolithisches Betriebssystem

Beispiel: Linux

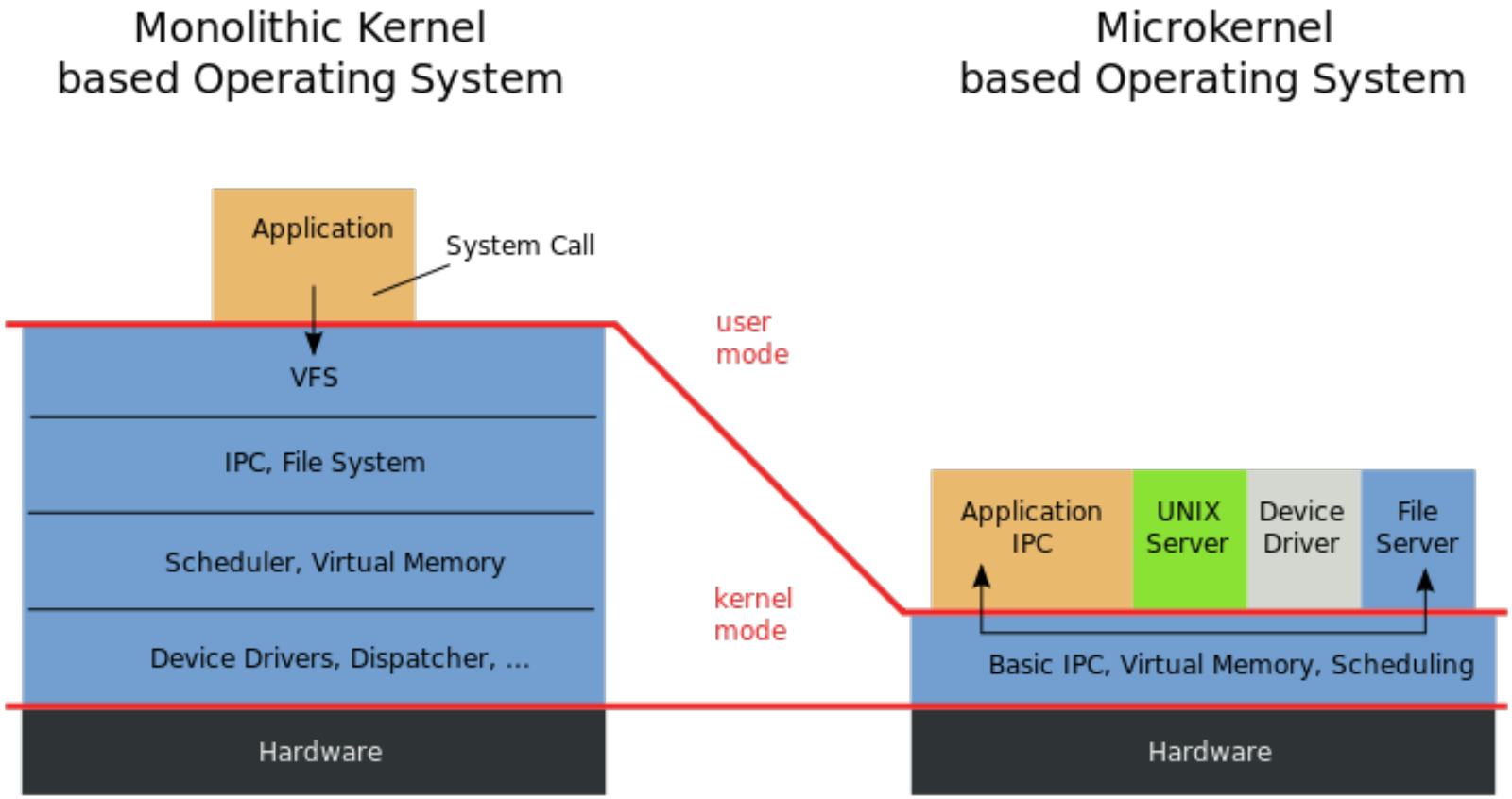


Quelle: Brendan Gregg: *Linux Performance Analysis and Tools*, Vortrag auf der SCaLE11x, 2013.

VFS	Virtual File System
ZFS	Zettabyte File System
LVM	Logical Volume Manager

Linux ist monolithisch. Innerhalb des Kerns findet man auch eine geschichtete Architektur (siehe Kommunikationsstack und Dateisystem).

Zum Vergleich: Microkernel-Architektur

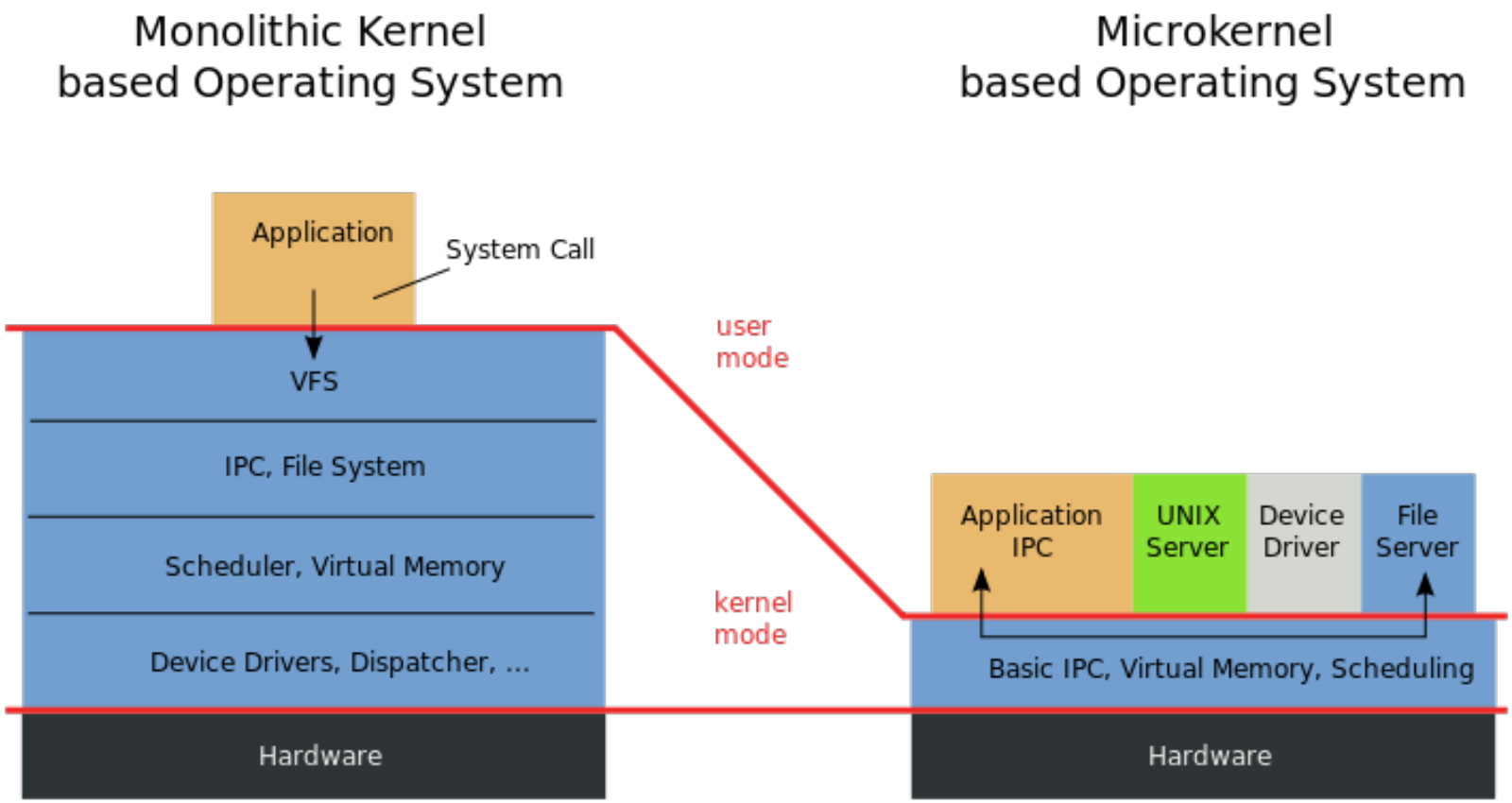


Quelle: `https://en.wikipedia.org/wiki/Microkernel`

Kennzeichen: Der Betriebssystem**kern** enthält nur die grundlegenden Funktionen (u.a. Speicher- und Prozessverwaltung). Jede weitere Funktionalität wird von eigenständigen Prozessen im User-Mode implementiert.

Vorteil einer Microkernel-Architektur: Der Betriebssystem**kern** ist deutlich kleiner. $\Longrightarrow$ Leichter zu verifizieren!

Nachteil einer Microkernel-Architektur:



Quelle: `https://en.wikipedia.org/wiki/Microkernel`

Wechsel zwischen Betriebssytem-Diensten im User-Mode bedeutet einen Context-Switch. $\Longrightarrow$ Zeitaufwendiger!

Architekturbeispiel Windows NT

zurück in das Jahr 1992: Windows NT 3.1

NT = New Technology

Chef-Entwickler: David N. Cutler, ehemals DEC (z.B. am VAX/VMS beteiligt)

(Marketing-)Ziele:

- Portabilität,
- Sicherheit (endlich Zugriffsrechte für Dateien),
- POSIX-Schnittstelle,
- 32-bit Adressen,
- Kompatibilität zu 16-bit Betriebssystemen,
- Multiprozessor-Support, z.B. Leichtgewichtprozesse

Definition 10: Ein Leichtgewichtprozeß (Thread) ist eine Ausführungseinheit mit minimalen Zustandsinformationen (z.B. Programmzähler, Registerinhalte, Thread-Priorität).

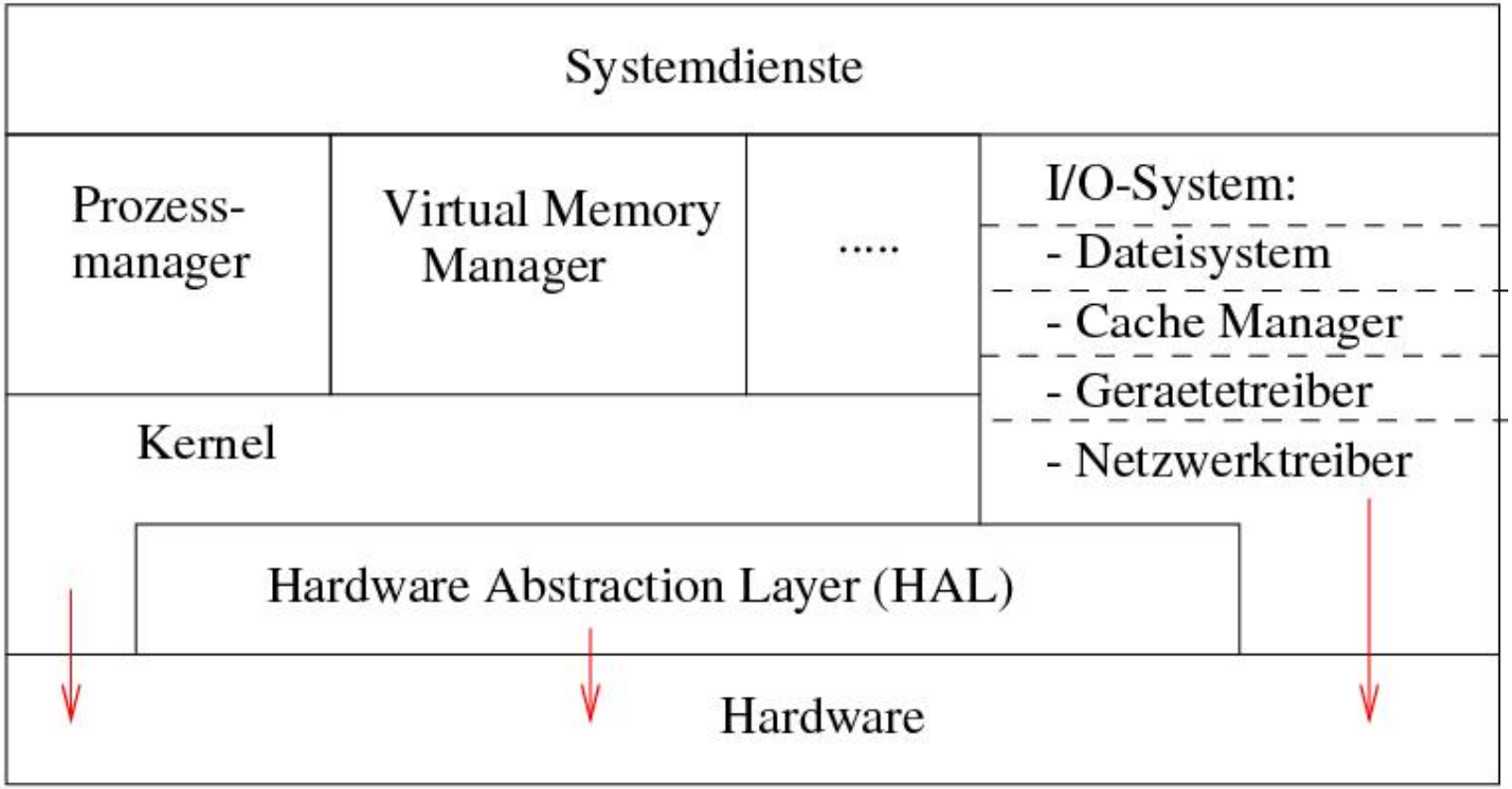
Definition 11: Eine Task ist eine Verwaltungseinheit, in der die Systemressourcen wie z.B. Hauptspeicher, I/O-Kanäle etc., die einer oder mehreren Threads zugeteilt sind, zusammengefaßt sind.

Ein UNIX-Prozeß entspricht einer Task mit einem Thread.

Achtung: Windows/NT-Terminologie: Mehrere Threads innerhalb eines „Prozesses"

Anwendung des Thread-Konzepts:

- Parallelverarbeitung auf Multicoremaschinen
- Lastverteilung: Multithreaded (Web-)Server



Windows NT ist modularisiert aufgebaut. Analog zur Microkernel-Architektur wird NT in einen *Kernel* und *Manager* genannte Komponenten strukturiert. Teilweise wird auch eine geschichtete Architektur benutzt.

Handelt es sich um eine Microkernel-Architektur?

Windows-Vista-Betriebssystem

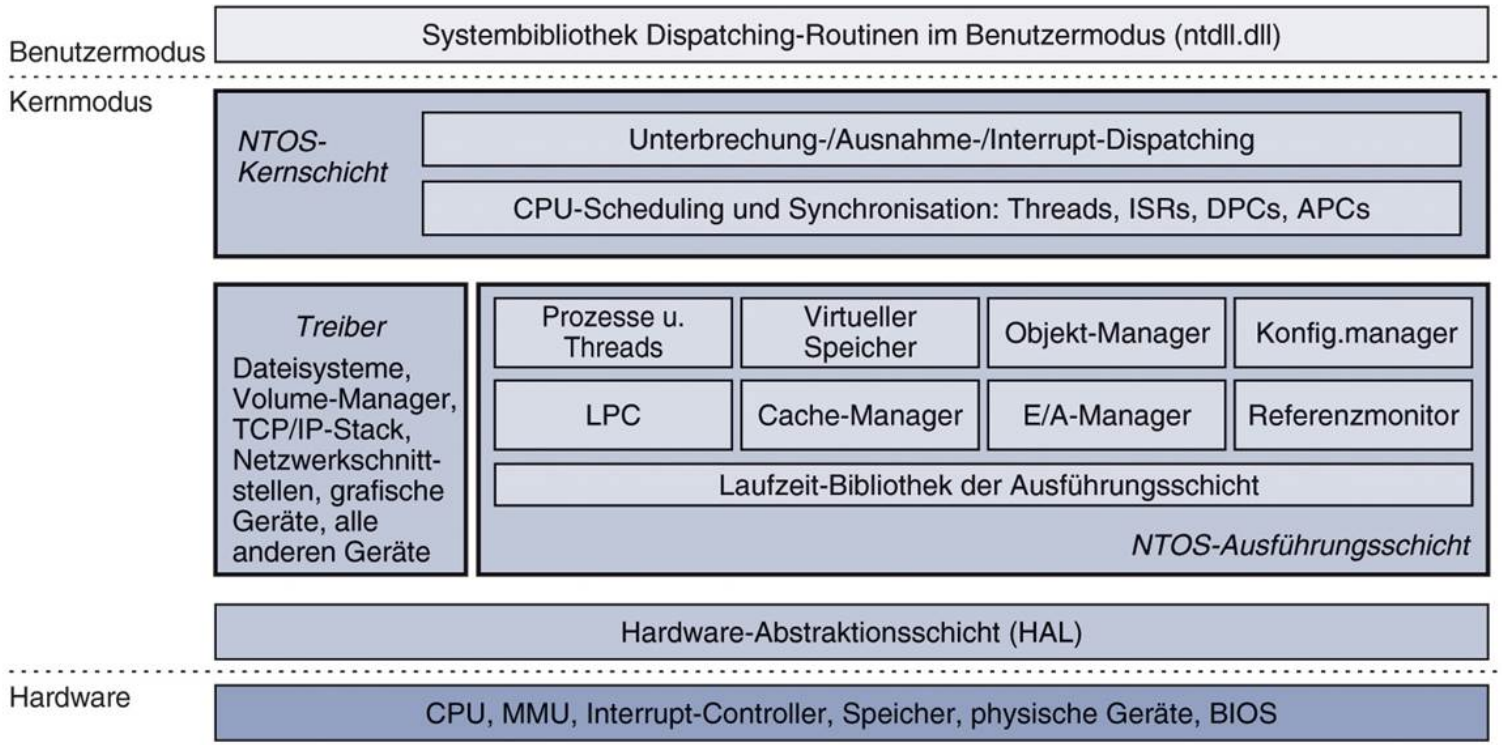


Abbildung 11.13: Organisation des Windows-Kernmodus

Quelle: Andrew S. Tanenbaum: *Moderne Betriebssysteme*, 3. Auflage, Pearson Studium