Vorlesung
Betriebssysteme und Systemsoftware

Kapitel 1:
Betriebssysteme: Aufbau und Aufgaben

Bastian Leibe

Visual Computing Institute
Computer Vision
RWTH Aachen University

http://www.vision.rwth-aachen.de

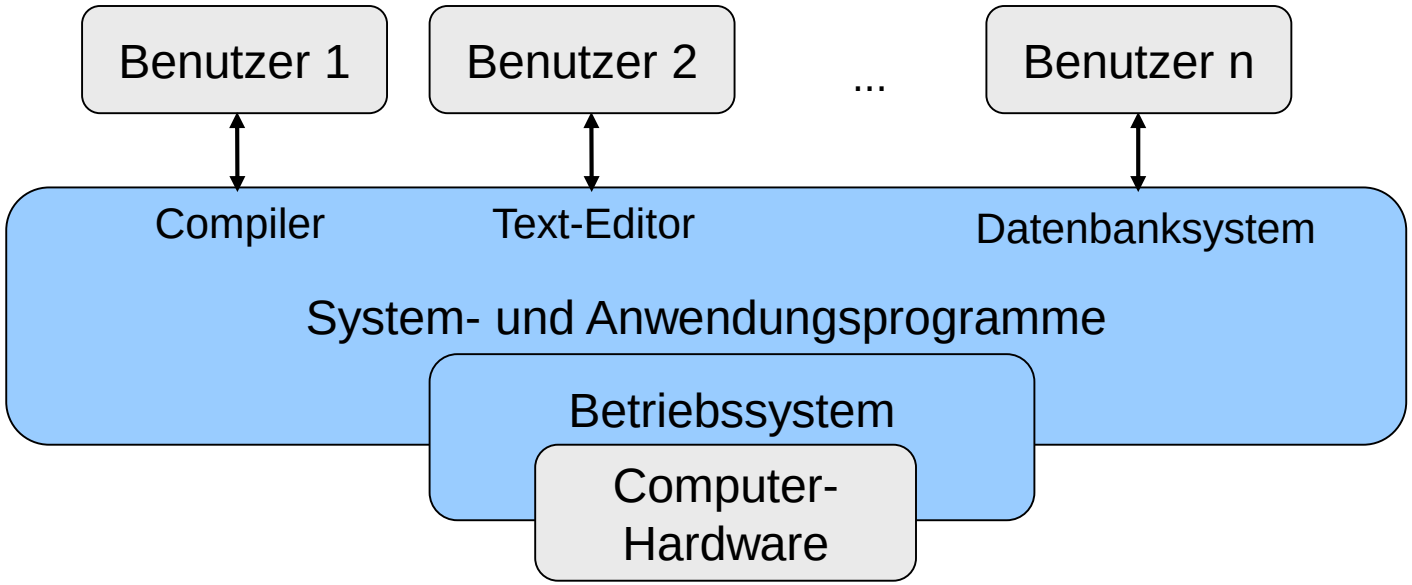
Viele Folien adaptiert von Klaus Wehrle und Dirk Thissen

Themenübersicht

- **Betriebssysteme und Systemsoftware**
  - Betriebssysteme: Aufbau und Aufgaben
  - Shell- und C-Programmierung
  - Prozesse und Threads, Prozessverwaltung und -kommunikation
  - CPU-Scheduling
  - Prozesssynchronisation, Deadlocks
  - Speicherverwaltung, virtueller Speicher
  - Dateisystem, Zugriffsrechte und I/O-System
  - Kommunikation, Verteilte Systeme

Computer-Hardware und Betriebssystem

- **Einteilung des gesamten Computers in vier Bereiche:**
  - *Computer-Hardware*: Ansammlung von Betriebsmitteln, welche die Ausführung von Programmen ermöglichen
  - *Betriebssystem*: Verwaltung und Koordination der Hardware
  - *System- und Anwendungsprogramme*
  - *Benutzer*: können Menschen sein, aber auch andere Computer



1.1 Aufbau von Rechnersystemen

- Hardware, Rechnerarchitektur, Betriebssystem

1.2 Aufbau und Aufgaben eines Betriebssystems

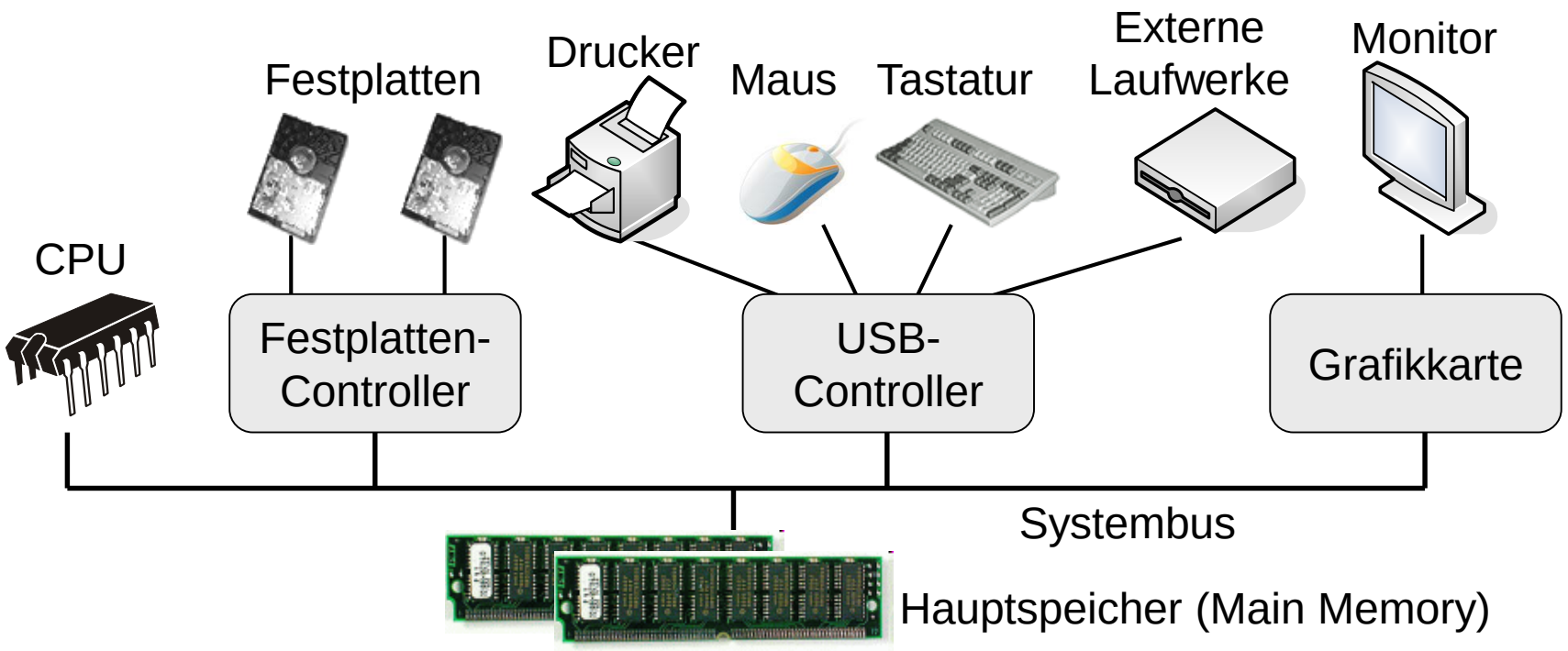
- Generelle Aufgaben, Betriebssystemstruktur, Betriebssystemarten

1.3 Multitasking und Interrupts

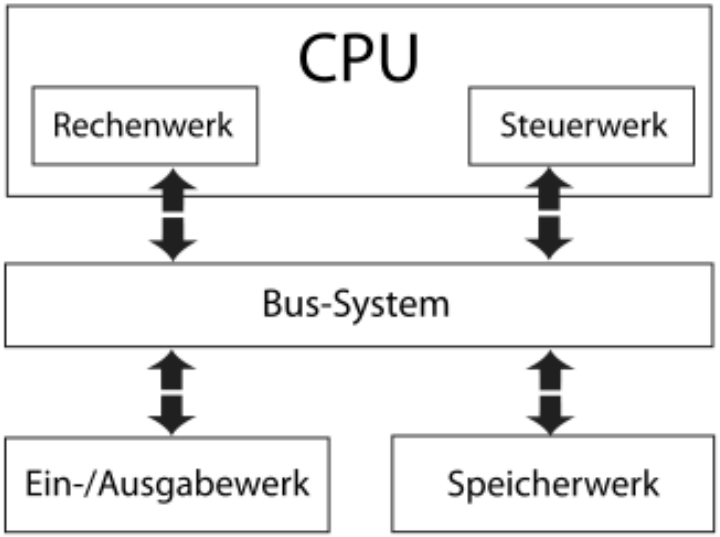
- Multitasking, Interrupt-Handling, Context Switches, Systemaufrufe

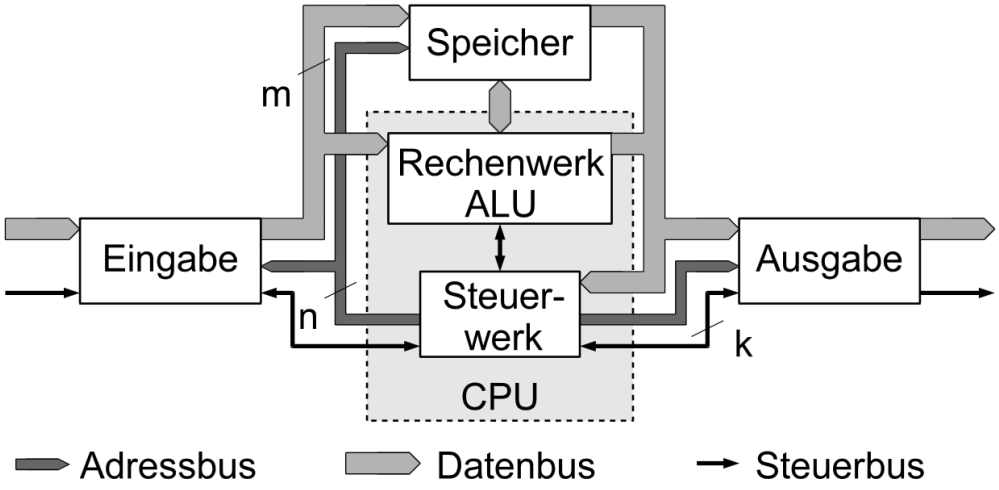
Computer-Hardware (vereinfacht)

- **Systembus verbindet alle Geräte des Computers**
  - Eine oder mehrere CPUs zur Ausführung von Programmen
  - Gemeinsamer Speicher für Aufgaben der CPU und anderer Geräte
  - Controller zum Anschluss von I/O-Geräten



- **Prominente Rechnerarchitektur: von-Neumann**
  - Klassisches Referenzmodell für Computer

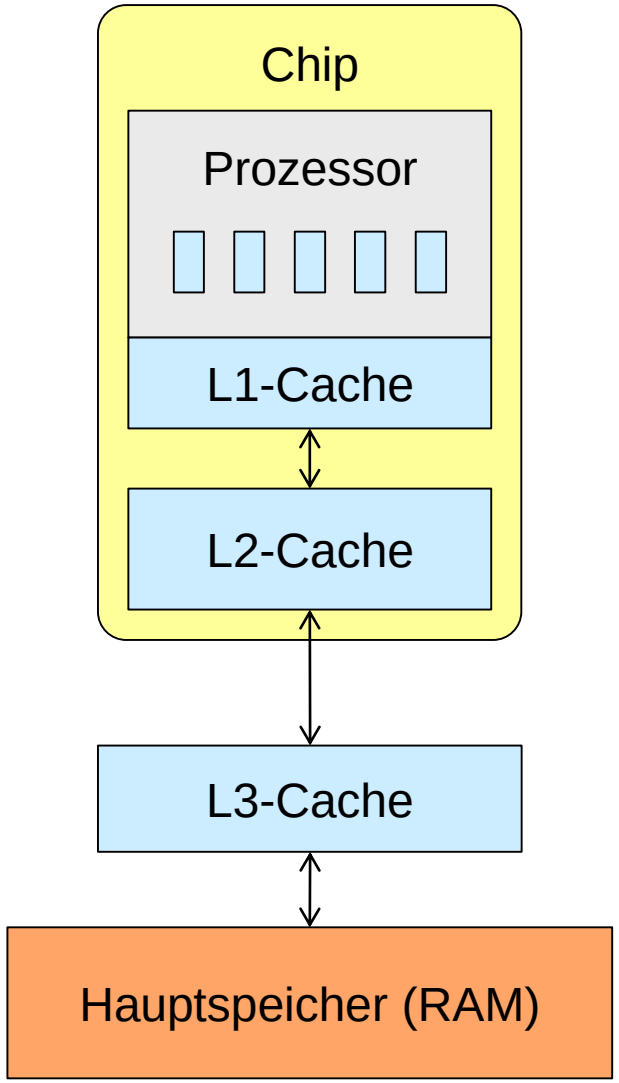




Quelle: http://de.wikipedia.org/wiki/Von-Neumann-Architektur

Rechnerarchitektur: Prozessor (CPU)

- **CPU**
  - Verfügt über *Register* zur Ausführung von Operationen
    - Datenregister, Addressregister, Spezialregister, ...
  - Zusätzlich: *Caches*
    - Schneller Pufferspeicher
    - Schnellerer Zugriff auf Caches als auf Hauptspeicher
    - Je kleiner desto geringere Zugriffszeiten
    - Caches sind transparent für Betriebssystem



Nebenbemerkung: Speicherstrukturen

Hierarchie-Ebene	Größe	Zugriffszeit (ns)
1: Register	8 – 64 Bit	<< 1
2: L1-Cache	16 – 64 KiB	1 – 2
3: L2-Cache	64 KiB – 4 MiB	2 – 4
4: L3-Cache	4 – 16 MiB	8 – 10
5: Hauptspeicher	> 1 GiB	~ 60
6: Festplatte (SSD)	> 100 GiB	~ 5.000.000 (250.000)

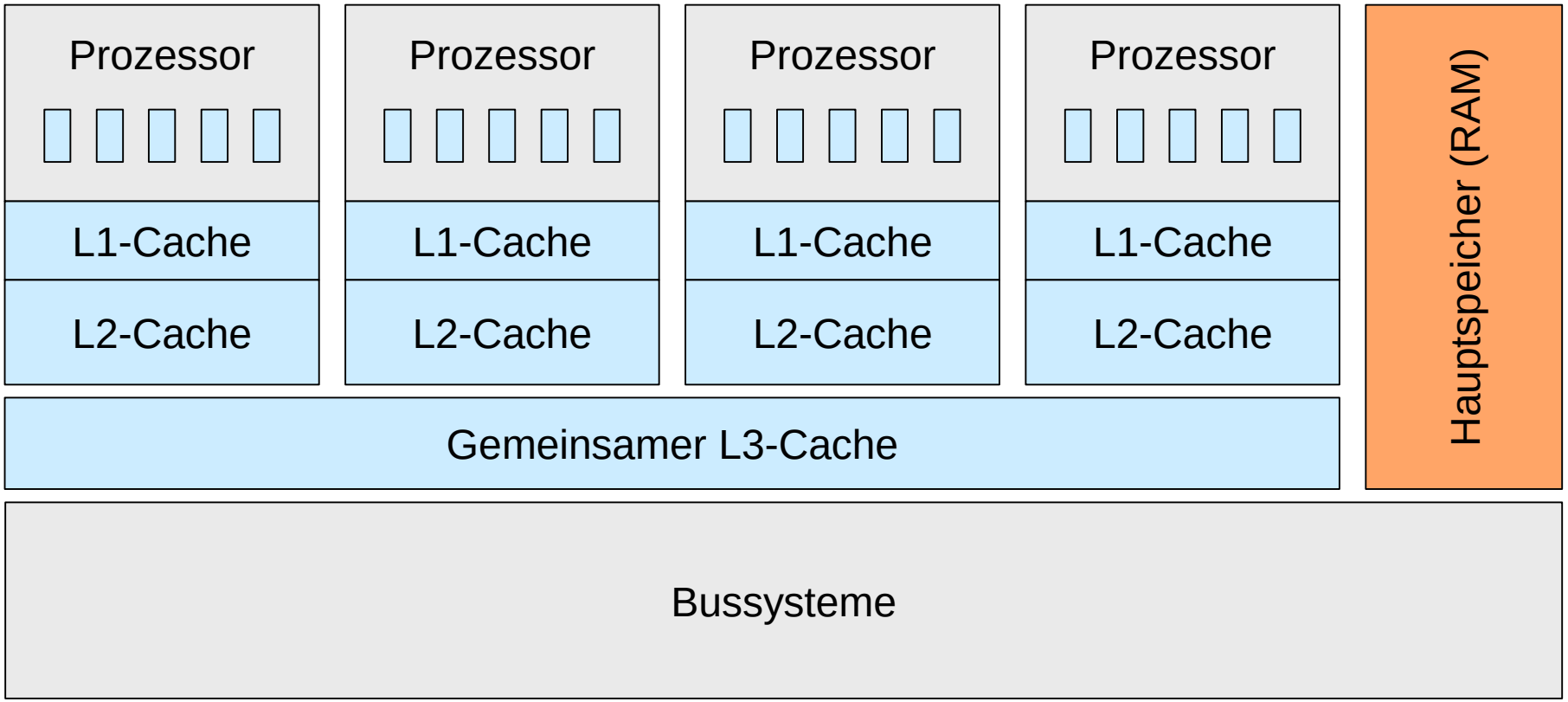
…

…

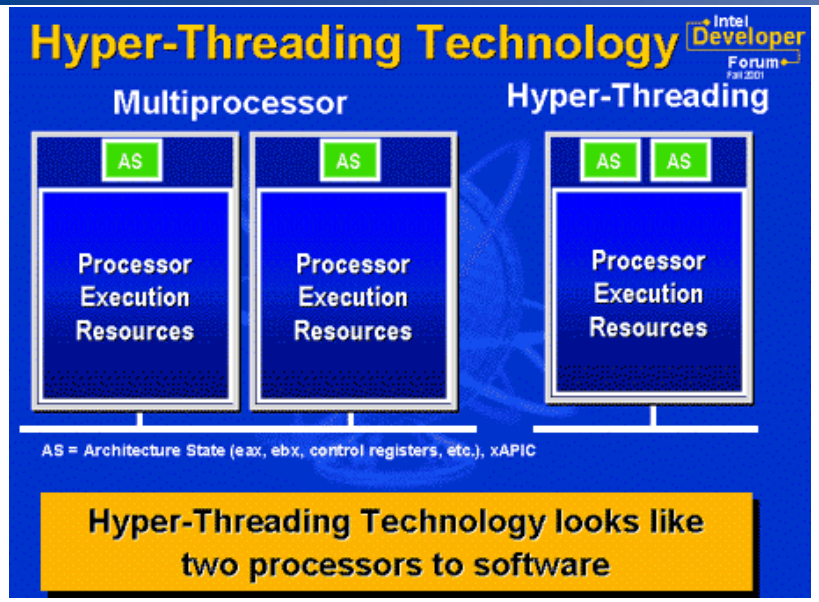
…

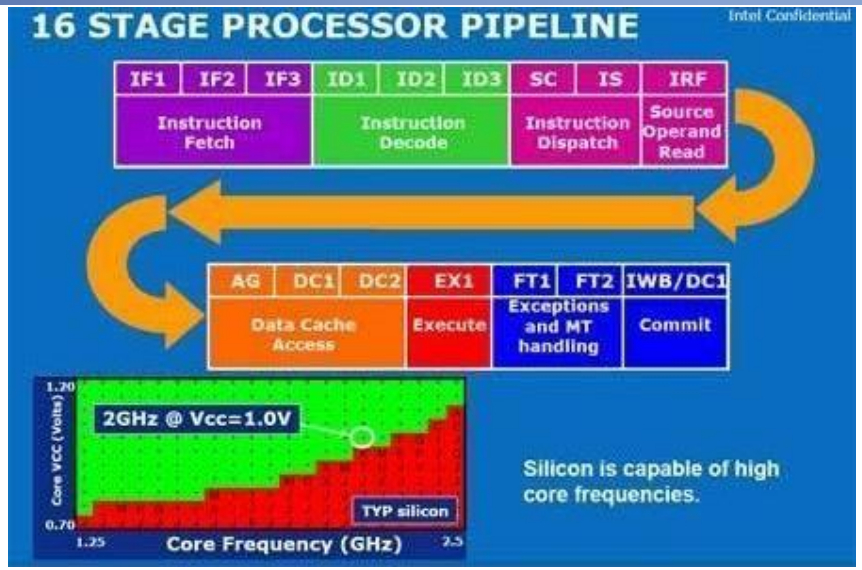
Prozessorkerne und Caches

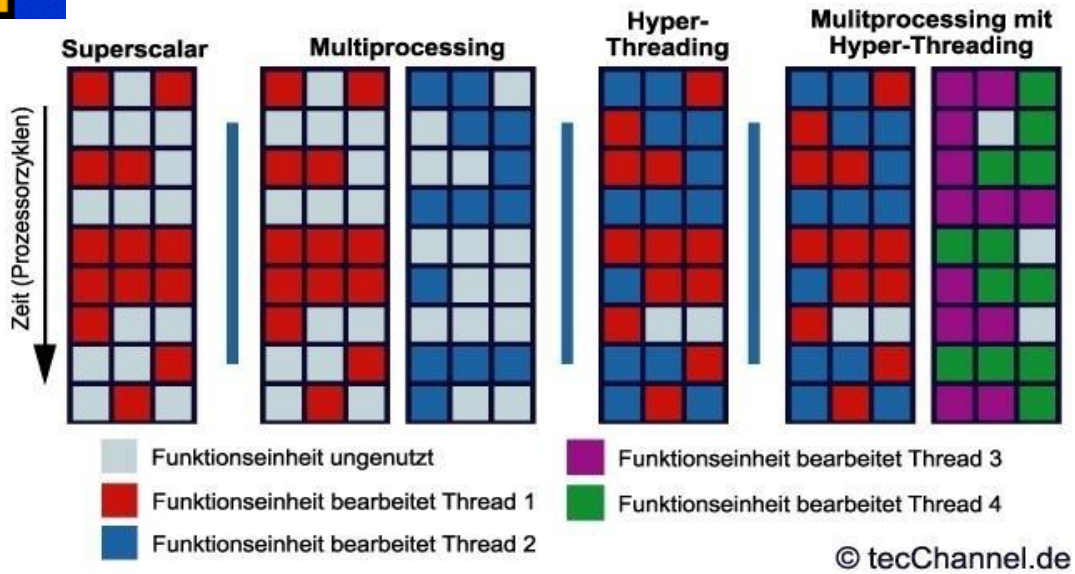
- **Grobe Architektur bei Verwendung mehrerer Prozessorkerne**



Hyperthreading







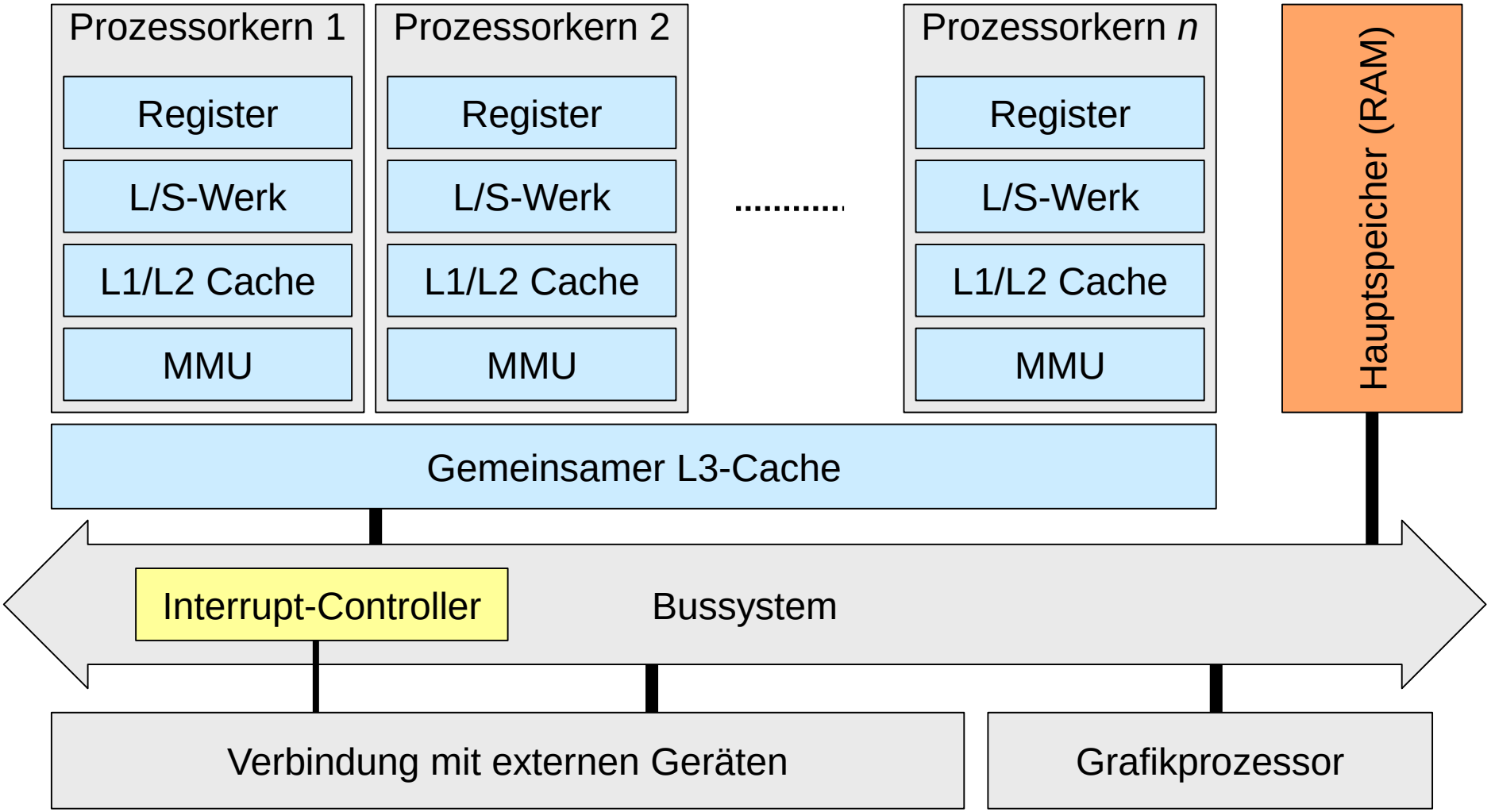
Bildquelle: http://www.pcstats.com/articleimages/122002/hyperthreading_image1.gif
http://www.tecchannel.de/pc_mobile/extra/401816/intel_developer_forum_2002/index3.html

Zusammenspiel der Hardware-Komponenten

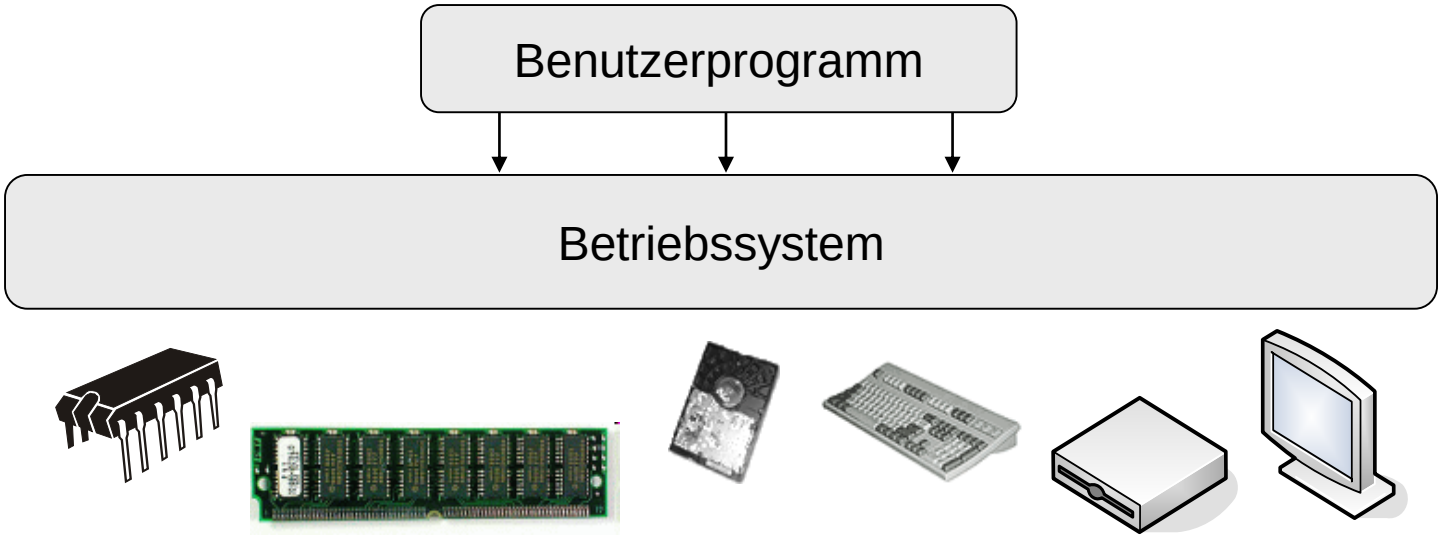
- **Kern: CPU – führt Operationen aus**
- **CPU und I/O-Geräte werden *nebenläufig* ausgeführt**
  - Jeder Controller ist für einen bestimmten Gerätetyp verantwortlich
  - Zur Ausführung einer Operation eines Gerätes wird die CPU benötigt:
    - Jeder Controller hat eigene Register und einen lokalen Buffer (*Cache*)
    - Die CPU verschiebt Daten zwischen Hauptspeicher und Caches
    - Nach Verschieben der Daten wird die Operation gestartet
  - Heutzutage: *Direct Memory Access, DMA*:
    - Separater Controller zum Verschieben von Daten zwischen CPU und Geräten
    - Entlastung der CPU

Vereinfachte Rechnerarchitektur

- **Wesentliche Komponenten des Rechnersystems:**



- ***Betriebssystem* (engl. Operating System, OS)**
  - Ansammlung von Programmen, die die effiziente und komfortable Nutzung eines Computers ermöglichen:
    - Plattform zur Ausführung von Programmen auf einer Computer-Hardware
    - Effiziente Aufteilung der *Betriebsmittel* (CPU, Festplatten, …) auf mehrere Benutzer bzw. Benutzerprogramme



Nebenbemerkung: Betriebsmittel

- ***Betriebsmittel*** **können Hardware- aber auch Softwareressourcen sein:**
  - Prozessoren, Prozesse, Threads
  - Speicher
    - Hauptspeicher, Caches, virtueller Speicher
  - Dateisystem
    - Verzeichnisse, Dateien
  - I/O-Geräte
    - Grafikkarte, Netzwerkkarte, Festplatte, Tastatur, Maus
  - Klassifikation:
    - Exklusive oder geteilte Nutzung?
    - Entziehbar oder nicht entziehbar?

Entwicklung der Betriebssysteme

- **Vielzahl von Betriebssystemen, entwickelt für**
  - Mainframes — 1950
  - “Mini-computer” — 1960
  - Desktop-Computer — 1970
  - Handheld-Computer — 1990
  - Access Points, Sensorknoten, Smartphones, Tablets… — 2000
  - Internet-of-Things, Cyber-physical systems, Smartwatches, ... — 2010+

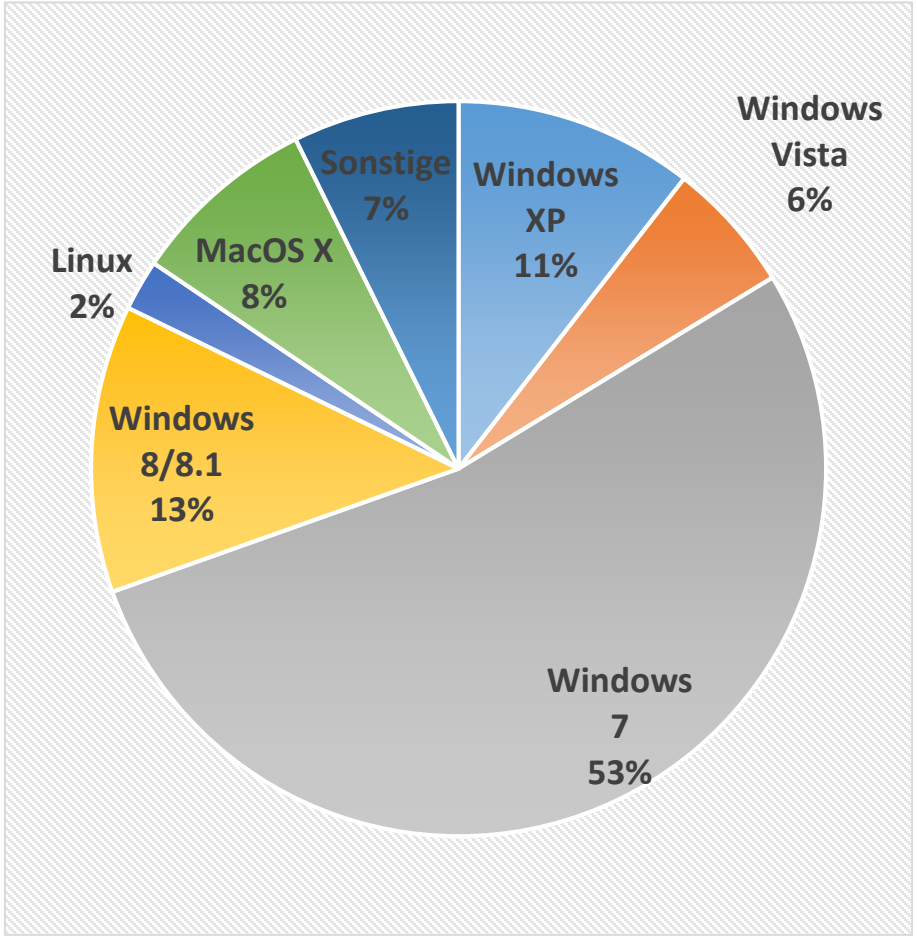
- http://en.wikipedia.org/wiki/Timeline_of_operating_systems

Verbreitung von Betriebssystemen

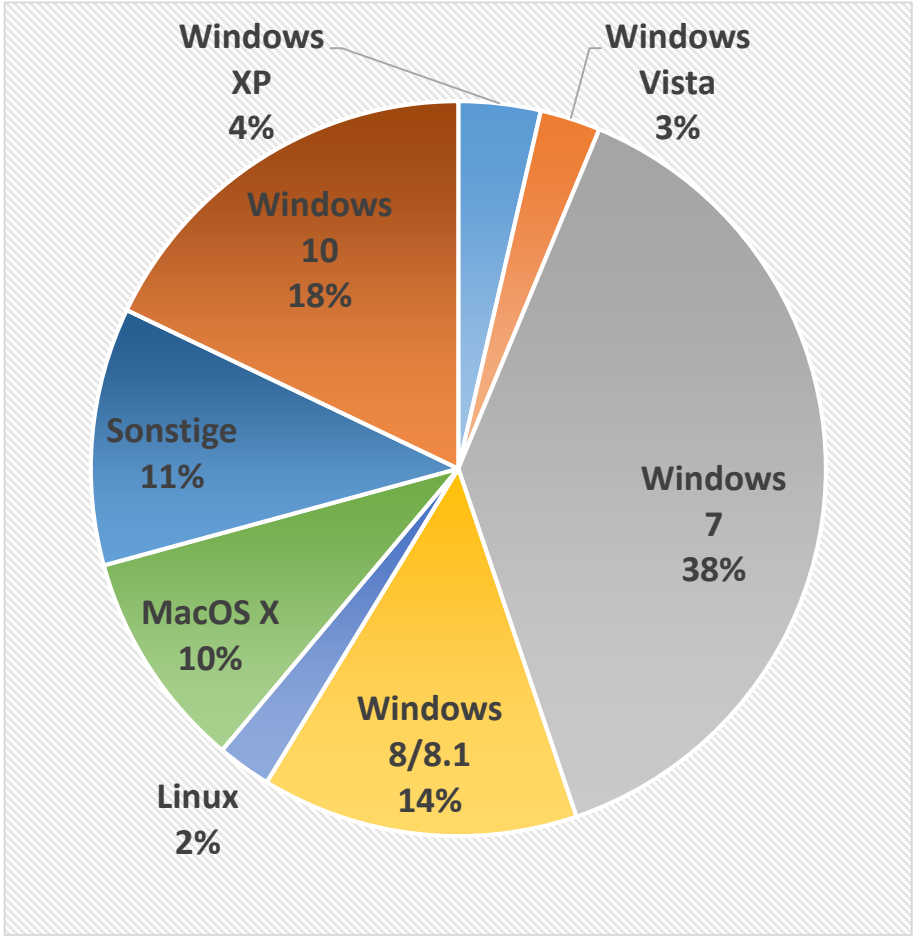
- **Heutige Betriebssysteme**
  - (MS-Dos, OS/2)
  - Windows
    - NT, 2000, XP, Vista, 7, 8, 8.1, 10, Server-Varianten
  - *Unix*
    - Sun Solaris, HP UX, ...
    - Linux
    - Mac OS X
    - Android
    - iOS
    - ...
  - TinyOS, Contiki,
  - ...

Verbreitung von (Desktop)-Betriebssystemen

- **Betriebssysteme in Deutschland März 2014**

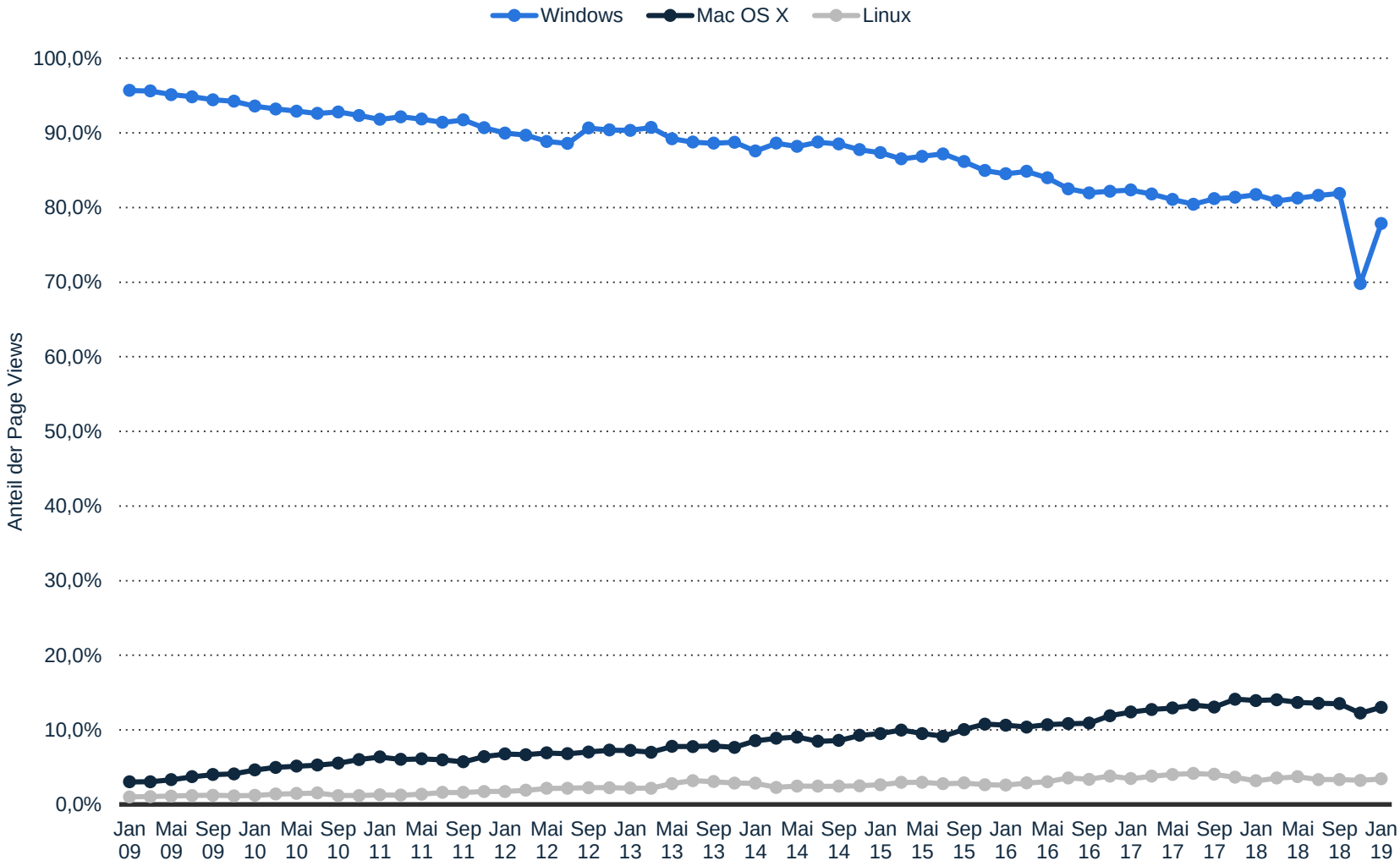


- **Betriebssysteme in Deutschland Januar 2016**



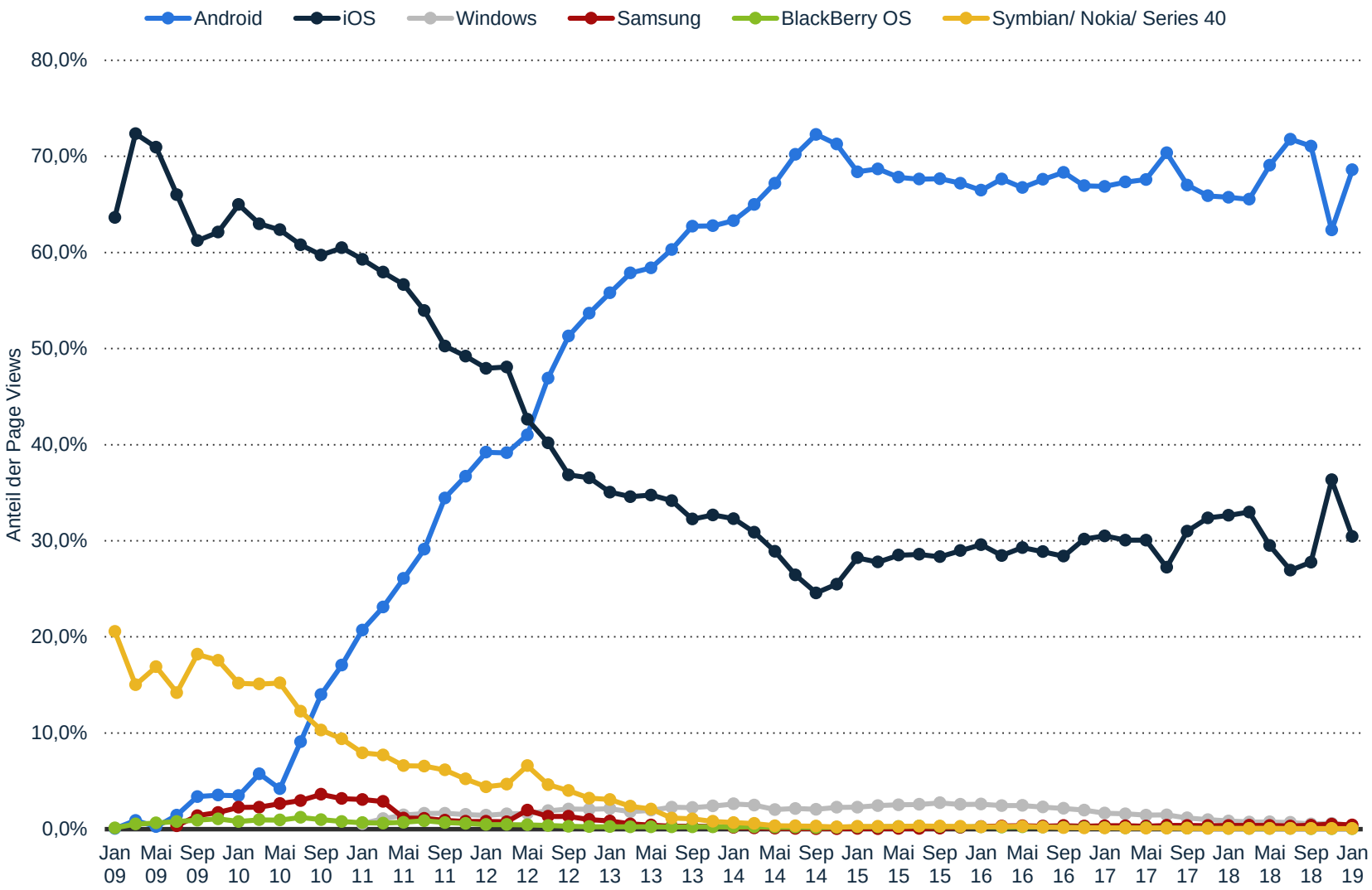
http://de.statista.com/statistik/daten/studie/158102/umfrage/marktanteile-von-betriebssystemen-in-deutschland-seit-2009/

Verbreitung von (Desktop)-Betriebssystemen



http://de.statista.com/statistik/daten/studie/158102/umfrage/marktanteile-von-betriebssystemen-in-deutschland-seit-2009/

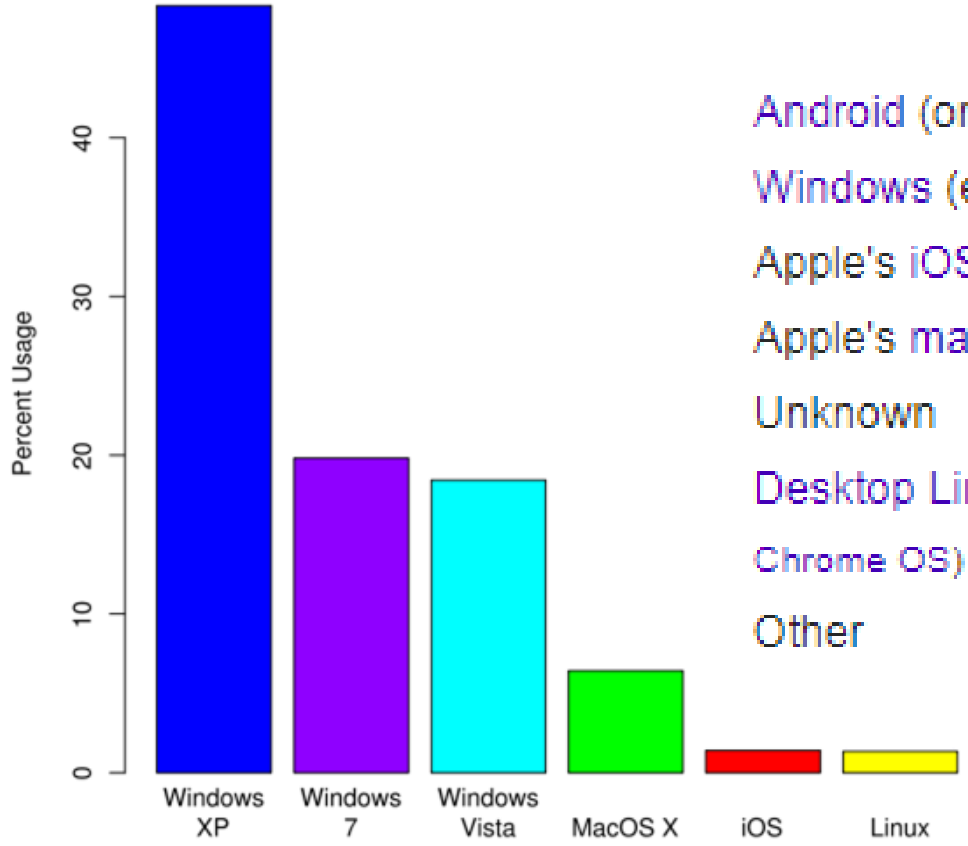
Verbreitung von (Mobilgeräte)-Betriebssystemen



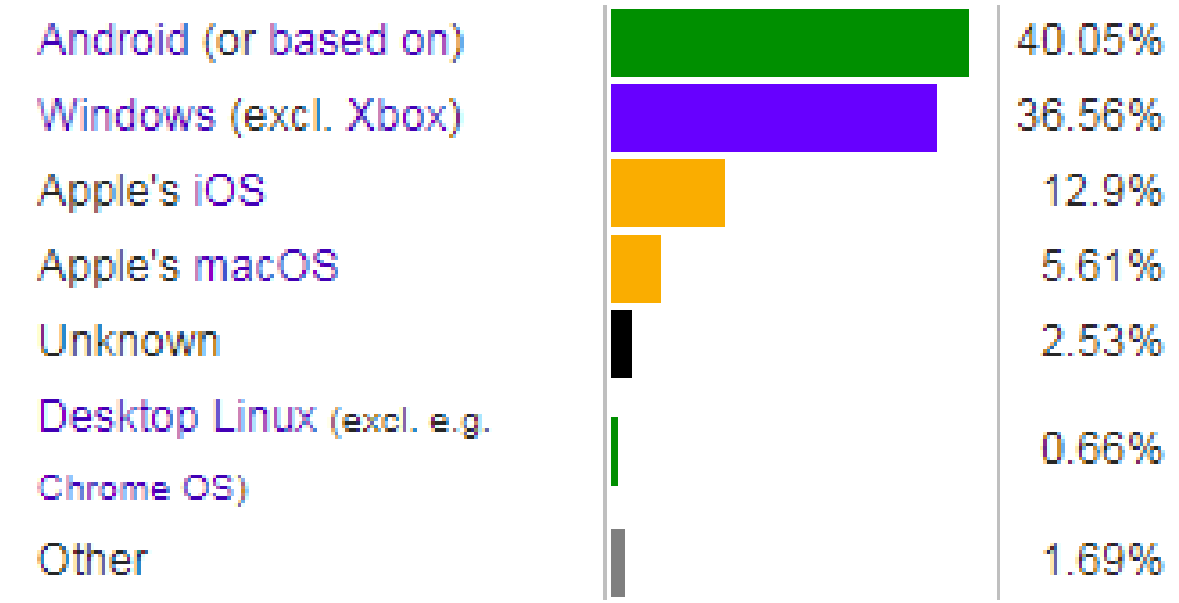
https://de.statista.com/statistik/daten/studie/184332/umfrage/marktanteil-der-mobilen-betriebssysteme-in-deutschland-seit-2009/

Verbreitung von Betriebssystemen

- **Web Client OS (September 2010)**



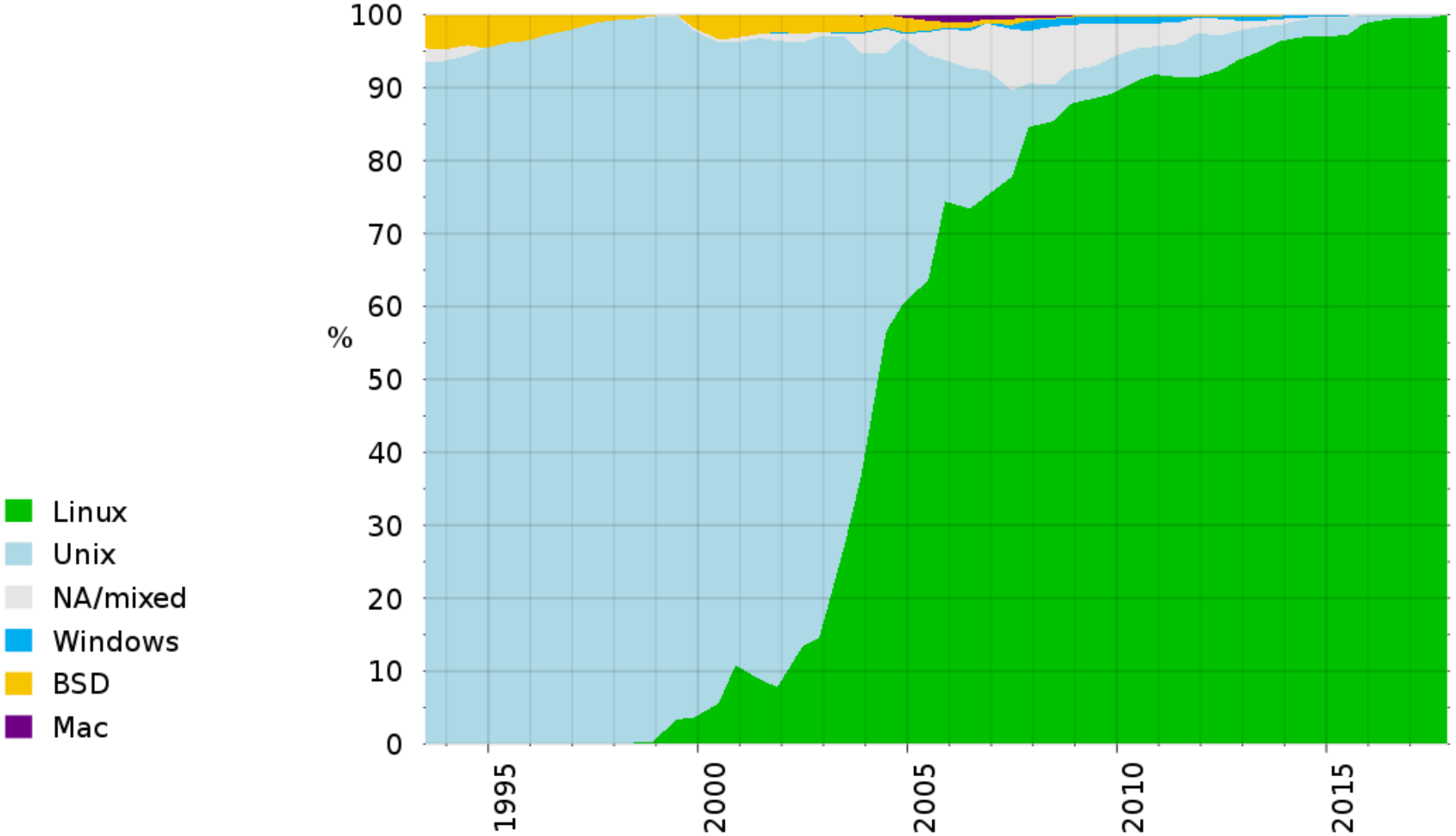
- **Web Client OS (Januar 2018)**



http://en.wikipedia.org/wiki/Usage_share_of_operating_systems

Verbreitung von Betriebssystemen

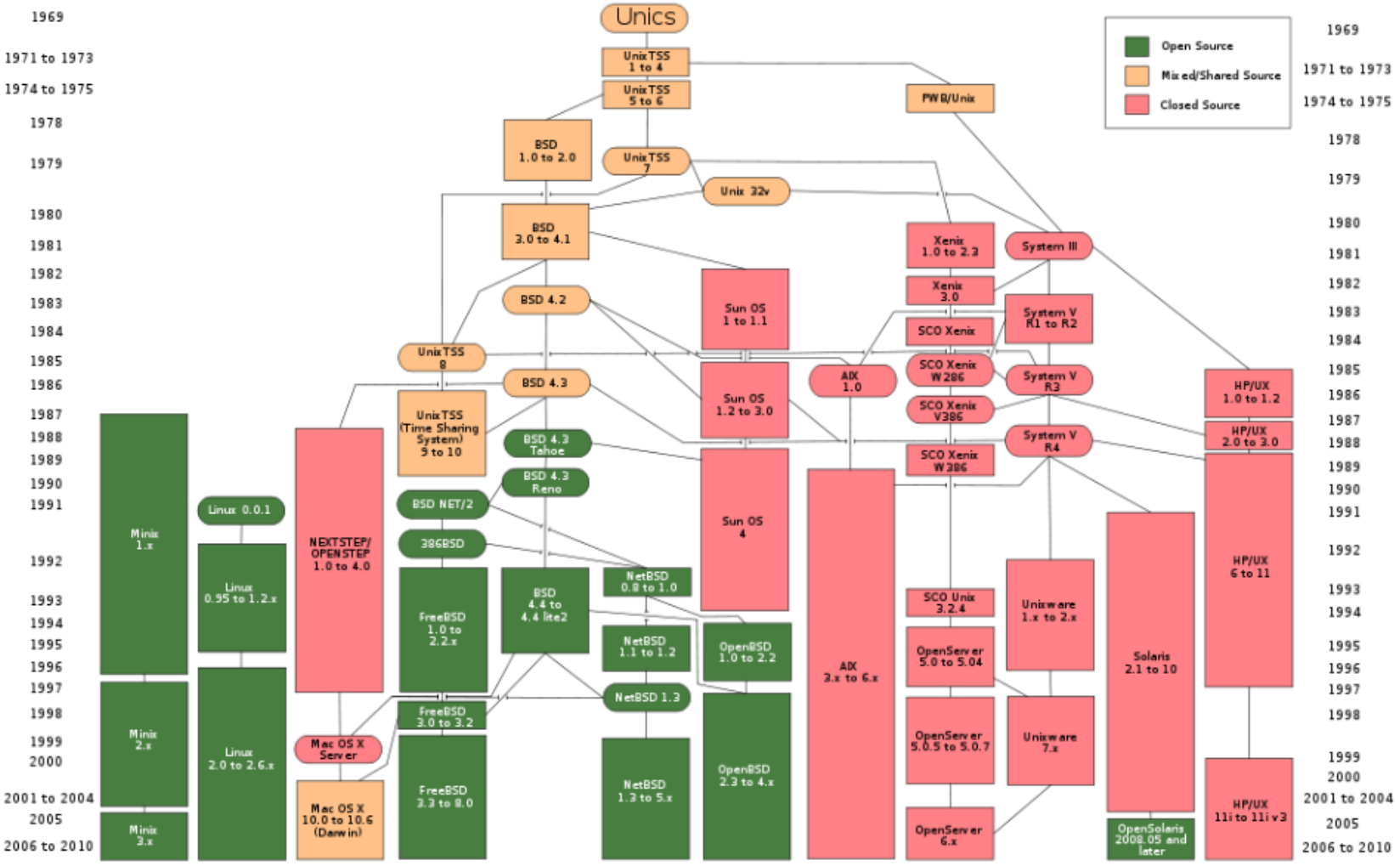
- **Supercomputer**



http://en.wikipedia.org/wiki/Usage_share_of_operating_systems

Vielzahl von Varianten

- **Beispiel UNIX**



http://en.wikipedia.org/wiki/File:Unix_history-simple.svg

1.1 Aufbau von Rechnersystemen

- Hardware, Rechnerarchitektur, Betriebssystem

1.2 Aufbau und Aufgaben eines Betriebssystems

- Generelle Aufgaben, Betriebssystemstruktur, Betriebssystemarten, Interrupts, Systemaufrufe

1.3 Multitasking und Interrupts

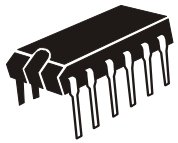
- Multitasking, Interrupt-Handling, Context Switches, Systemaufrufe

Aufgaben des Betriebssystems

- **Wichtige Aufgaben**
  - Verwaltung aktiver Programme (= Prozesse)
  - Verwaltung des Hauptspeichers und der Caches,
    Laden der Prozesse in den Hauptspeicher
  - Steuerung der Abarbeitungsreihenfolge der Prozesse auf der CPU,
    Interrupt-Bearbeitung: Verdrängen von Prozessen von der CPU
  - Kommunikation zwischen Prozessen
  - Zuteilung von Betriebsmitteln (z.B. I/O-Geräte) zu Prozessen,
    Vermeidung bzw. Auflösung von Konflikten bei Betriebsmittelanfragen
  - Dateiverwaltung
  - …
- **Aufgaben/Prinzipien:**
  - *Abstraktion* und *Koordination* (Verwaltung)

Zentrale Aufgaben des Betriebssystems

- **Prozessverwaltung**
  - *Prozess: ein in Ausführung befindliches Programm, welches CPU-Zeit und Hauptspeicher benötigt sowie I/O-Operationen durchführt*
  - Erzeugung/Löschen von Prozessen
  - Aufteilung der CPU-Zeit auf mehrere Prozesse („gleichzeitige“ Ausführung)
  - Interprozess-Kommunikation
  - Synchronisation von Prozessen, Vermeidung/Behebung von Konflikten



Process Control Block

- Prozesszustand
- Prozessnummer
- Programmzähler
- Registerwerte
- Speicherbereiche
- Liste offener Dateien
- …

Zentrale Aufgaben des Betriebssystems

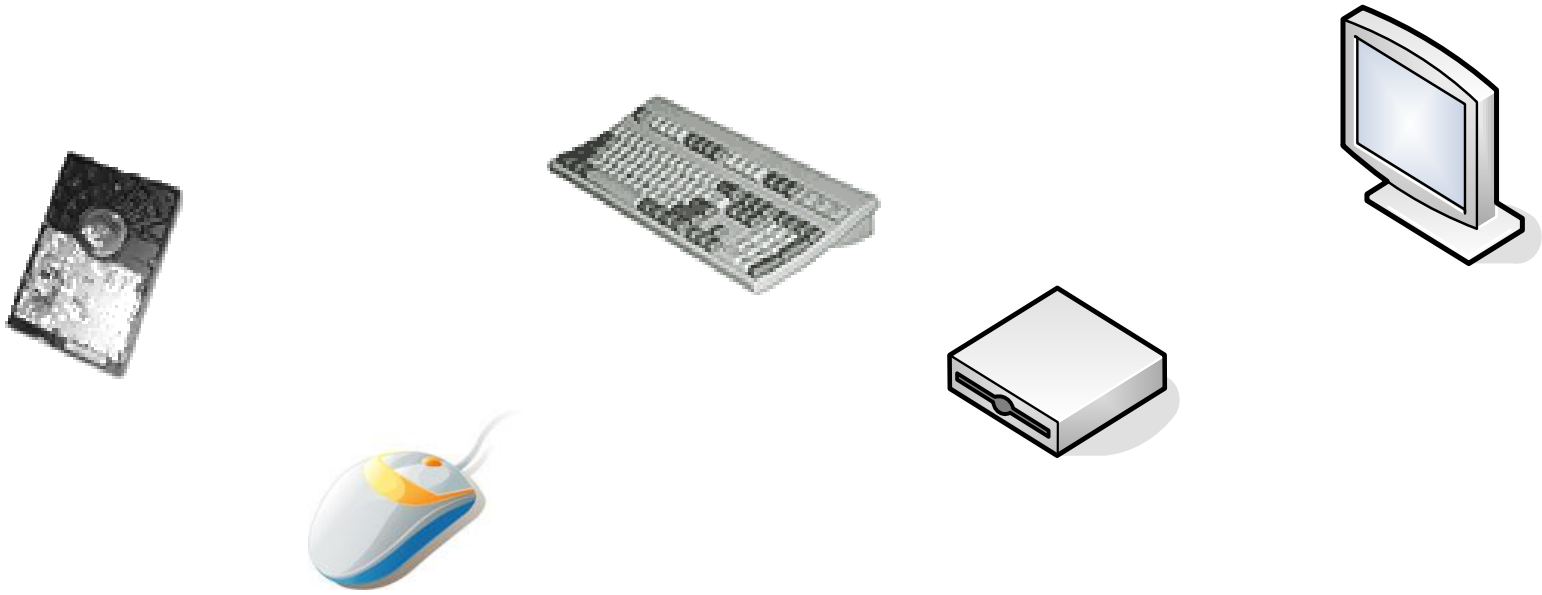
- **Speicherverwaltung**
  - *Alle Informationen/Instruktionen eines Prozesses müssen vor der Ausführung in den Hauptspeicher geladen werden*
  - Verwaltung freier und belegter Speicherbereiche
  - Entscheidung, *welche Prozesse wann* in den Hauptspeicher geladen werden
  - Entscheidung, *wie viel* Speicherplatz einem Prozess zugewiesen/entzogen wird

Zentrale Aufgaben des Betriebssystems

- **Dateiverwaltung**
  - *Das Betriebssystem stellt eine einheitliche, logische Sicht auf einen dauerhaften Speicher (Festplatte) bereit*
  - Geeignete *Abbildung der logischen Struktur auf die physikalische Struktur* des Speichers
  - Verwaltung freier Speicherbereiche, Reservieren von Speicher
  - *Koordination des Zugriffs* mehrerer Prozesse auf den Speicher (Disk-Scheduling)
  - Erstellen, Modifizieren und Löschen von Dateien
  - Überwachung von *Zugriffsrechten*
  - Eventuell automatische Sicherung auf ein weiteres Speichermedium (Backup)

Zentrale Aufgaben des Betriebssystems

- **I/O-System**
  - *Verbergen von Besonderheiten der einzelnen Geräte vor dem Nutzer*
  - Speicherverwaltung für I/O-Geräte (Caching)
  - *Interaktion* eines I/O-Geräts mit der CPU
  - Verwaltung und Bereitstellung von *Gerätetreibern*



- **Zugriffskontrolle**
  - *Das Betriebssystem überwacht, welche Prozesse/Nutzer auf welche Betriebsmittel zugreifen dürfen*
  - Vergabe von Zugriffsrechten für einzelne Benutzer, Zuordnung zu Prozessen/Dateien/…
  - Zugriffsrechte für Gruppen, um einer ganzen Benutzergruppe die gleichen Rechte zuteilen zu können
  - Zusätzlich: Schutz gegen interne und externe Angriffe (Würmer, Viren, …)

Zentrale Aufgaben des Betriebssystems

- **Schnittstelle**
  - Implementierung der bisherigen (hardwarenahen) Funktionalitäten als Betriebssystem-Kernel
  - Zugriff auf den Kernel über definierte Schnittstelle
  - Ausführung von *Systemaufrufen* über diese Schnittstelle: Programmausführung, Dateimanipulation, Kommunikation (auch über ein Netzwerk), Zugriffsrechte, …
  - Realisierung als Kommando-Interpreter (Shell), graphisches Interface (Desktop/Icons), …
  - *Systemprogramme* (z.B. Texteditor) als komfortable Nutzungsumgebung für Benutzer

Kernel

- **“Kern” des Betriebssystems: *Kernel***
  - Umfasst wesentliche Dienste des Betriebssystems (s.o.)
  - Sollte möglichst immer im Hauptspeicher sein
- **Implementierung**
  - Monolithisch
  - Mikrokernel
  - Hybridkernel
- **Generell: Unterscheidung zwischen**
  - Kernel-Mode
  - User-Mode

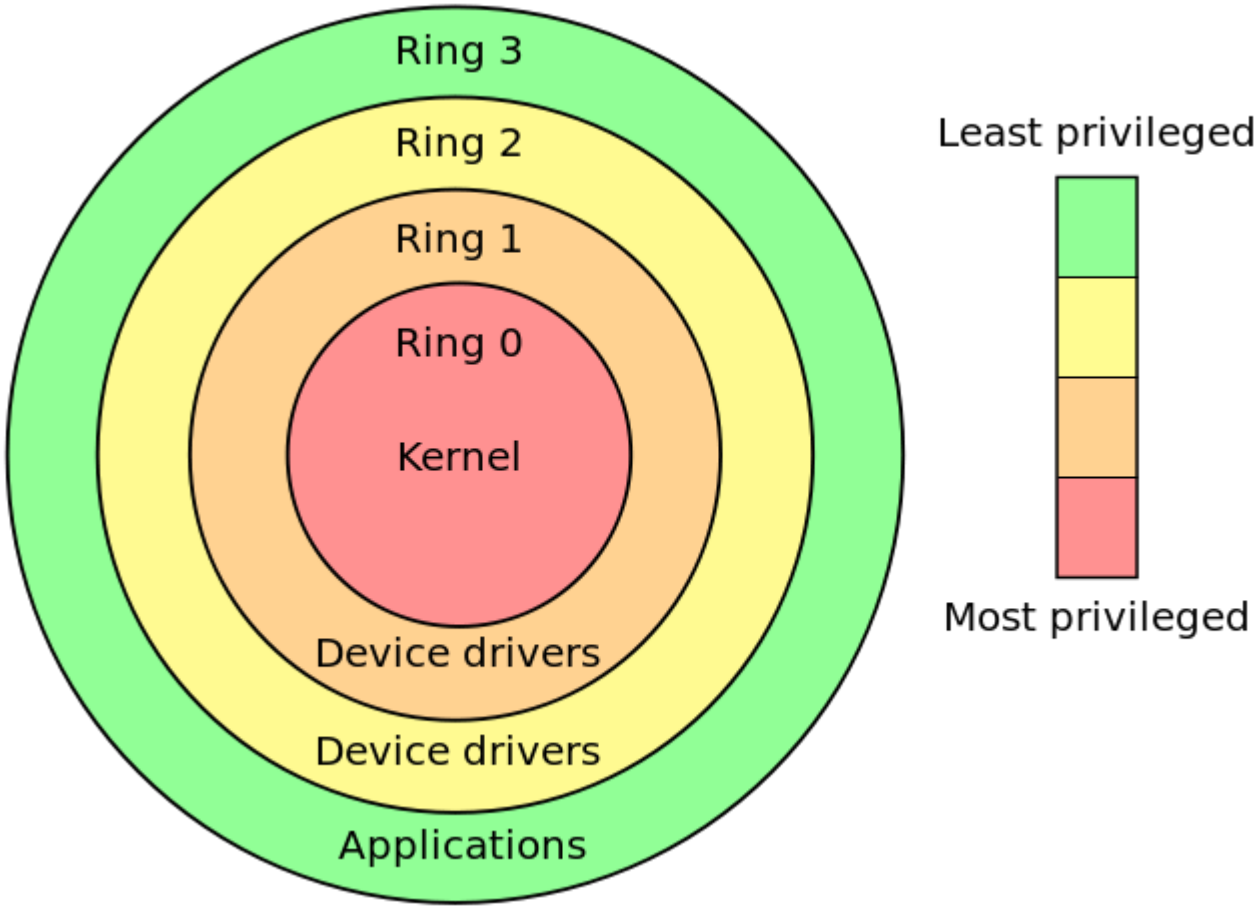
Kernel- und User-Mode

- ***Kernel-Mode***
  - Privilegierter Modus
  - Ausführung der Module des Kernels
  - Schutz von Datenstrukturen des Kernels
    (exklusiver Zugriff auf bestimmte Speicherbereiche, *Kernel-Space*)

- ***User-Mode***
  - Ausführung von Anwendungsprogrammen
  - Kein Zugriff auf Kernel-Space

- **Umschaltung zwischen Kernel- und User-Mode**
  - Über spezielle Befehle
  - Kostet Zeit

- **Gruppierung von Zugriffsrechten in *Ringe***



http://en.wikipedia.org/wiki/File:Priv_rings.svg

Monolithischer Kernel

- **Implementierung einzelner Funktionalitäten als Module**
  - Alle Module laufen im Kernel-Mode
  - Relativ fehleranfällig, da alle Betriebssystemkomponenten im privilegierten Modus laufen

- **Gängiges Implementierungskonzept: *Schichten***
  - Modularität durch Strukturierung der gesamten Funktionalität in unabhängige Bereiche
  - Eine Schicht kann nur auf die tiefer liegenden Schichten zugreifen und stellt den höheren Schichten selbst Dienste zur Nutzung bereit
    - Aber: alle Schichten im privilegierten Modus
  - Komplexität des Gesamtsystems wird reduziert

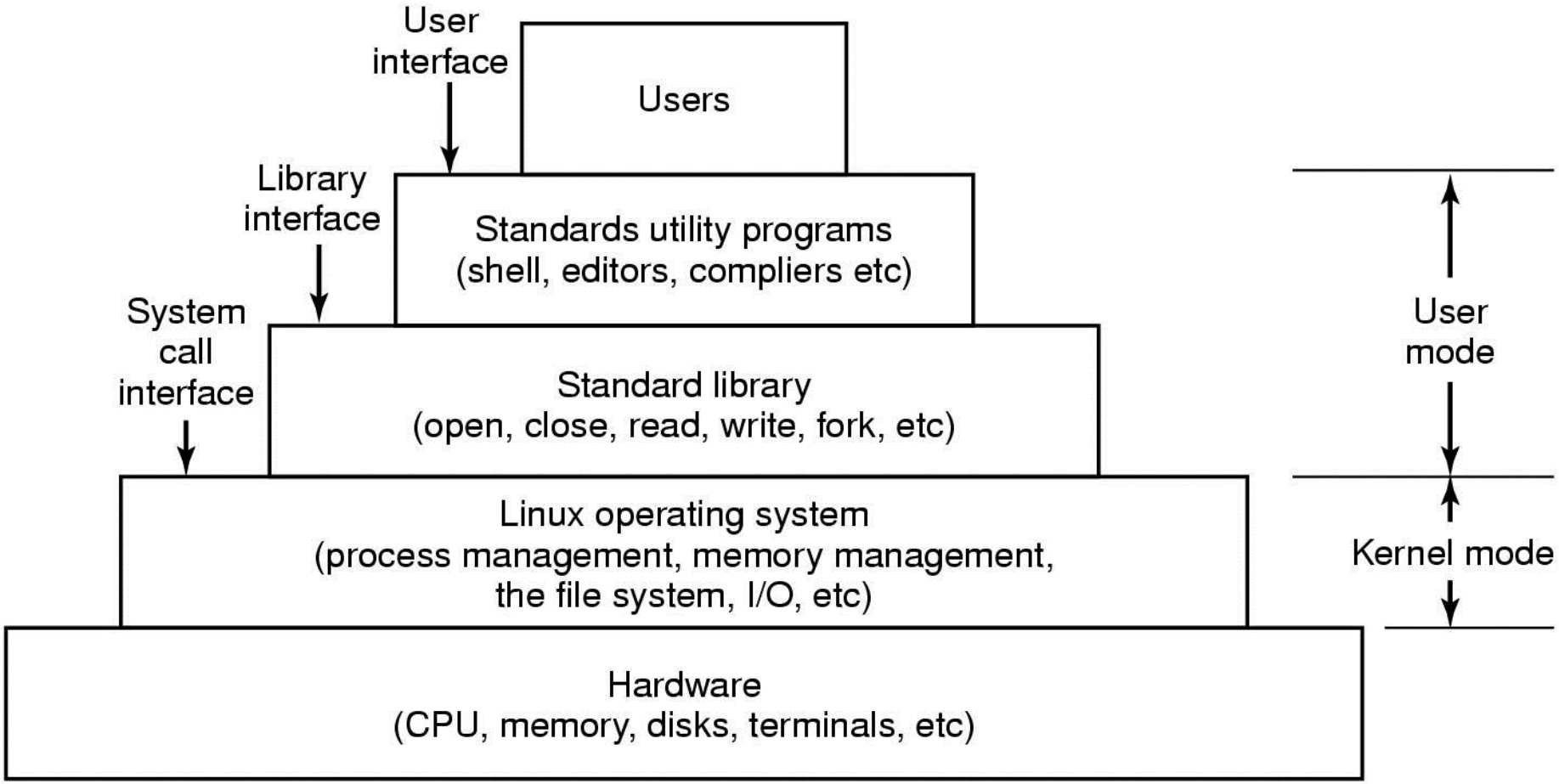
- **Erste Umsetzung des Schichtenkonzepts: THE**
  - THE = Technische Hogeschool Eindhoven, Dijkstra, 1968
  - 5 Schichten:

n	Funktionalität
1	Prozessverwaltung, CPU-Scheduling
0	Hardware

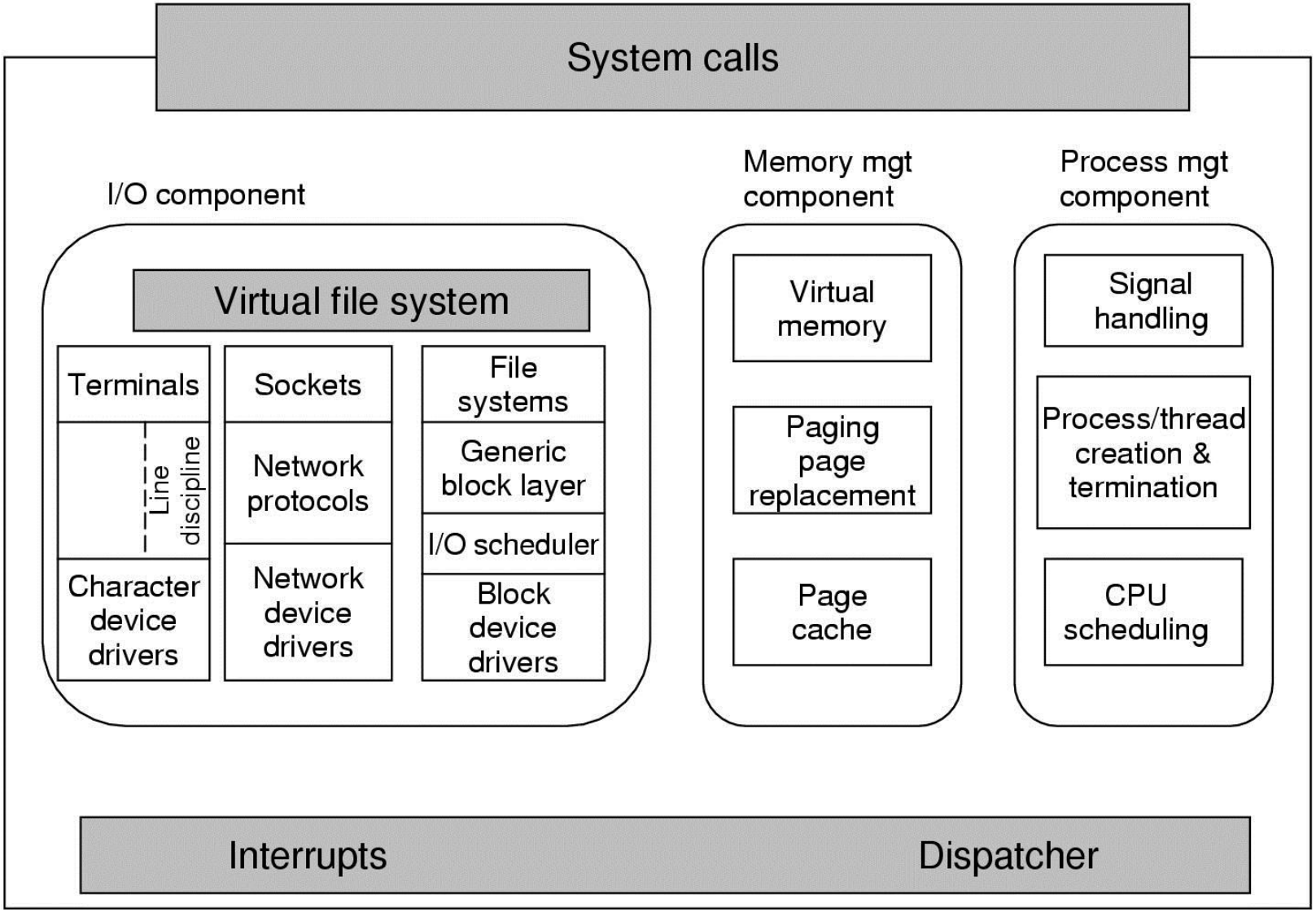
- **Erste Umsetzung des Schichtenkonzepts: THE**
  - THE = Technische Hogeschool Eindhoven, Dijkstra, 1968
  - 5 Schichten:

n	Funktionalität
5	Benutzerprogramme
4	I/O-Verwaltung (Lochstreifen, Drucker)
3	Administratorkonsole
2	Speicherverwaltung
1	Prozessverwaltung, CPU-Scheduling
0	Hardware

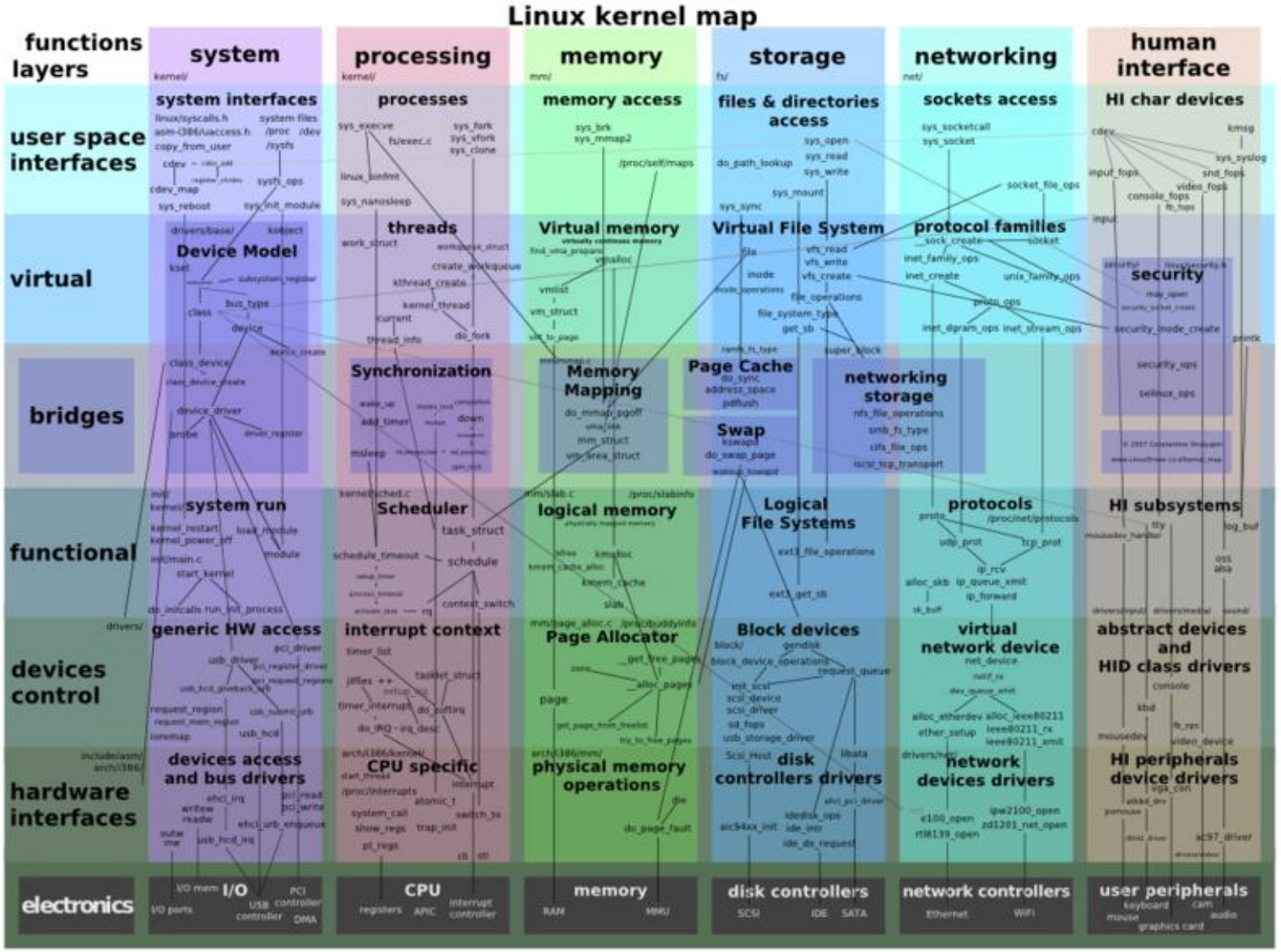
Betriebssystemstruktur bei Unix



Betriebssystemstruktur bei Unix

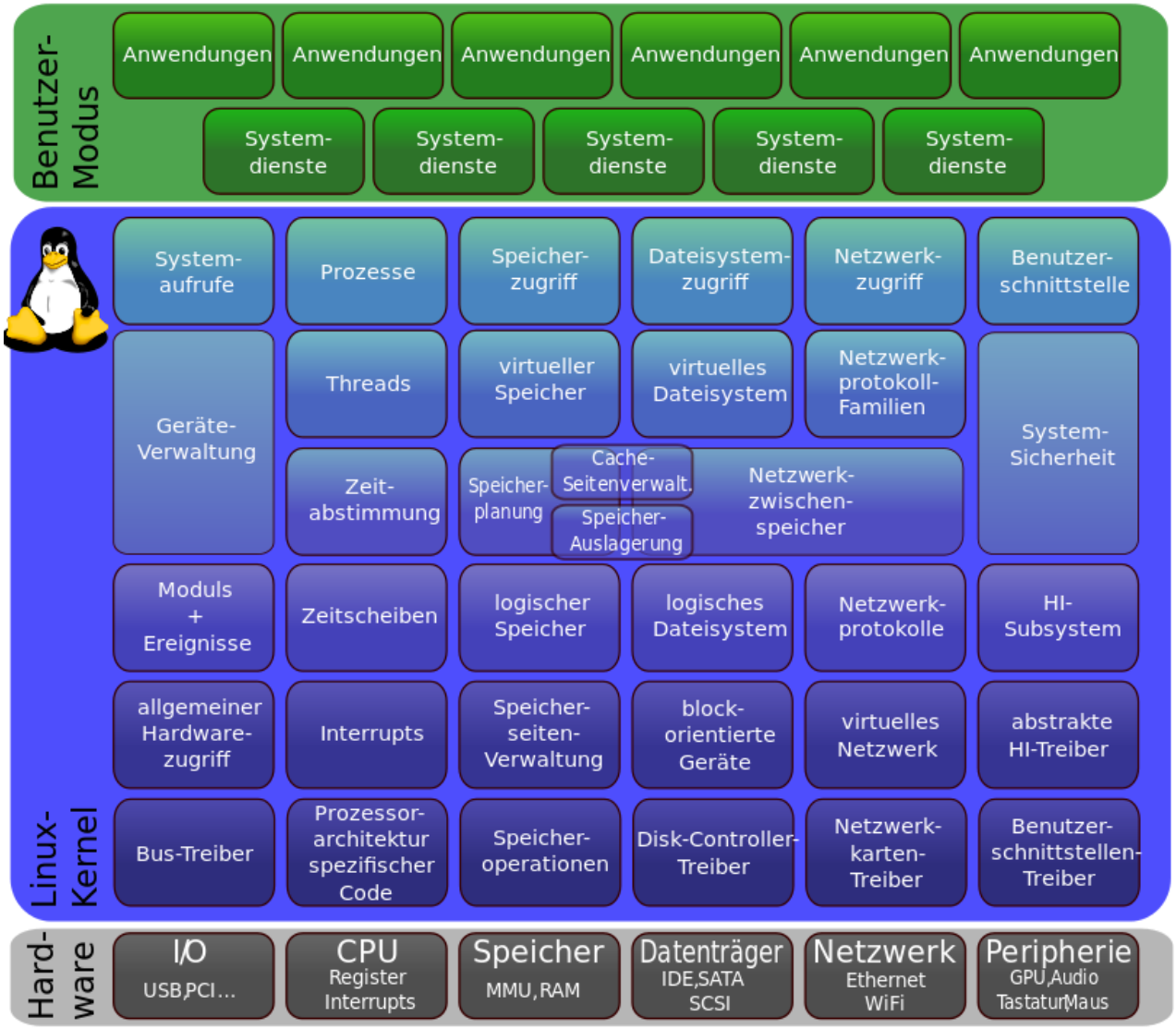


Grobstruktur Linux-Kernel



http://en.wikipedia.org/wiki/File:Linux_kernel_map.png

Grobstruktur Linux-Kernel



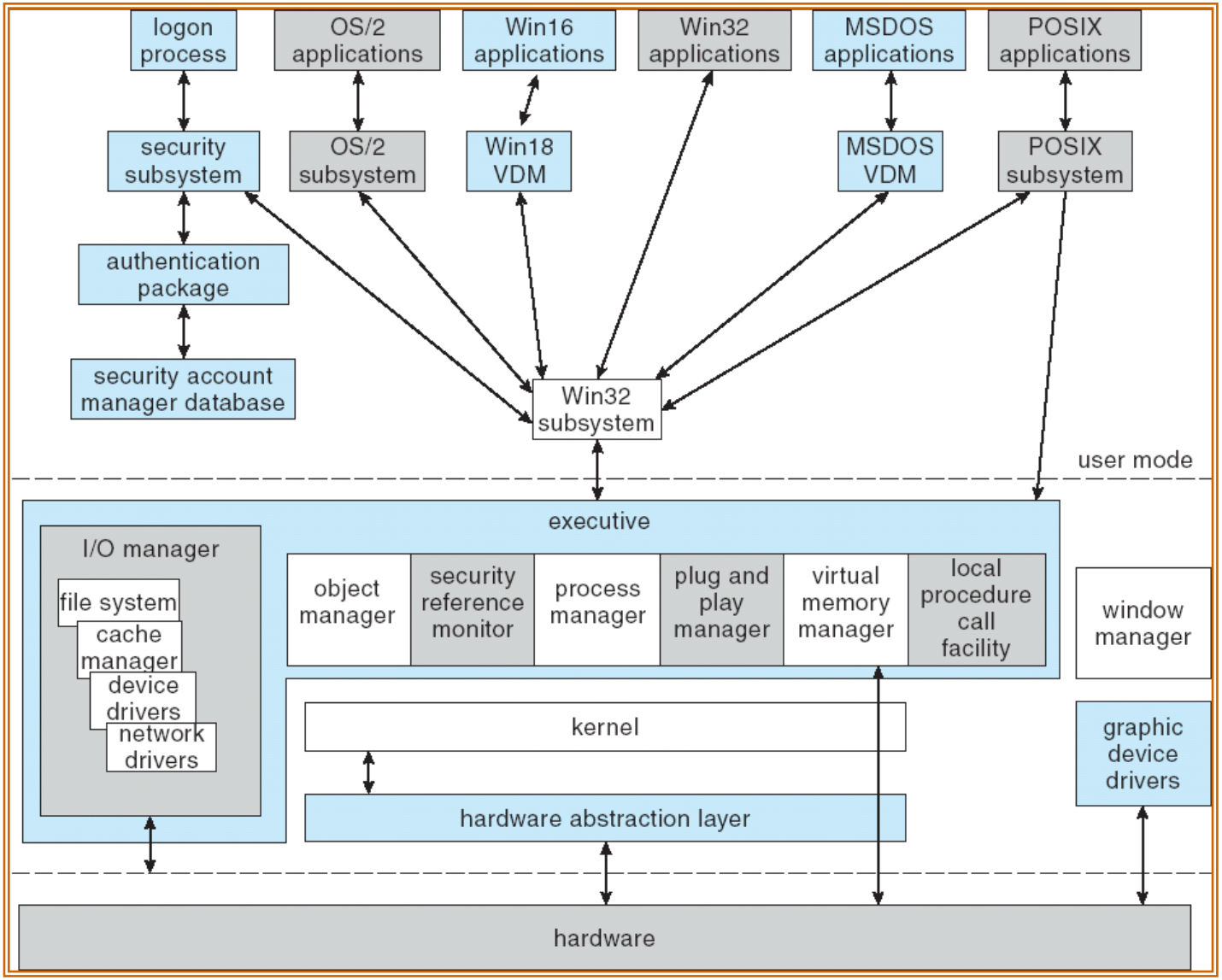
http://commons.wikimedia.org/wiki/File:Linux_Kernel_Struktur.svg

Mikrokernel Und Hybridkernel

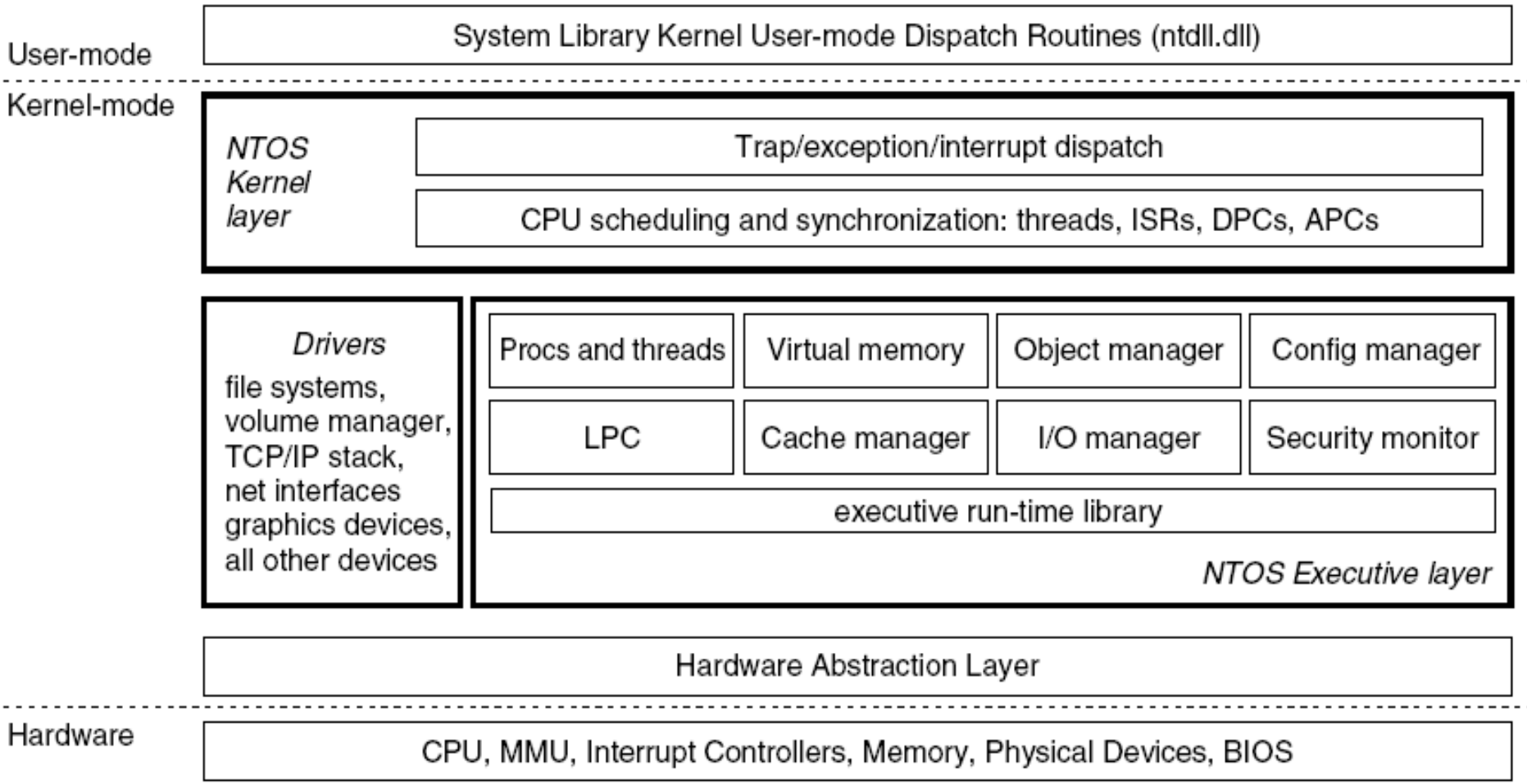
- ***Mikrokernel***
  - Verfügt nur über grundlegende Funktionalitäten
  - Weitere Funktionalitäten werden als eigene Prozesse oder Programmbibliotheken in den User-Mode ausgelagert
  - Weniger fehleranfällig, aber langsamer

- ***Hybridkernel***
  - Mischung aus monolithischem und Mikrokernel
    - “Weniger” Funktionalitäten in User-Mode ausgelagert
    - Vereint Vorteile beider Welten

Betriebssystemstruktur bei Windows XP

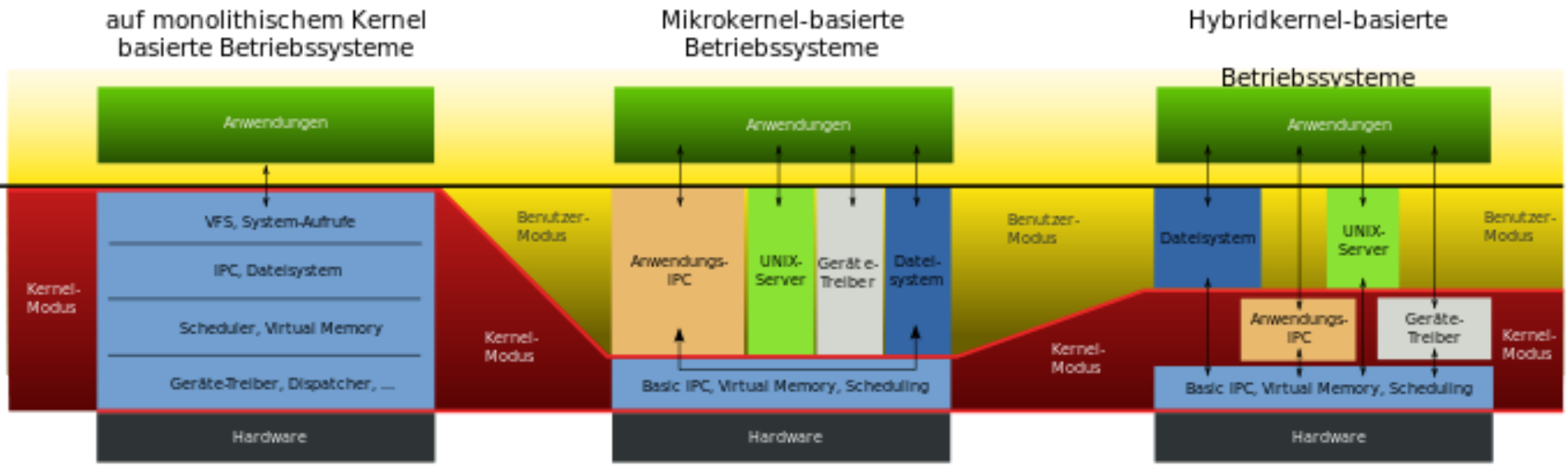


Betriebssystemstruktur bei Windows Vista



Monolithischer, Mikro- und Hybridkernel

- **Vergleich der drei Varianten**

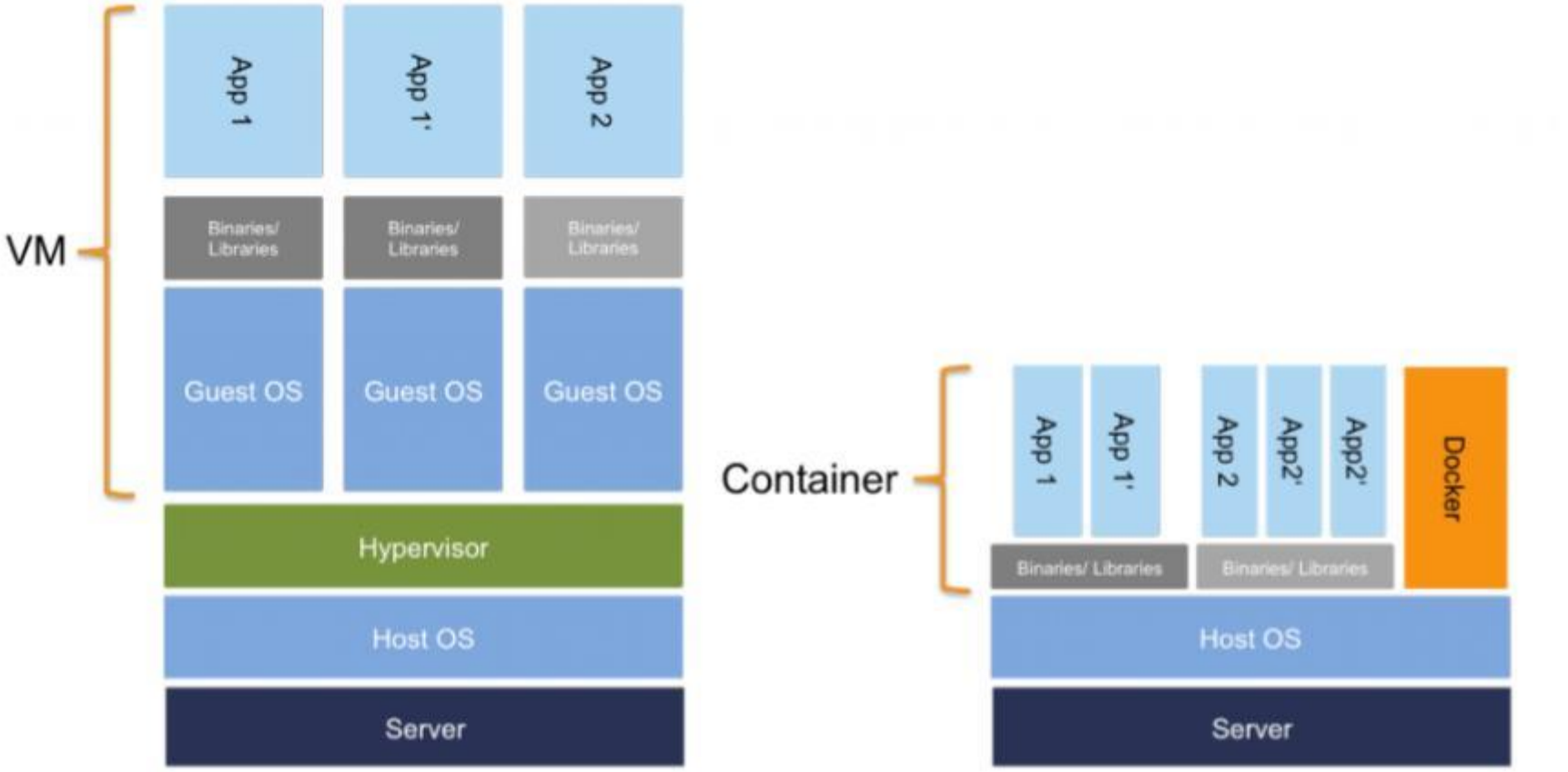


http://commons.wikimedia.org/wiki/File:OS-structure.de.svg

- **Neuere Technologie: *Virtualisierung***
  - “Abstraktion von Ressourcen mit Hilfe von Software”
  - Schaffen einer Softwareumgebung, die sich wie eine Hardwareumgebung verhält: *virtuelle Maschine*
  - Erlaubt die Betriebssystemvirtualisierung
    - Schaffen virtueller Computer
    - Unabhängige Computer können auf einer einzelnen Maschine laufen

- ***Container* als andere Form der Virtualisierung**
  - Zielt auf Bereitstellung einer Ausführungsumgebung für einzelne Anwendungen ohne Abhängigkeiten vom Gastsystem ab
    - Ermöglicht Ausführung von Anwendungen unabhängig vom System, einfache Portierung zwischen Gastrechnern

Betriebssystem, Virtualisierung und Container



Quelle: Docker, Crisp Research, 2014

Weitere Betriebssystemarten

- **Viele weitere Betriebssystemkonzepte**
  - Verteilte Betriebssysteme
  - Middleware
  - Cloud Computing

Performanz & Zuverlässigkeit

- **Wichtige Kriterien bei der Implementierung von Betriebssystemen**
  - *Performanz (Performance)*
    - Leistungsfähigkeit/Geschwindigkeit des gesamten Rechners hängt vom Betriebssystem ab
  - *Zuverlässigkeit (Reliability)*
    - Häufigkeit von Systemabstürzen, Systemstillstand, Systemneustart

- **Performanz und Zuverlässigkeit hängen von Art des Betriebssystems ab**
  - Mikrokernel zuverlässiger, aber auch langsamer
  - Monolithische Kernel unzuverlässiger, aber schneller

Zuverlässigkeit: Fehlerursachen

- **Hauptursache: *Kernel-Erweiterungen***
  - Z.B. Treiber
    - Treiberprogrammierer haben oft weniger Erfahrung als Kernel-Experten
    - Fehler in Treibern verursachen…
      - fehlerhafte Speicherzugriffe – Reboot, Bluescreen
      - Konflikte beim Zugriff auf Ressourcen – Einfrieren des Systems
    - Vielzahl an Geräten/Treibern macht vollständiges Testen sehr schwer
- **Aber auch: fehlerhafte Implementierung von Kernel / Betriebssystemdiensten**
  - *Mehrere Millionen Zeilen Code*
  - *Keine Fehler-Isolation*
    - Kernel-Code kann z.B. wichtige Datenstrukturen anderer Komponenten überschreiben

Zuverlässigkeit: Lösungsansätze

- **Verwendung eines Mikrokernels**
  - Kernel-Erweiterungen haben eingeschränkten Zugriff
  - Reduktion der Performanz
- **Isolierte Ausführung einzelner Prozesse**
  - Verhindert Fehler-Propagation, aber immer noch Crash einzelner Prozesse möglich
- **Error Recovery**
  - Z.B. Neustart abgestürzter Komponenten
  - Problem: Wiederherstellung des aktuellen Zustands
- **Code-Analyse / Laufzeit-Tests**
  - Finden nur einen Teil der Fehler
- **Andere Möglichkeiten?**

1.1 Aufbau von Rechnersystemen

- Hardware, Rechnerarchitektur, Betriebssystem

1.2 Aufbau und Aufgaben eines Betriebssystems

- Generelle Aufgaben, Betriebssystemstruktur, Betriebssystemarten, Interrupts, Systemaufrufe

1.3 Multitasking und Interrupts

- Multitasking, Interrupt-Handling, Context Switches, Systemaufrufe

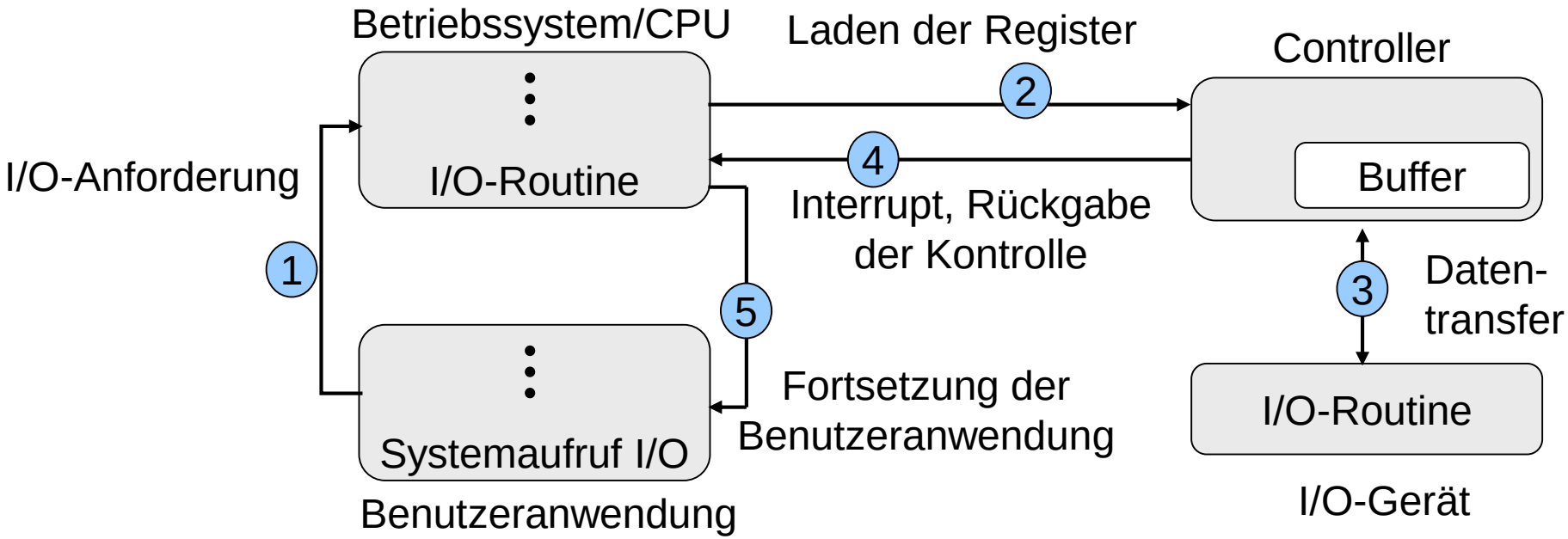
- **„Klassische“ Systeme: Singletasking**
  - Die CPU kann nur einen Prozess nach dem anderen ausführen
- **Multiprogrammierung**
  - Verwaltung mehrerer Prozesse gleichzeitig durch das Betriebssystem
  - Das Betriebssystem bedient bei Eintreten bestimmter Ereignisse andere Prozesse
- **Timesharing-Systeme (*Multitasking*)**
  - Interaktion mit dem Computer im Vordergrund
  - Verarbeite mehrere Prozesse „gleichzeitig“
  - Prozesse können nach Ablauf einer bestimmten Zeiteinheit unterbrochen werden

- **Offene Punkte:**
  - Wie kann die Ausführung eines Prozesses auf der CPU unterbrochen werden, wenn die Zeitscheibe abgelaufen ist?
  - Wie kann ein I/O-Gerät die CPU informieren, dass Daten vorliegen, die verarbeitet werden müssen?

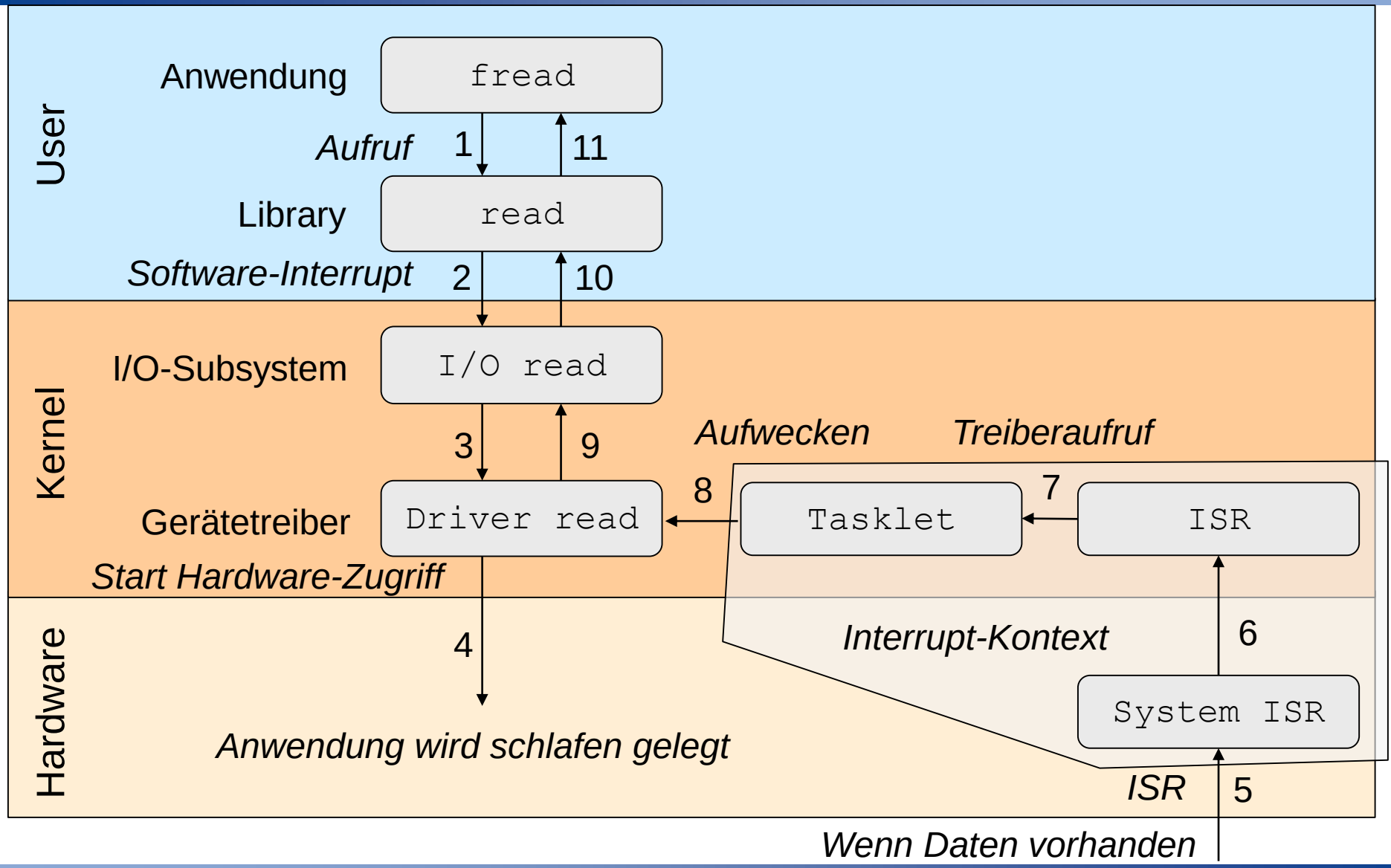
- **Lösungsmöglichkeiten generell:**
  - *Polling*
  - *Interrupts*

Ablauf eines Interrupts

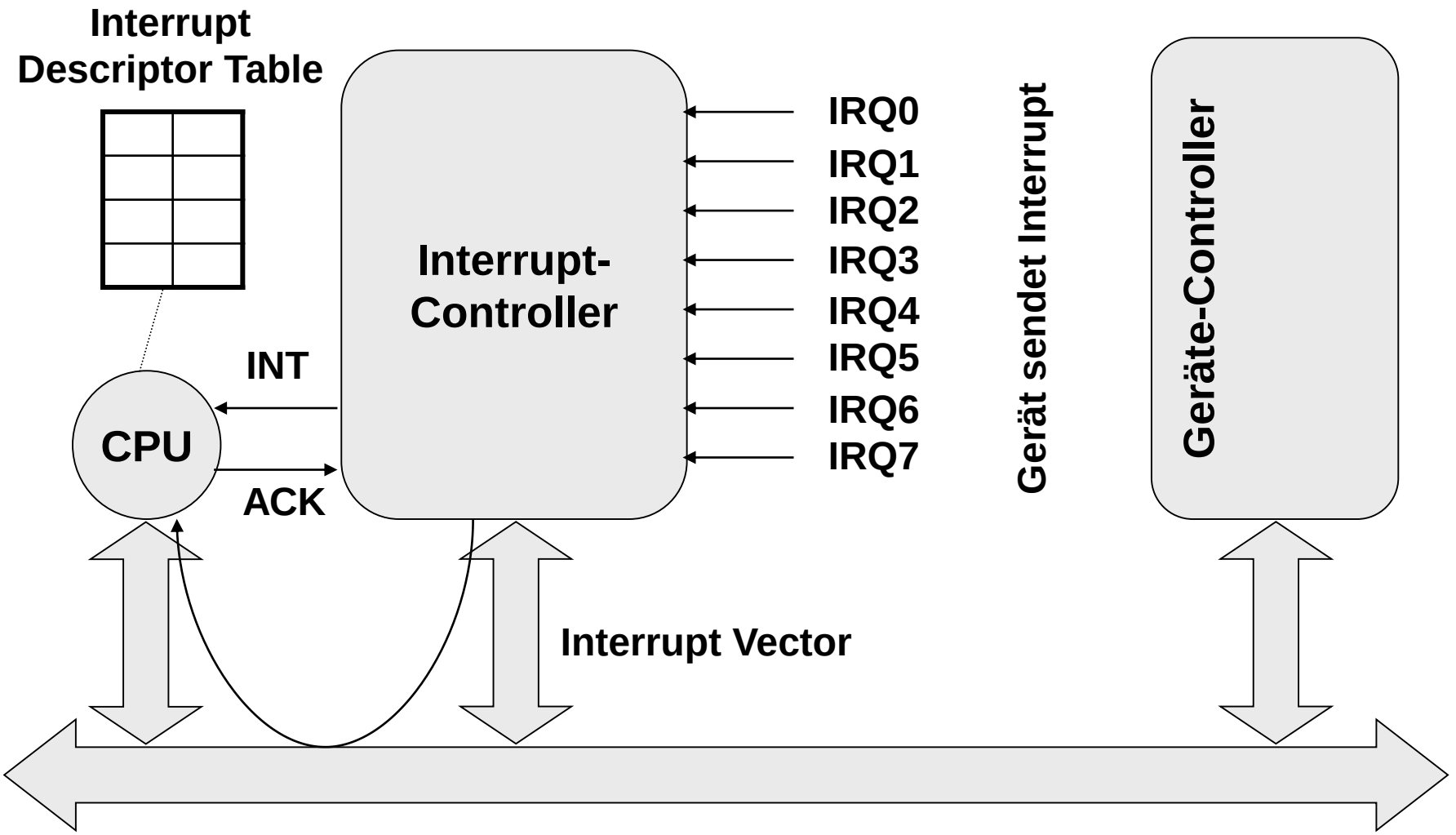
- **Anwendungen und Controller informieren die CPU durch *Interrupts* über Ereignisse:**
  - CPU stoppt die aktuelle Operation, speichert ihren Zustand
  - Starten der neuen Routine (*Interrupt Service Routine, ISR*)
  - Danach: CPU setzt unterbrochene Operation an gleicher Stelle fort



Ablauf eines Interrupts (Beispiel)



- **CPU muss auf Interrupts geeignet reagieren**
  - *Interrupt-Controller* auf Motherboard
  - Interrupt Controller verfügt über *Interrupt Request Lines (IRQs)*, über welche Geräte Interrupts signalisieren können
  - Signalisierung von Interrupt-Controller an CPU über Interrupt Control Line mittels Index (*Interrupt Vector*) zu einer Tabelle von Adressen, die auf *Interrupt-Handler* zeigen
    - CPU verfügt über spezielles Register, was bei Anliegen eines Interrupts gesetzt wird
    - Register wird am Ende jedes Maschinenbefehls geprüft
    - Interrupt Handler: Routine, die die Art des Interrupts bestimmt, das auslösende Gerät ermittelt, die entsprechenden Operationen ausführt und anschließend den Zustand der CPU vor dem Interrupt wiederherstellt

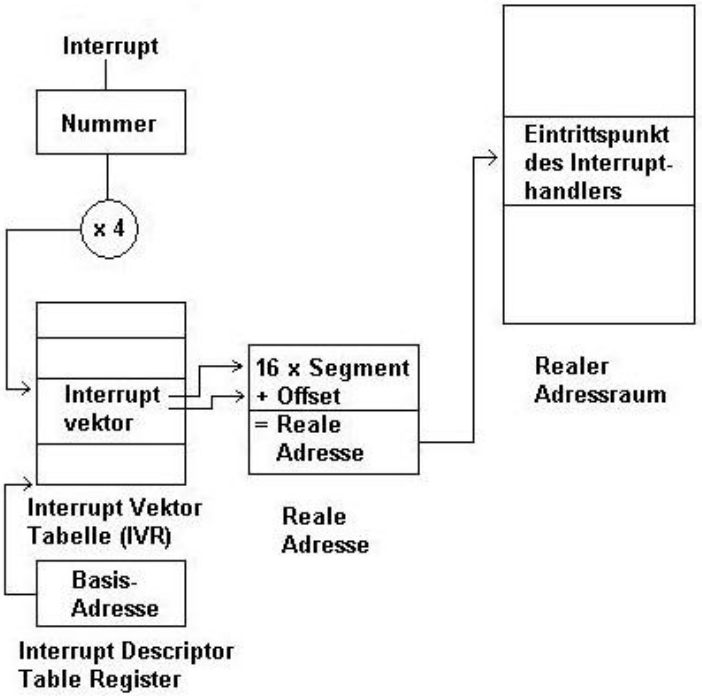

Interrupt
Descriptor Table
Interrupt-
Controller
IRQ0
IRQ1
IRQ2
IRQ3
IRQ4
IRQ5
IRQ6
IRQ7
Gerät sendet Interrupt
Geräte-Controller
INT
CPU
ACK
Interrupt Vector

IRQ-Belegung – Beispiel

Master	Slave	Priorität	Verwendung / Belegung
IRQ0		höchste	Timer
IRQ1			Tastatur-Controller
IRQ2			Verbindung zum Slave-Controller
	IRQ8		Echtzeituhr
	IRQ9		Peripherie-Geräte
	IRQ10		Peripherie-Geräte
	IRQ11		Peripherie-Geräte
	IRQ12		PS/2-Maus
	IRQ13		Coprozessor 80x87
	IRQ14		Festplattencontroller
	IRQ15		Zweiter Festplattencontroller
IRQ3			COM2 (zweite serielle Schnittstelle)
IRQ4			COM1 (erste serielle Schnittstelle)
IRQ5			LPT2 (zweite parallele Schnittstelle) / Soundkarte
IRQ6			Diskettencontroller
IRQ7		niedrigste	LPT1

Software-Interrupts

- **Interrupt Vectors**
  - Silberschatz



Vector Number	Description
0	Divide Error
1	Debug Exception
2	Null Interrupt
3	Breakpoint
4	INTO-detected Overflow
5	Bound Range Exception
6	Invalid Opcode
7	Device Not Available
8	Double Fault
9	CoProcessor Segment Overrun (reserved)
10	Invalid Task State Segment
11	Segment Not Present
12	Stack Fault
13	General Protection
14	Page Fault
15	(Intel reserved, do not use)
16	Floating-Point Error
17	Alignment Check
18	Machine Check
19 - 31	(Intel reserved, do not use)
32 – 255	*Maskable Interrupts*

- ***Komplexere Interrupt-Behandlung*:**
  - Während der Ausführung zeitkritischer Operationen sollten keine Interrupts bearbeitet werden
  - Verwendung von *Prioritätsleveln* zu Interrupts, damit unwichtige Interrupts verzögert werden können oder ein Interrupt durch einen wichtigeren unterbrochen werden kann
  - Zwei Arten von Interrupts
    - *Nicht-maskierbare Interrupts* – Interrupt-Vektor 0 – 31; reserviert für Systemereignisse, die immer bearbeitet werden müssen
    - *Maskierbare Interrupts* – ab Interupt-Vektor 32; kann von der CPU vor der Ausführung kritischer Befehlssequenzen ausgeschaltet werden

- **Zugangspunkte zum Betriebssystem wohldefiniert**
  - Zugriff auf den Kernel über *Systemaufrufe (System Calls)*
    - Programmausführung, Dateimanipulation, Kommunikation (auch über ein Netzwerk), Zugriffsrechte, …
    - Systemaufruf = Software-Interrupt (*Trap*)
      - Umschalten vom User-Mode in den Kernel-Mode
    - Werden meist in Bibliotheken bereitgestellt
  - Realisierung als Kommando-Interpreter (Shell), graphisches Interface (Desktop/Icons), …
  - *Systemprogramme* (z.B. Texteditor) als komfortable Nutzungsumgebung für Benutzer

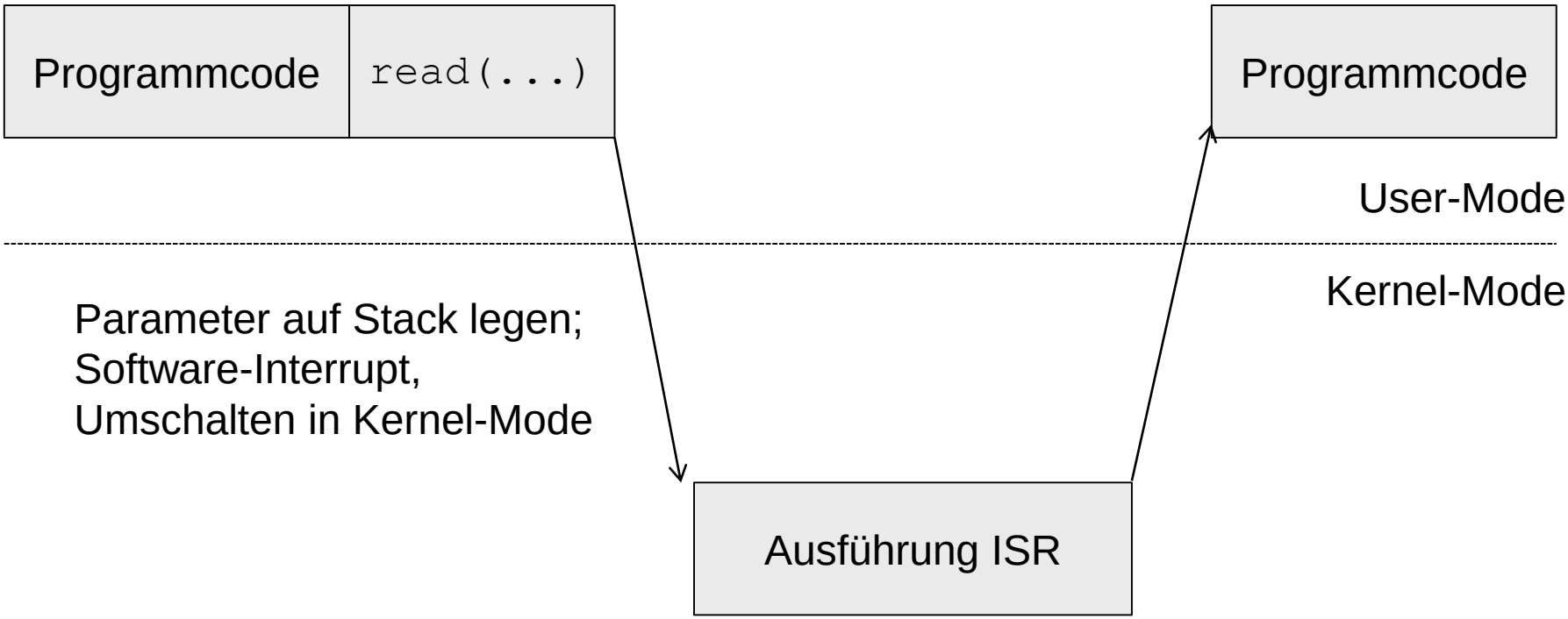
- **Kommunikation eines Benutzerprozesses mit dem Betriebssystem durch Systemaufrufe**
  - Systemaufrufe sind als Prozeduren (in C oder C++, früher Assembler) in einer *Bibliothek* verfügbar
  - Beispiel: Lesen einer Datei

```
count = read (filename, buffer, n_bytes)
```

  - Effekt: lese eine Anzahl von `n_bytes` Bytes aus der Datei `filename` in den Bufferspeicher `buffer`. Nach dem Lesen sollte `count` = `n_bytes` sein, andernfalls ist ein Fehler aufgetreten

Bearbeitung des Systemaufrufs `count = read(...)`

- **Bearbeitung des Aufrufs:**



- **Weitere Beispiele für Systemaufrufe**
  - Prozesskontrolle (create/terminate process, wait/signal event, allocate/free memory)
  - Prozesskommunikation (create/delete communication connection, send/receive message)
  - Dateisystem (create/delete files, open/close file, read/write file)
  - Gerätemanipulation (request/release device, read/write/reposition device)
  - Abruf von Informationen (get attributes of process/file/date, get/set time/date)
  - …

Systemsoftware?

- **Verschiedene Definitionen:**
  - BIOS und Firmware
  - Betriebssystem
  - Shell: Prozesskontrolle, Dateimanipulation, Gerätemanipulation, Kommunikation, Informationsabruf
  - Compiler, Linker, Debugger
  - Utilities (Festplattenverwaltung, Backups, Antivirenprogramme, …)

Themenübersicht

- **Betriebssysteme und Systemsoftware**
  - Betriebssysteme: Aufbau und Aufgaben
  - Shell- und C-Programmierung
  - Prozesse und Threads, Prozessverwaltung und -kommunikation
  - CPU-Scheduling
  - Prozesssynchronisation, Deadlocks
  - Speicherverwaltung, virtueller Speicher
  - Dateisystem, Zugriffsrechte und I/O-System
  - Kommunikation, Verteilte Systeme