TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Betriebssysteme

BETRIEBSSYSTEME UND SICHERHEIT

mit Material von Olaf Spinczyk,
Universität Osnabrück

Abstraktionen und Strukturen

https://tud.de/inf/os/studium/vorlesungen/bs

HORST SCHIRMEIER

Inhalt

- Prozesse
  - CPU-Zuteilung
  - Synchronisation und Verklemmungen
  - Interprozesskommunikation
- Speicherverwaltung
  - Arbeitsspeicher
  - Hintergrundspeicher
- Systemsicherheit
- Multiprozessorsysteme
- *Cloud-Computing* und Virtualisierung

Literatur

Silberschatz, Chap. 1, „Introduction“

Tanenbaum, Kap. 1, „Einführung“

Inhalt

- **Prozesse**
  - CPU-Zuteilung
  - Synchronisation und Verklemmungen
  - Interprozesskommunikation
- Speicherverwaltung
  - Arbeitsspeicher
  - Hintergrundspeicher
- Systemsicherheit
- Multiprozessorsysteme
- *Cloud-Computing* und Virtualisierung

Literatur

Silberschatz, Chap. 1, „Introduction“

Tanenbaum, Kap. 1, „Einführung“

Ein Prozess …

- Horning/Randell, *Process Structuring*

> *„… P ist ein Tripel (S, f, s), wobei S einen Zustandsraum, f eine Aktionsfunktion und* **s** ⊂ **S** *die Anfangszustände des Prozesses P bezeichnen. Ein Prozess erzeugt Abläufe, die durch die Aktionsfunktion generiert werden können.“*

- Dennis/van Horn, *Programming Semantics for Multiprogrammed Computations*

> *„… ist das Aktivitätszentrum innerhalb einer Folge von Elementaroperationen. Damit wird ein Prozess zu einer abstrakten Einheit, die sich durch die Instruktionen eines abstrakten Programms bewegt, wenn dieses auf einem Rechner ausgeführt wird.“*

- Habermann, *Introduction to Operating System Design*

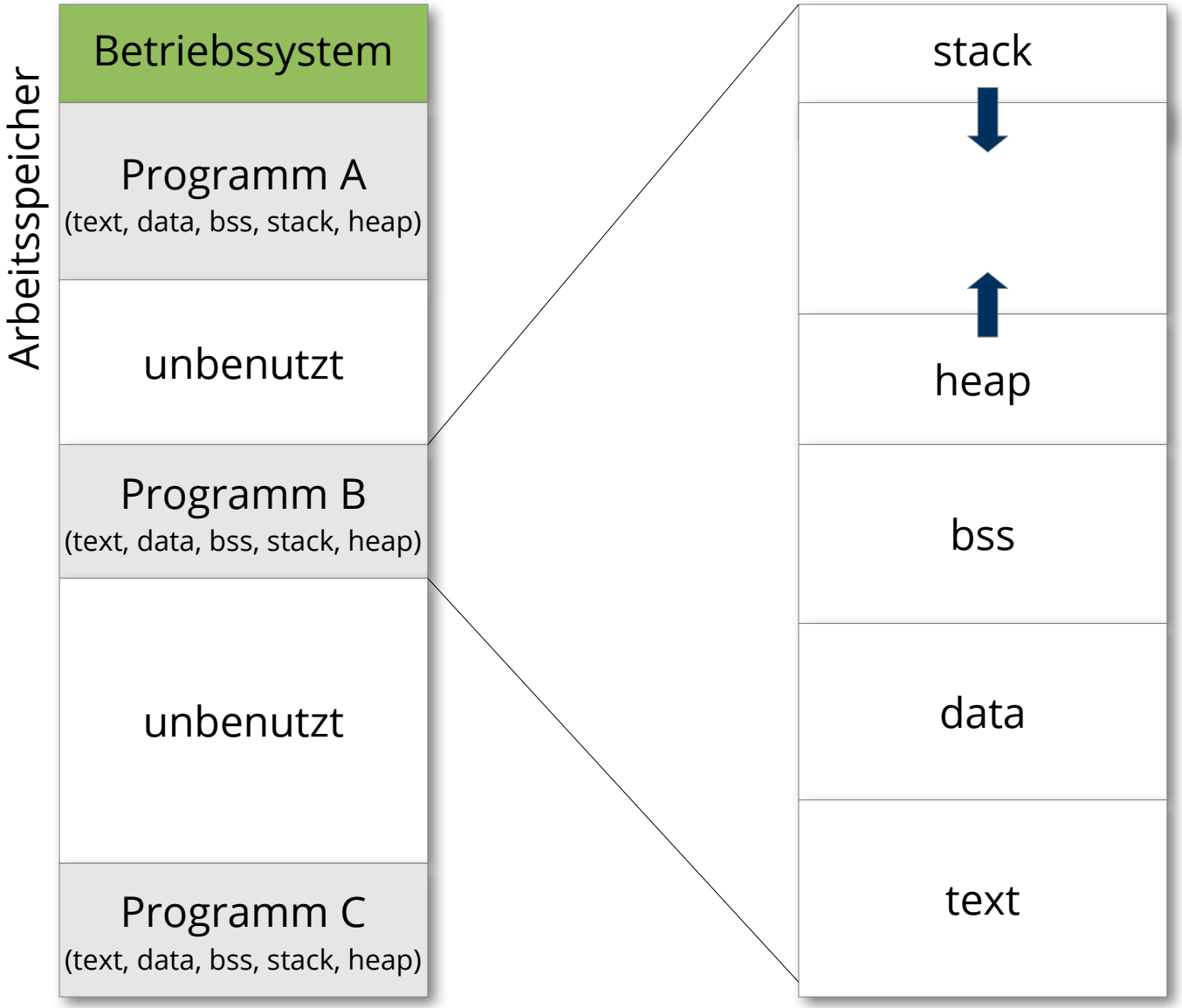
> *„… wird durch ein Programm kontrolliert und benötigt zur Ausführung dieses Programms einen Prozessor.“*

Ein Prozess …

- **„ist ein Programm in Ausführung.“**
  - unbekannte Referenz, „Mundart“

- Dazu gehört ein **Prozesskontext**, i.d.R. …
  - Speicher: Code-, Daten und Stapelsegment (*text*, *data*, *bss*, *stack*, *heap*)
  - Prozessorregisterinhalte
    - Instruktionszeiger
    - Stapelzeiger
    - Vielzweckregister
    - …
  - Prozesszustand
  - Benutzerkennung
  - Zugriffsrechte
  - Aktuell belegte Betriebsmittel
    - Dateien, E/A-Geräte, u.s.w.
  - …

Prozesse im Speicher



Stapel für
- lokale Variablen
- Rücksprungadressen
- Funktionsparameter

„Halde“: new / malloc

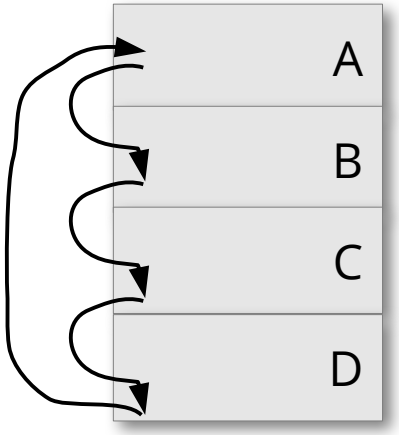
uninitialisierte globale Variablen

initialisierte globale Variablen

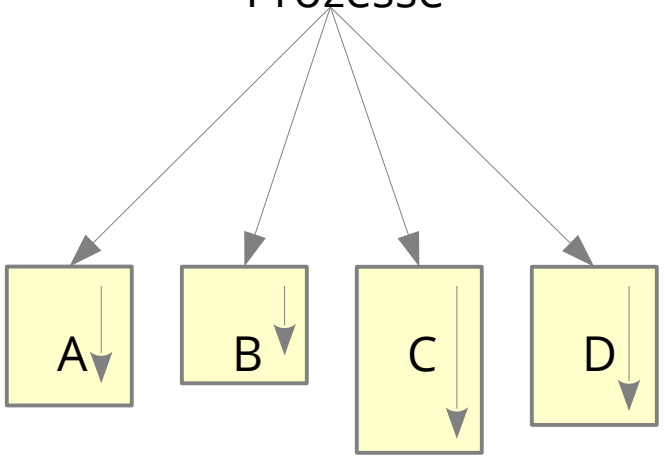
Maschinencode

Prozessmodell

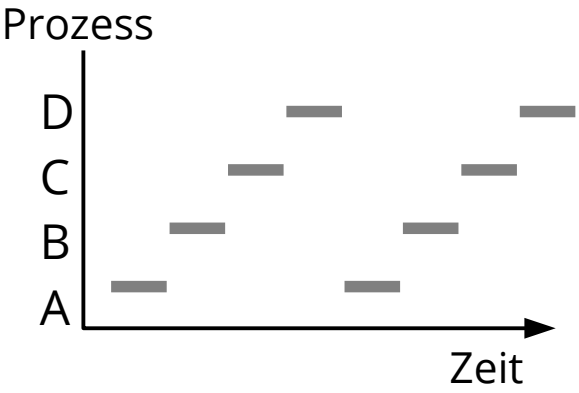
Mehrprogrammbetrieb



Nebenläufige Prozesse



Multiplexing der CPU

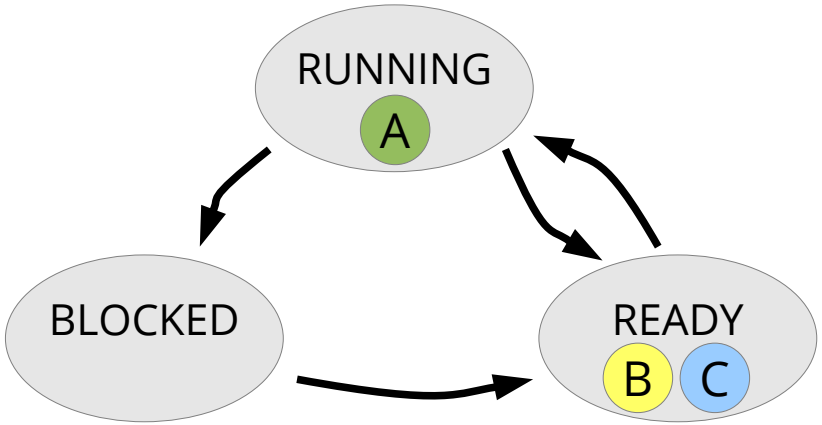


Technische Sicht
- 1 Instruktionszeiger
- Kontextwechsel

Konzeptionelle Sicht
- 4 unabhängige sequentielle Kontrollflüsse

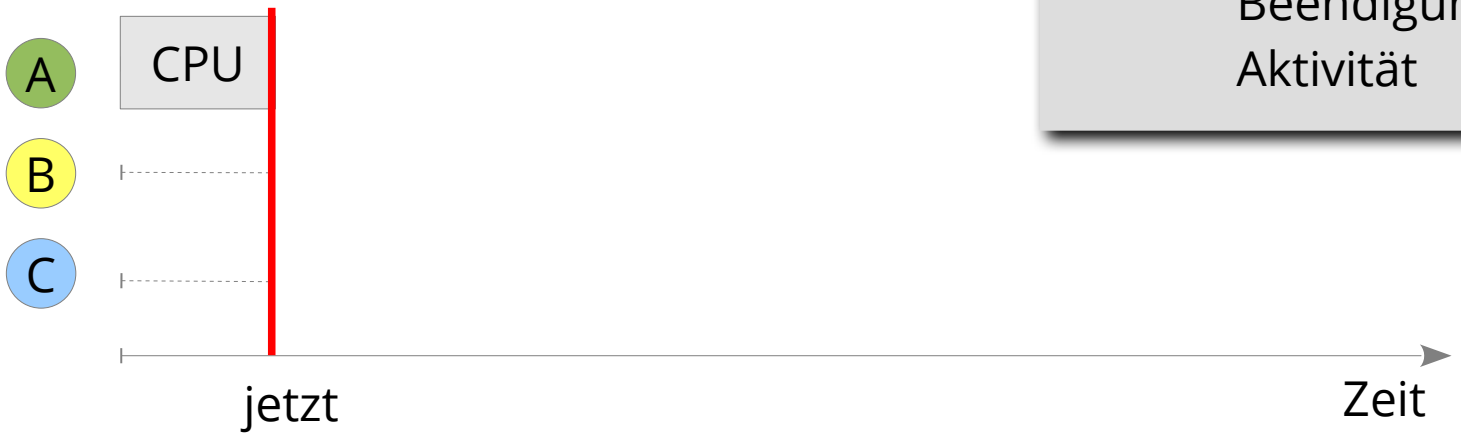
Realzeit-Sicht (Gantt-Diagramm)
- Zu jedem Zeitpunkt ist nur ein Prozess aktiv (Uni-Prozessor-HW)

Prozessverhalten und -zustände (3)

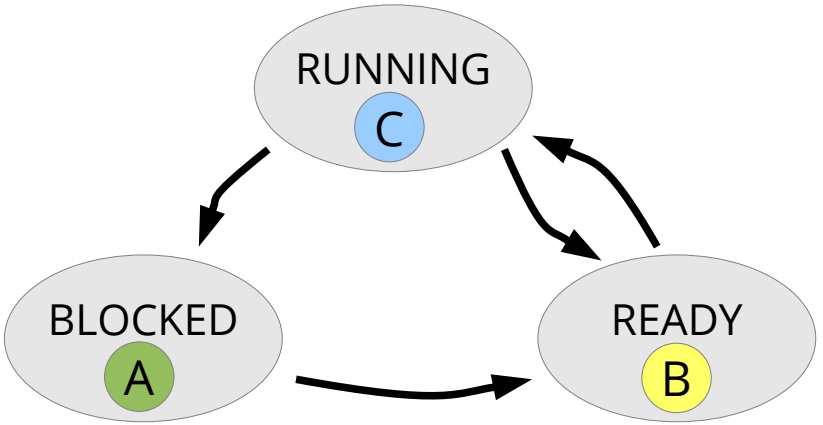


Prozesszustände

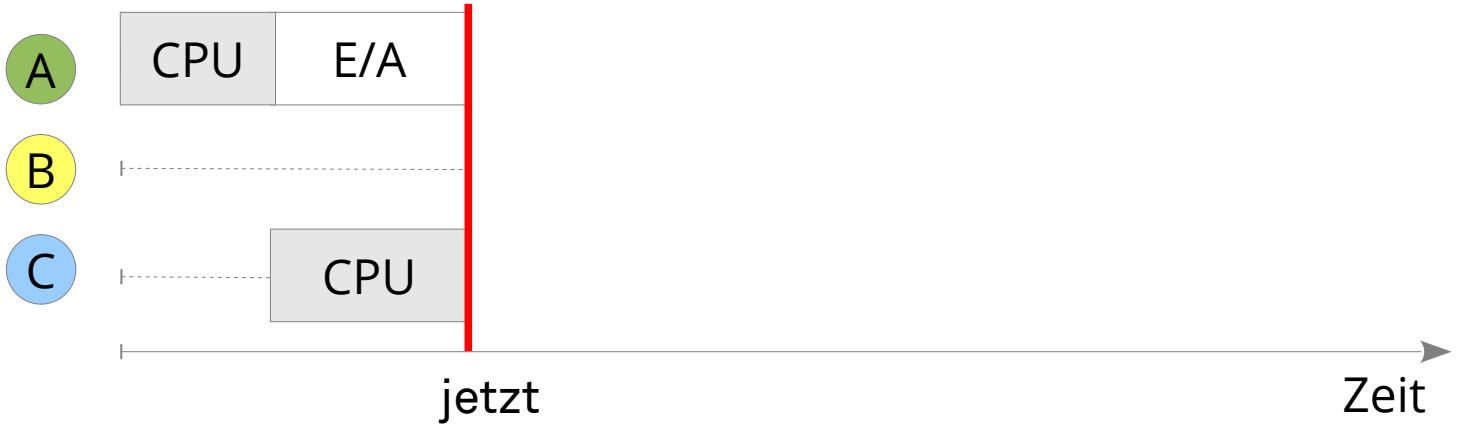
- **RUNNING**
  - Prozess wird gerade ausgeführt
- **READY**
  - Prozess ist rechenbereit, wartet auf die CPU
- **BLOCKED**
  - Prozess wartet auf die Beendigung einer E/A-Aktivität



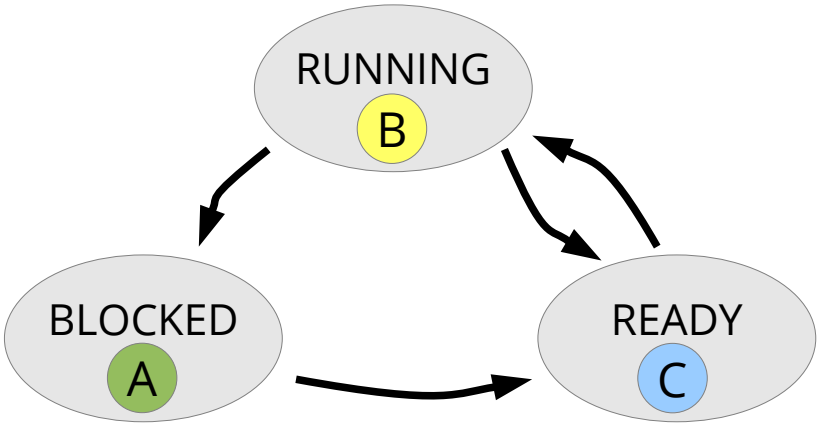
Prozessverhalten und -zustände (3)



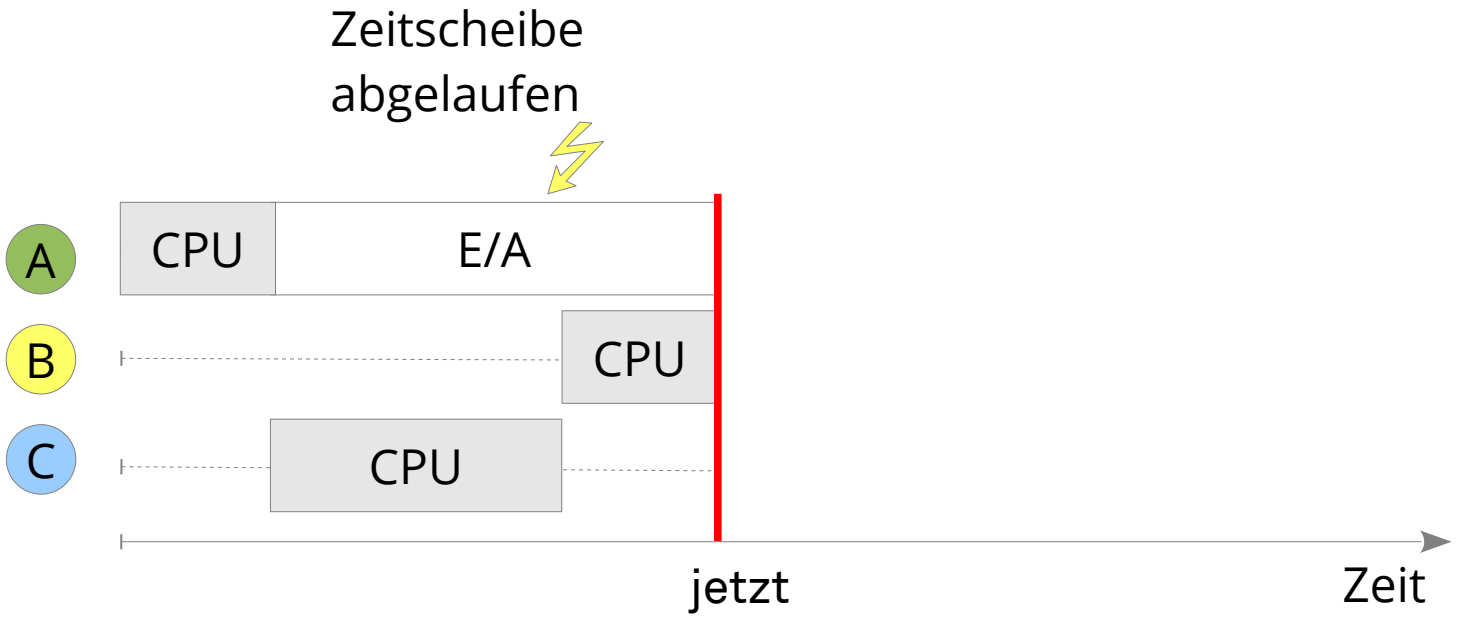
Prozess A hat einen E/A-Vorgang gestartet und ist in den Zustand BLOCKED übergegangen. Da A die CPU nun nicht benötigt, hat das Betriebssystem den Prozess C ausgewählt und von READY in RUNNING überführt. Es fand ein **Kontextwechsel** von A zu C statt.



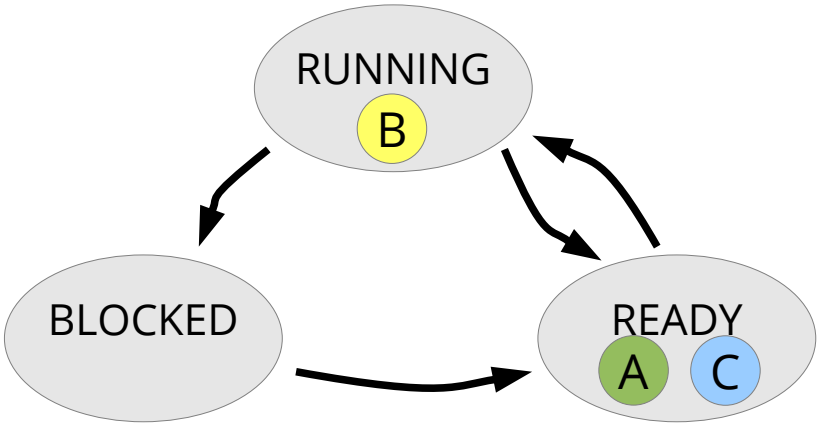
Prozessverhalten und -zustände (3)



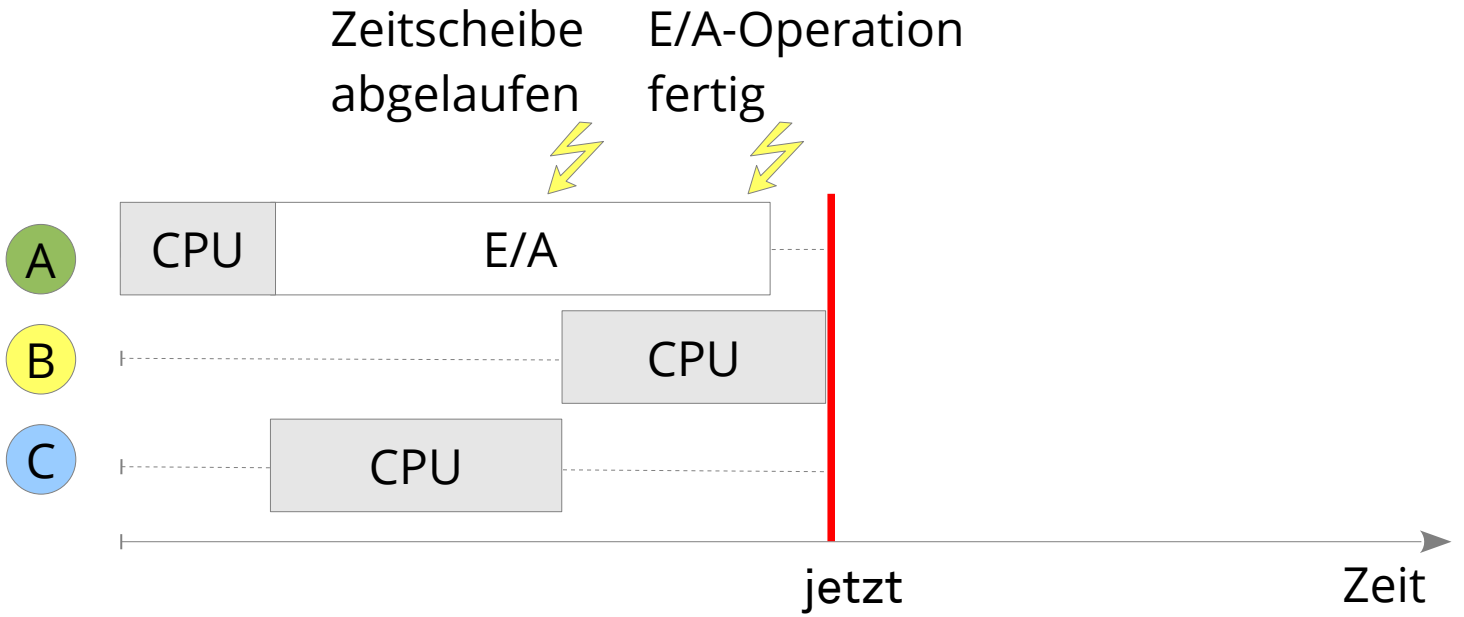
C hat die CPU zu lange „besessen“, wurde **„verdrängt“** und ist daher nun wieder im Zustand READY. Damit kann jetzt endlich auch B bearbeitet werden und wird RUNNING.



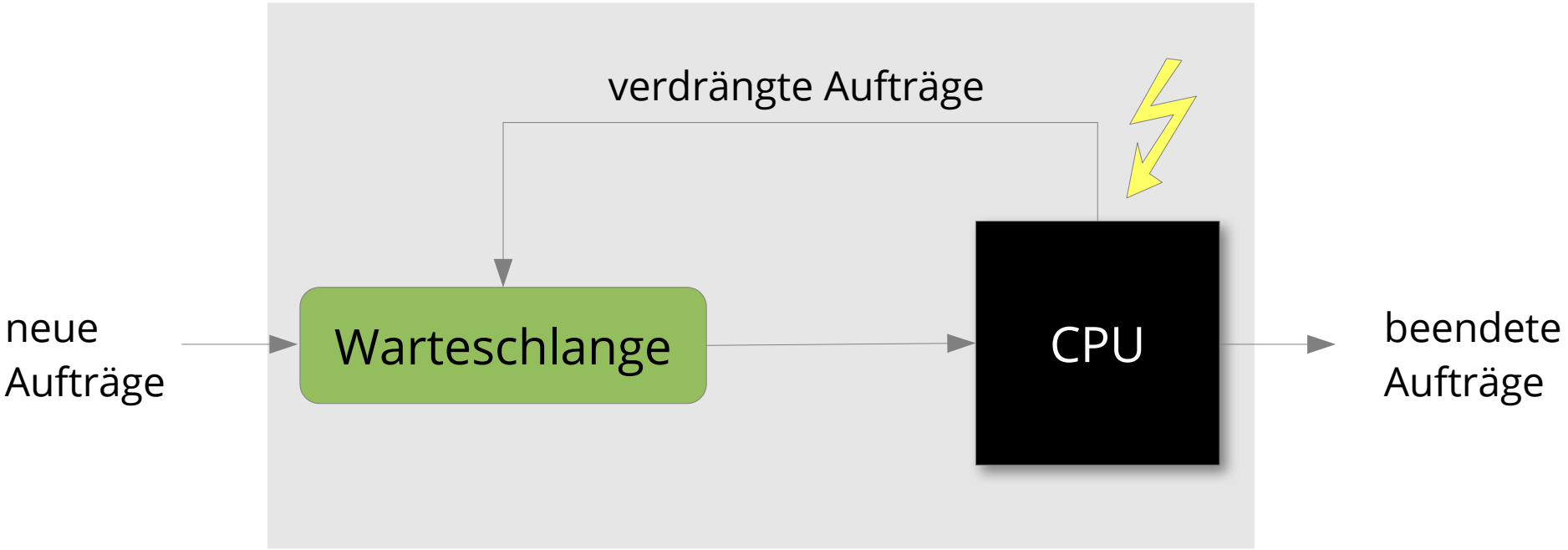
Prozessverhalten und -zustände (3)



Die E/A-Operation von A ist nun abgeschlossen. Daraufhin wird A nun READY und wartet auf die Zuteilung der CPU.



CPU-Zuteilung (*Scheduling*)



Ein einzelner **Scheduling-Algorithmus** charakterisiert sich durch die Reihenfolge von Prozessen in der Warteschlange und die Bedingungen, unter denen die Prozesse der Warteschlange zugeführt werden.

CPU-Zuteilung (*Scheduling*)

(auch „Ablaufplanung“)

- Sorgt für den geordneten Ablauf konkurrierender Prozesse
- Grundsätzliche Fragestellungen
  - Welche Arten von Ereignissen führen zur **Verdrängung**?
  - In welcher **Reihenfolge** sollen Prozesse ablaufen?
- Ziele eines *Scheduling*-Algorithmus
  - benutzerorientiert, z.B. kurze Antwortzeiten
  - systemorientiert, z.B. optimale CPU-Auslastung
- Kein *Scheduling*-Algorithmus kann alle Bedürfnisse erfüllen.

Prozesssynchronisation

- Beispiel: unkoordinierter Druckerzugriff

Prozess A

```
...
print("Hallo Otto\n");
print("Ruf mich an.\n");
print("Tel.: 420815\n");
...
```

Prozess B

```
...
print("Karl-");
print("Ich mag dich.\n");
...
```

Prozesssynchronisation

- Ursache: **kritische Abschnitte**
- Lösungsmöglichkeit: **Gegenseitiger Ausschluss**
  - *Mutex*-Abstraktion

Prozess A

```
...
lock(&printer_mutex);
print("Hallo Otto\n");
print("Ruf mich an.\n");
print("Tel.: 420815\n");
unlock(&printer_mutex);
...
```

Prozess B

```
...
lock(&printer_mutex);
print("Karl-");
print("Ich mag dich.\n");
unlock(&printer_mutex);
...
```

Wenn sich einer der Prozesse A oder B zwischen **lock** und **unlock** befindet, kann der jeweils andere das **lock** nicht passieren und blockiert dort, bis der kritische Abschnitt wieder frei ist (**unlock**).

Verklemmungen (*Deadlocks*)

Es gilt: **„Rechts vor links!“**
Kein Auto darf fahren.

Verklemmungssituationen wie diese kann es auch bei Prozessen geben.

Interprozesskommunikation

- … ermöglicht die Zusammenarbeit mehrerer Prozesse
  - lokal (*local*), z.B. Drucker-Dämon, *X-Server*
  - entfernt (*remote*), z.B. *Webserver*, Datenbank-*Server*, ftp-*Server*
    - „*Client/Server*-Systeme“
- Abstraktionen/Programmiermodelle
  - Gemeinsamer Speicher
    - mehrere Prozesse dürfen gleichzeitig denselben Speicherbereich nutzen
    - zusätzlich Synchronisation notwendig
  - Nachrichtenaustausch
    - Semantik eines Faxes (verschickt wird die Kopie einer Nachricht)
    - synchron oder asynchron

Inhalt

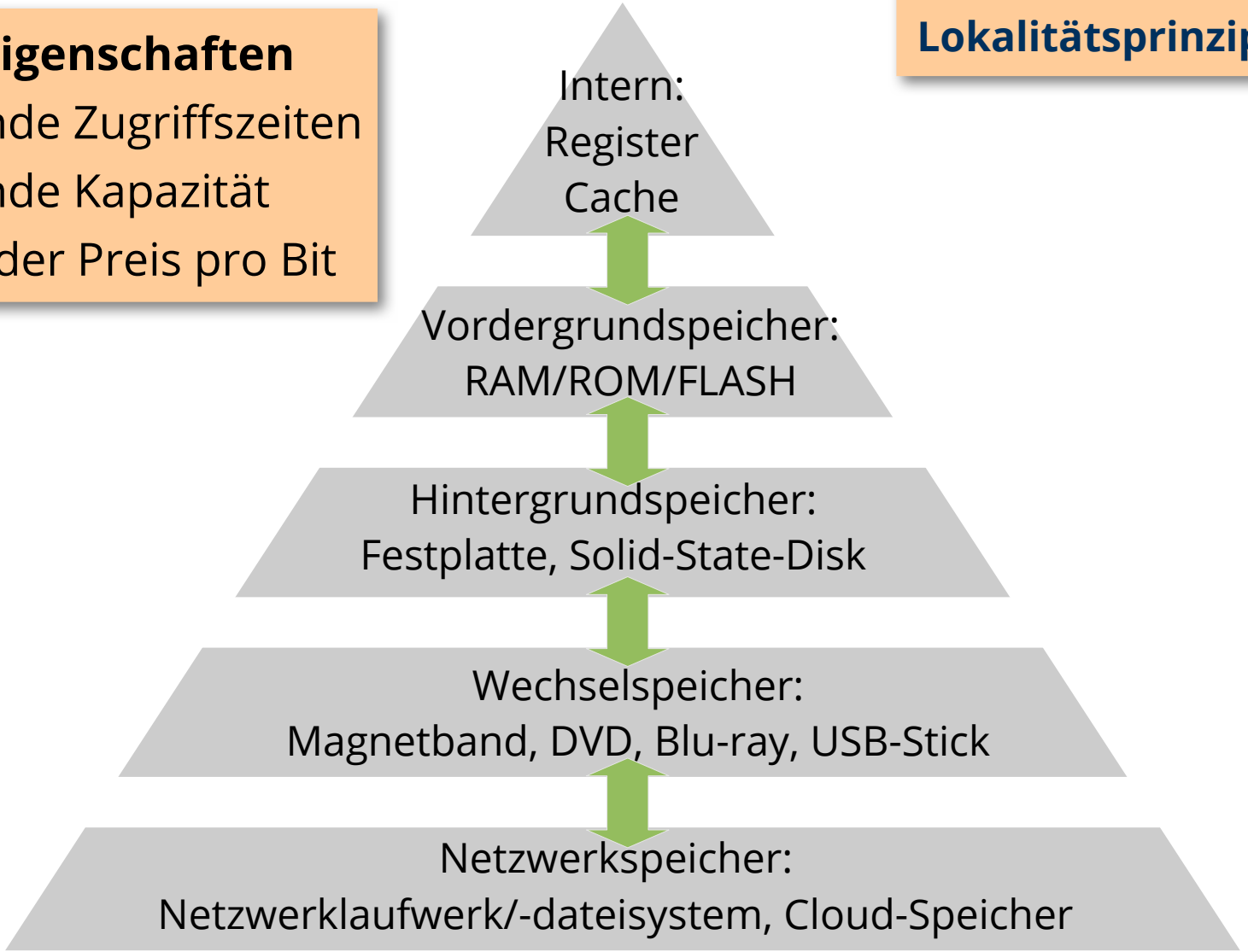
- Prozesse
  - CPU-Zuteilung
  - Synchronisation und Verklemmungen
  - Interprozesskommunikation
- **Speicherverwaltung**
  - Arbeitsspeicher
  - Hintergrundspeicher
- Systemsicherheit
- Multiprozessorsysteme
- *Cloud-Computing* und Virtualisierung

Die Speicherhierarchie

Speichereigenschaften

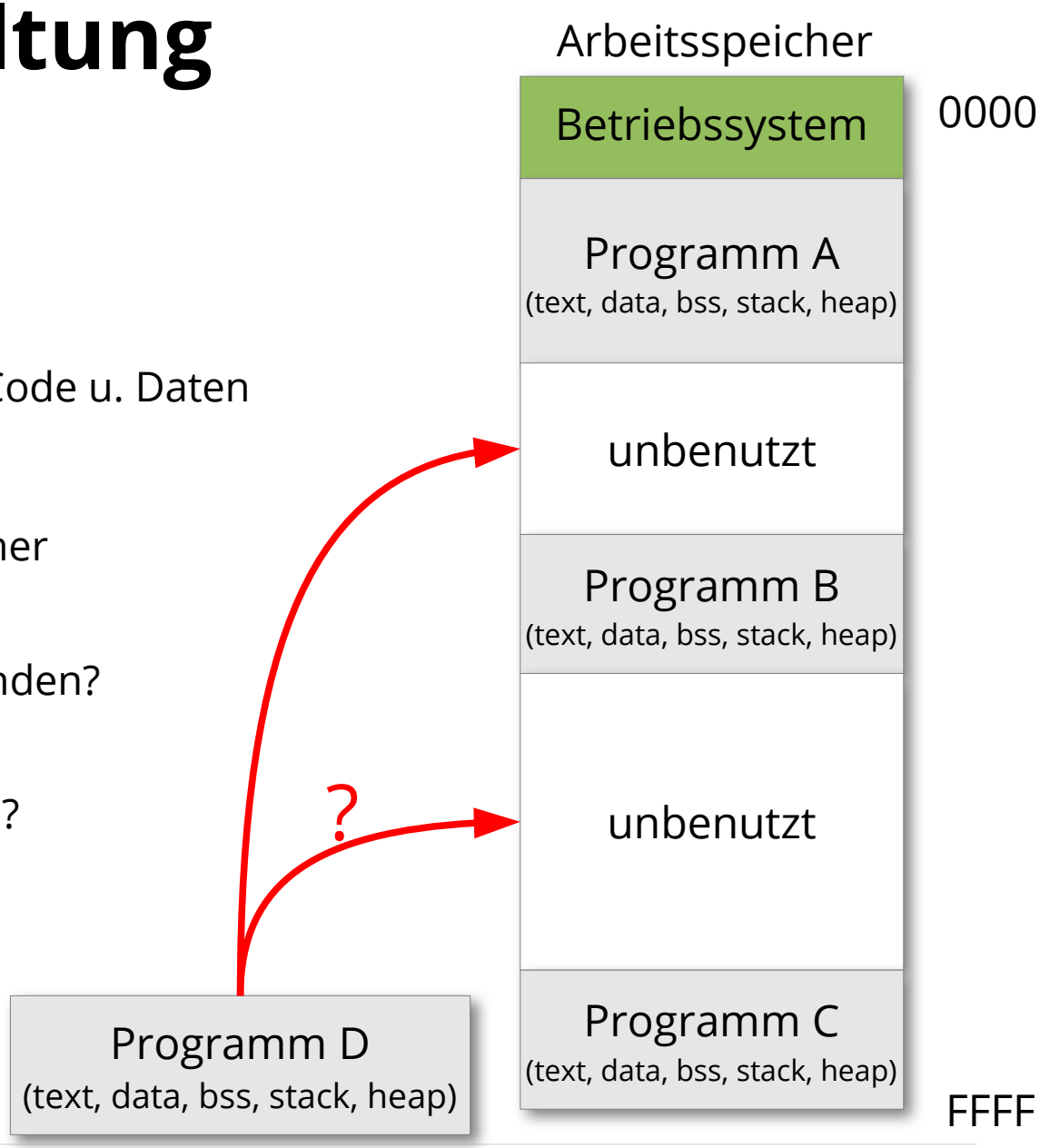
- steigende Zugriffszeiten
- steigende Kapazität
- sinkender Preis pro Bit

Sehr wirtschaftlich durch das **Lokalitätsprinzip**!

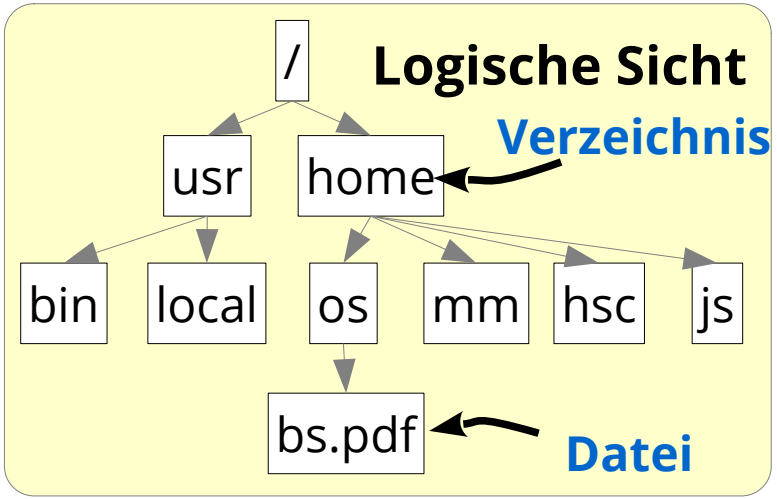


Speicherverwaltung

- **Adressabbildung**
  - **Logische Adressen** auf physische Adressen
  - Gestattet **Relokation** von Code u. Daten
- **Platzierungsstrategie**
  - In welcher Lücke soll Speicher reserviert werden?
  - **Kompaktifizierung** verwenden?
  - Wie minimiert man das **Fragmentierungsproblem**?
- **Ersetzungsstrategie**
  - Welcher Speicherbereich könnte sinnvoll ausgelagert werden?

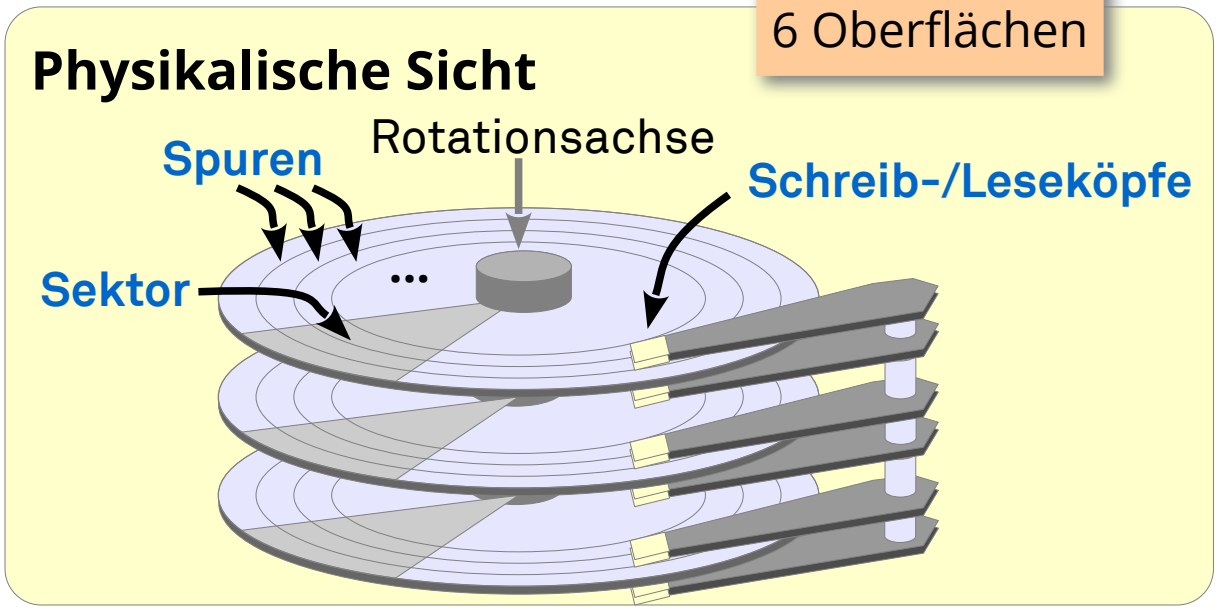


Hintergrundspeicher



Dateisysteme erlauben die dauerhafte Speicherung großer Datenmengen.

Abbildung



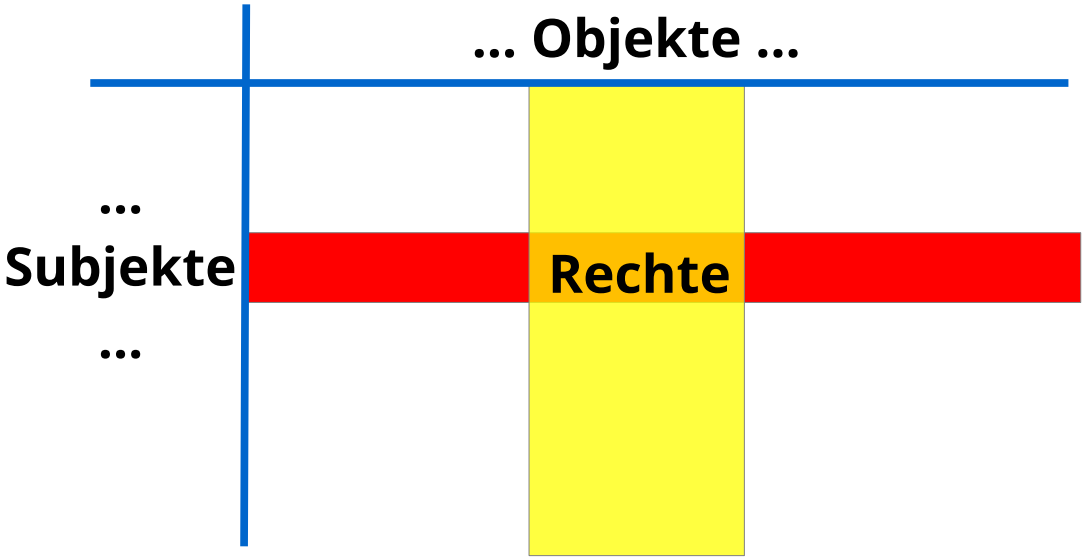
Das Betriebssystem stellt den Anwendungen die logische Sicht zur Verfügung und muss diese effizient realisieren.

Inhalt

- Prozesse
  - CPU-Zuteilung
  - Synchronisation und Verklemmungen
  - Interprozesskommunikation
- Speicherverwaltung
  - Arbeitsspeicher
  - Hintergrundspeicher
- **Systemsicherheit**
- Multiprozessorsysteme
- *Cloud-Computing* und Virtualisierung

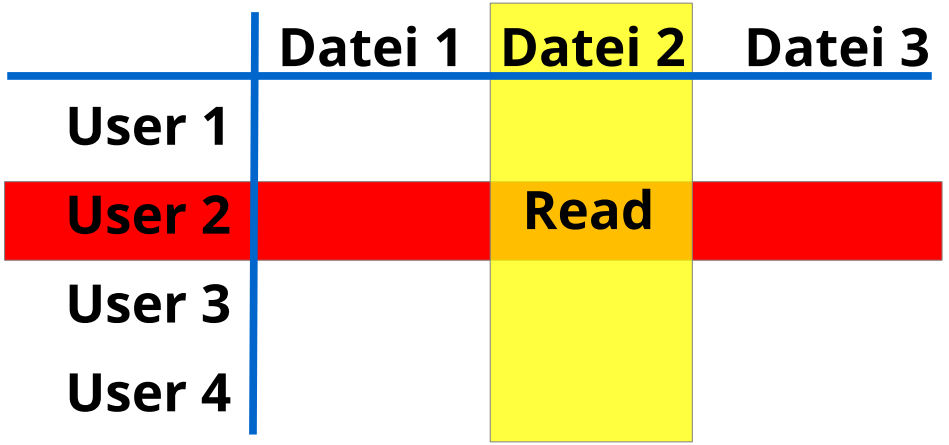
Zugriffsmatrix

- Begriffe:
  - **Subjekte** (Benutzer, Personen, Prozesse)
  - **Objekte** (Daten, Geräte, Prozesse, Speicher …)
  - **Operationen** (Lesen, Schreiben, Löschen, Ausführen …)
- Frage: Ist Operation(Subjekt, Objekt) zulässig?



Basismodell: Datei-/Prozessattribute

- Festlegungen in Bezug auf Benutzer:
  - Für welchen Benutzer arbeitet ein Prozess?
  - Welchem Benutzer gehört eine Datei (*owner*)?
  - Welche Rechte räumt ein Benutzer anderen und sich selbst an „seiner“ Datei ein?
- Rechte eines Prozesses an einer Datei
  - Attribute von Prozessen:
    User ID
  - Attribute von Dateien:
    Owner ID



Unix-Zugriffsrechte

- Unix: rudimentäre Zugriffssteuerlisten
  - Prozess: *User ID*, (eine oder mehrere) *Group IDs*
  - Datei: *Owner*, *Group*
  - Rechte in Bezug auf *User* (*Owner*), *Group*, *Others*

Datei.tex		
`rw-`	`r--`	`---`
		Others
	Group: staff	
User: me		

Dateiattribute:

rwx

- Ausführen (*e**X**ecute*): ja/nein
- Schreiben (***W**rite*): ja/nein
- Lesen (***R**ead*): ja/nein

Inhalt

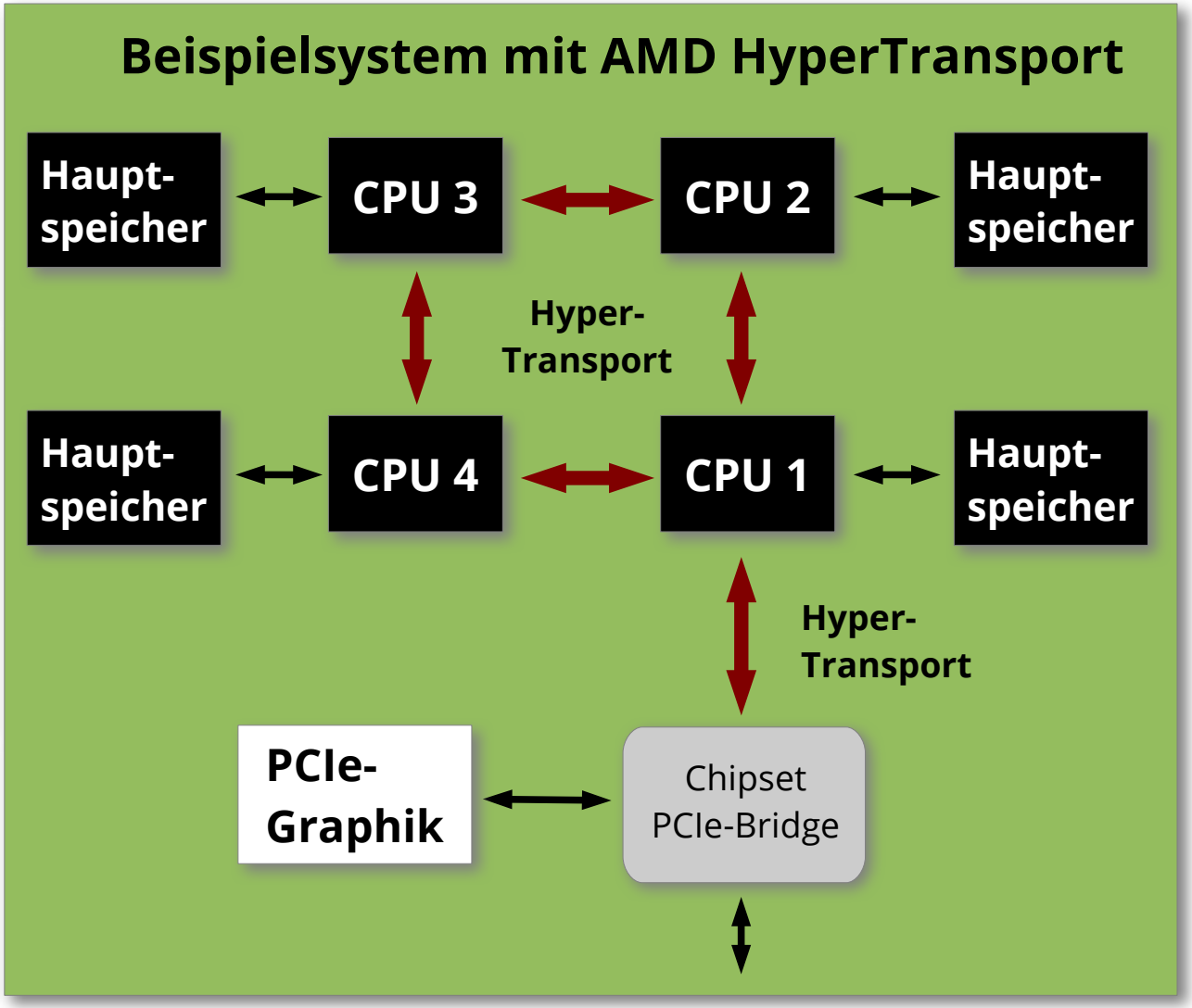
- Prozesse
  - CPU-Zuteilung
  - Synchronisation und Verklemmungen
  - Interprozesskommunikation
- Speicherverwaltung
  - Arbeitsspeicher
  - Hintergrundspeicher
- Systemsicherheit
- **Multiprozessorsysteme**
- *Cloud-Computing* und Virtualisierung

NUMA-Architekturen *(Non-Uniform Memory Architecture)*

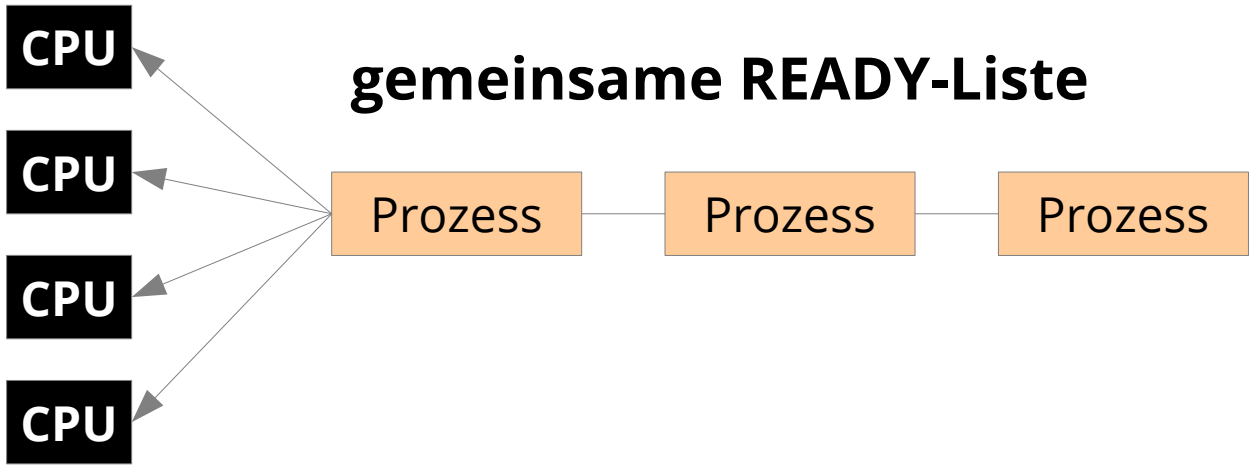
Die CPUs (u.U. mit mehreren Cores) kommunizieren untereinander via HyperTransport.

Globaler Adressraum: An andere CPUs angebundener Hauptspeicher kann adressiert werden, die **Latenz ist jedoch höher**.

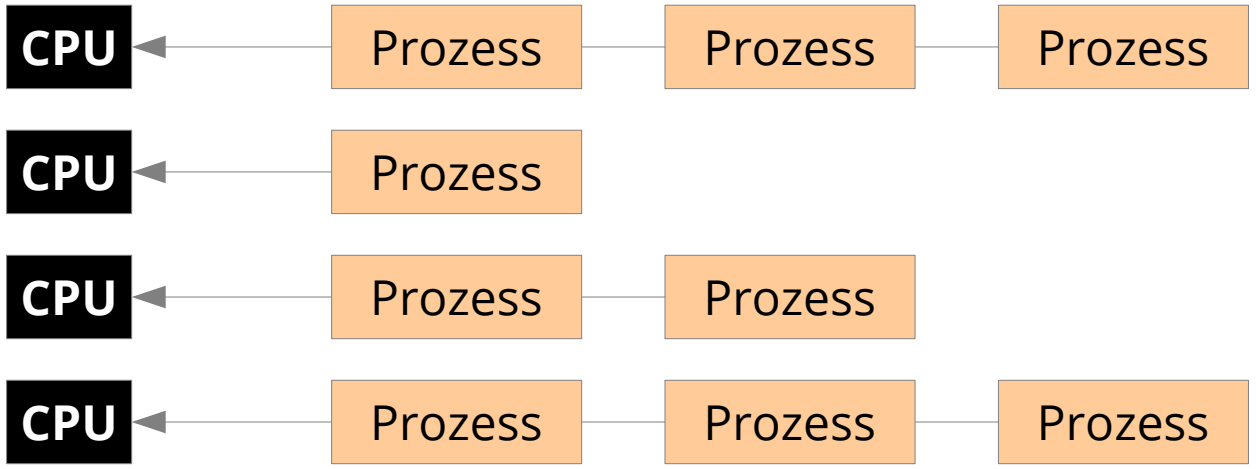
Ansatz skaliert besser, da parallele Speicherzugriffe möglich sind.



CPU-Zuteilung im Multiprozessor



oder ...

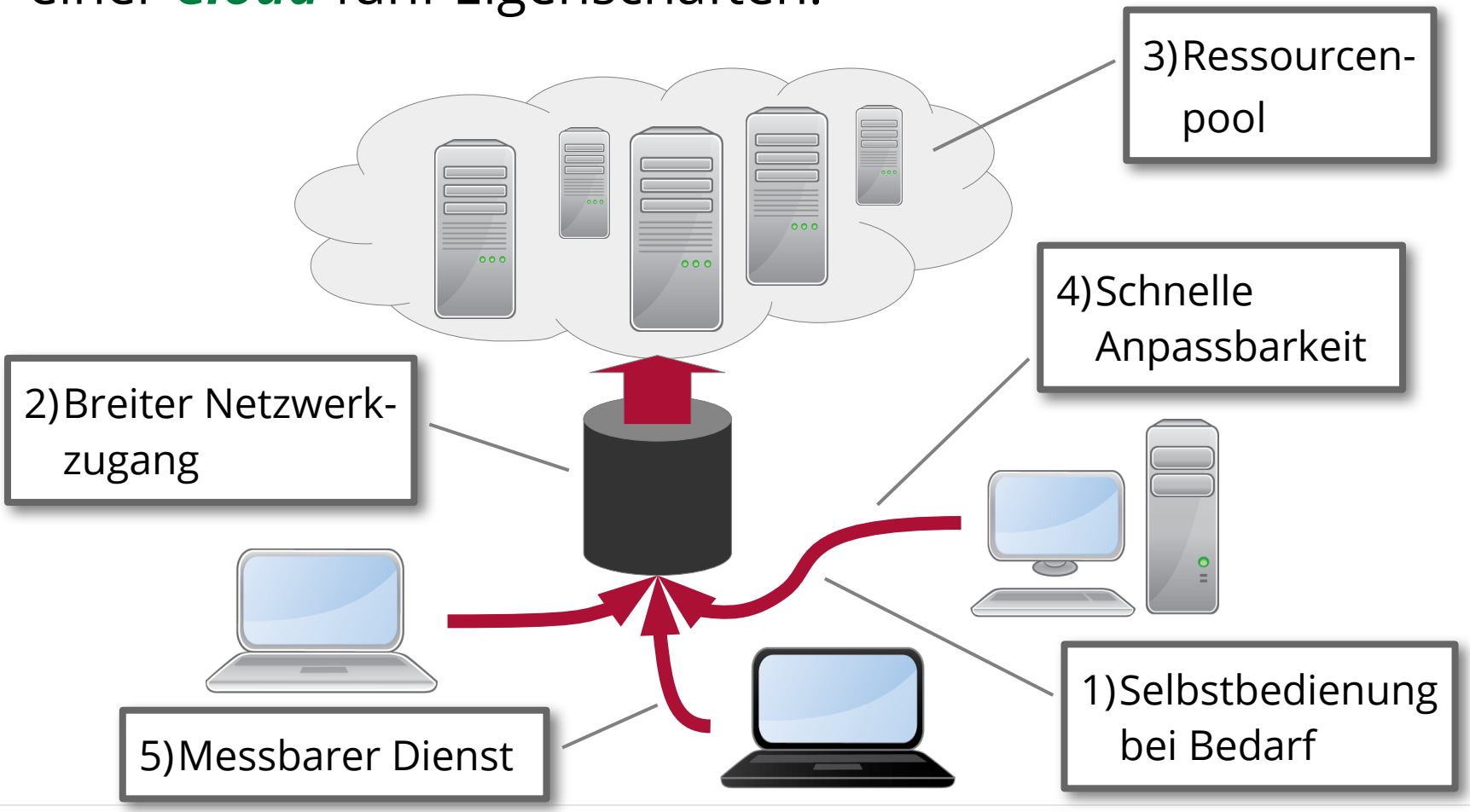


Inhalt

- Prozesse
  - CPU-Zuteilung
  - Synchronisation und Verklemmungen
  - Interprozesskommunikation
- Speicherverwaltung
  - Arbeitsspeicher
  - Hintergrundspeicher
- Systemsicherheit
- Multiprozessorsysteme
- ***Cloud-Computing* und Virtualisierung**

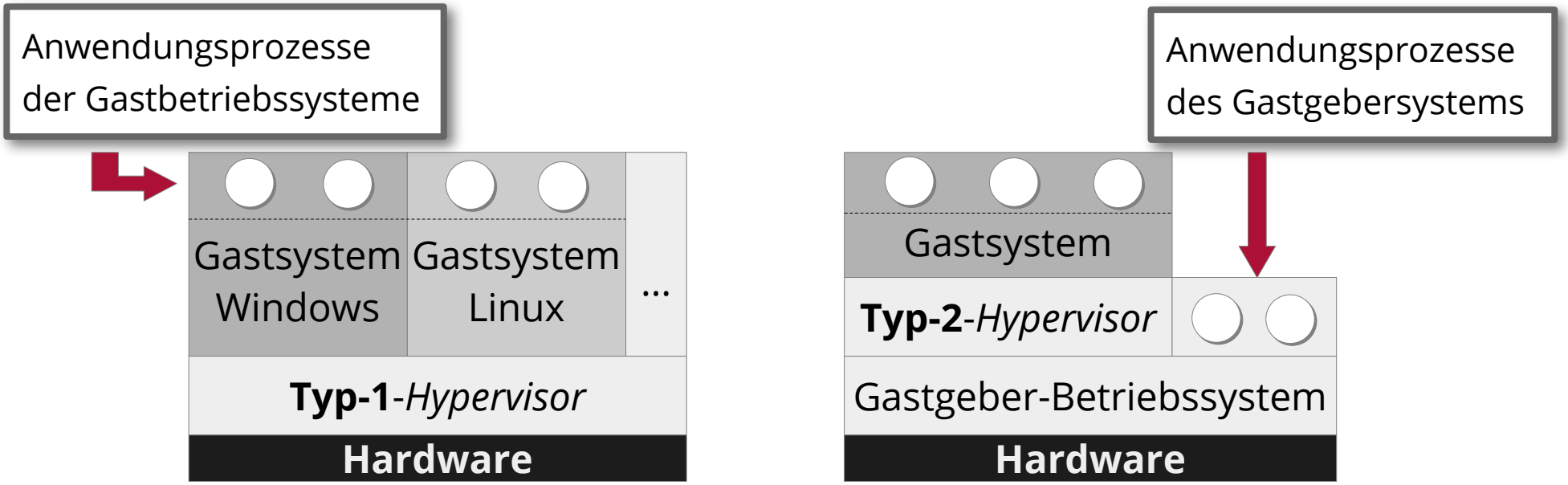
Cloud Computing

- Laut *National Institute of Standards and Technology* gehören zu einer ***Cloud*** fünf Eigenschaften:



Hardware-Virtualisierung

- … ermöglicht es in einer realen Maschine mehrere virtuelle Maschinen zu schaffen, die alle ihr eigenes Betriebssystem haben können.
  - Wichtige Grundlage für *Cloud-Computing* und *Server*-Konsolidierung.
- Technische Basis: ***Hypervisor*** / ***Virtual Machine Monitor***



Fazit: Das Betriebssystem …

- verwaltet **Betriebsmittel**, insbesondere CPU und Speicher
- stellt **Abstraktionen** zur Verfügung, z.B. …
  - Prozesskonzept
  - Dateien und Verzeichnisse
  - Rechtekonzept
- ist für ein bestimmtes **Anwendungsprofil** optimiert
  - Allen Anwendungen 100% gerecht zu werden ist unmöglich.

Betriebssysteme, typische Anwendungen und Hardware haben sich Hand in Hand im Laufe der vergangenen Jahrzehnte entwickelt. Die heute üblichen **Systemabstraktionen sind das Ergebnis einer Evolution**, deren Ende nicht in Sicht ist.