UNIVERSITÄT BERN

# 2405 Betriebssysteme
# II. Aufgaben und Strukturen

Thomas Staub, Markus Anwander
Universität Bern

# Inhalt

1. Systemkomponenten
2. Dienste und Funktionen
   1. Dienste
   2. Funktionen
      1. Ressourcenzuweisung, Accounting, Sicherheit
      2. Schutzmechanismen
         1. Dual-Mode Operation
         2. E/A-Schutz
         3. Speicherschutz
         4. CPU-Schutz
3. Schnittstelle zwischen Anwendungen und Betriebssystem
   1. Systemaufrufe
   2. Systembibliotheken
   3. Systemprogramme
4. Betriebssystemarchitekturen
   1. Monolithische Systeme
      1. MS-DOS
      2. UNIX
   2. Geschichtete Systeme und Abstrakte Maschinen
   3. Mikrokerne
   4. Module
   5. Ereignisgesteuerte Betriebssysteme
   6. Virtuelle Maschinen
   7. Hybride Systeme
      1. Linux
      2. Windows
      3. Mac OS X
      4. iOS
      5. Android

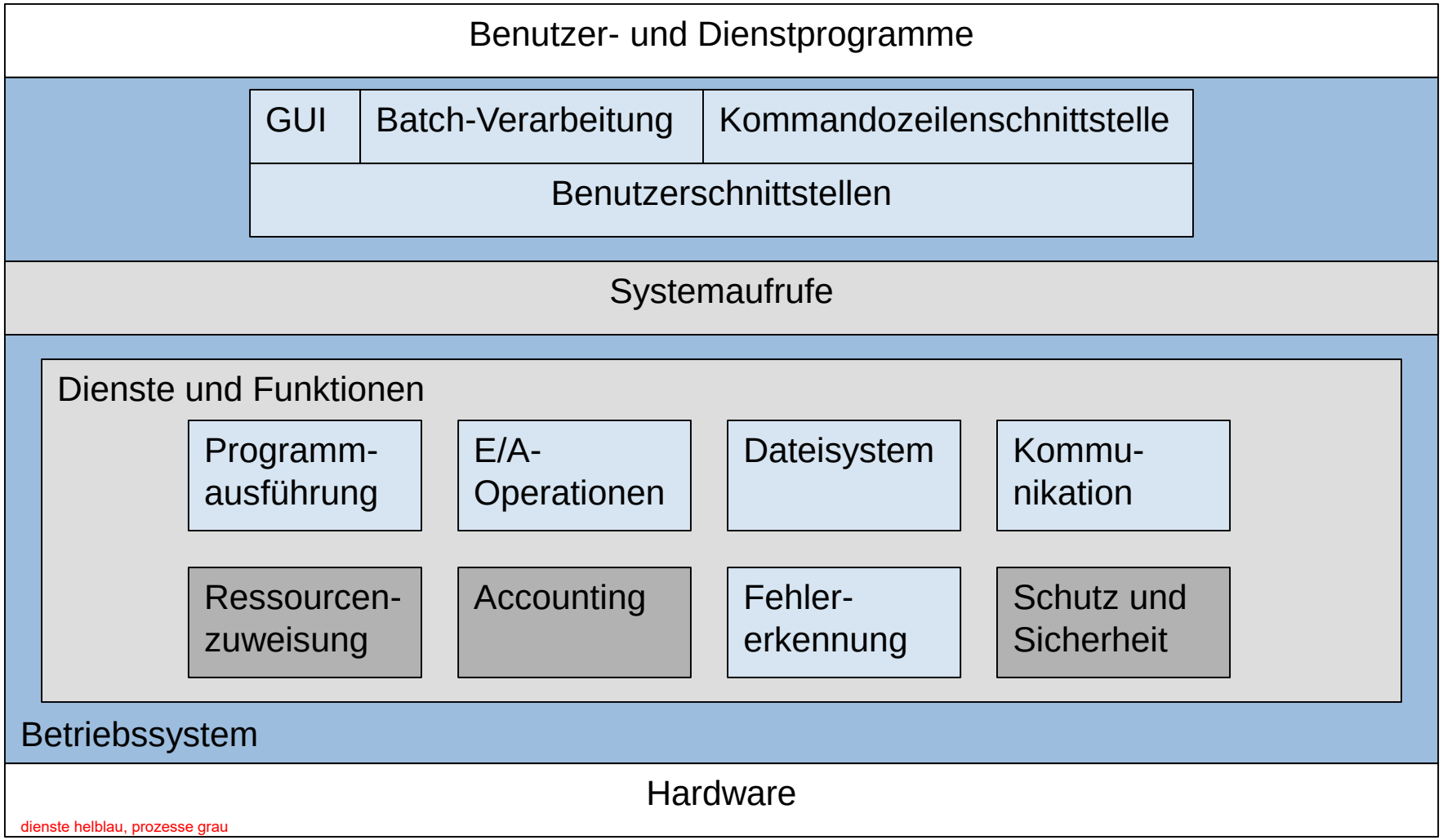
# 1.1 Systemkomponenten

- Prozessverwaltung
  - Erzeugen, Löschen, Ausführen, Anhalten und Weiterführen von Prozessen
  - Synchronisation von Prozessen
  - Kommunikation zwischen Prozessen
  - Behandlung von Verklemmungen
- Hauptspeicherverwaltung
  - Belegen und Freigabe von Hauptspeicher
  - Zuordnung des Hauptspeichers zu Prozessen
- Ein-/Ausgabe-System
  - Speicherverwaltung (Puffer, Caches, Spooling)
  - allgemeine Gerätetreiberschnittstelle
  - Treiber für spezielle Hardware-Geräte
- Kommunikationssystem
  - s. Vorlesung Computernetze

# 1.2 Systemkomponenten

- Sekundärspeicherverwaltung
  - Dateisystem-Verwaltung
    - Erzeugen, Öffnen, Schliessen, Lesen, Schreiben, Löschen von Dateien und Verzeichnissen
    - Abbildung von Dateien auf Sekundärspeicher
    - Sicherung auf nicht-flüchtigen Speichermedien
  - Massenspeicherverwaltung
    - Auslagern von Daten im Hauptspeicher auf Sekundärspeicher
    - Zuweisen von Speicherplatz
    - Freispeicherverwaltung
    - Scheduling bei Festplatten
  - Caching
- Schutz- und Sicherheitssysteme
  - Zugriffskontrolle auf System- oder Benutzerressourcen
- Kommando-Interpreter und grafische Benutzerschnittstellen

# 2. Dienste und Funktionen



# 2.1 Dienste

→ für Programme und Benutzer
→ zur Vereinfachung der Programmierung

- Benutzerschnittstelle
  - Command Line Interface
  - Grafische Benutzerschnittstelle
- Programmausführung
  - Laden und Beenden von Programmen
- Ein-/Ausgabe-Operationen
  - Datei- oder Gerätezugriff
  - Abwicklung über Systemaufrufe
- Dateisystem-Manipulationen
  - Erzeugen, Verändern und Löschen von Dateien und Verzeichnissen
- Kommunikation zwischen Prozessen
  - Implementierung über gemeinsamen Speicher oder Nachrichtenaustausch
- Fehlererkennung
  - Abfangen von Hardware- oder Programmfehlern

# 2.2.1 Funktionen



→ zur effizienten Ausführung des Systems, speziell bei mehreren Benutzern

- Zuweisen von Ressourcen
  - CPU
  - Hauptspeicher
  - Sekundärspeicher
  - Ein-/Ausgabe-Geräte
- Accounting
  - Statistiken zur Systemoptimierung oder fairer Ressourcennutzung
- Schutzmechanismen
  - Vermeiden von Wechselwirkungen zwischen Prozessen
- Sicherheitsmechanismen
  - Authentifizierung
  - Verschlüsselung

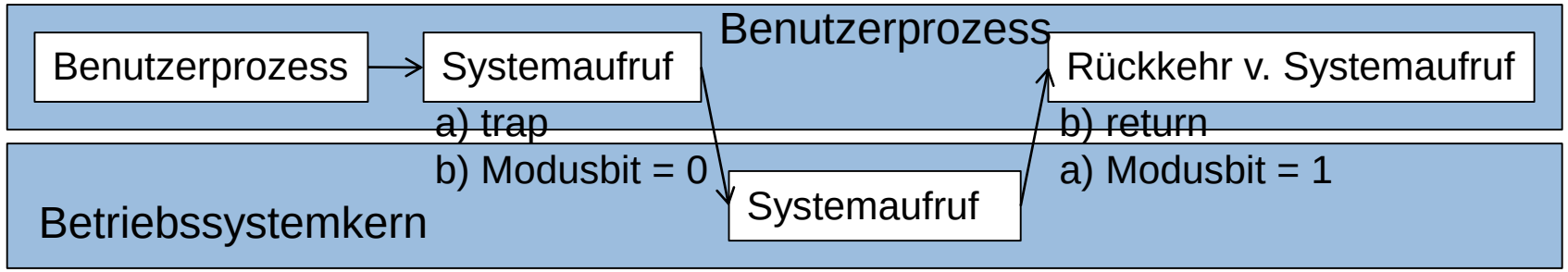
# 2.2.2 Schutzmechanismen

- Schwerwiegende Folgen durch Programmierfehler in Anwendungen, speziell in Mehrprogrammsystemen
- Fehlererkennung häufig durch Hardware, Fehlerbehandlung durch Betriebssystem interrput wird geworfen, der dann vom OS behandelt wird
- Schutzmechanismen
  - Dual-Mode Operation
  - E/A-Schutz
  - Speicherschutz
  - CPU-Schutz

# 2.2.2.1 Dual-Mode Operation

- Dual-Mode Operation
  - Benutzermodus für Anwendungen (1)
  - Systemmodus für Betriebssystem (0)
    - Ausführung privilegierter Instruktionen nur im Systemmodus
    - Ausführung privilegierter Instruktionen im Benutzermodus verursacht Systemaufruf (Trap)
- Wechsel zwischen Benutzer- und Systemmodus
  - Benutzermodus → Systemmodus
    - durch Hardware nach Systemaufruf (Trap) oder Interrupt
  - Systemmodus → Benutzermodus
    - durch Rücksetzen des Modusbits vor Rückkehr in Benutzerprozess

priv Instruktionen können nicht von Anwendungen asugeführt werden




trap ist softwaregesteurter interrput

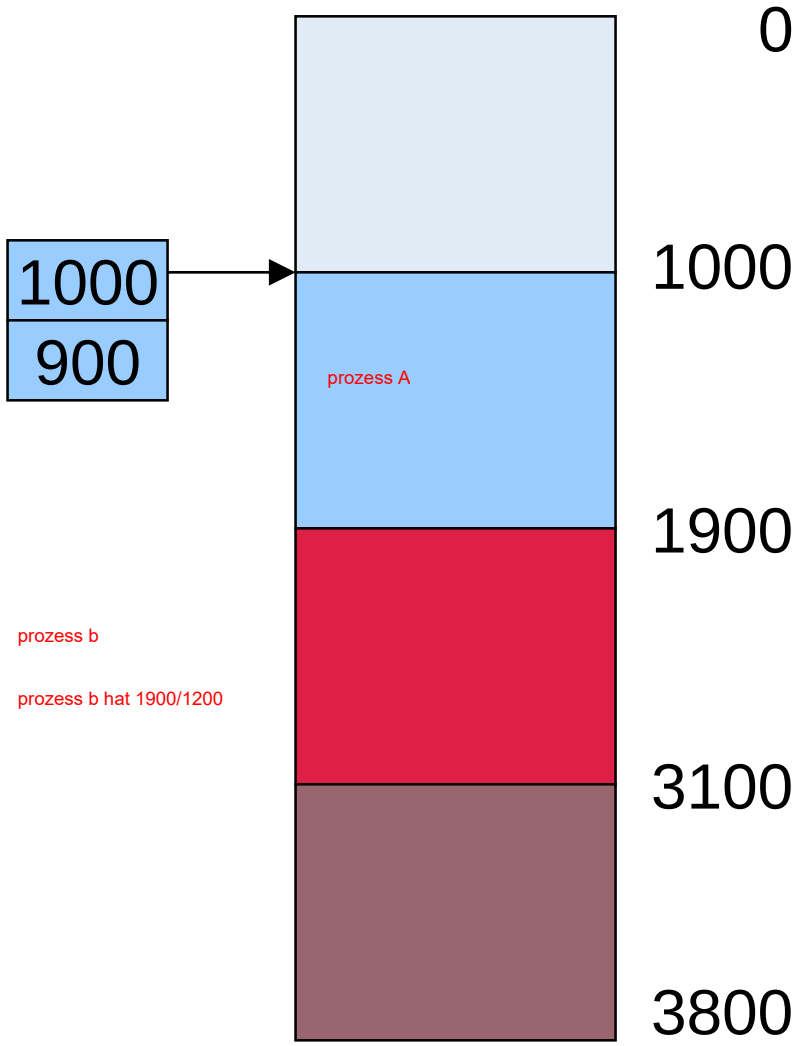
# 2.2.2.2 E/A-Schutz

Ein/Ausgabe

- E/A-Instruktionen sind privilegiert und können nur vom Betriebssystem aufgerufen werden.
- Ausführen der E/A-Operationen durch Anwendungen über Systemaufrufe

# 2.2.2.3 Speicherschutz

- Schutz der Prozesse untereinander vor inkorrektem Speicherzugriff
- Beschreibung des zulässigen Speicherbereichs durch Basis- (kleinste Adresse) und Limit-Register (Grösse des Speicherbereichs)
- Vergleich von Speicheradressen mit Registern durch CPU-Hardware
- Laden der Register durch Betriebssystem über privilegierte Instruktionen



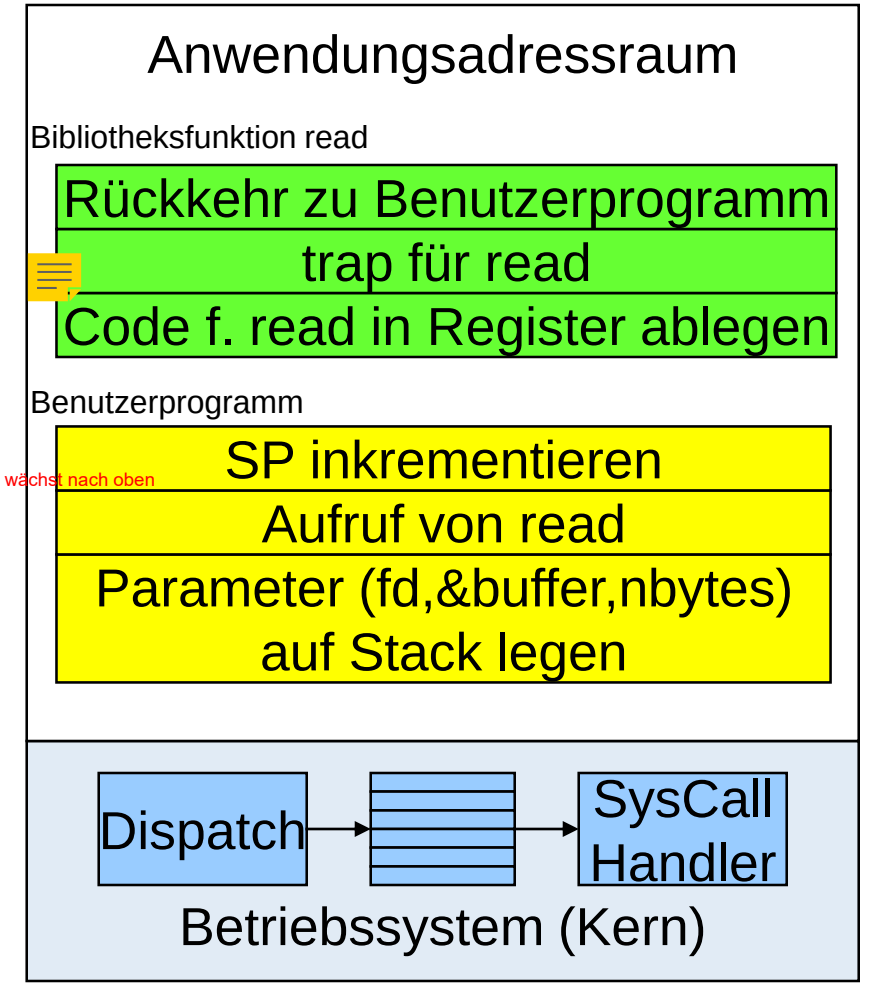
# 2.2.2.4 CPU-Schutz

- Vermeiden von Monopolisierung der CPU durch eine Anwendung
- Betriebssystem setzt Zeitgeber zur Auslösung eines Interrupts und übergibt danach die Kontrolle der CPU an die Anwendung.
- Time-Sharing
  - Unterbrechung nach bestimmtem Intervall, z.B. N ms, und Übergabe der CPU an andere Anwendung
  - Ggf. Überprüfung, ob Anwendung maximale Laufzeit erreicht hat.

# 3.1.1 Systemaufrufe

- Schnittstelle zwischen Anwendungsprogrammen und Betriebssystem
- vom Betriebssystem bereitgestellte Aufrufe von Systemfunktionen
- Maschinen-Instruktionen (**`trap`**) zum Wechsel vom Benutzer- in Systemmodus
- Parameterübergabe an Betriebssystem
  - Register
  - Speicherbereich
  - Stack
- Beispiel: **`count = read (fd, buffer, nbytes)`**

zeiger auf file, buffer

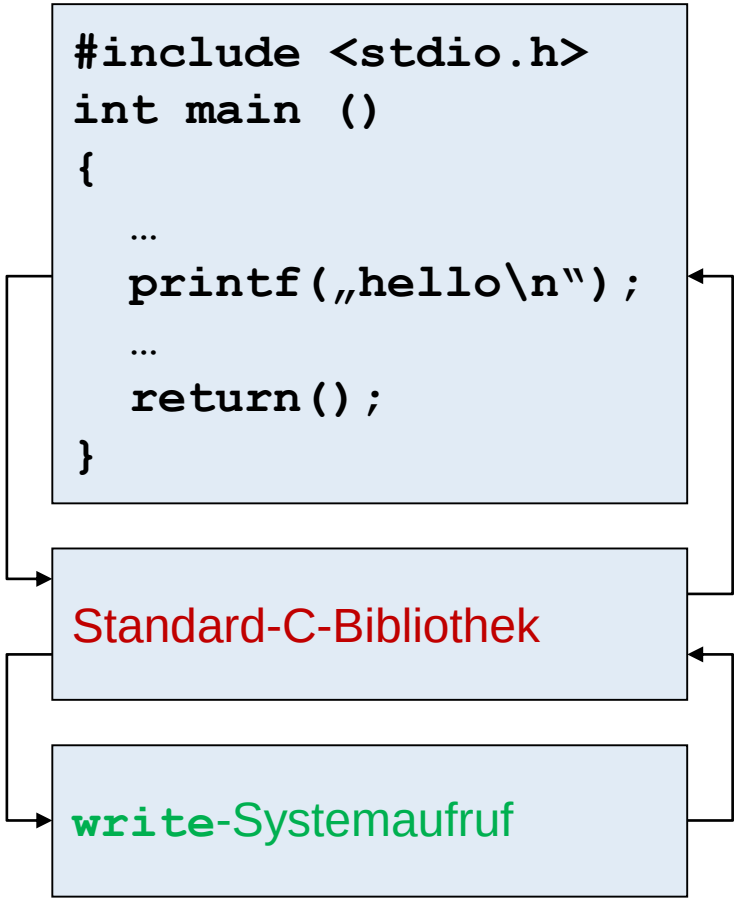


# 3.1.2 Beispiele für Systemaufrufe

- Prozessverwaltung
  - **`pid = fork():`** Erzeugen eines Kindprozess
  - **`s = execve(name, argv, environp):`** Ersetzen des Prozess-Image
- Datei- und Dateisystemverwaltung
  - **`fd = open():`** Öffnen einer Datei
  - **`n = write(fd, buffer, nbytes):`** Schreiben eines Puffers
- Gerätemanipulation
  - **`i = ioctl(fd, request, argp):`** Übergabe spezieller Kommandos
- Informationsmanagement
  - **`i = gettimeofday(timeval):`** Abrufen der Systemzeit
- Kommunikation
  - Socket-Calls (siehe Vorlesung Computernetze)

# 3.2 Systembibliotheken

- Systemaufrufe bilden die Schnittstelle zwischen Prozessen und Betriebssystemen.
- Systembibliotheken erweitern die Systemaufrufe um weitere Funktionen.
  - Definition von komplexeren Datenstrukturen in Header-Dateien
  - Bibliotheksfunktionen, z.B.
    - Kommunikationsfunktionen
    - Datenkonvertierung
    - grafische Funktionen



# 3.3 Systemprogramme

- Systemprogramme bieten Umgebung für Programmentwicklung und -ausführung.
- Kategorien
  - Dateimanipulation und -modifikation
  - Anzeige von Statusinformationen
  - Programmiersprachenunterstützung
  - Laden und Ausführen von Programmen
  - Kommunikationsprogramme
  - Anwendungsprogramme, z.B. Browser, Spiele
- Beispiele
  - **`delete, copy, vi; ls; cc; loader; rlogin, ftp`**

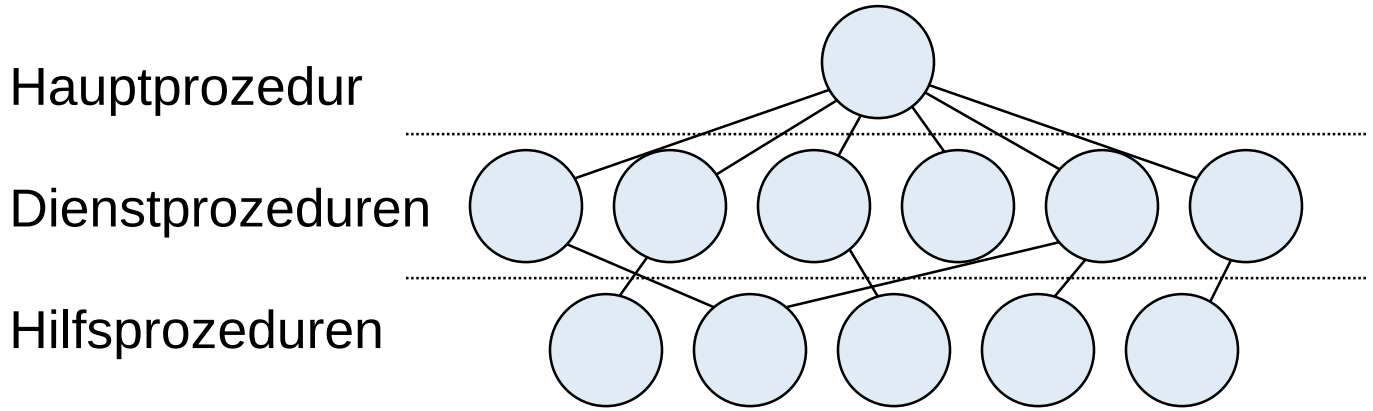
# 4. Betriebssystemarchitekturen

1. Monolithische Systeme
2. Geschichtete Systeme
3. Mikrokerne
4. Modulare Systeme
5. Ereignisgesteuerte Betriebssysteme
6. Virtuelle Maschinen
7. Hybride Systeme

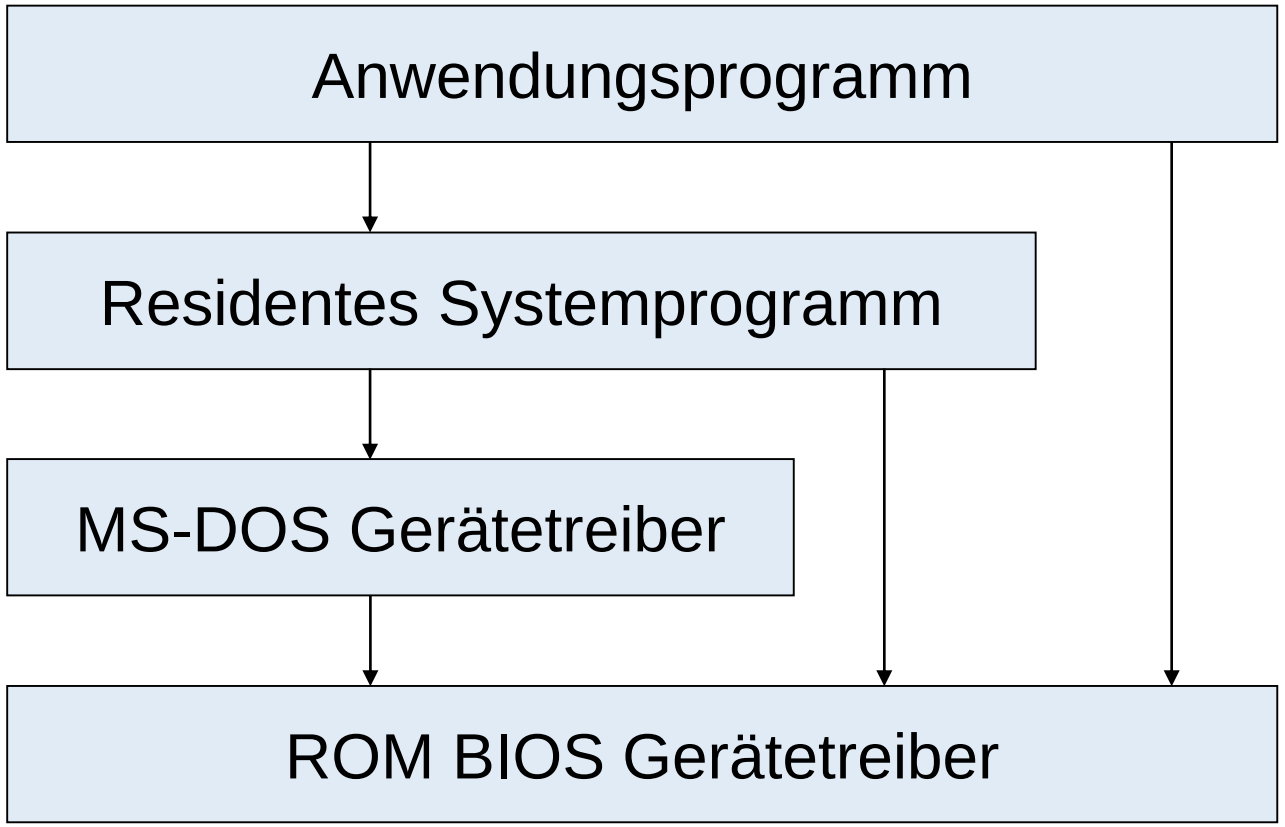
# 4.1 Monolithische Systeme

keine hierarchie, jeder kann jeden aufrufen

- strukturlose Sammlung von Prozeduren und Funktionen
- Prozeduren können beliebig andere aufrufen und Datenstrukturen ändern.
- Gruppierung von Funktionen, aber keine Modularisierung, kein Information Hiding
- früher: Übersetzen der Prozeduren, Binden zu einer ausführbaren Datei
- Prozeduren
  - Hauptprozedur ruft erforderliche Dienstprozedur auf.
  - Dienstprozeduren führen Systemaufrufe durch. (1:1-Abbildung)
  - Hilfsprozeduren unterstützen Dienstprozeduren.



# 4.1.1 MS-DOS


Anwendungsprogramm
Residentes Systemprogramm
MS-DOS Gerätetreiber
ROM BIOS Gerätetreiber

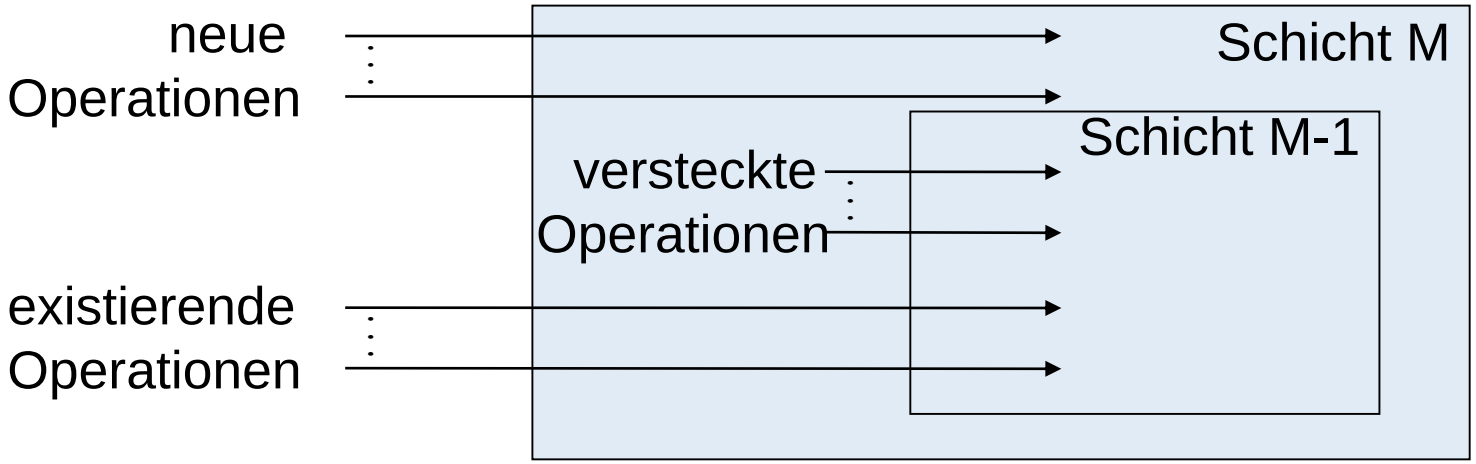
# 4.1.2 UNIX



| Benutzer | | |
|---|---|---|
| Shells, Compiler, Interpreter, Systembibliotheken | | |
| Systemaufruf-Interface des Kerns | | |
| Signale<br>Terminal-Steuerung<br>Character I/O<br>Terminal-Treiber | Dateisystem<br>Swapping-<br>Block I/O<br>Disk-,<br>Band-Treiber | CPU-Scheduling<br>Seitenersetzung<br>Demand Paging<br>virtueller Speicher |
| Hardware-Interface des Kerns | | |
| Terminal-Controller<br>Terminals | Device Controller<br>Disk-, Band-Laufwerke | Speicher-Contr.<br>physikalischer Speicher |

# 4.2.1 Geschichtete Systeme

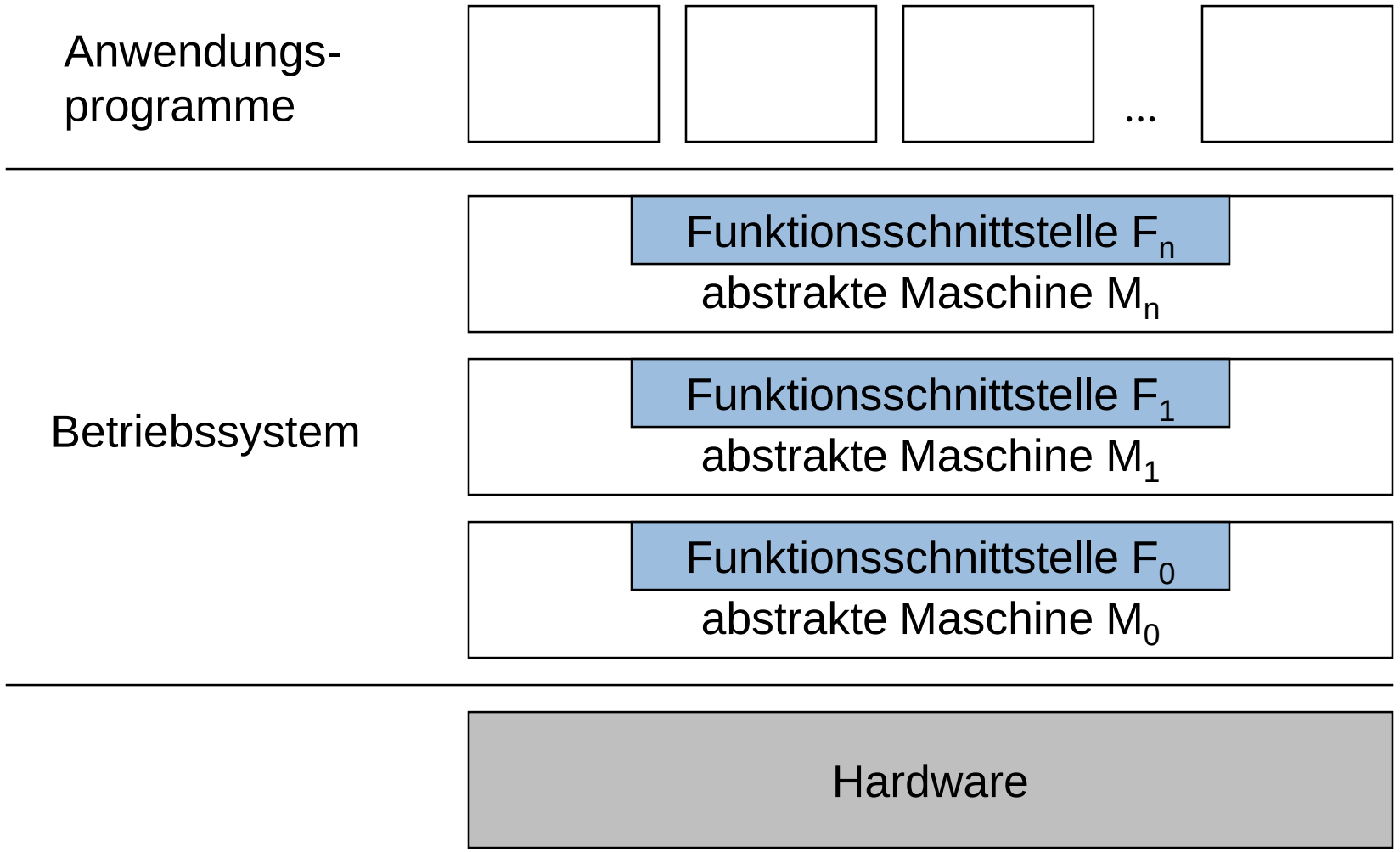
- Betriebssystem besteht aus mehreren Schichten.
- Schichten bauen aufeinander auf und benutzen Funktionen der darunter liegenden Schichten.
- kein Zugriff von Benutzerprozessen auf Funktionen der unteren Ebenen
- Vorteile: Modularität, Testbarkeit / Debugging
- Nachteil: höherer Overhead



# 4.2.2 Abstrakte Maschinen

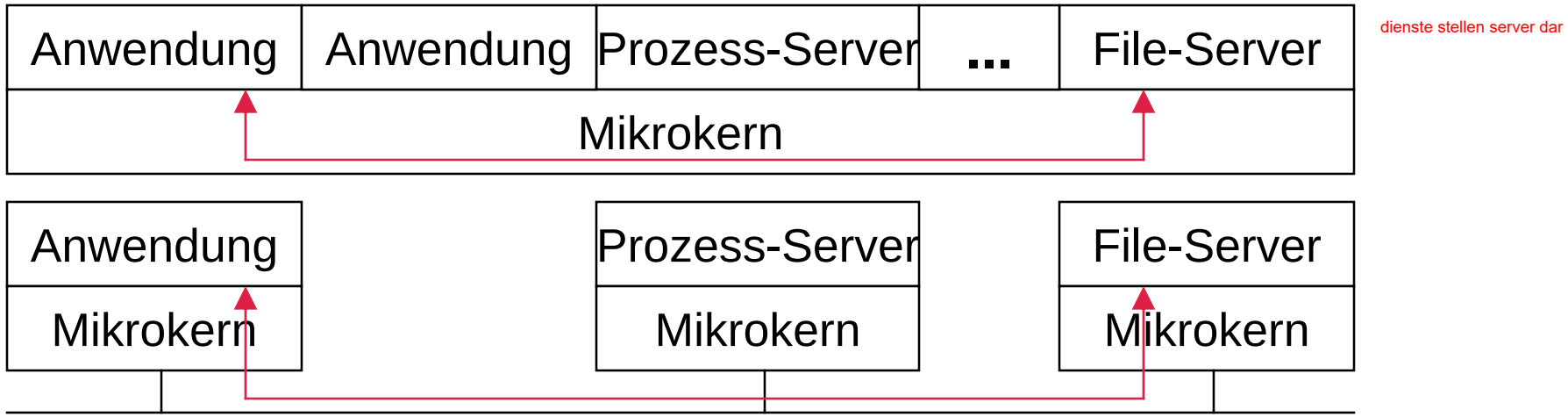


nachteil: overhead



# 4.3 Mikrokerne

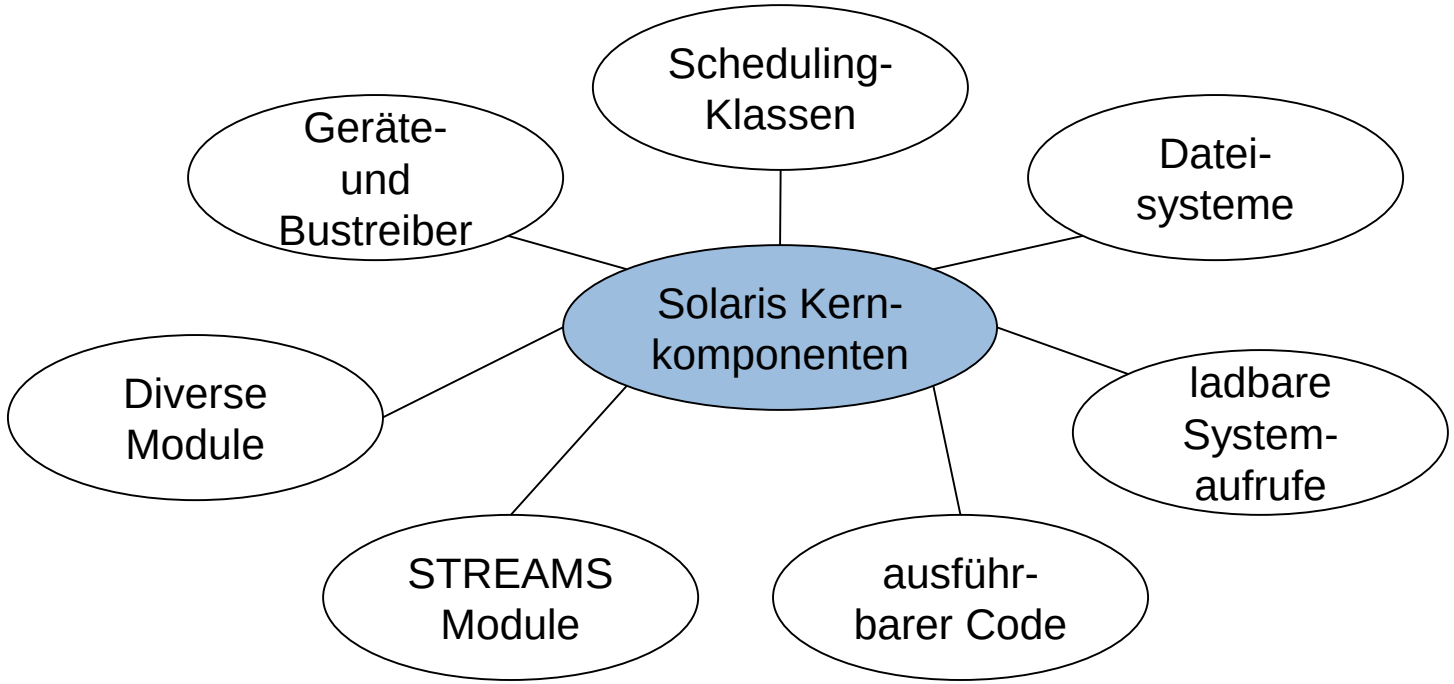
- kleinere Betriebssysteme durch Auslagerung von Teilfunktionen in Server (ggf. im Benutzermodus)
- Weiterleiten von Systemaufrufen an zuständigen Server
- Mikrokern: Kommunikationssystem, elementarer Mehrprogrammbetrieb, Ein-/Ausgabe
- Option: Verteilung der Server über ein Rechnernetz
- Beispiel: Mach (CMU, 1980er Jahre)




dienste stellen server dar

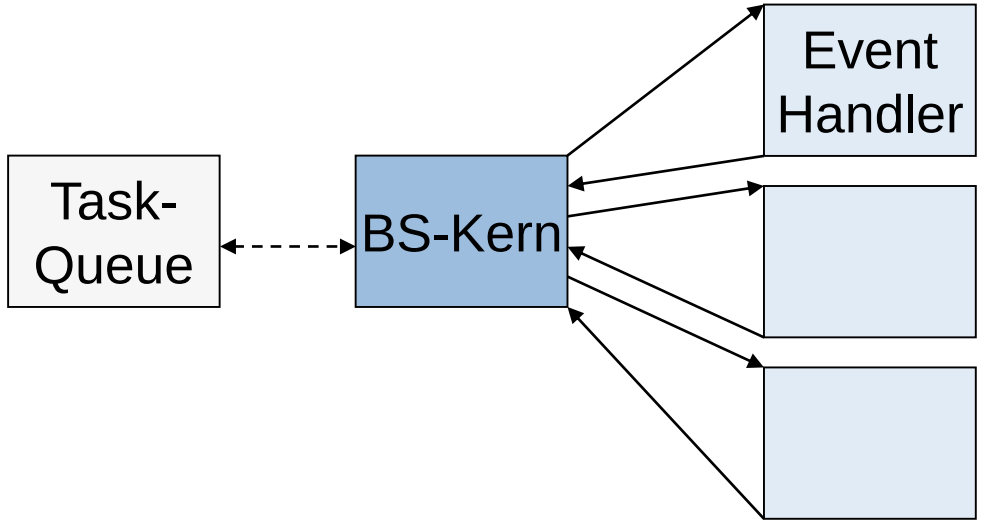
# 4.4 Module

- Objektorientierte Programmierung zur Erzeugung modularer Betriebssystemkerne
- Betriebssystemkern mit Kernkomponenten und
  Einbinden von zusätzlichen Diensten beim Booten oder zur Laufzeit
- Beispiel: Solaris



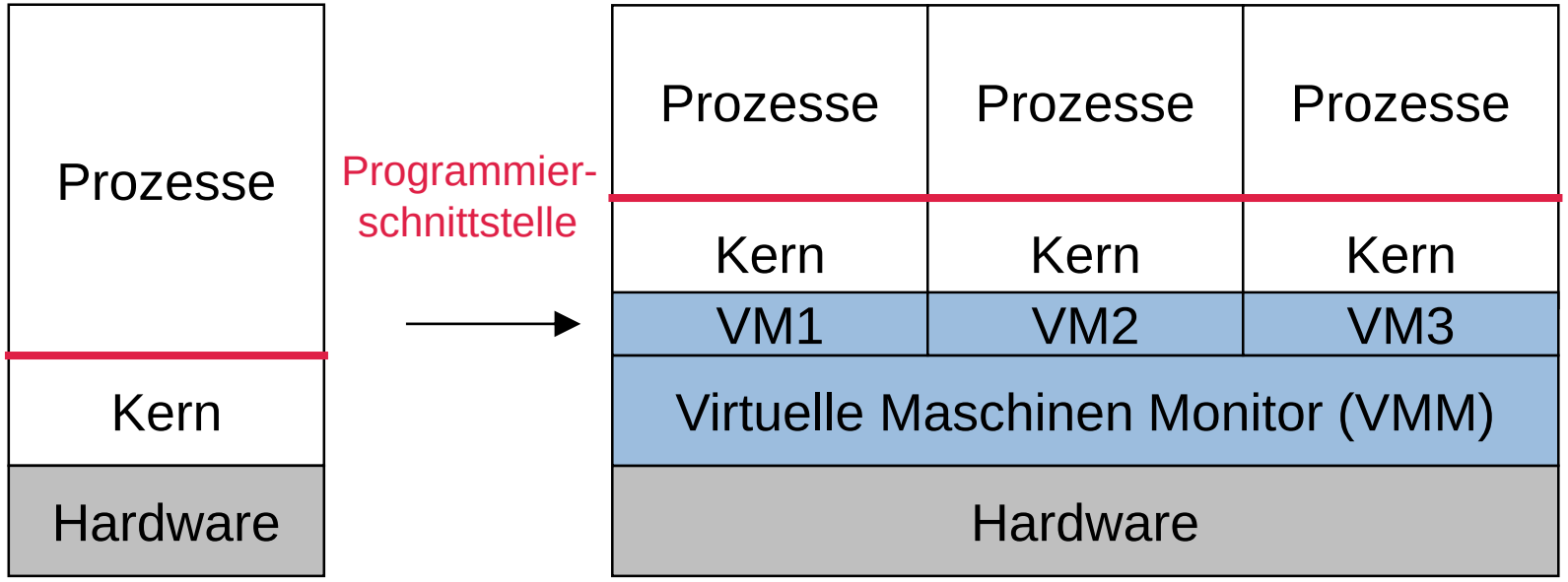
# 4.5 Ereignisgesteuerte Betriebssysteme

- Betriebssystem reagiert nur auf bestimmte Ereignisse (z.B. Benutzereingaben, Nachrichten, Zeitgeber) und arbeitet diese ab.
- Unterbrechbare Tasks für länger andauernde Berechnungen
- Mögliches Abschalten von Systemkomponenten nach der Ereignisverarbeitung
- Anwendung: Realzeitsysteme, eingebettete Systeme, PDAs
- Beispiel: TinyOS



# 4.6.1 Virtuelle Maschinen

- Abstrahieren der Computer-Hardware in verschiedene Ausführungsumgebungen und Erzeugen der Illusion, dass jede Ausführungsumgebung eigenen Computer zur Verfügung hat.
- Emulation verschiedener virtueller Maschinen (Prozessor und Speicher) durch virtuellen Speicher, CPU-Scheduling, Disk-Partitionen, virtuelle Geräte
- Anbieten einer zur Hardware identischen Schnittstelle ermöglicht gleichzeitige Unterstützung mehrerer, verschiedener Betriebssysteme.
- Beispiele: VM/370 (1972), MS-DOS Emulationen in Windows, Java VM, VMWare




hardware vom rechner abstrahieren, dann wird bei einem OS simuliert, es sässe direkt auf der hardware

# 4.6.2. Eigenschaften virtueller Maschinen

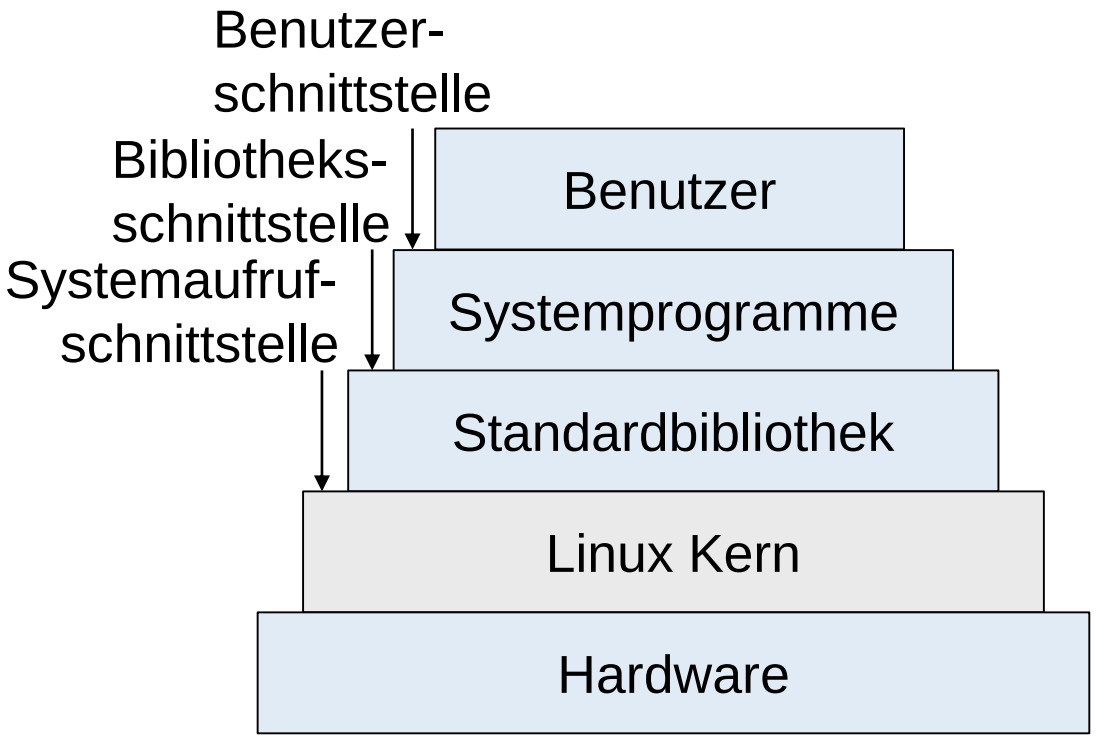


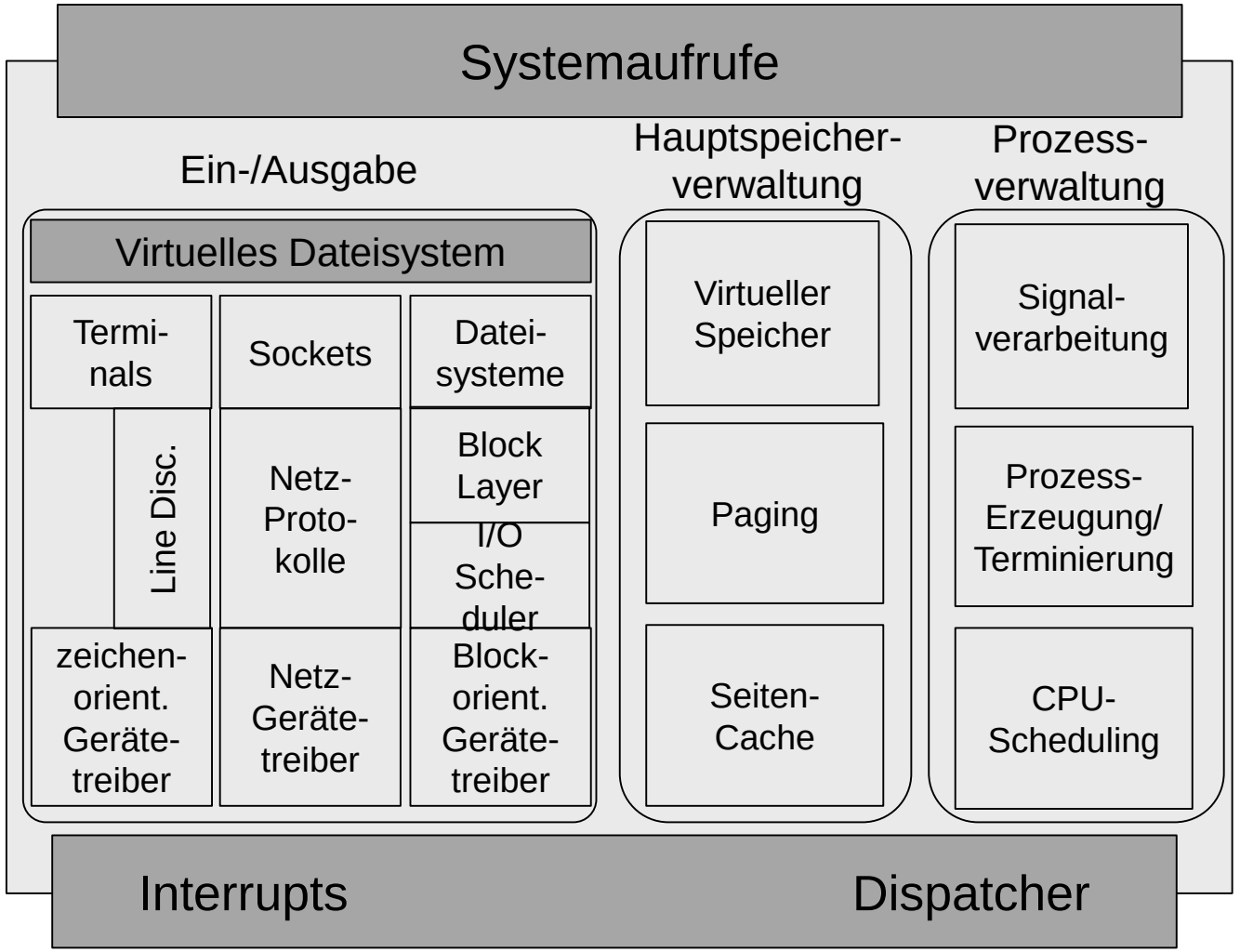
- Vorteile
  - Sicherheit und Robustheit durch Isolation der virtuellen Maschinen
  - Betriebssystementwicklung
  - Bequemes Wechseln von Betriebssystemen
  - Emulation diverser Betriebssysteme auf einem Rechner
  - Management- und Hardware-Kosten
- Nachteil
  - Kontextwechsel und Emulationsoperationen erzeugen Overhead
  - erfordert Hardwareunterstützung, z.B. zusätzliche Betriebsmodi, Speicherverwaltung

# 4.7 Hybride Systeme

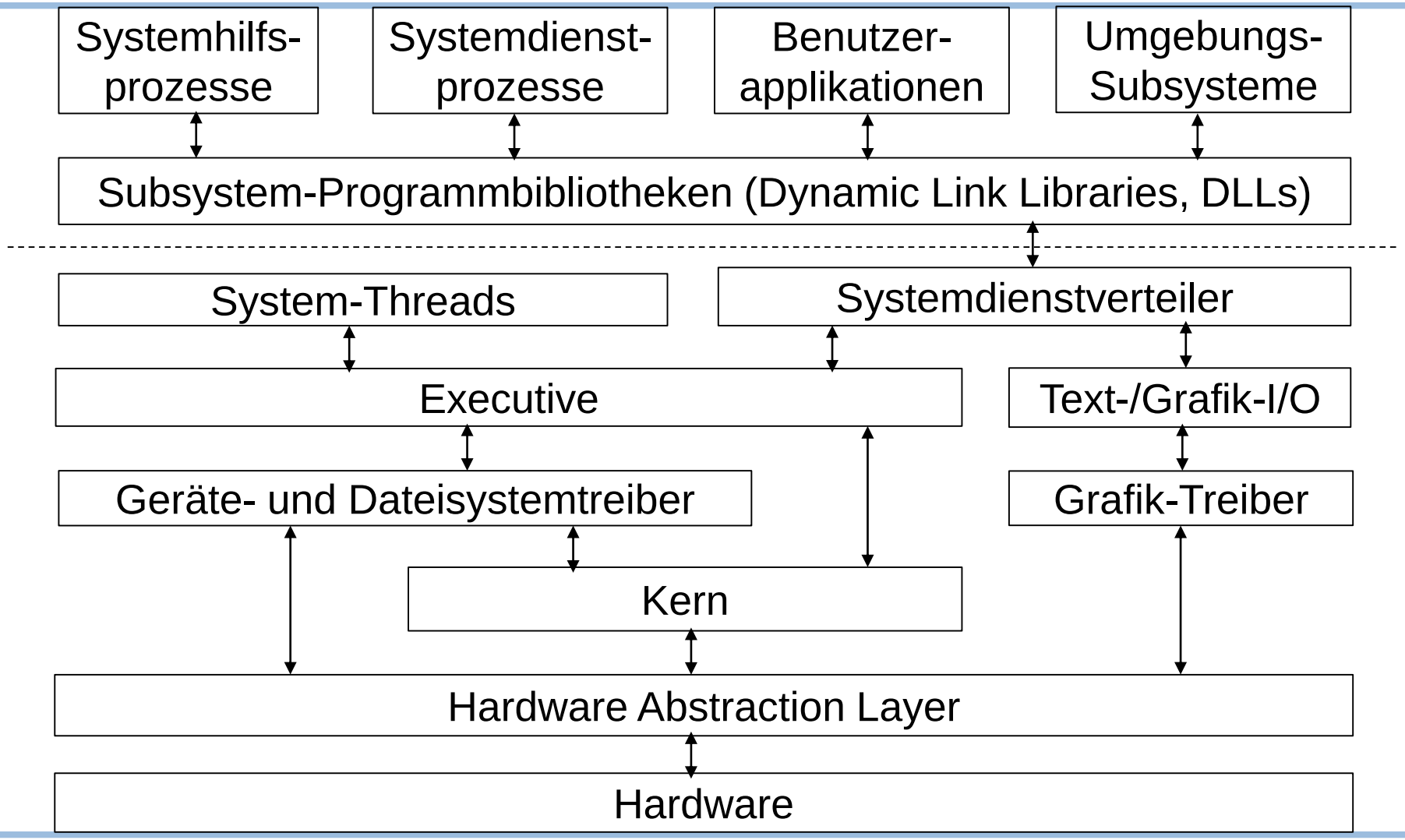
- In der Praxis werden grundlegende Architekturkonzepte oft kombiniert.
- Beispiele:
  - Linux und Solaris sind monolithisch (Leistung),
    aber auch modular um dynamisch Funktionen nachzuladen.
  - Windows (ebenfalls monolithisch) hat Konzepte von Mikrokernen und
    dynamisch ladbaren Kernmodulen.
  - Mac OS X ist ein geschichtetes System
    mit einer auf Mikrokernen basierenden Kern-Umgebung.
  - iOS basiert auf Max OS X mit Funktionalitäten für mobile Dienste
  - Android hat eine geschichtete Software-Struktur mit einem Linux-Kern.

# 4.7.1 Linux

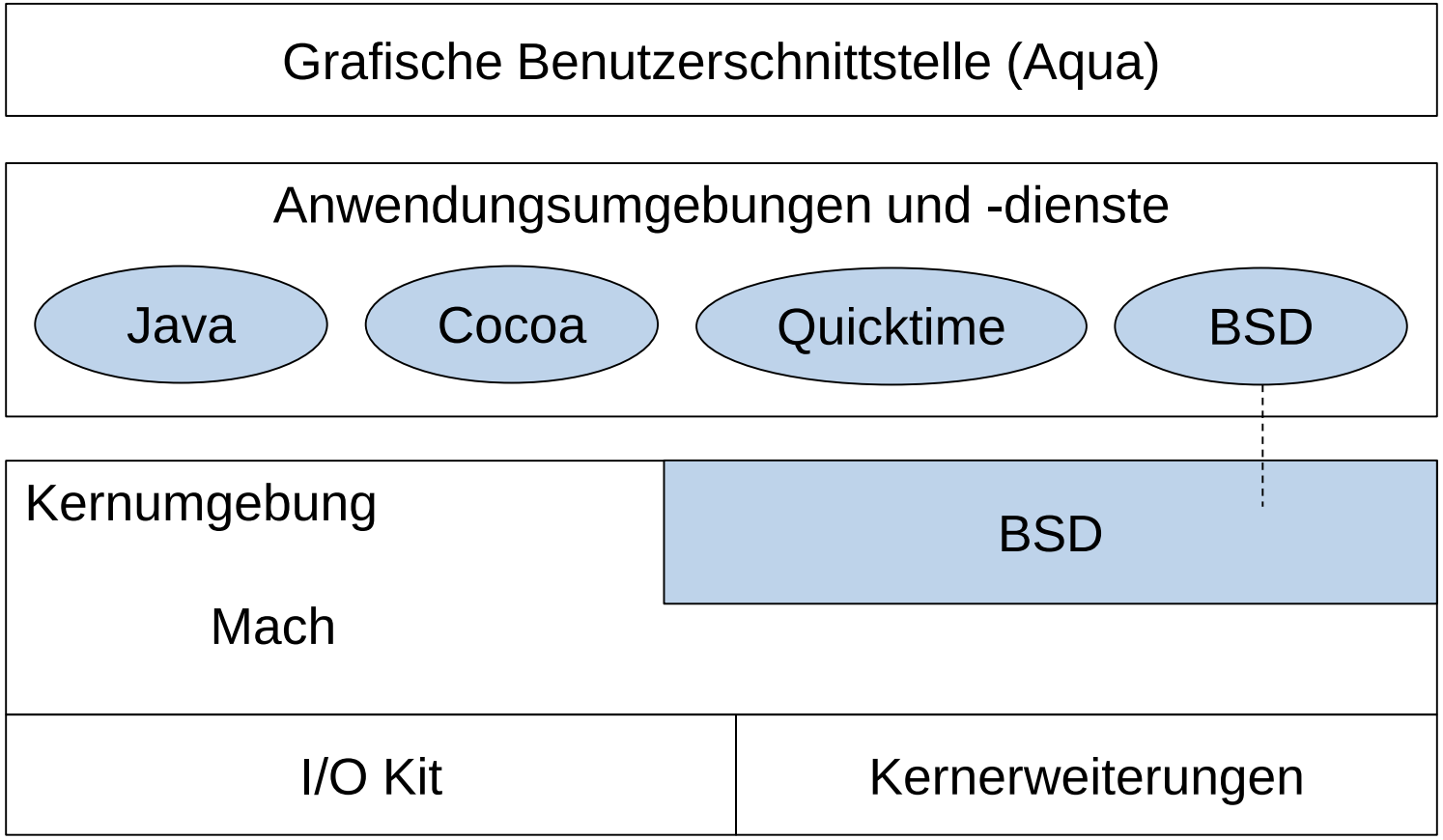

Benutzer-
schnittstelle
Bibliotheks-
schnittstelle
Systemaufruf-
schnittstelle
Benutzer
Systemprogramme
Standardbibliothek
Linux Kern
Hardware


Systemaufrufe
Ein-/Ausgabe
Hauptspeicher-
verwaltung
Prozess-
verwaltung
Virtuelles Dateisystem
Termi-
nals
Sockets
Datei-
systeme
Line Disc.
Netz-
Proto-
kolle
Block
Layer
I/O
Sche-
duler
zeichen-
orient.
Geräte-
treiber
Netz-
Geräte-
treiber
Block-
orient.
Geräte-
treiber
Virtueller
Speicher
Paging
Seiten-
Cache
Signal-
verarbeitung
Prozess-
Erzeugung/
Terminierung
CPU-
Scheduling
Interrupts
Dispatcher

# 4.7.2 Windows 10

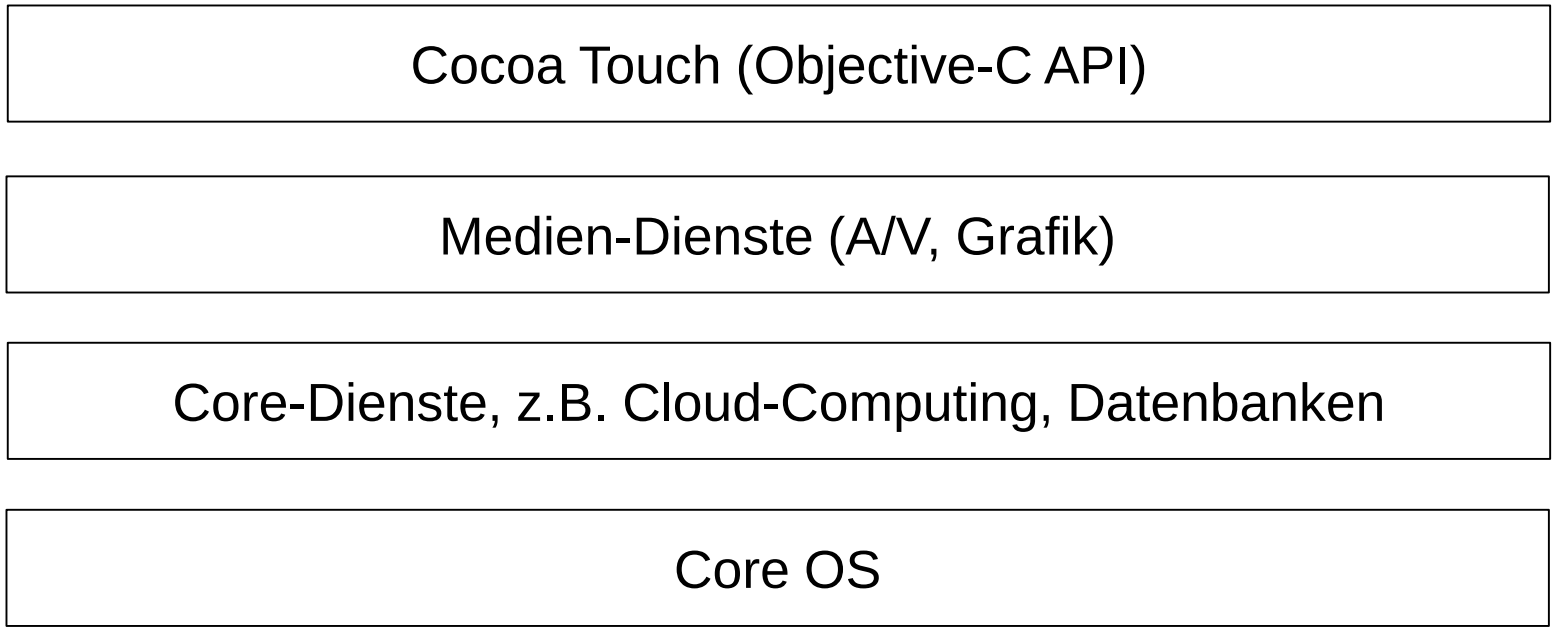


# 4.7.3 Mac OS X

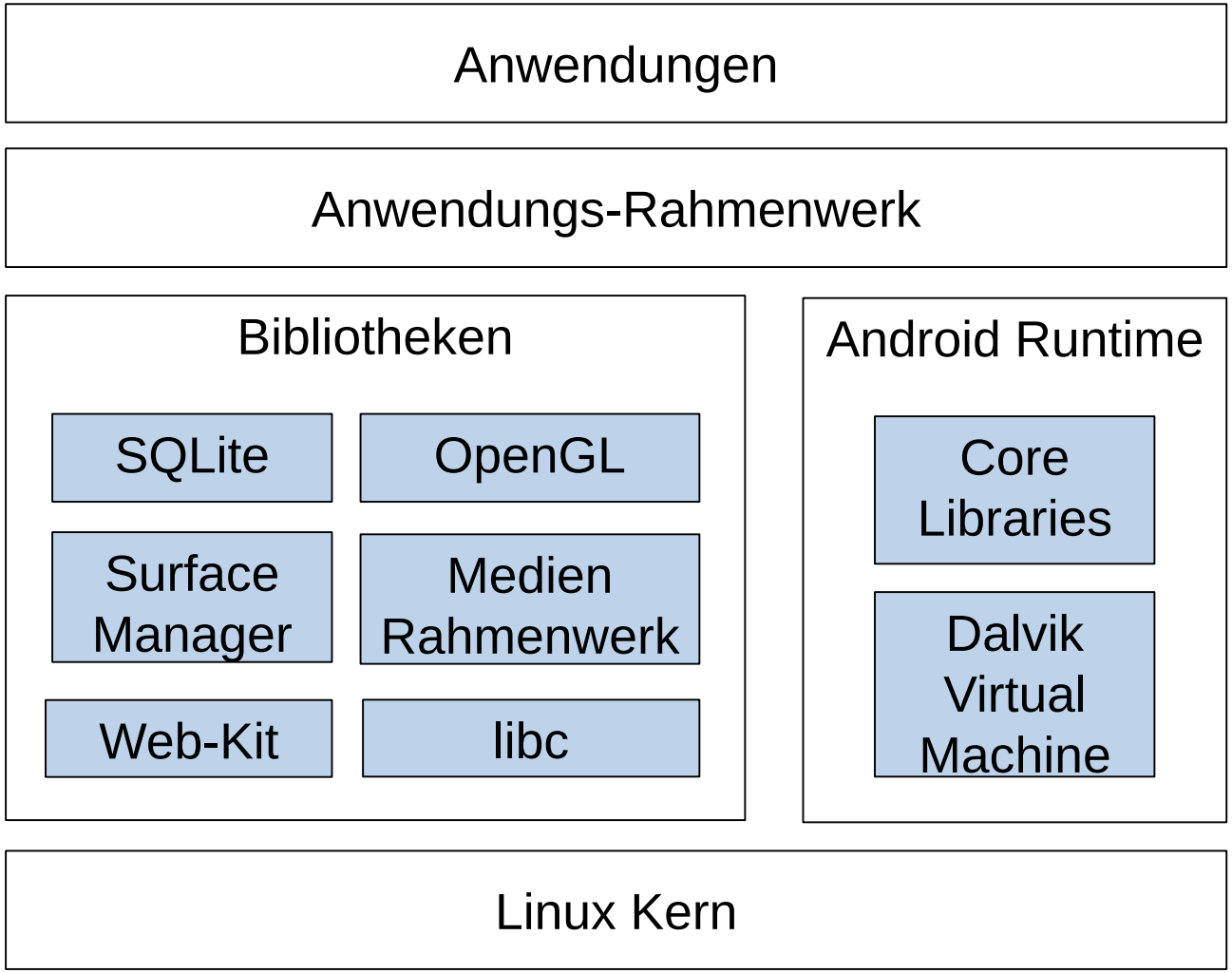

Grafische Benutzerschnittstelle (Aqua)
Anwendungsumgebungen und -dienste
Java
Cocoa
Quicktime
BSD
Kernumgebung
BSD
Mach
I/O Kit
Kernerweiterungen

# 4.7.4 iOS




Cocoa Touch (Objective-C API)
Medien-Dienste (A/V, Grafik)
Core-Dienste, z.B. Cloud-Computing, Datenbanken
Core OS

# 4.7.5 Android


Anwendungen
Anwendungs-Rahmenwerk
Bibliotheken
SQLite
OpenGL
Surface Manager
Medien Rahmenwerk
Web-Kit
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kern