HAW HAMBURG

Betriebssysteme

Prof. Dr.-Ing. Christian Lins

Einführung

Lernziele

HAW HAMBURG

Am Ende der VL-Einheit können Sie…

- die **Aufgaben eines Betriebssystems** beschreiben,
- heutige BS in den **historischen Kontext** einordnen und
- die wesentliche **BS-Konzepte und -Strukturen** erläutern.
- Sie kennen die wichtigsten **Grundlagen von Linux** und
- besitzen das nötige Rüstzeug für die erste Praktikumsaufgabe.

https://www.youtube.com/watch?v=7QWosNm5xV8

Was ist ein Betriebssystem?

Einführung

Struktur eines Rechnersystems

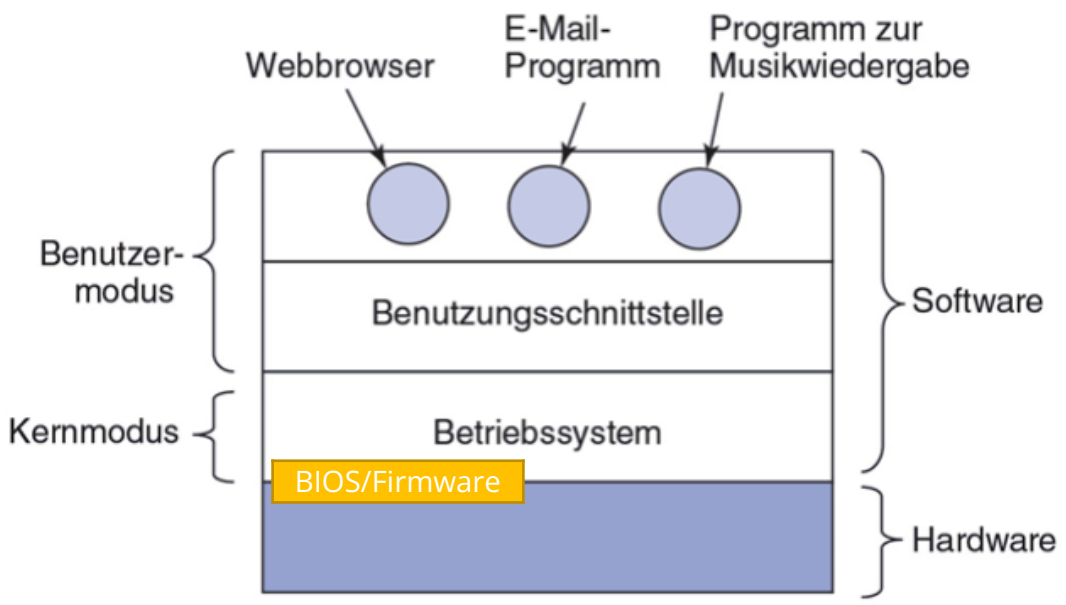


Abb. aus [TB]

Ein **Betriebssystem** ist

- Schicht zwischen Hardware und Software
- Abstrahiert und vereinfacht die Hardwarenutzung
- Organisiert die Nutzung der Hardware

Betriebssystem als Schnittstelle zur Hardware

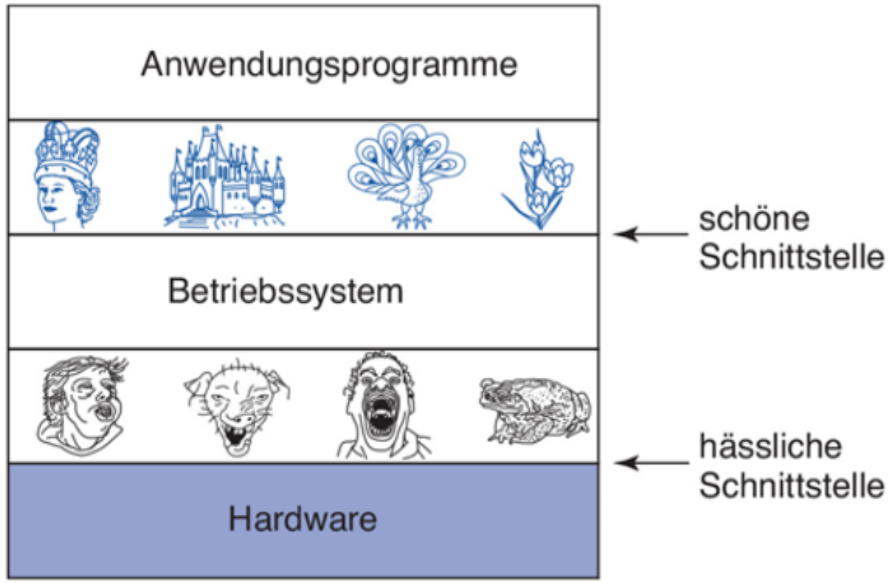


Abb. aus [TB]

Schnittstellen der Hardware sind oft

- komplex
- schwierig zu benutzen
- individuell

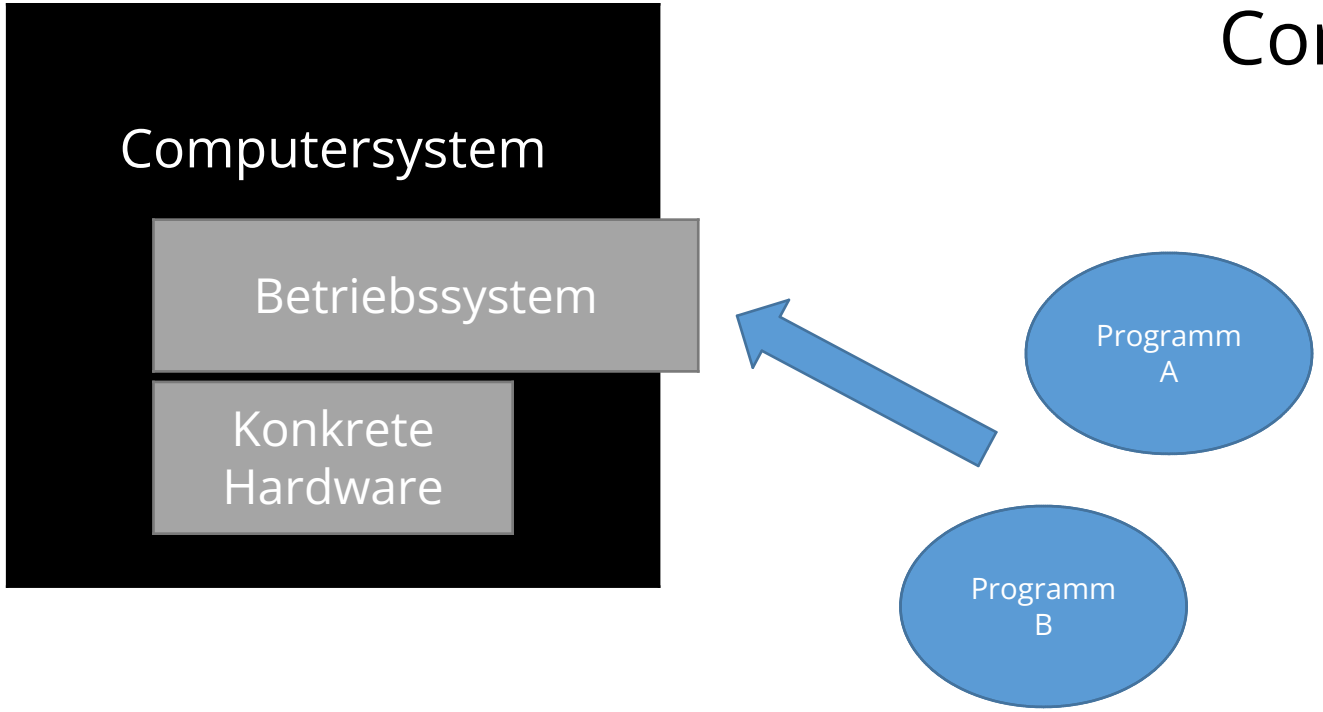
Schnittstellen des BS

- vereinfachen und vereinheitlichen

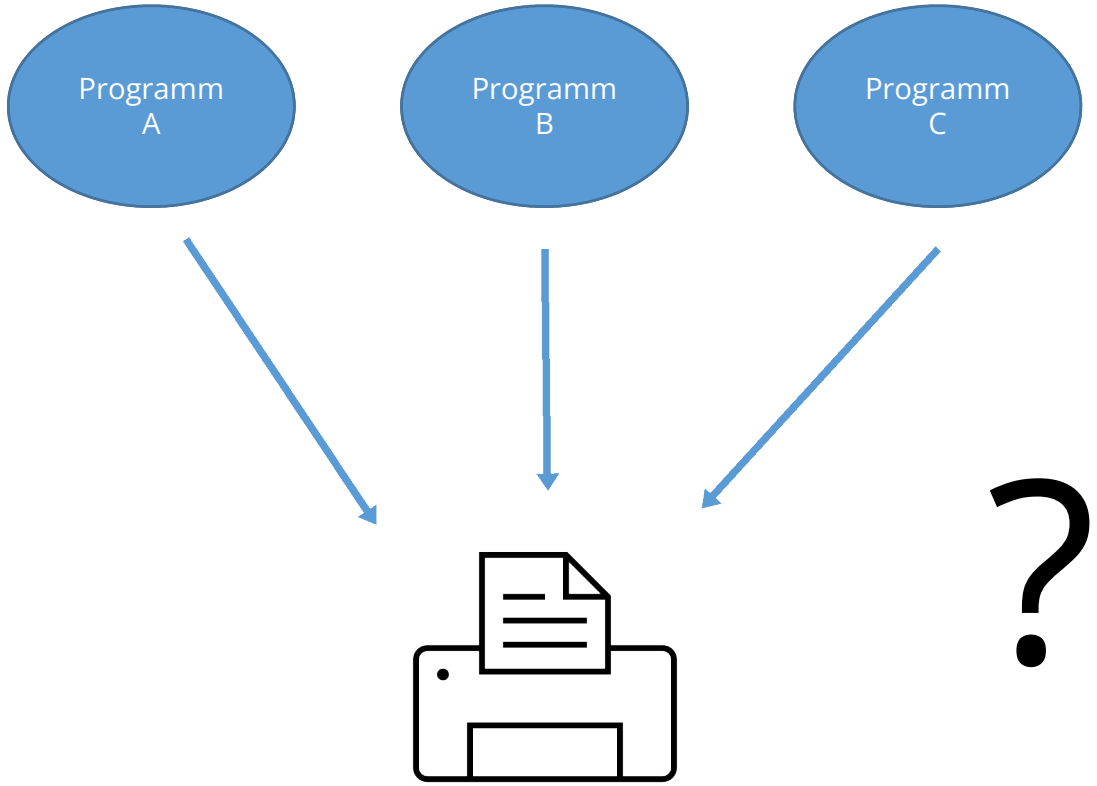
Abstraktion der Hardware

Das Betriebssystem als erweiterte Maschine

Computer als Black Box



Ressourcenverwaltung

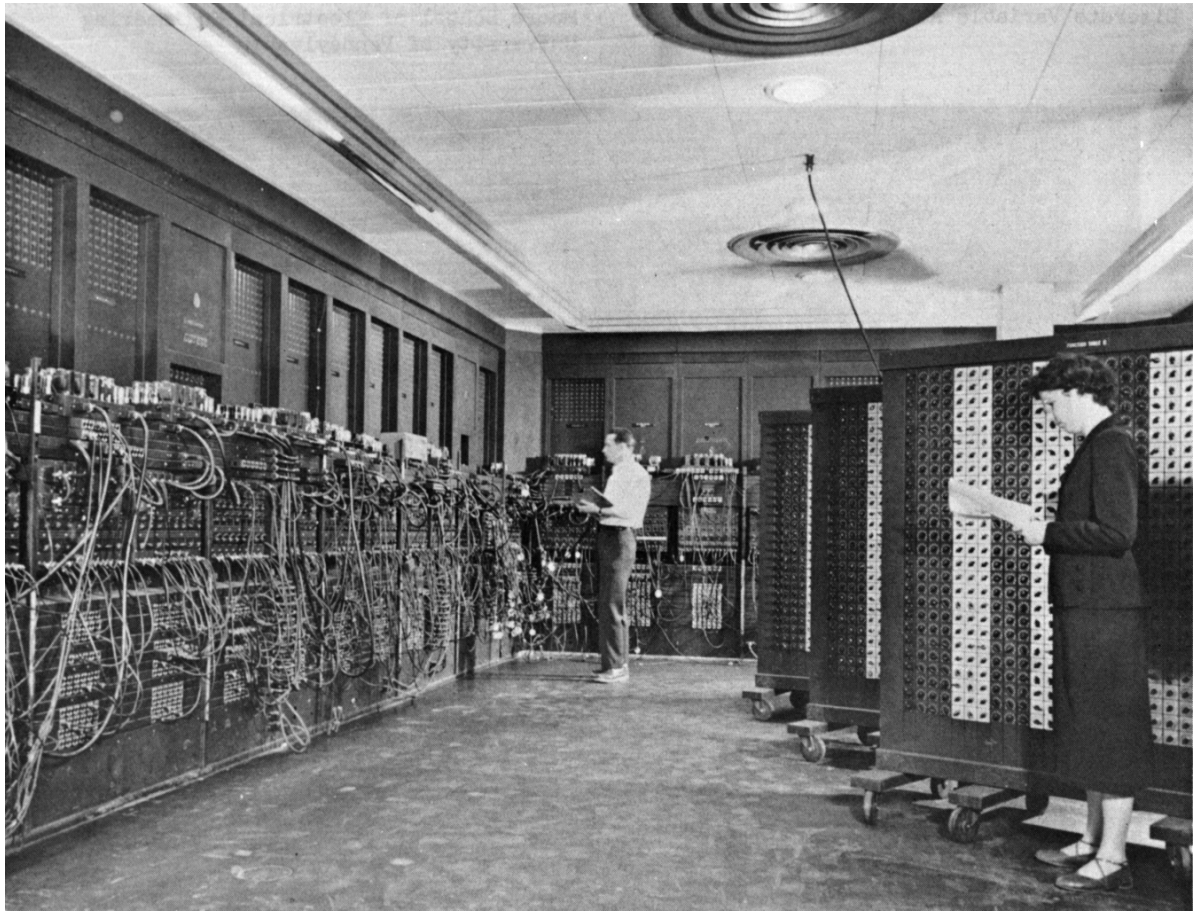


HAW HAMBURG

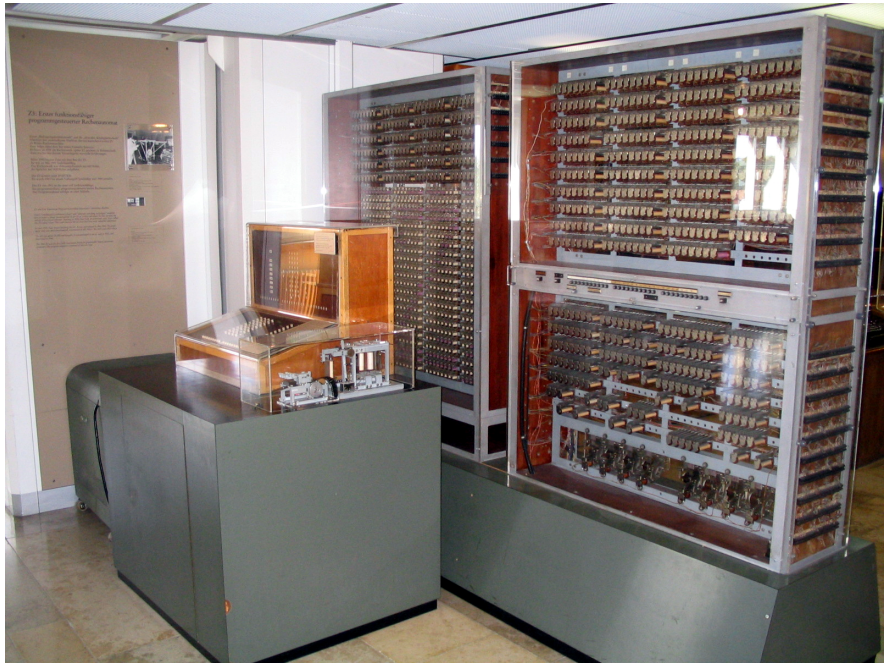
Geschichte der Betriebssysteme

Einführung

Die erste Generation (1945 – 1955): Röhren und Steckkarten



ENIAC



Zuse Z3

Von Venusianer, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=3632073

Die zweite Generation (1955 – 1965): Transistoren und Stapelverarbeitung

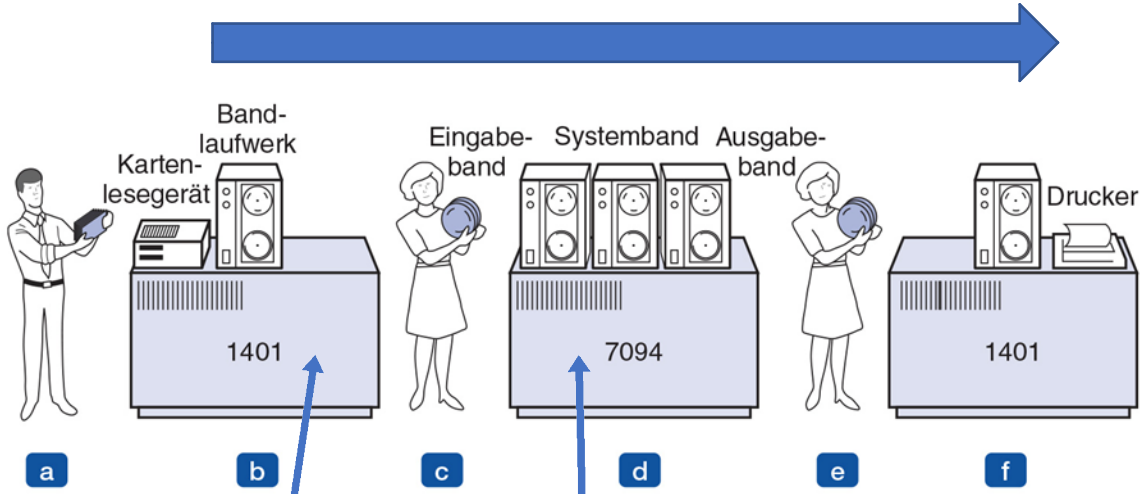


Abbildung 1.3: Ein frühes Stapelverarbeitungssystem. (a) Die Programmierer bringen die Stapel zur 1401. (b) Die 1401 liest den Stapel von Jobs auf ein Band. (c) Ein Operator trägt das Eingabeband zur 7094. (d) Die 7094 führt die Berechnung durch. (e) Ein Operator trägt das Ausgabeband zur 1401. (f) Die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

Vorrechner

Großrechner

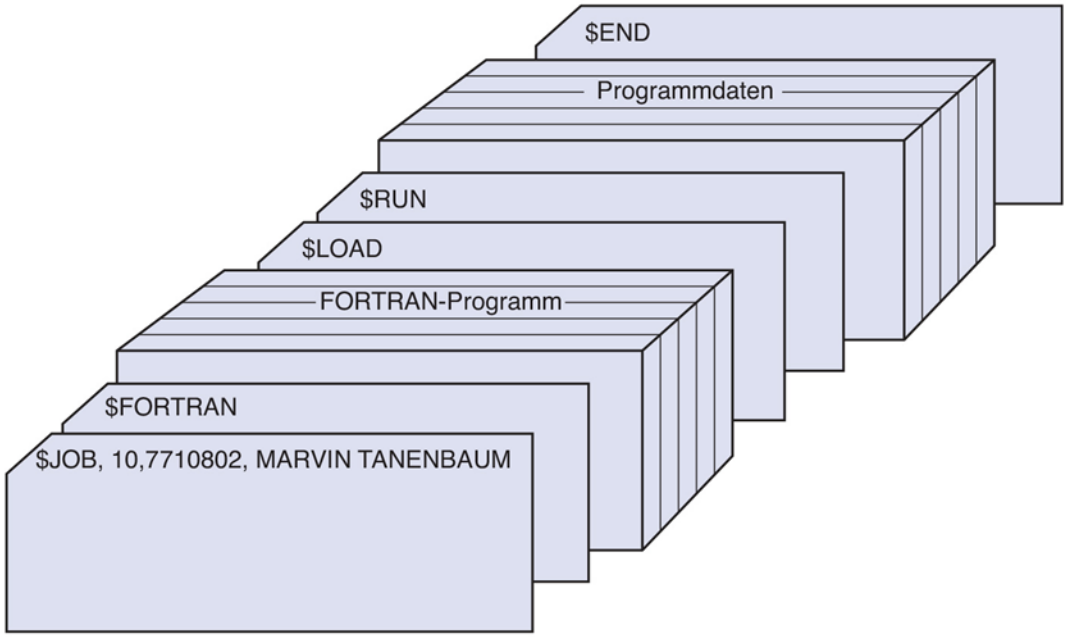


Abbildung 1.4: Struktur eines typischen FMS-Jobs.

Abb. aus [TB]

Die dritte Generation (1965 – 1980): ICs und Mehrprogrammbetrieb

HAW HAMBURG

- Integrierte Schaltkreise
- Parallele Jobs auf einem System?
- IBM System/360 Serie
  - verschiedene CPUs
  - gleichartige Architektur
- Erstes BS: OS/360
- Time Sharing Systeme
  - CTSS (MIT)
  - MULTICS

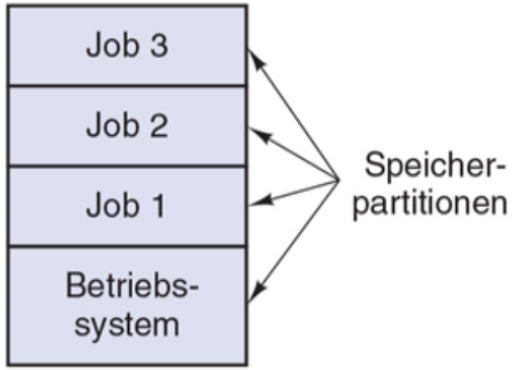
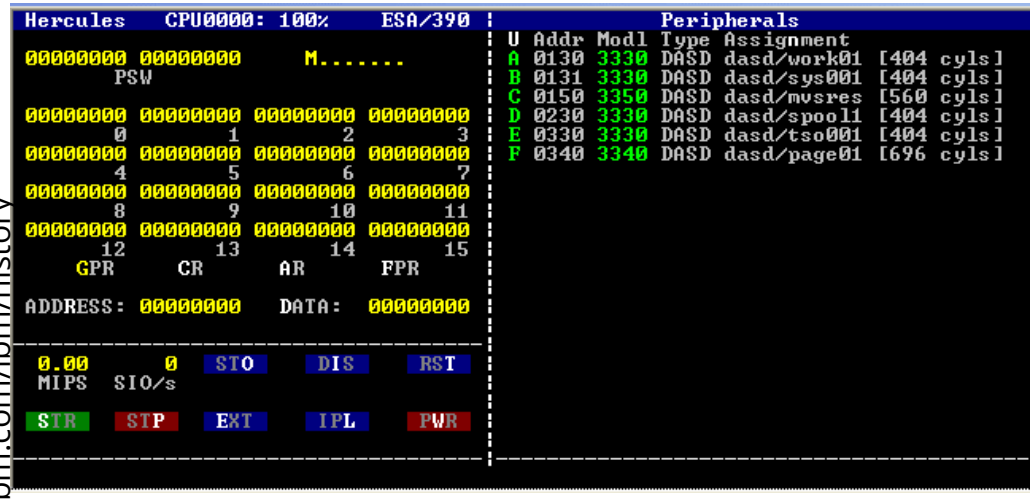


Abb. aus [TB]

Quelle: http://www-03.ibm.com/ibm/history



OS7360 auf Hercules-Emulator [HE]

Die vierte Generation (seit 1980): Personal Computer

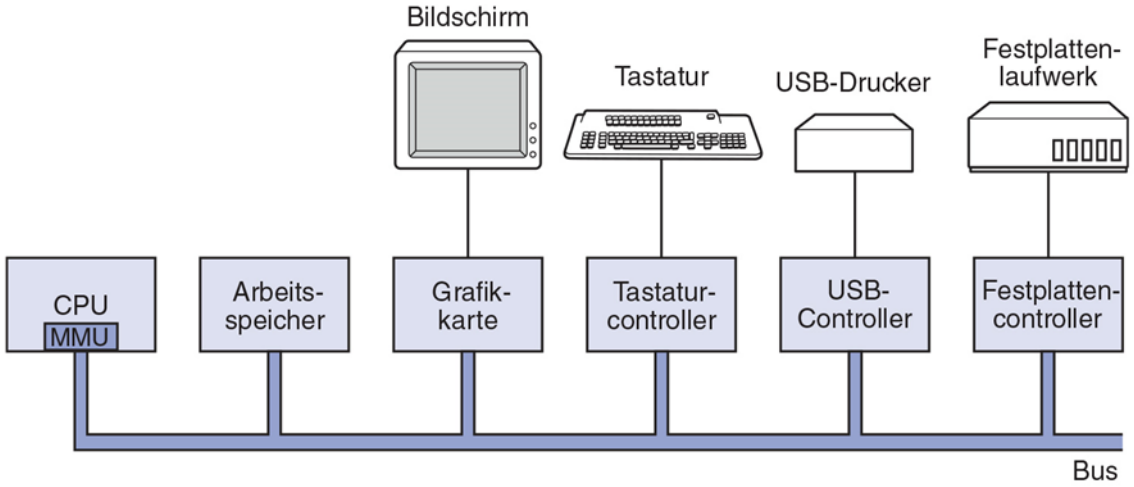
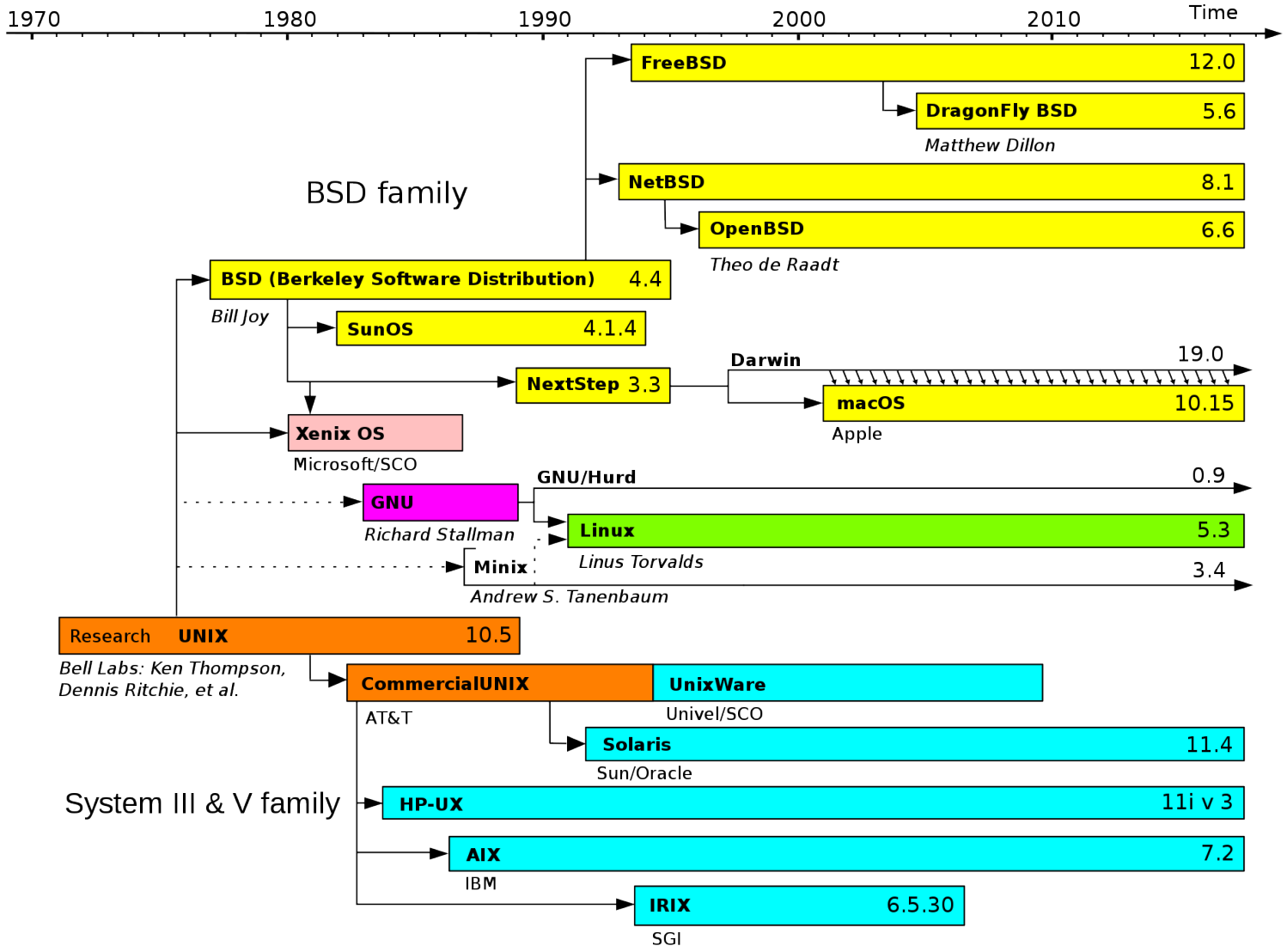


Abbildung 1.6: Einige Komponenten eines einfachen PCs.

Abb. aus [TB]

- Hochintegrierte Schaltkreise mit Mio. Transistoren
- Allzweck CPUs, beginnend mit Intel 8080
- CP/M
- DOS, später MS-DOS
- Windows, Mac OS
- Linux, Unix

Entwicklung unixoider Betriebssysteme



Die fünfte Generation (seit 1990)

HAW HAMBURG

SYMBIAN

android

- PCs (→ 4. Generation)
- High-Performance-Computing (HPC)
- Mobile Geräte
- Ubiquitous computing

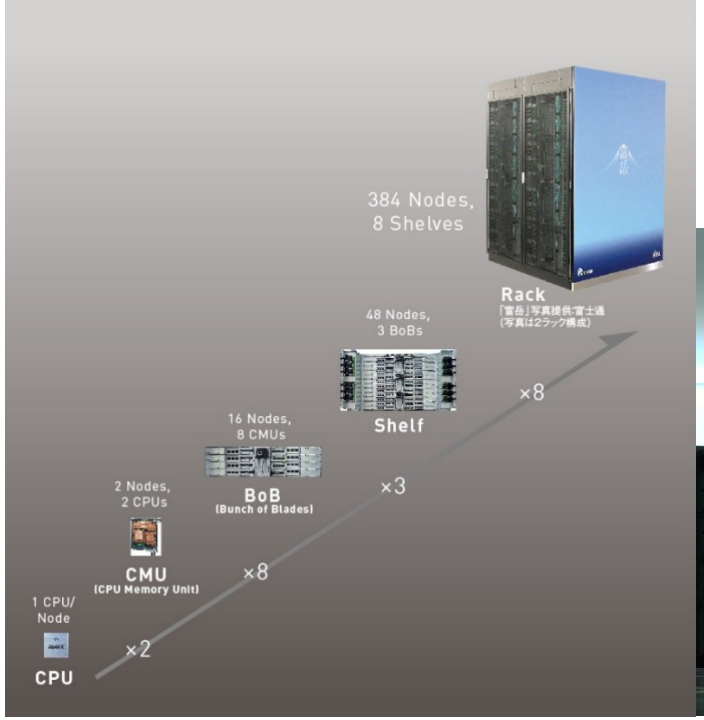


Abb. https://www.r-ccs.riken.jp/en/fugaku/

Betriebssystem-Konzepte

Einführung

Wiederholung: Computerhardware

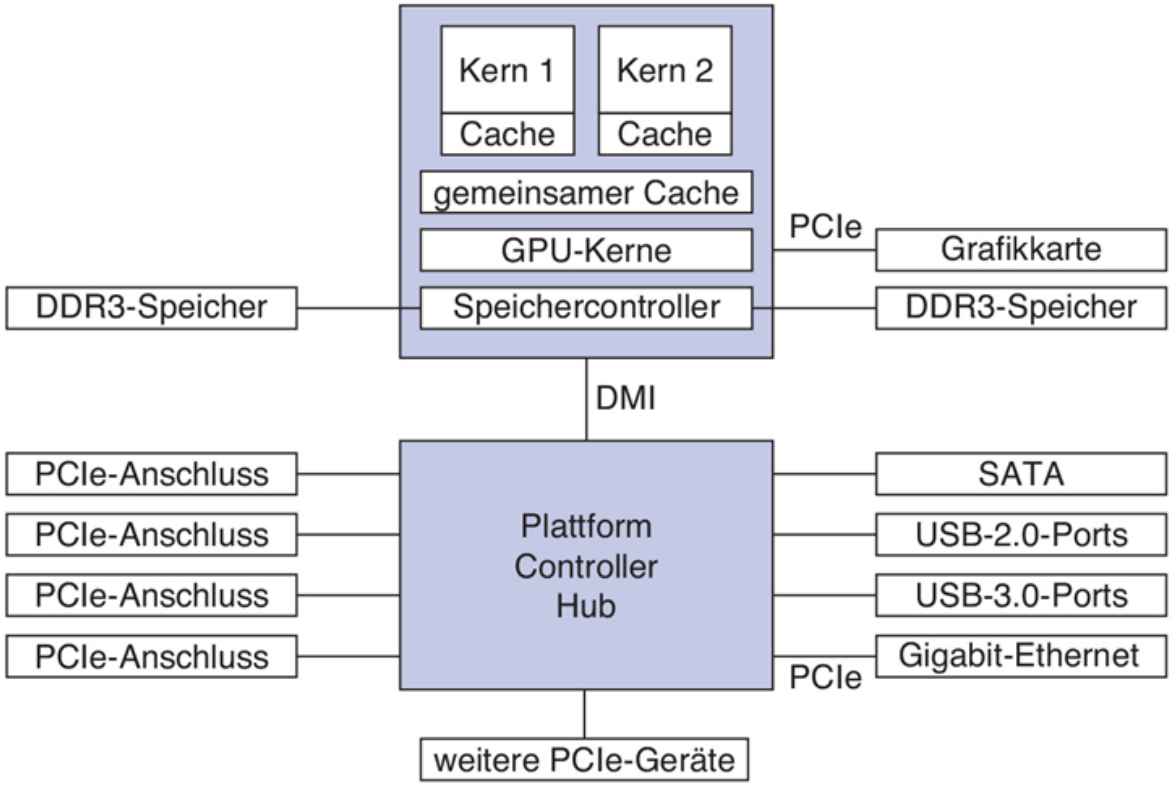
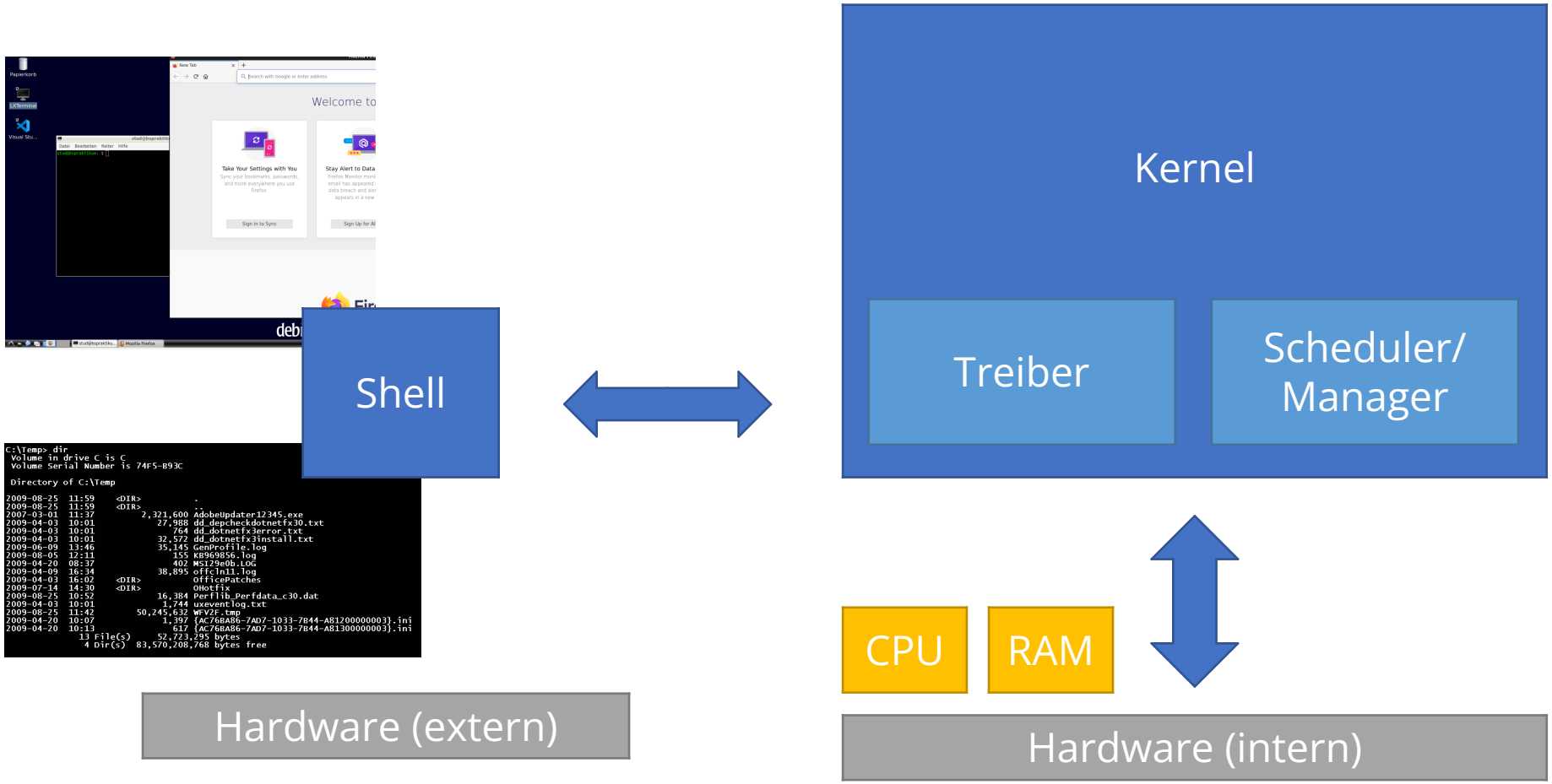


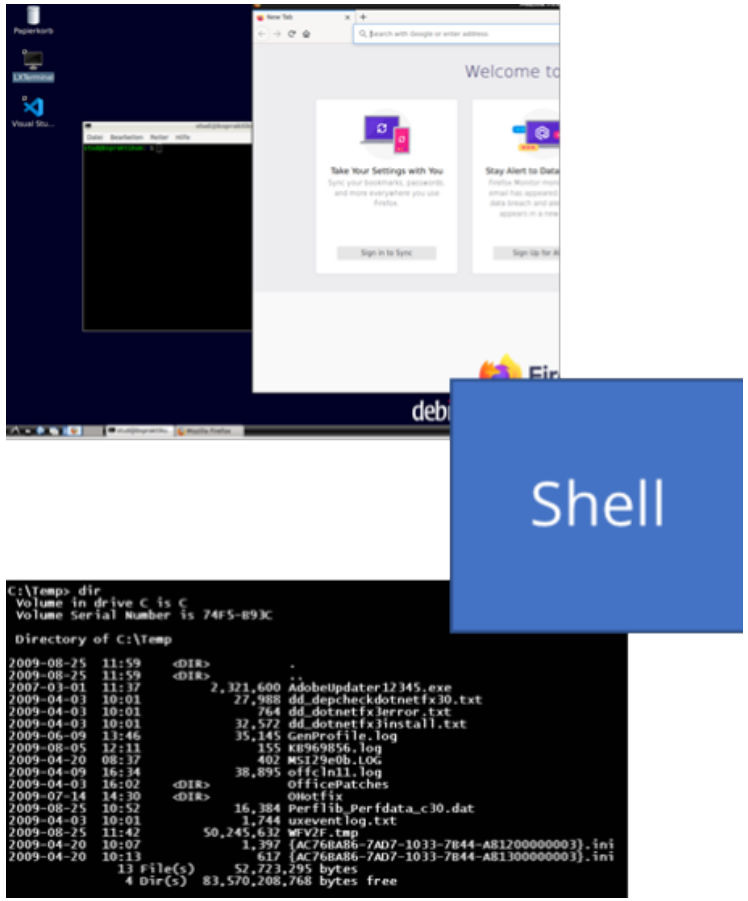
Abbildung 1.12: Der Aufbau eines ausgebauten x86-Systems.

Abb. aus [TB]

Bestandteile des Betriebssystems



Die Shell



Benutzerschnittstelle

- Grafisch (GUI)
- Zeilenbasiert (Kommandozeileninterpreter, Konsole)

Nimmt Kommandos des Benutzers entgegen

Startet und stoppt Prozesse auf Anweisung

Prozesse

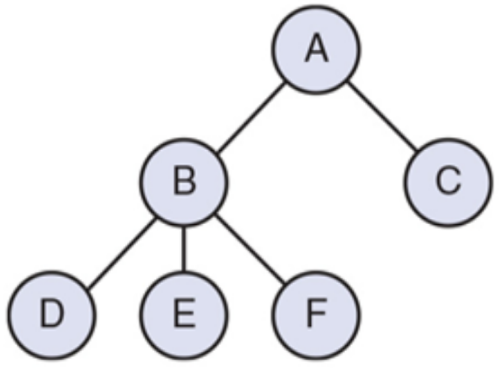


Abb. aus [TB]

Prozess

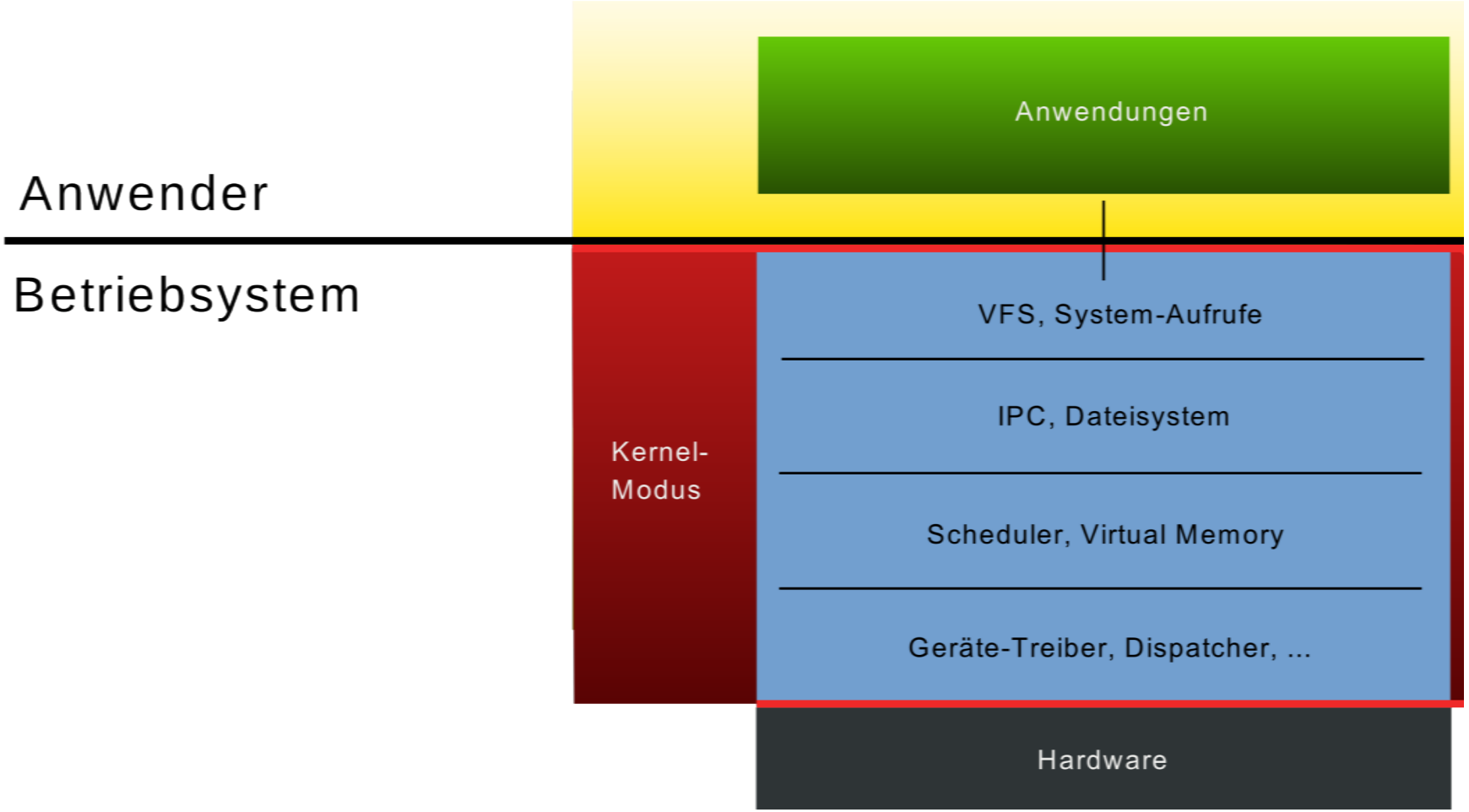
- Programm wurde in den Speicher geladen
- Programm ist ausführbar
- **Addressraum**

Prozesse können wiederum andere Prozesse starten
→ **Prozessbaum** entsteht

Prozesse können mit dem Kernel über **Systemaufrufe** kommunizieren

Prozesse können untereinander kommunizieren → **Interprozesskommunikation** (IPC)

Kernel


Anwender
Betriebsystem
Anwendungen
Kernel-
Modus
VFS, System-Aufrufe
IPC, Dateisystem
Scheduler, Virtual Memory
Geräte-Treiber, Dispatcher, ...
Hardware

Dateien

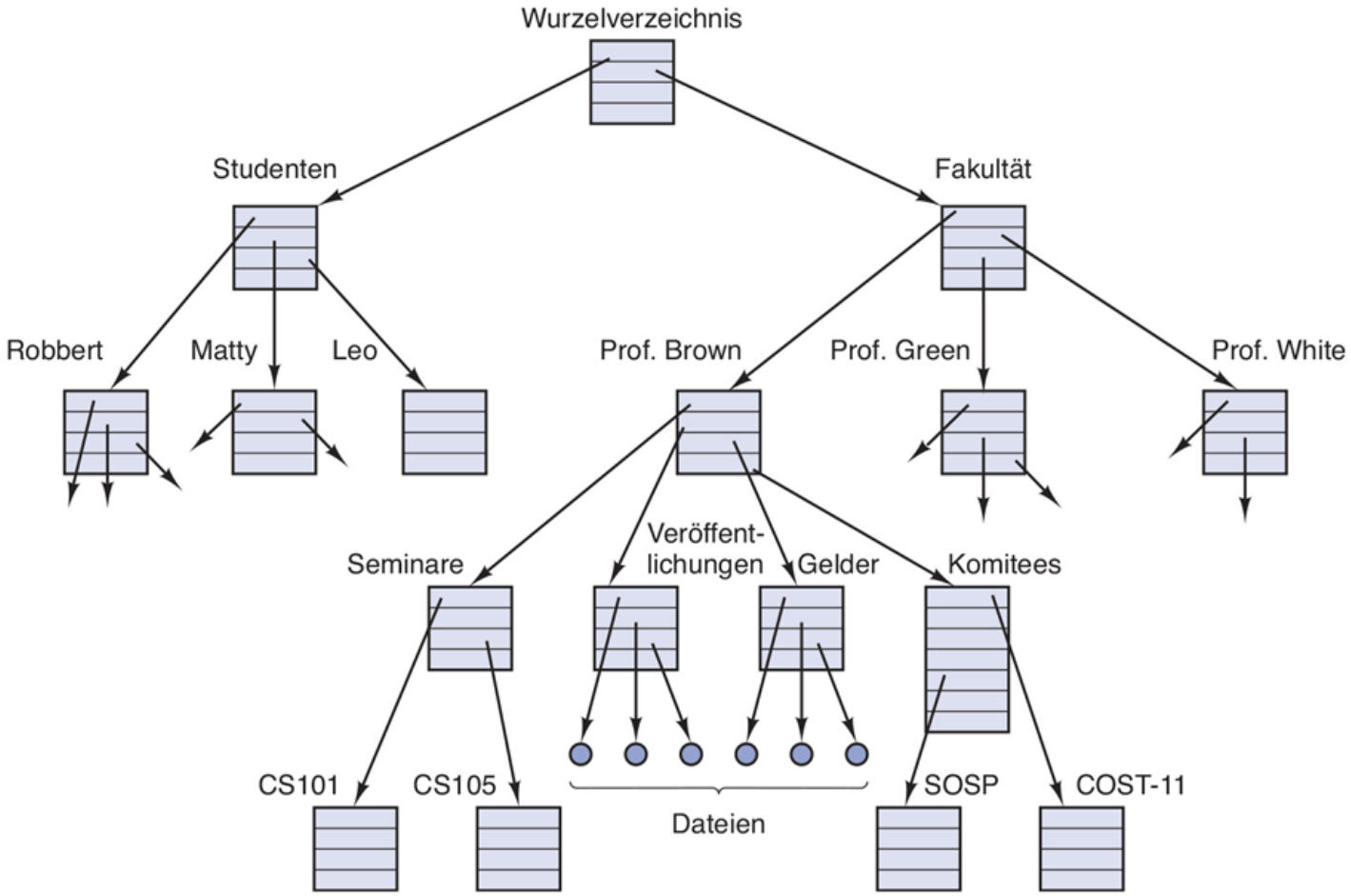


Abbildung 1.14: Das Dateisystem einer Fakultät in einer Universität.

Abb. aus [TB]

Dateien und Prozesse

Prozesse können über Pseudodateien kommunizieren

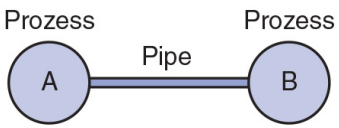


Abbildung 1.16: Zwei Prozesse, die durch eine Pipe verbunden sind.

Abb. aus [TB]

Einem Prozess ist immer ein **Arbeitsverzeichnis** zugewiesen.

Prozesse können durch einen **Systemaufruf** ihr Arbeitsverzeichnis ändern.

Systemaufrufe

```
...
read(datei, meinPuffer, pufferLen);
```

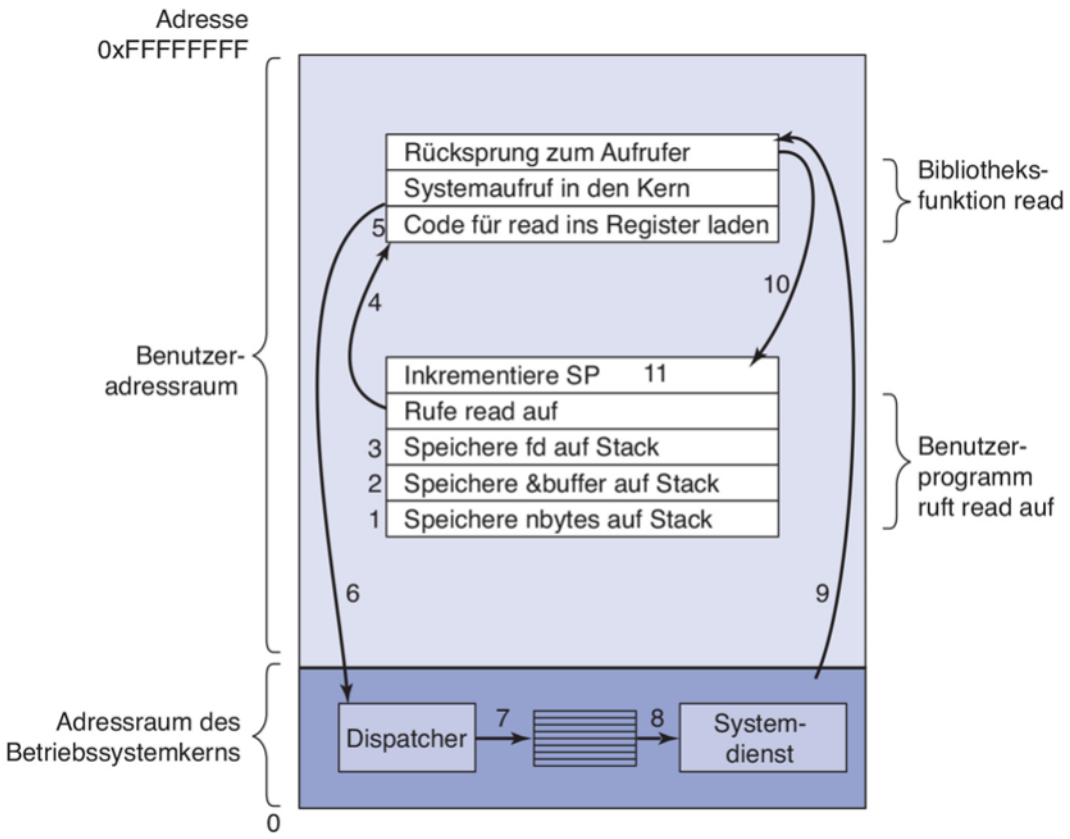


Abbildung 1.17: Die elf Schritte für den Systemaufruf `read(fd, buffer, nbytes)`.

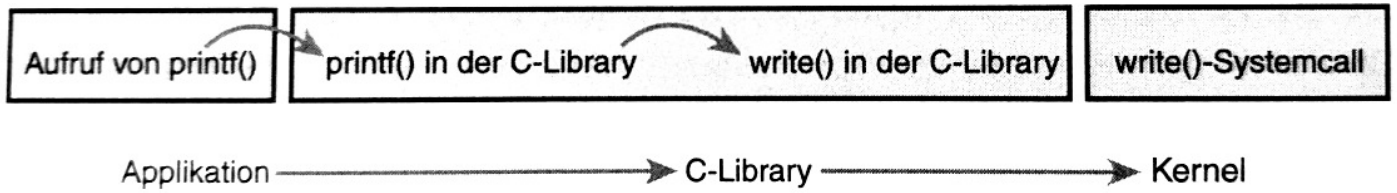
Abb. aus [TB]

Systemaufrufe

- abstrahieren Anforderungen an die Hardware/System
- Sicherheitsfunktion

Nur Systemaufrufe dürfen in den Kernmodus wechseln

Syscalls



Auf x86-CPUs:

```
mov $6,  %eax     ; close() ist Systemaufruf 6
mov $15, %ebx     ; Dateideskriptor als erstes Argument
int $0x80         ; Softwareinterrupt auslösen
```

POSIX-Syscalls

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

Abb. aus [TB]

POSIX-Syscalls

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

Abb. aus [TB]

Win32-API

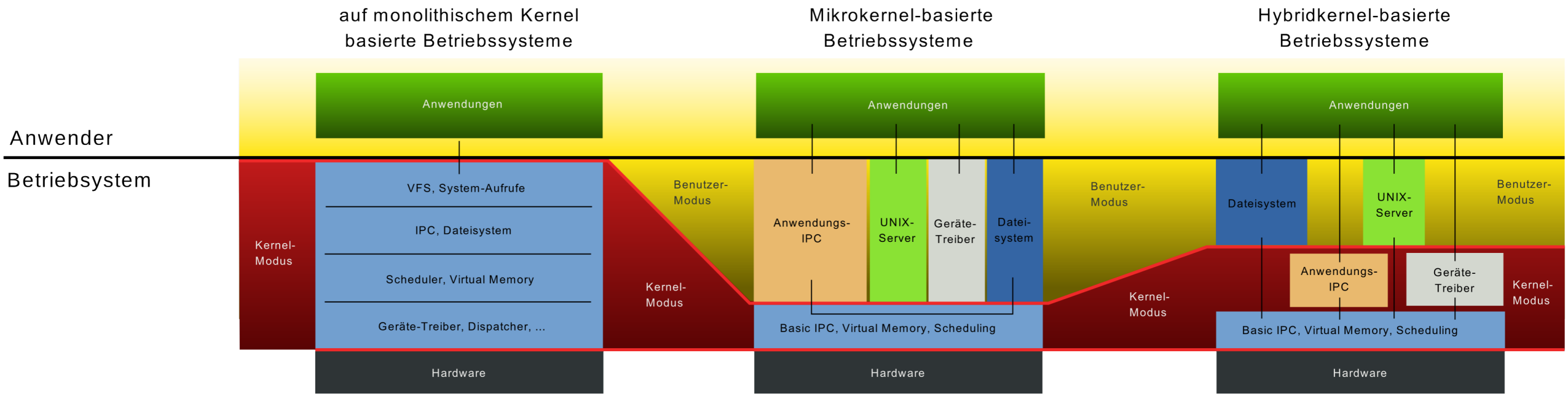
UNIX	Win32	Beschreibung
fork	CreateProcess	Erzeugen eines neuen Prozesses
waitpid	WaitForSingleObject	Warten auf das Ende eines Prozesses
execve	(nicht vorhanden)	CreateProcess = fork + execve
exit	ExitProcess	Ausführung beenden
open	CreateFile	Erzeugen einer Datei oder Öffnen einer existierenden Datei
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

Abb. aus [TB]

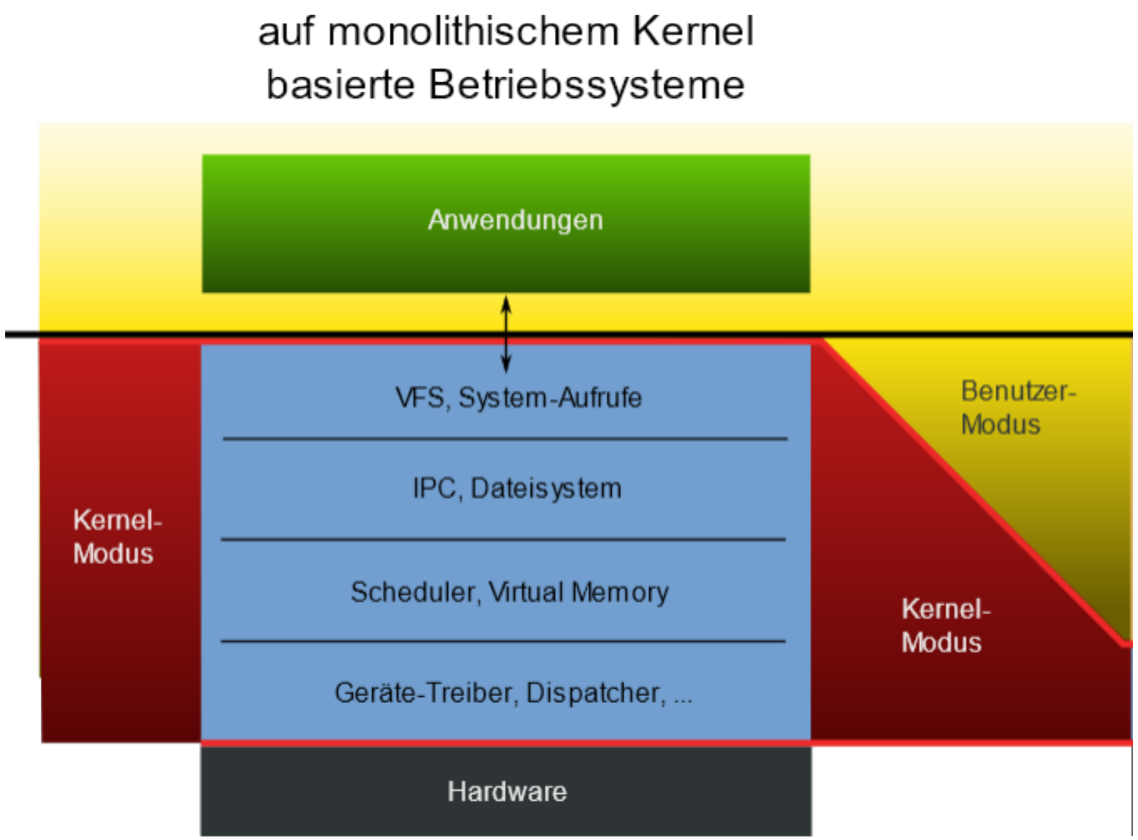
Betriebssystem-Strukturen

Einführung

Übersicht Betriebssystem-Strukturen



Monolithische Systeme



Das Betriebssystem ist „ein Programm“ im Kernmodus

Vielfältige Abhängigkeiten der Funktionen

Vor- und Nachteile?

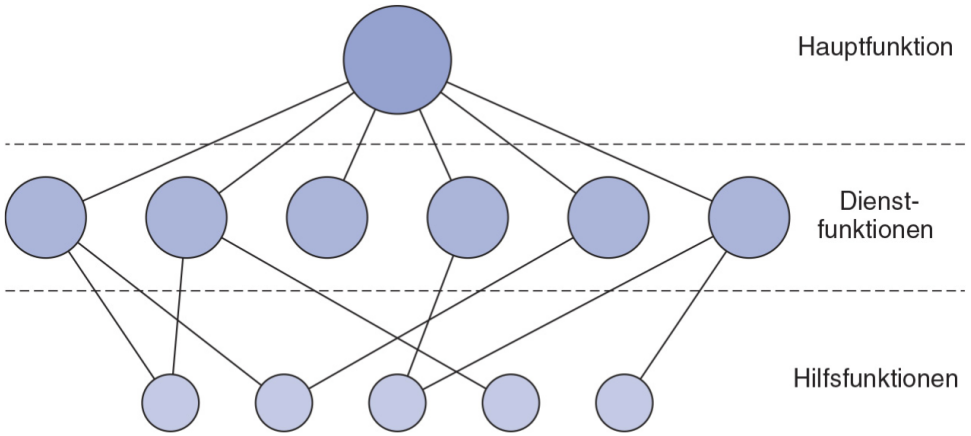
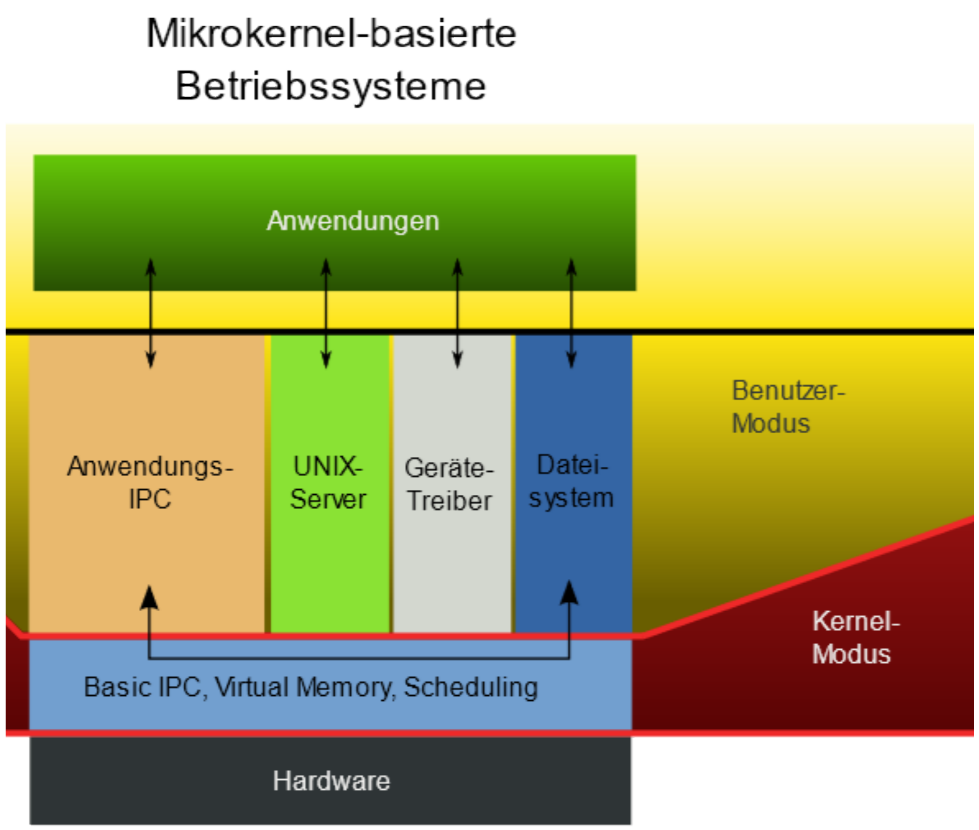


Abbildung 1.24: Ein einfaches Strukturmodell für ein monolithisches System.

Abb. aus [TB]

Mikrokerne



Modularer Aufbau, vor allem im Benutzermodus

Kern enthält nur minimal notwendige Funktionen

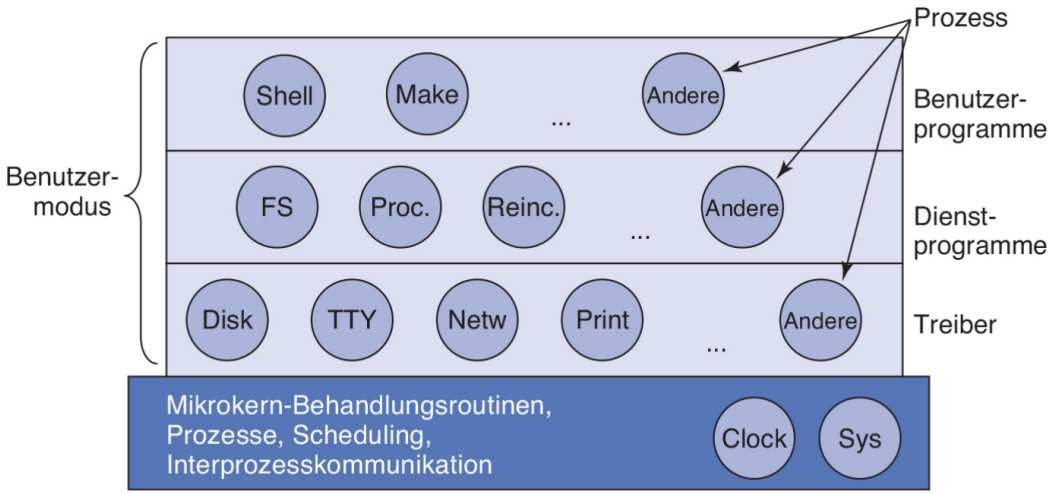


Abbildung 1.26: Vereinfachte Struktur des MINIX-Systems.

Abb. aus [TB]

Client-Server-Modell

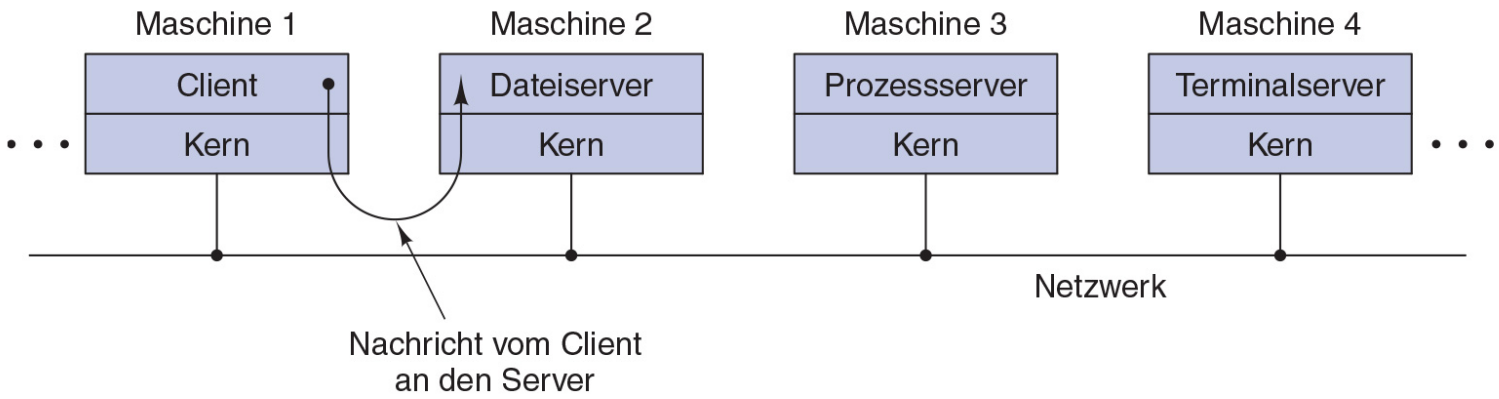


Abbildung 1.27: Das Client-Server-Modell über einem Netzwerk.

Abb. aus [TB]

Microkernel mit Diensten über Netzwerk

Geeignet z. B. für große Cluster-Rechner

Virtuelle Maschinen

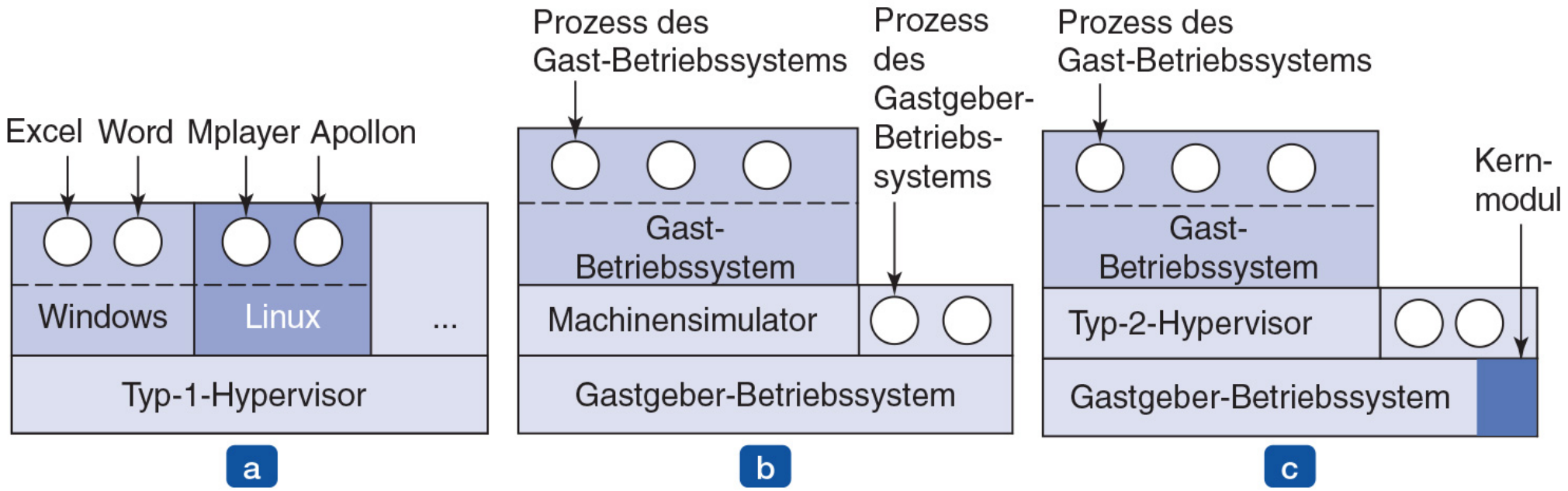


Abbildung 1.29: (a) Typ-1-Hypervisor; (b) reiner Typ-2-Hypervisor; (c) Typ-2-Hypervisor in der Praxis.

Abb. aus [TB]

Wie sieht es in der Realität aus?

Microkernel

Minix

GNU/Mach

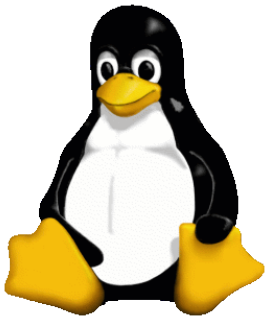
Hybride Kernel

Darwin

Monolithen

Unix/BSD

Linux



Einführung

Ursprünge von Linux

- 1991 als Terminal Emulator gestartet
  - Linus Torvalds wollte die Unix-Server der Universität Helsinki nutzen
  - Unzufrieden mit MINIX und DOS
- Zunächst für den Intel 80386 Prozessor
- Freie Software unter GPLv2 Lizenz
- Schnelle Verbreitung und Weiterentwicklung mit vielen Freiwilligen
- Newsgroup comp.os.minix

Linus Torvalds

Von Unknown photographer who sold rights to the picture to linuxmag.com - Linuxmag.com; The image is from an article in a December 2002 issue of Linux Magazine[1], CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17991

Überblick Linux-System

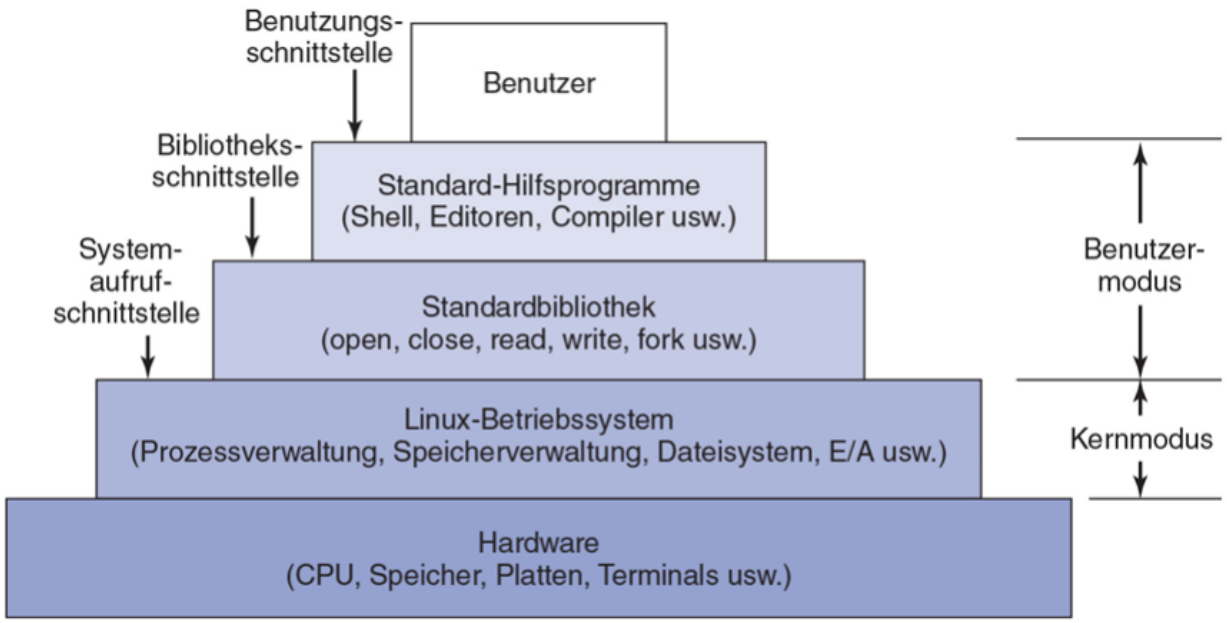
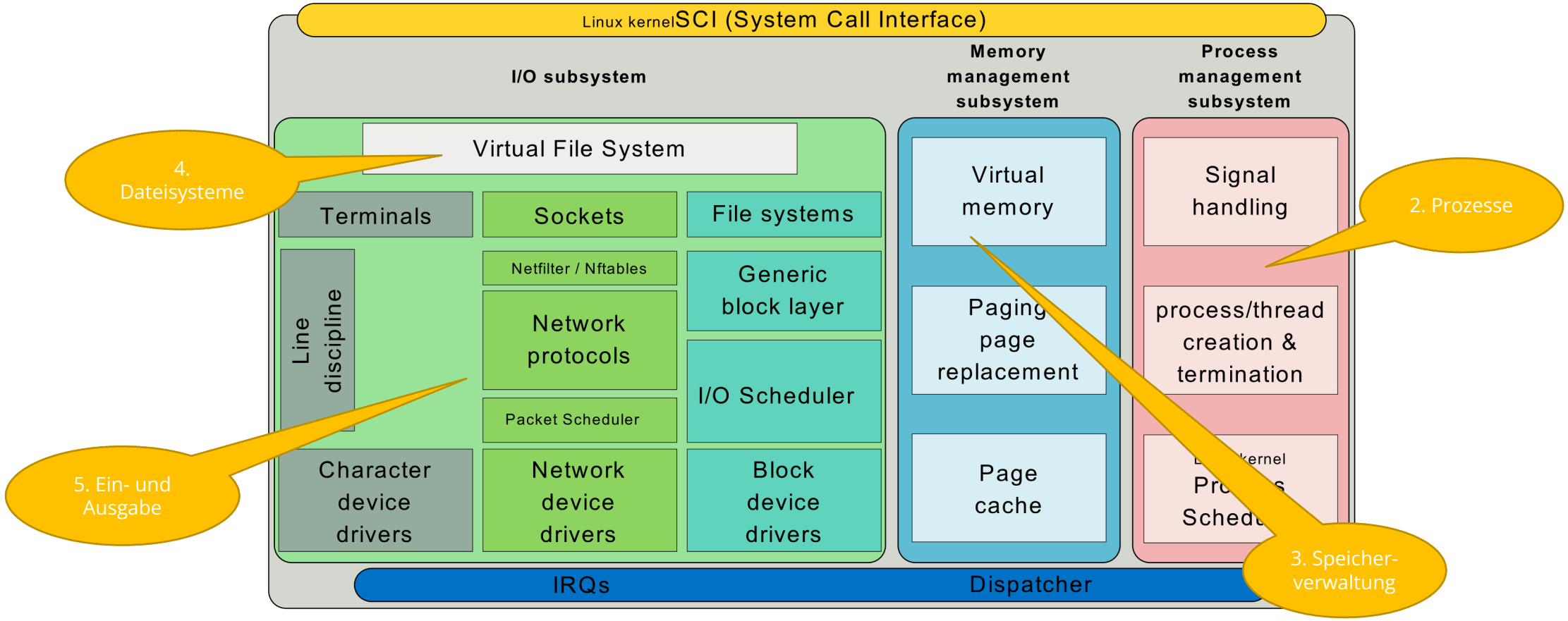


Abbildung 10.1: Die Schichten in einem Linux-System.

Abb. aus [TB]

- Linux ist <u>kein</u> „echtes“ Unix, sondern Unix-Klon
- Quasi für jede Plattform verfügbar
- Linux ist eigentlich ein Kernel
- Linux-Distribution = komplettes System
  - Red Hat/Fedora
  - SuSE
  - Debian (Ubuntu, Mint, ...)
  - diverse andere

Überblick Linux-Kernel



ScotXW at English Wikipedia
CC BY SA 4.0

Linux Kernel Development Community

HAW HAMBURG

- Aktuelle Linux-Sourcen unter www.kernel.org
- Kommunikation der Community läuft über die *Linux-Kernel Mailing List* (lkml)
- Codebase ist aufgeteilt in Subsysteme
  - Jedes Subsystem hat eine*n Maintainer*in
  - Hierarchische Struktur → Linus Torvalds an der Spitze
- Jede*r kann einen Patch submitten

Rüstzeug für das Praktikum (1)

Einführung

Was Sie benötigen

HAW HAMBURG

- VirtualBox VM
  - Vorbereitetes Image → Link in EMIL
  - Lesen Sie die beiligende README
- Alternativ: natives Linux-System auf Ihrem Rechner

- Aufgabe 1 im Praktikum
  - Kennenlernen Linux und wichtiger Kommandozeilenwerkzeuge
  - Einführung in die Bash-Programmierung

Bash, die *Bourne-again shell*

https://de.wikibooks.org/wiki/Linux-Praxisbuch/_Shellprogrammierung

Live-Demo jetzt ☺

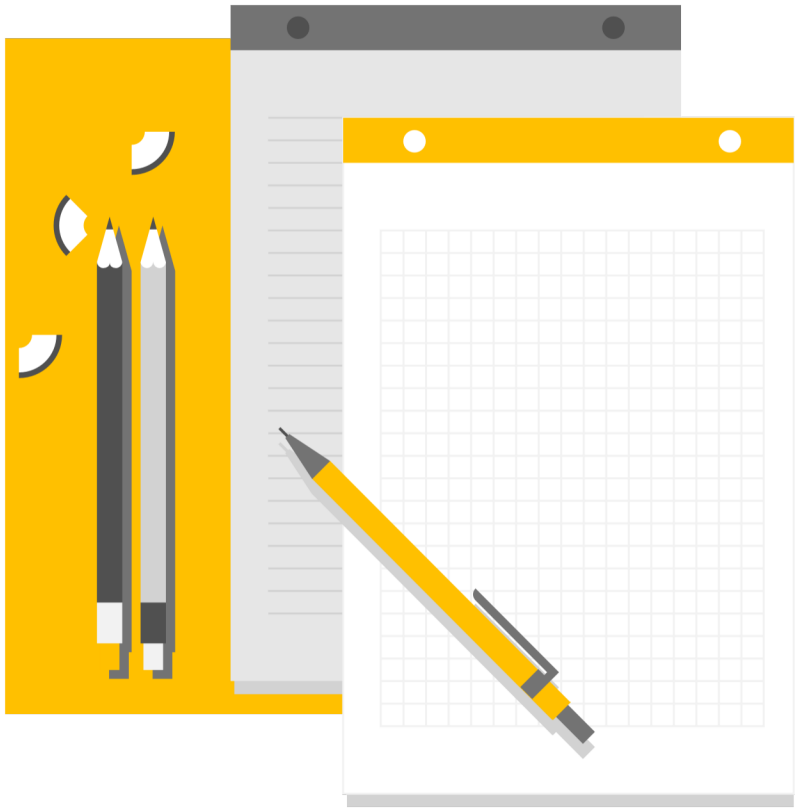
Zusammenfassung

Einführung

Ein kleiner Test zum Schluss…

In EMIL

https://emil.haw-hamburg.de/mod/quiz/view.php?id=3499004



Zusammenfassung

HAW HAMBURG

Wir haben heute…

- die **Aufgaben eines Betriebssystems** kennengelernt,
  - u.a. Abstraktion und Management der Hardware-Ressourcen
- heutige BS in den **historischen Kontext** eingeordnet und
- die wesentliche **BS-Konzepte und -Strukturen** erläutert:
  - Bestandteile eines Betriebssystems: Dateien, Prozesse, Shell, Systemaufrufe…
  - Monolithische, hybride Kernel oder Microkernel
- Wir haben **Linux** und die **Bash** kennengelernt und
- besitzen damit das nötige Rüstzeug für die erste Praktikumsaufgabe.

Literatur und Links

[TB 2017] Tanenbaum; Bos: Moderne Betriebssystem, Pearson 2017

[W1] Wikipedia-Artikel „Linux kernel“ https://en.wikipedia.org/w/index.php?title=Linux_kernel

[HE 2017] Herclues Emulator, http://www.hercules-390.eu/ mit OS/360 http://www.ibiblio.org/jmaynard/

Prozesse

Next time on…