

# Strukturierung von Betriebssystemen

---

## Betriebssysteme WS 2009/2010



**Jörg Kaiser**  
**IVS – EOS**

**Otto-von-Guericke-Universität Magdeburg**

# Allgemeines Entwurfsprinzip: "Weniger ist mehr"

---

Occam's Razor: "Plurality should not be assumed without necessity."  
(William of Ockham, ca. 1285-1349)



C.A.R. Hoare: "There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult."



A. Einstein: "Mache die Dinge so einfach wie möglich - aber nicht einfacher"

A. de Saint Exupérie: ".. la perfection n'est pas atteinte quand il n'y a plus rien à ajouter mais quand il n'y a plus rien à enlever!"



# Allgemeine Entwurfs- und Strukturierungsprinzipien und -techniken

---

**Abstraktion:** Reduktion der Komplexität

**Trennung von Strategie und Mechanismus:** Flexibilisierung

**Orthogonalität:** Unabhängigkeit und Komponierbarkeit

**Modularisierung:** Kapselung und Schnittstelle

**Hierarchisierung:** Beziehungen zw. Komponenten

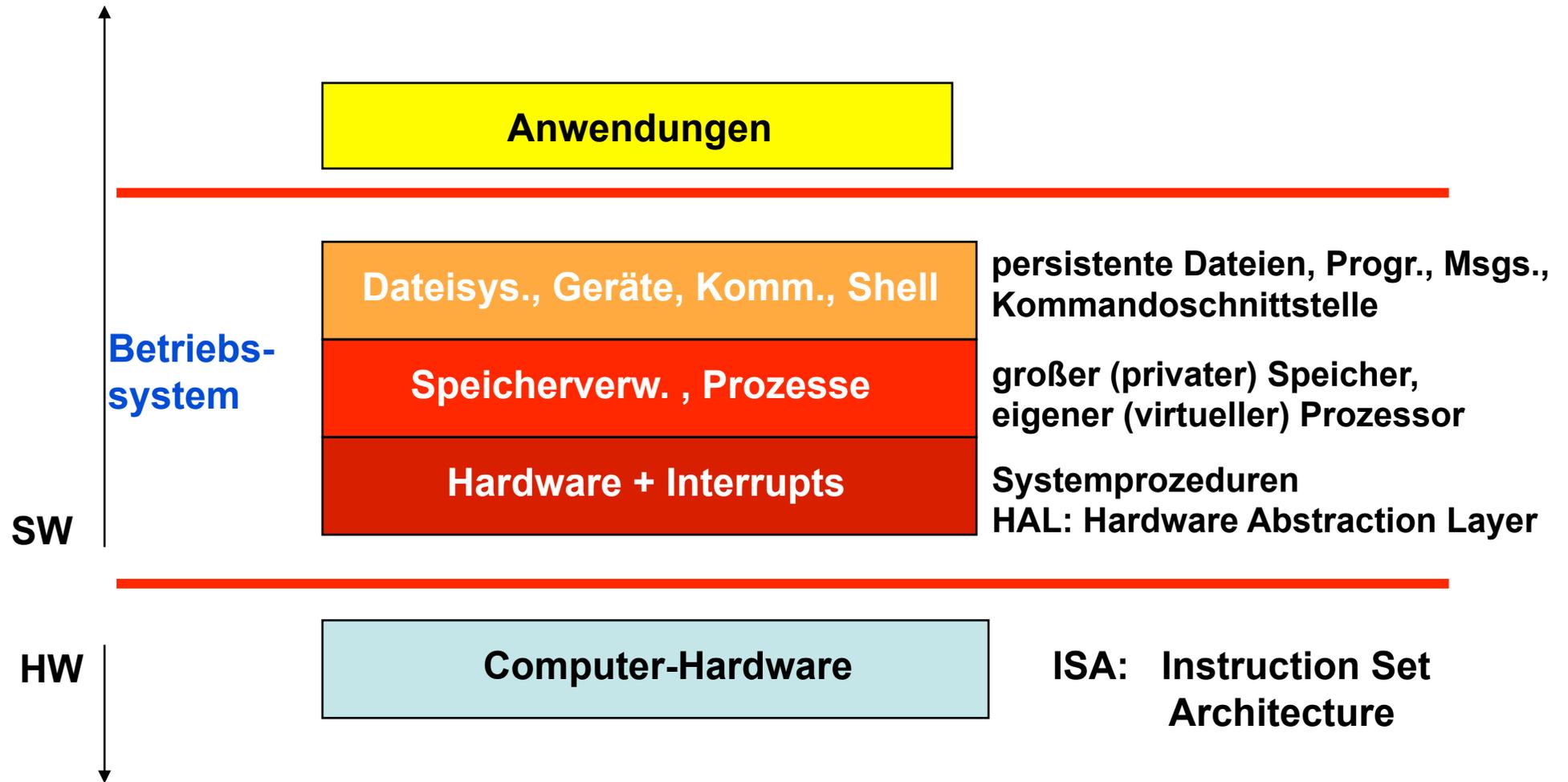
**Schichtung:** gängige Form der Hierarchisierung

Prinzipien

Techniken



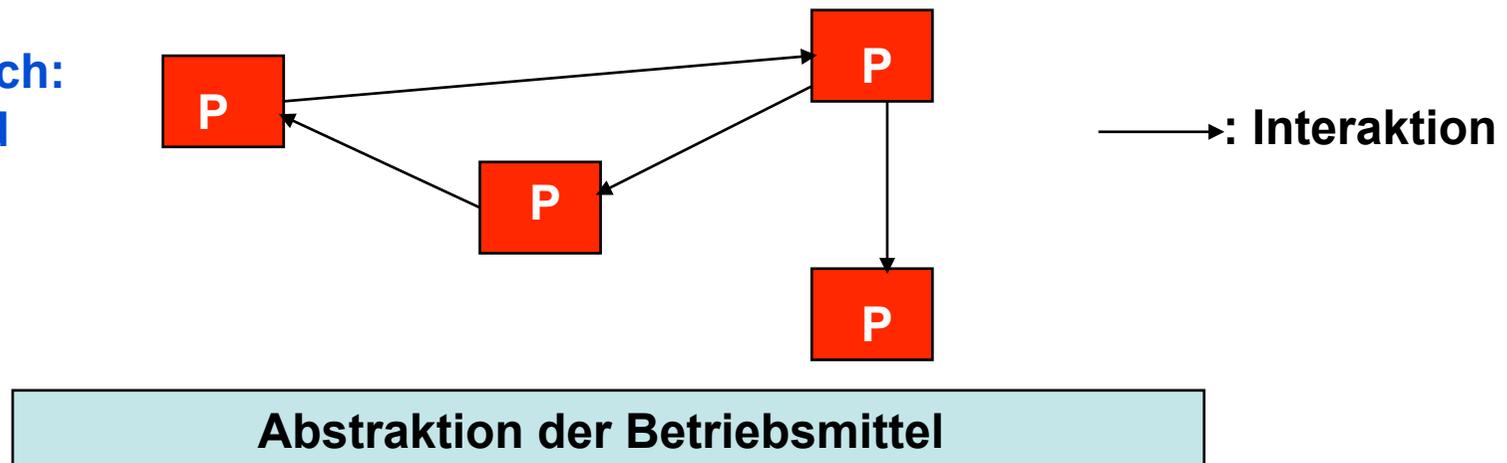
# Schichtenmodell



# Benutzerorientierte Sicht: Abstraktion - Prozess

Prozesse sind die Komponenten zur Repräsentation und Strukturierung der Aktivität im System !

Prozessbereich:  
Funktion wird  
hier erbracht



Kernbereich

Infrastruktur zur Realisierung von Prozessen

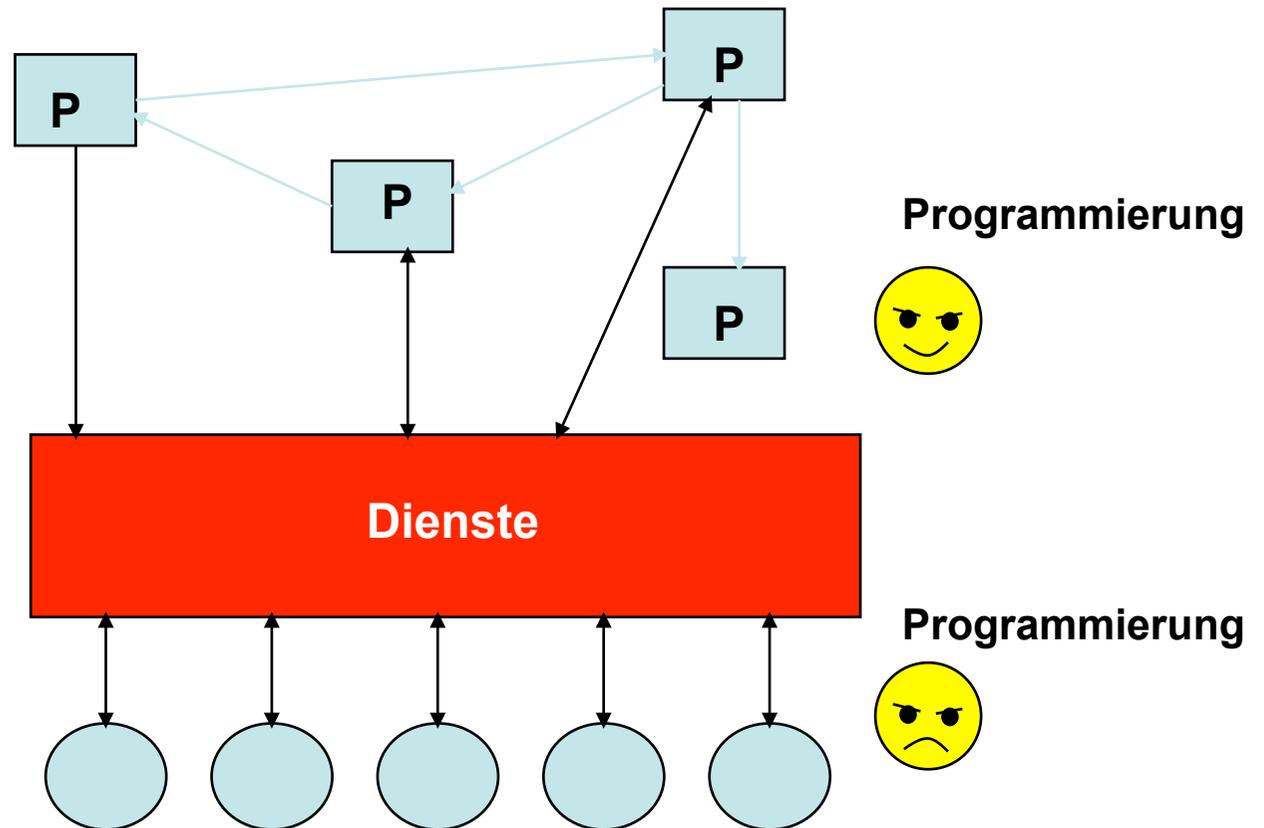


# Benutzerorientierte Sicht: Abstraktion - Dienst

Dienste dienen dem Betrieb und der Verwaltung von Ressourcen (Betriebsmittel).  
Sie stellen eine benutzergerechte Schnittstelle zu Betriebsmittel zur Verfügung.

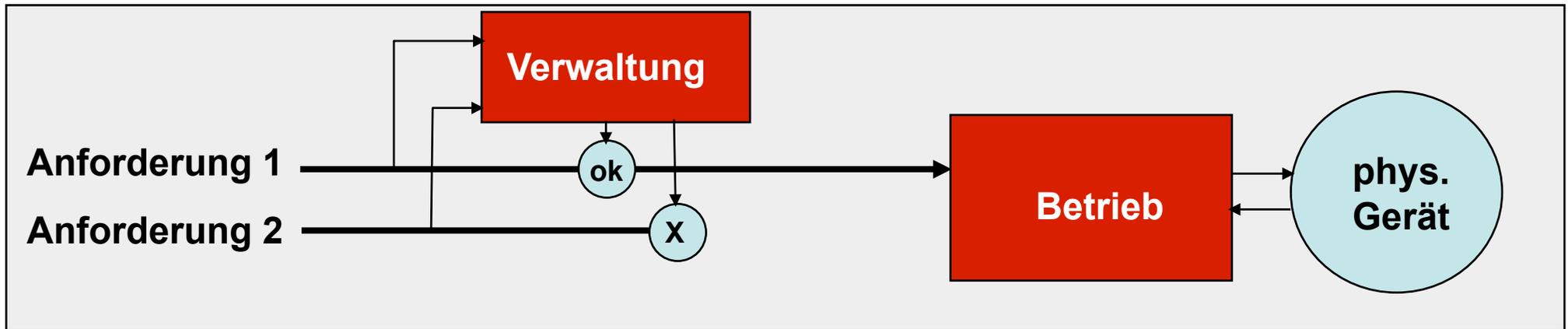
Logische Betriebsmittel:  
Datei, Fenster, Mouse-Zeiger,...

Reale Betriebsmittel:  
Platte, Video RAM, Serielles  
E/A Gerät, ...

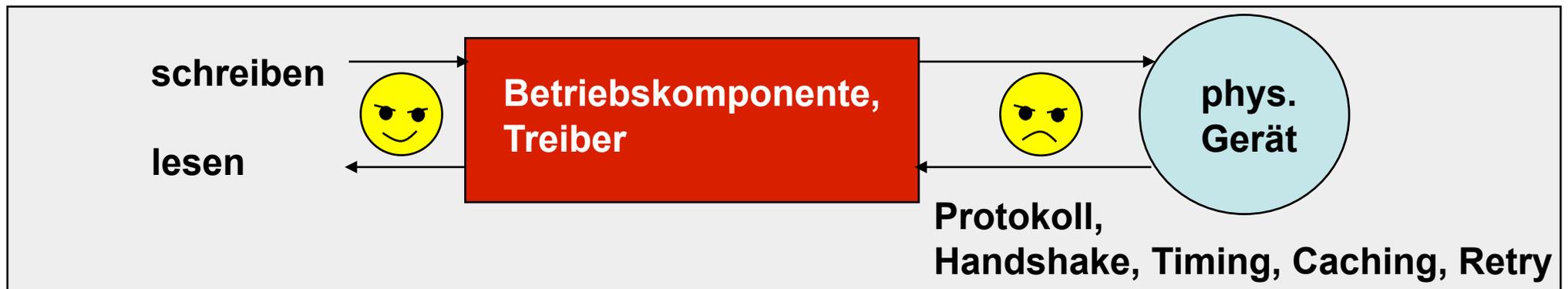


# Dienste: Betrieb und Verwaltung

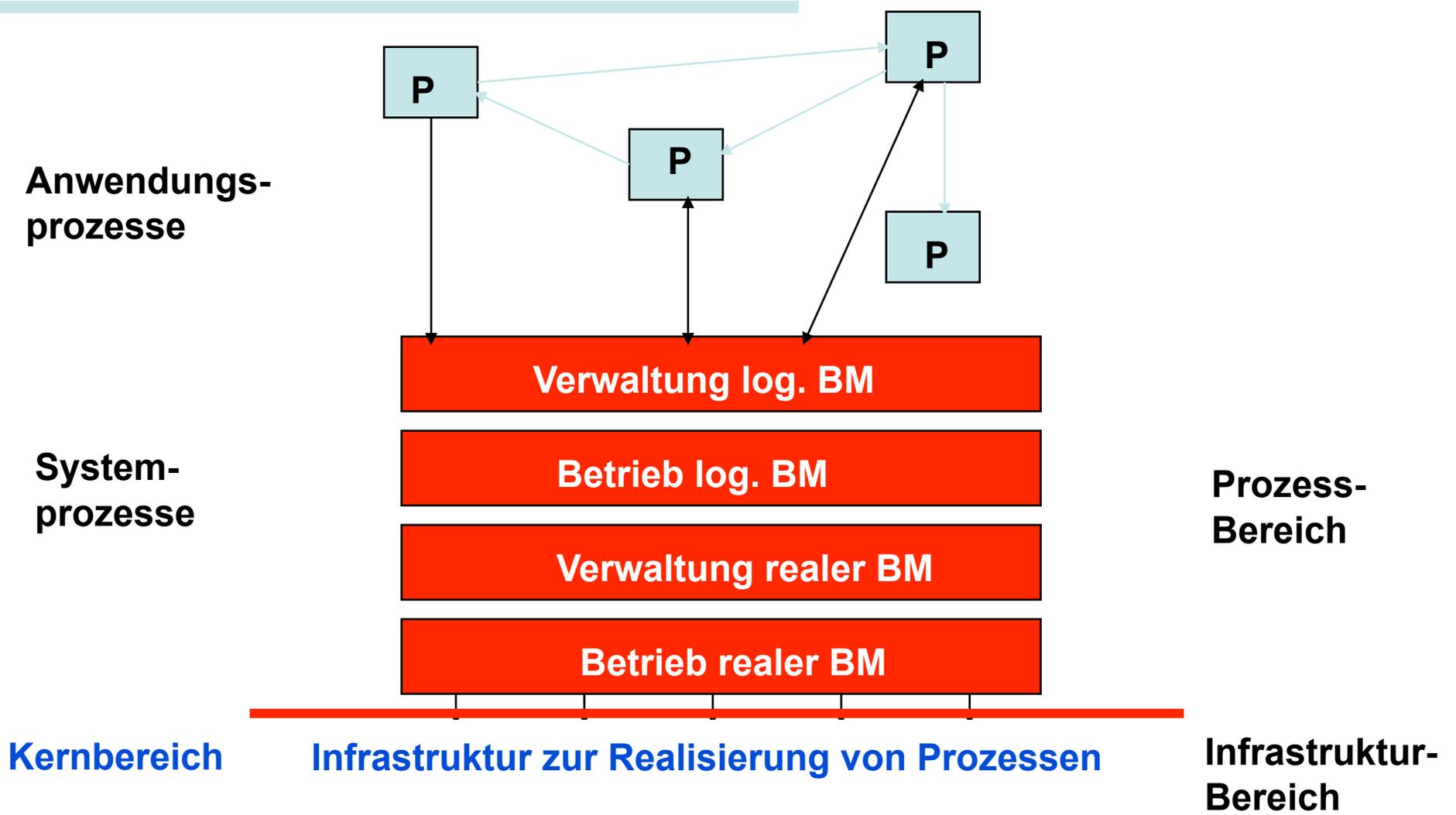
## BM-Verwaltung: Zuteilung von Nutzungsanforderungen.



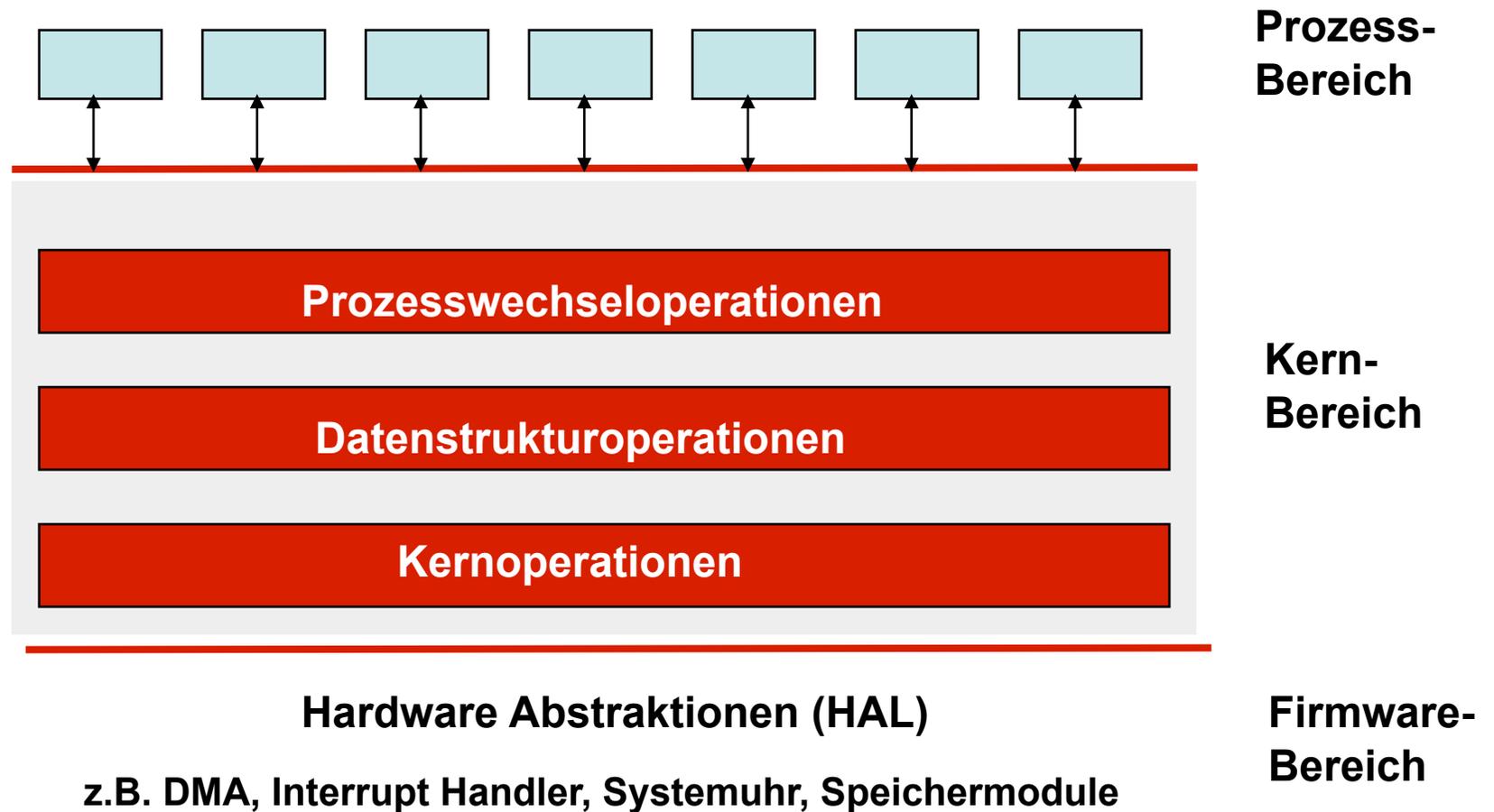
## BM-Betrieb: Steuerung und Kontrolle von Betriebsmitteln



# Schichtung der Dienste



# Strukturierung des Kernbereichs



---

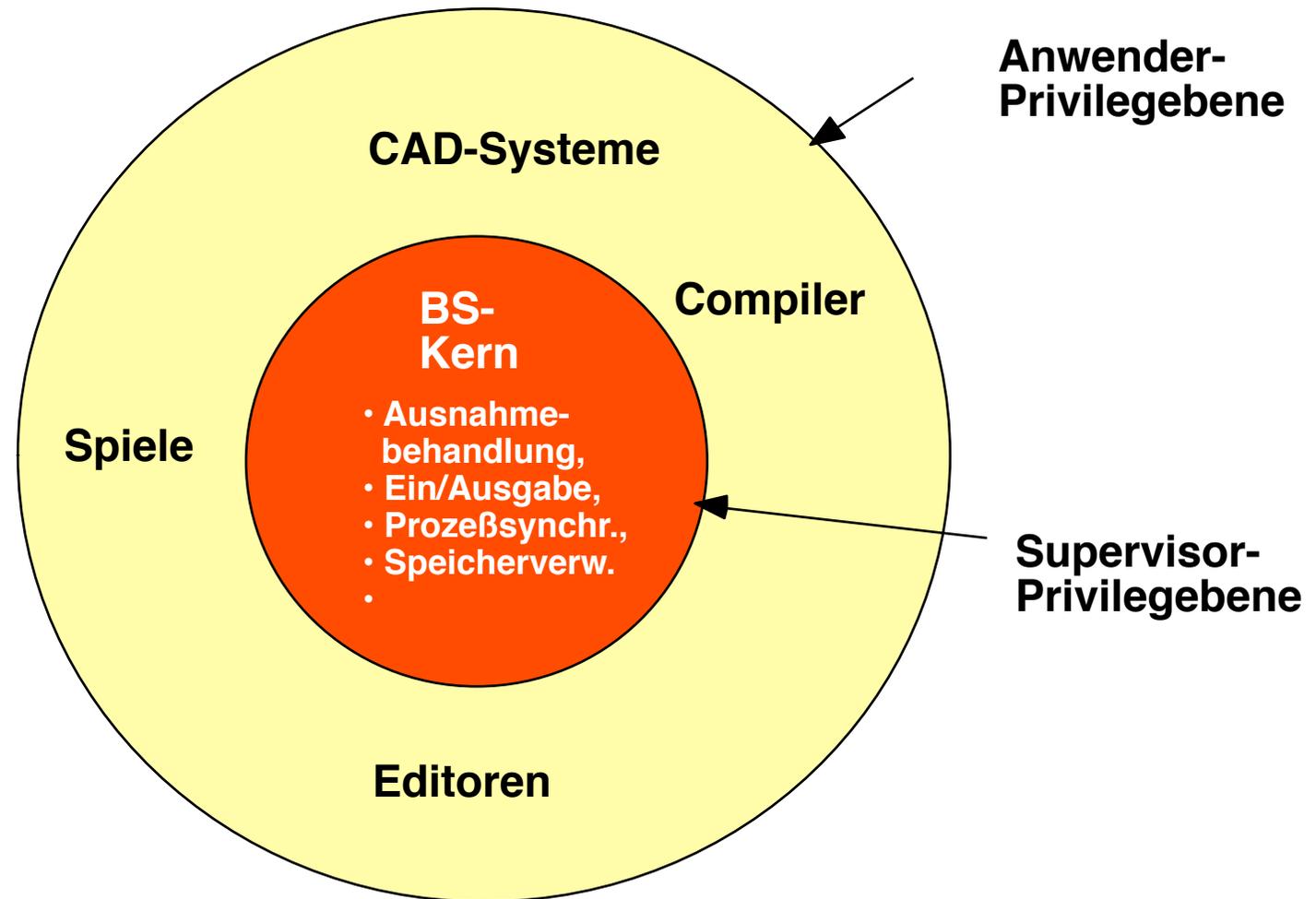
**Was ist Anwendung ?**  
**Was ist System ?**  
**Was ist Kern ?**

**Wie werden Benutzerebenen und Systemebenen  
voreinander geschützt?**

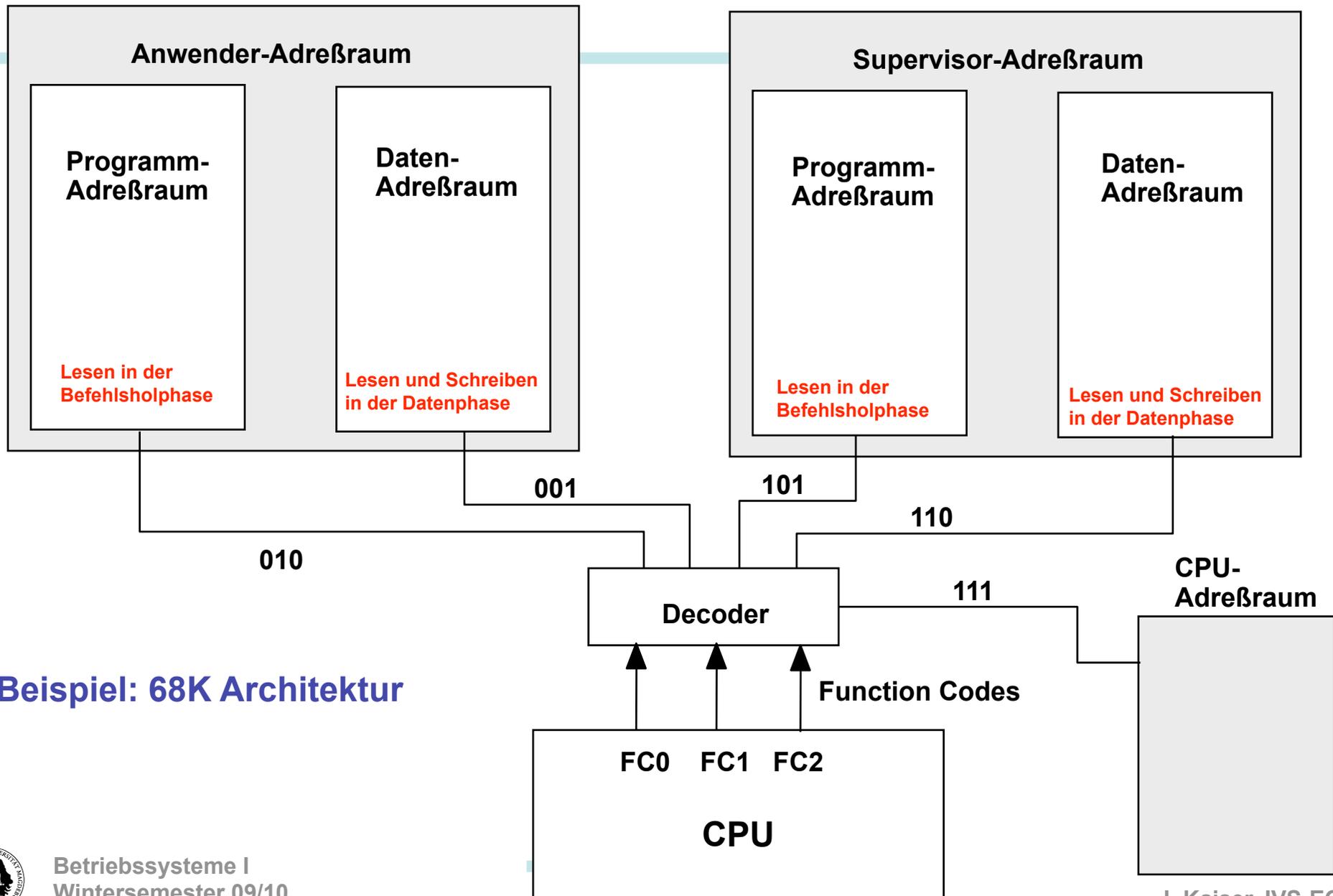


# Trennung von Anwendermodus und Systemmodus

Motivation:  
Zugriffsschutz!



# Trennung der Adreßräume durch die CPU-Hardware



Beispiel: 68K Architektur

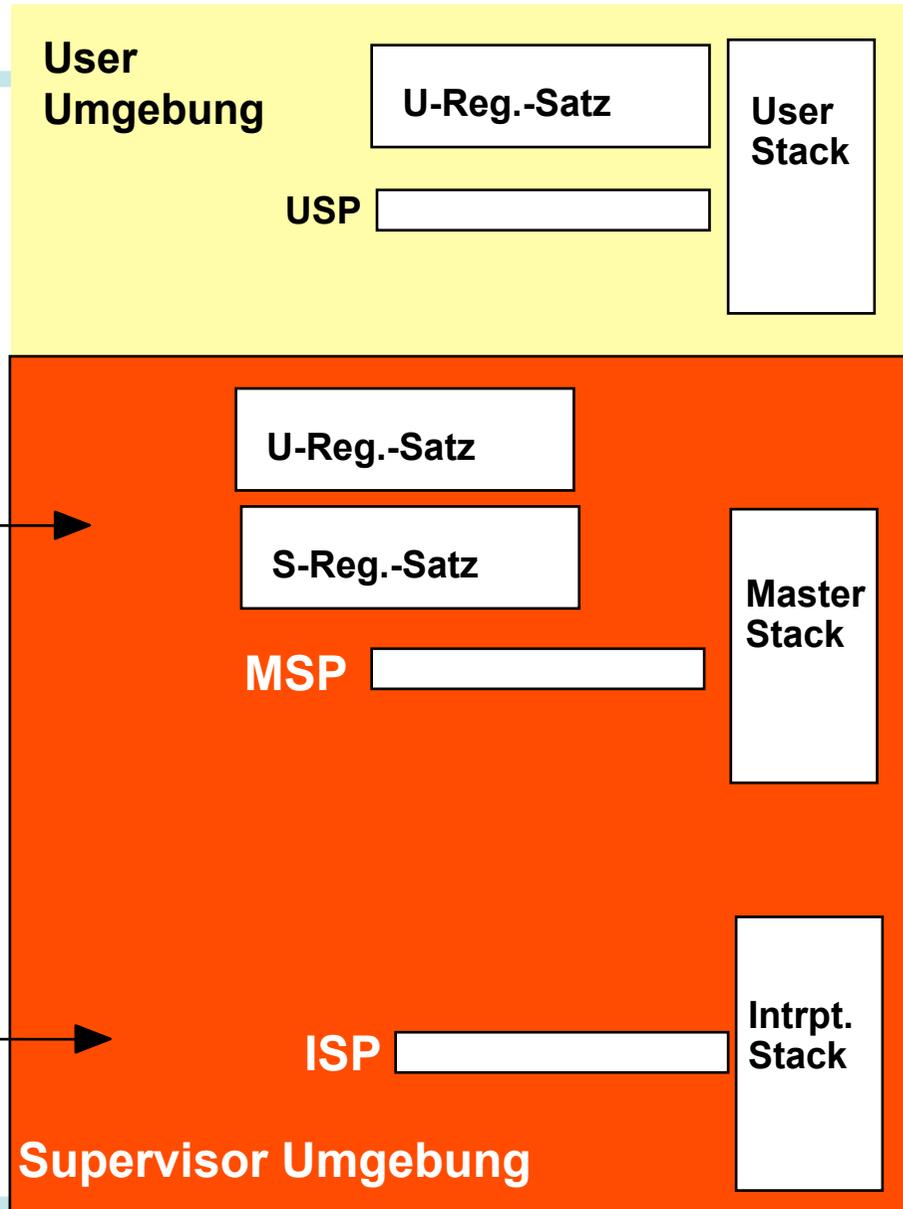


# Übergang vom Anwender in den Supervisor Zustand

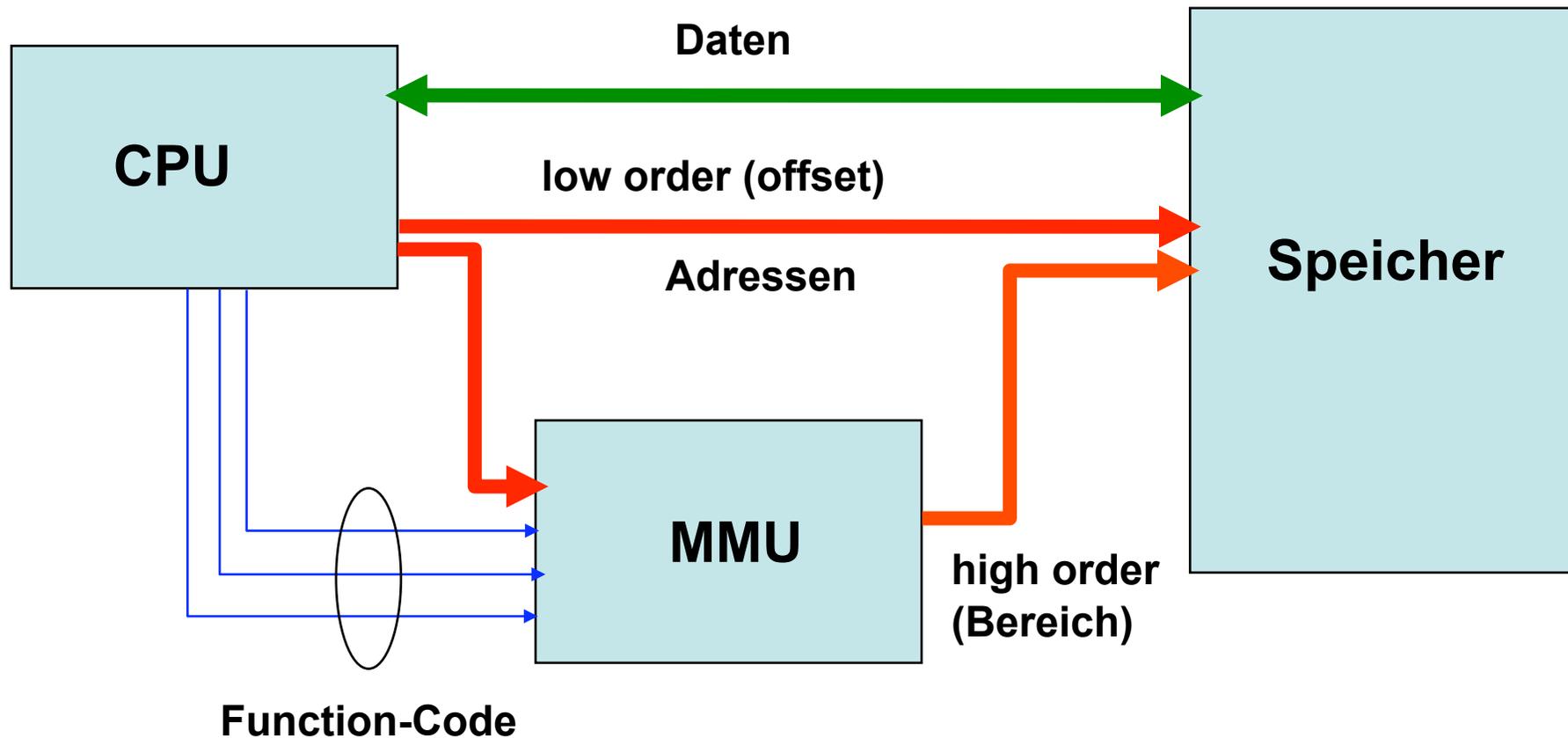
Der Übergang vom Anwendermodus in den Supervisormodus ist nur durch Interrupts oder Ausnahmebehandlung möglich.

Ausnahmen sind:

- explizite Software Traps:  
z.B. TRAP, CHK,
- Ausnahme, die bei der Befehlsabarbeitung auftreten:  
z.B. Illegale Instruktion,  
Division durch 0  
Adressierungsfehler  
Privilegverletzung
- Interrupts

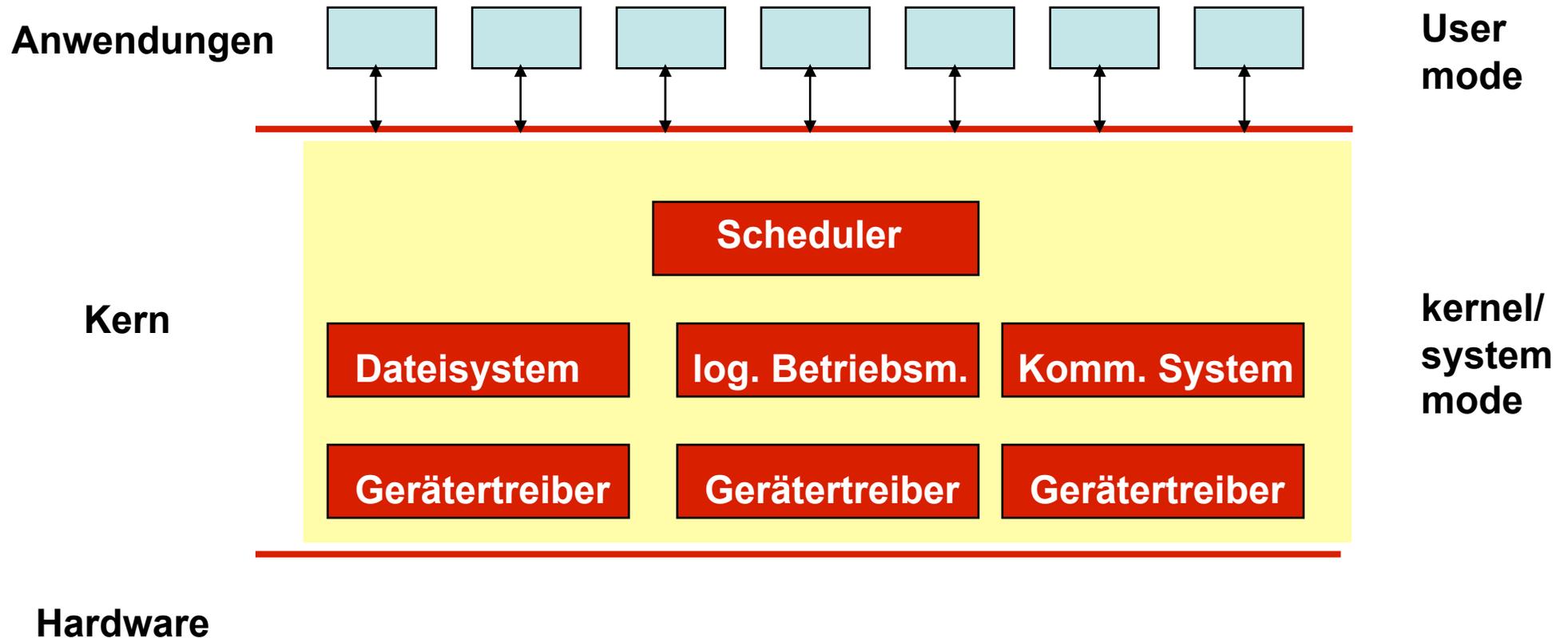


# Trennung der Adressräume durch MMU

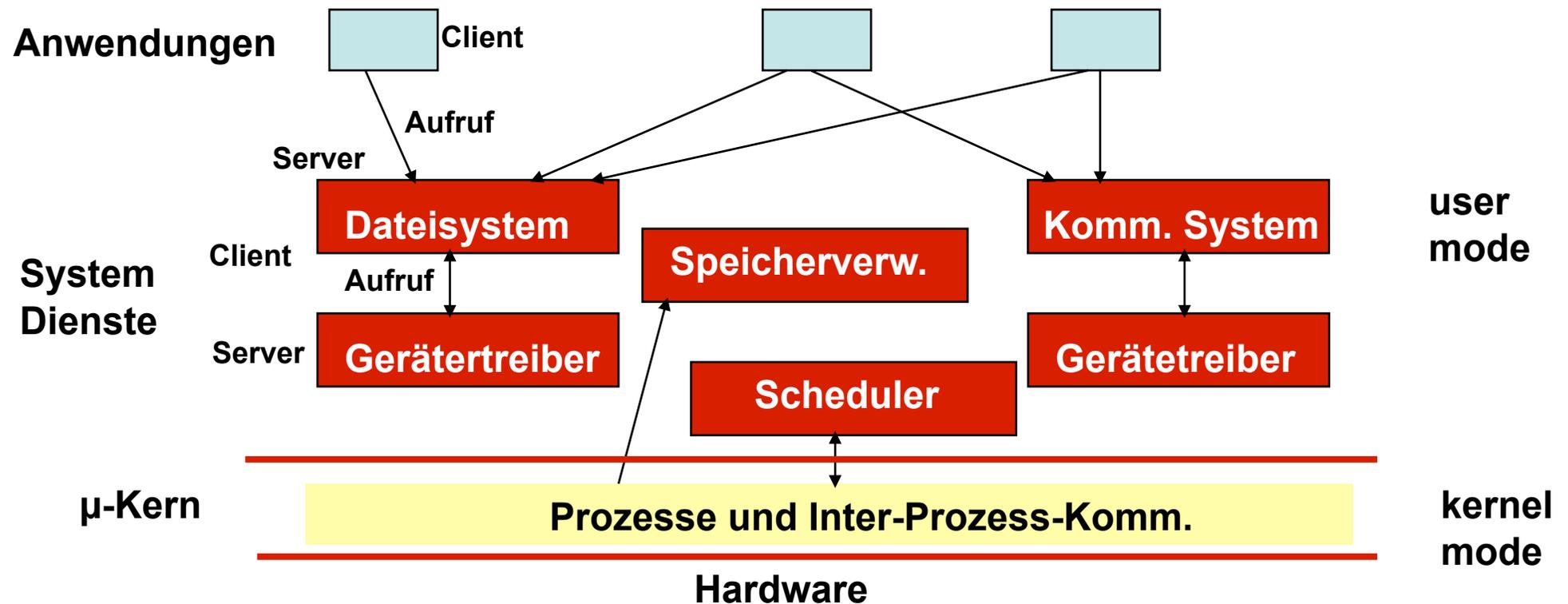


# Monolithische Betriebssystemstruktur

---

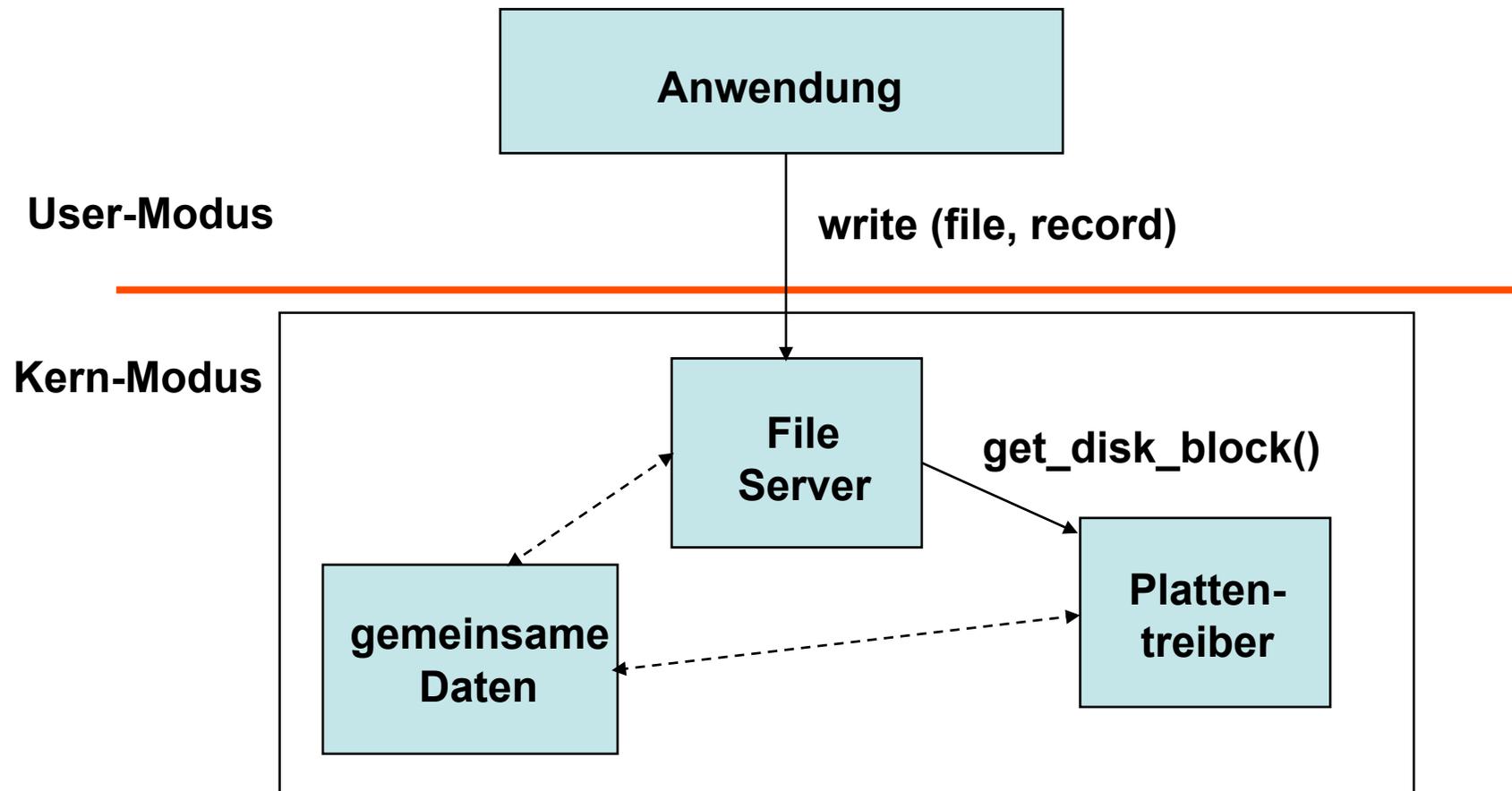


# Micro-Kern Betriebssystemstruktur

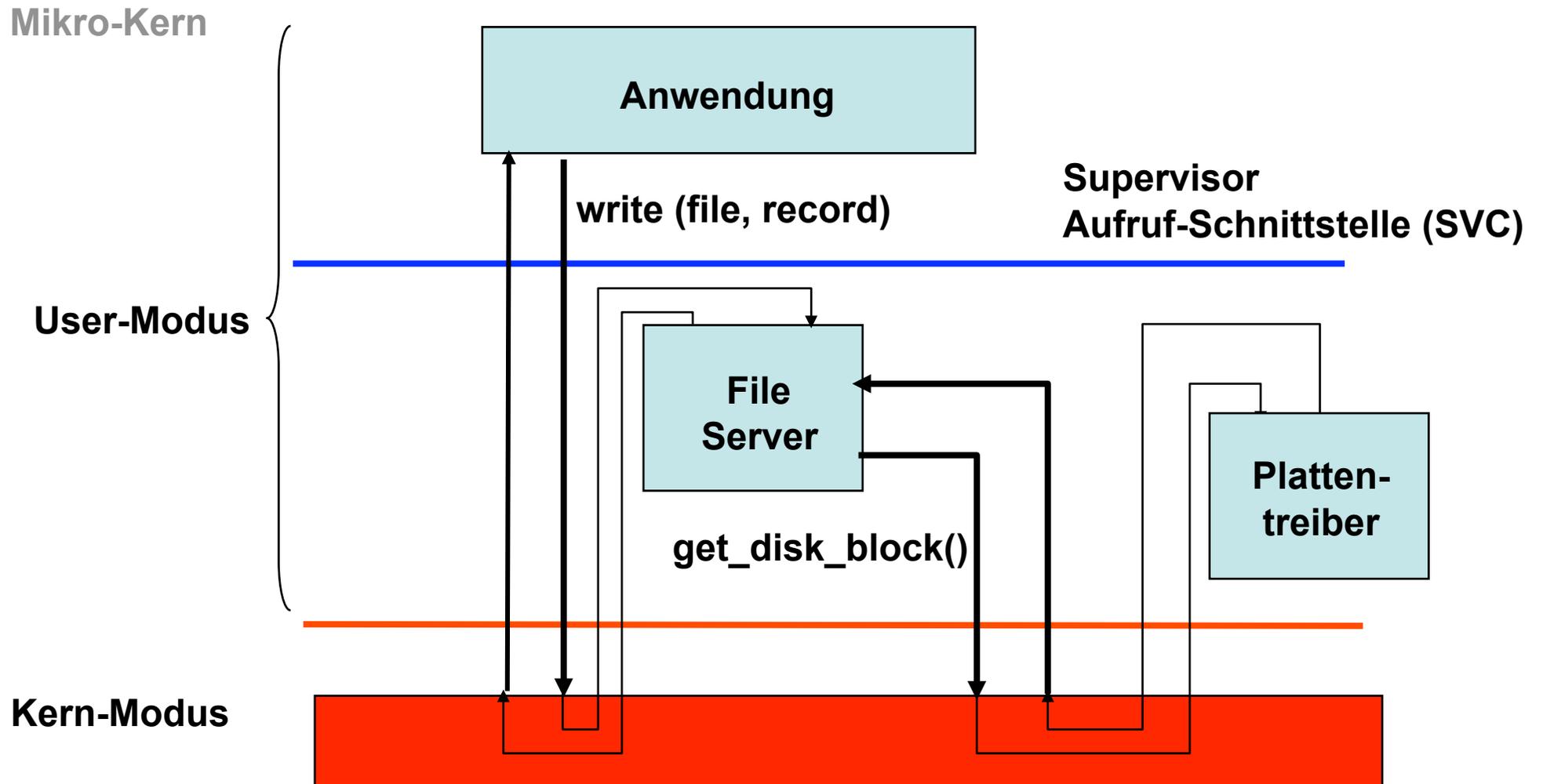


# Kosten des Aufrufs einer Systemfunktion

## Monolithischer Kern



# Kosten des Aufrufs einer Systemfunktion



# Monolithisch vs. $\mu$ -Kern: Pro und Con

	Monolith	$\mu$ -Kern
<b>Modularität</b>		 Fehlereingrenzung
<b>Erweiterbarkeit</b>	 	 neue Dienste können einfach hinzugefügt werden.
<b>Schutz</b>	User/System. Kein Schutz der Kernfkt. gegeneinander	User-Prozesse gegeneinander geschützt. kl. Kern (besser verifizierbar)
<b>Flexibilität</b>		Emulation versch. BS.
<b>Performance</b>		  viele Prozess- und Moduswechsel



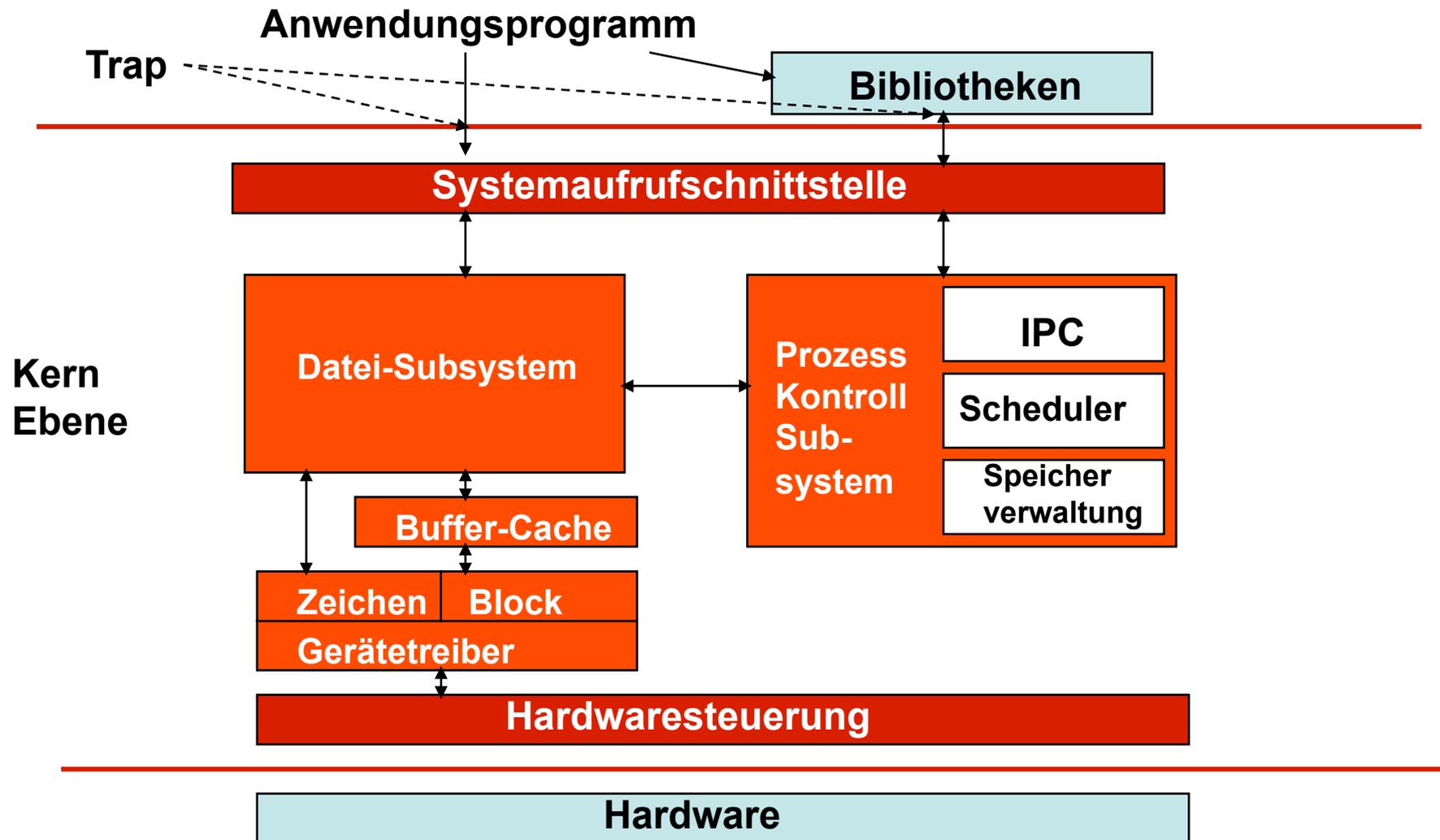
---

**The Unix kernel is an I/O Multiplexer more than a complete operating system.  
This is as it should be.**

**Ken Thompson: Unix Implementation, Bell Systems Technical Journal, 57(6), 1978**



# Herkömmlicher Unix-Kern (Bach 1986)



# Betriebssystemstrukturen

---

**Probleme mit einem monolithischen Kern:**

- ➔ **alle funktionalen Komponenten des Kerns haben Zugriff auf sämtliche internen Datenstrukturen und Routinen.**
- ➔ **werden an einem beliebigen Teil Änderungen vorgenommen, müssen 1. alle Module und Routinen neu gebunden und installiert werden und 2. das System neu gebootet werden.**
- ➔ **Problembehebung (z.B. bei Linux):  
Strukturierung des BS in relativ unabhängige Funktionsblöcke, sogenannte "Ladbare Module", die dynamisch geladen und gebunden werden können.**

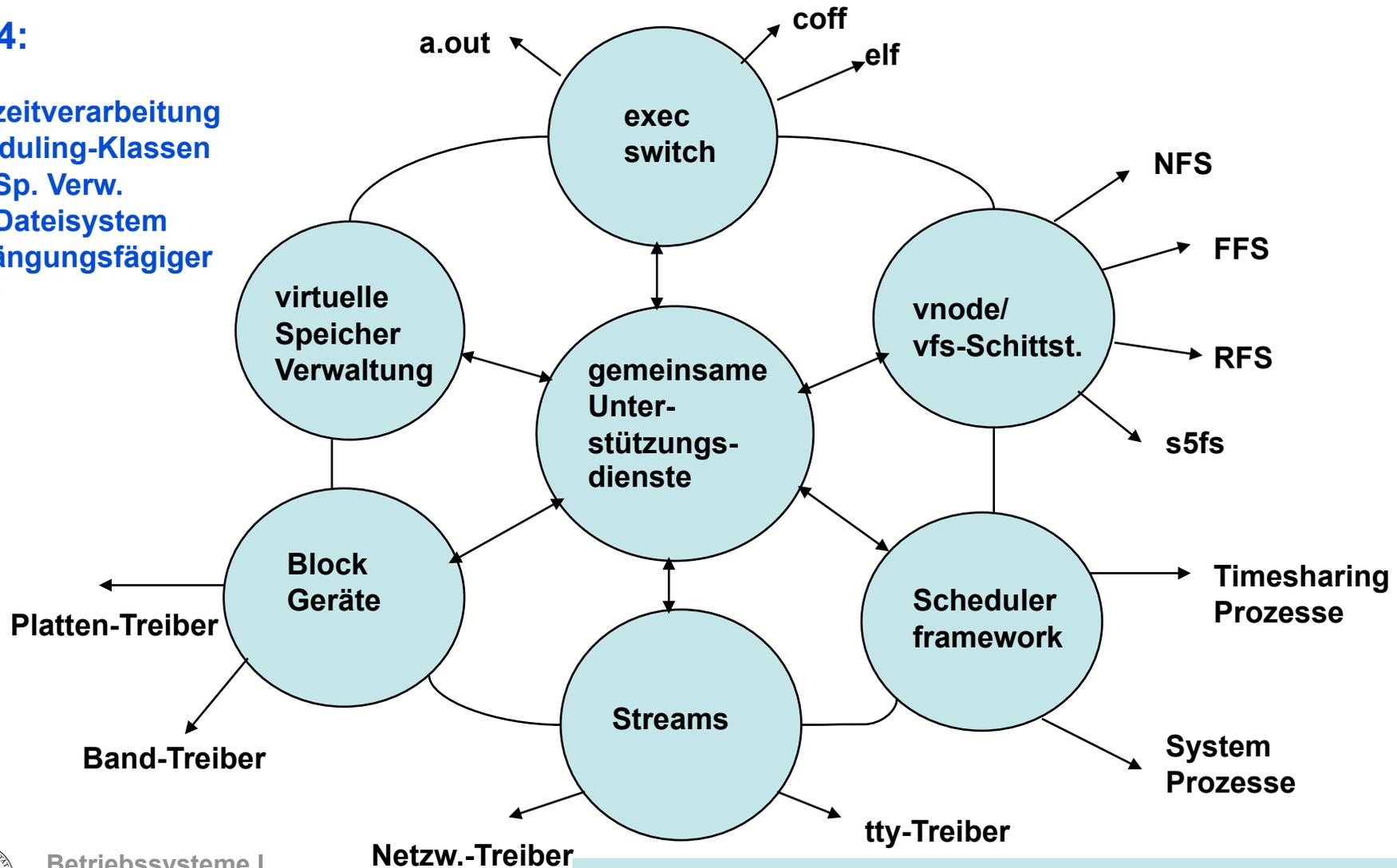
**Es bleibt das Problem des nicht vorhandenen Schutzes der Funktionsblöcke untereinander.**



# Moderner Unix-Kern (Vahalia 1996)

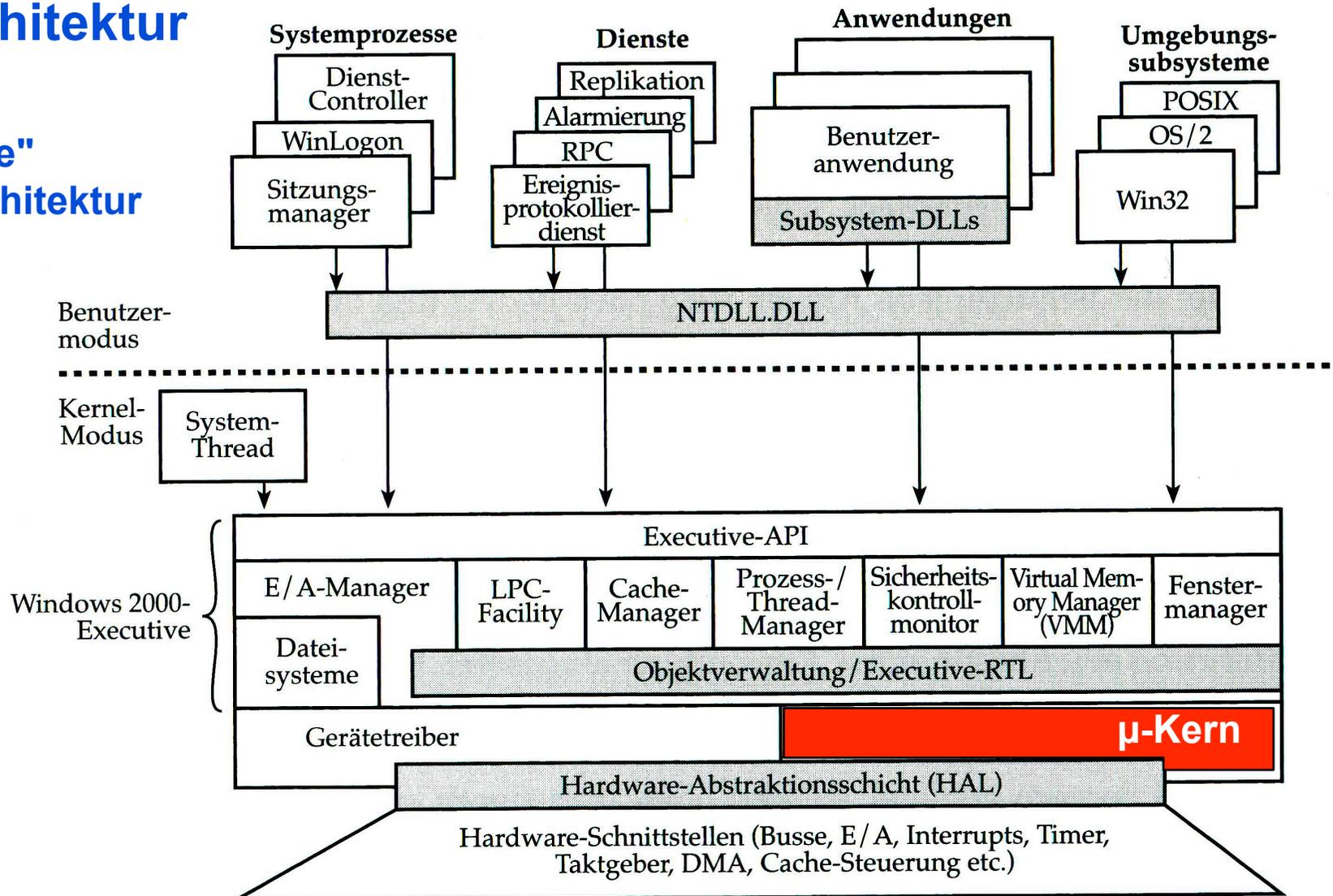
## SVR4:

- Echtzeitverarbeitung
- Scheduling-Klassen
- Virt. Sp. Verw.
- Virt. Dateisystem
- verdängungsfähiger Kern



# W2K Architektur

## "modifizierte" $\mu$ -Kern Architektur



Stallings: Betriebssysteme



# "modifizierte" $\mu$ -Kern Architektur

---

**$\mu$ -Kern:** Scheduling von Threads, Prozessumschaltung, Ausnahme- und Interrupt-Behandlung, Synchronisation (resident im Speicher, wird nicht verdrängt).

**Moduln der Exekutive (laufen als Threads im Systemprozess):**

**E/A-Manager:** Einheitlicher Rahmen für E/A, API für alle E/A, Sicherheitsziele, Zuordnung logischer Namen.

**Objektmanager:** Verwaltung, Erstellung, Löschen von Objekten der Exekutive wie: Prozesse, Threads, Dateien, Adressräume, E/A-Geräte, Sync.-Obj. , Namenszuordnung, Sicherheitsaspekte.

**Prozess/Thread Manager:** verwaltet Prozesse und Threads.

**Sicherheitskontrollmonitor:** Durchsetzung von Zugriffskontrollen.

**LPC Facility:** Local Procedure Call unterstützt lokale Client/Server Beziehungen.

**Virtual Memory Manager:** unterstützt virtuellen Speicher.

**Cache Manager:** verbessert die Leistung dateibasierter E/A .

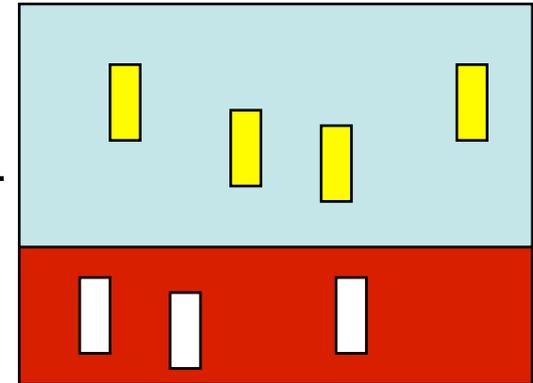
**Grafik-Modul:** unterstützt die fensterorientierte Bildschirmschnittstelle, verwaltet Grafikgeräte.



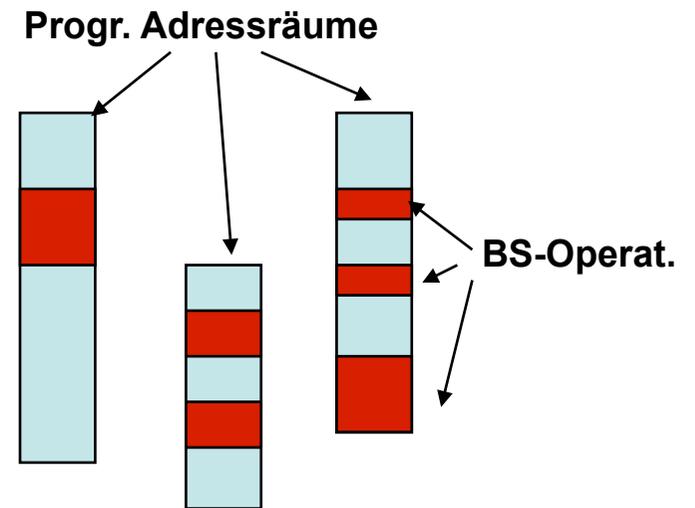
# Realisierungsformen

**Residente Realisierung:**  
Betriebssystem besitzt einen eigenen Adressraum und ist (mehr oder weniger) vollständig im Speicher unabhängig von Anwendungsprogrammen.  
**Standard bei General Purpose BS.**

Prozesse im Anwenderadressraum  
Kernprogr. als Prozeduren im Kernadressraum



**Realisierung als Funktionsbibliothek:**  
Benötigten Betriebssystemfunktionen werden beim Compilieren oder Binden zur Anwendung hinzugebunden.  
**Standard bei BS für eingebettete Systeme**



# Diskussion und Zusammenfassung

---

**Ein Betriebssystem ist ein komplexes Softwaresystem und Abstraktion und Strukturierungsmassnahmen sind wesentlich zum Verständnis und zur Realisierung.**

**Abstraktion wird eingesetzt, um den Umgang mit Betriebsmitteln zu erleichtern.**

**Schichtung ist eine Prinzip, um Abstraktionen zu strukturieren.**

**Modularisierung ist ein wesentliches Konzept, um eine adäquate Realisierung zu erreichen.**

**Qualitätsziele:**

**Dynamische Erweiterbarkeit und Einschränkung, Flexibilität, Testbarkeit, Wartbarkeit, Sicherheit und Zuverlässigkeit, Effizienz, Portierbarkeit.**

