CORONA-EDITION

Betriebssysteme (BS)

02. Abstraktionen und Strukturen

https://sys.cs.tu-dortmund.de/DE/Teaching/SS2021/BS/

21.04.2021

Peter Ulbrich

peter.ulbrich@tu-dortmund.de

Basierend auf *Betriebssysteme* von Olaf Spinczyk, Universität Osnabrück

technische universität
dortmund

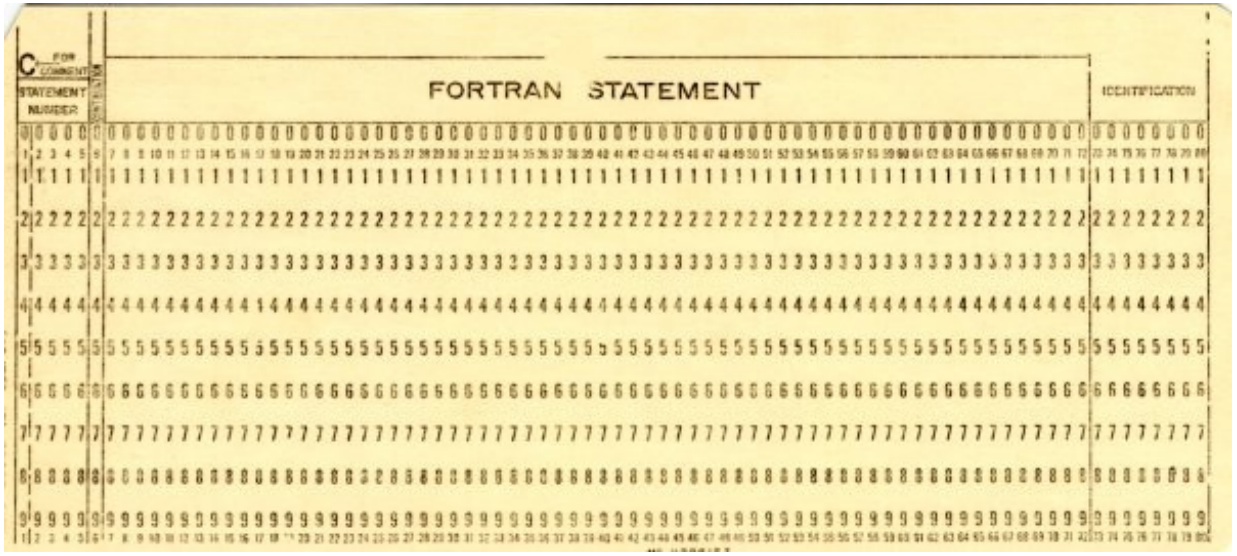
arbeitsgruppe
systemsoftware

Inhalt

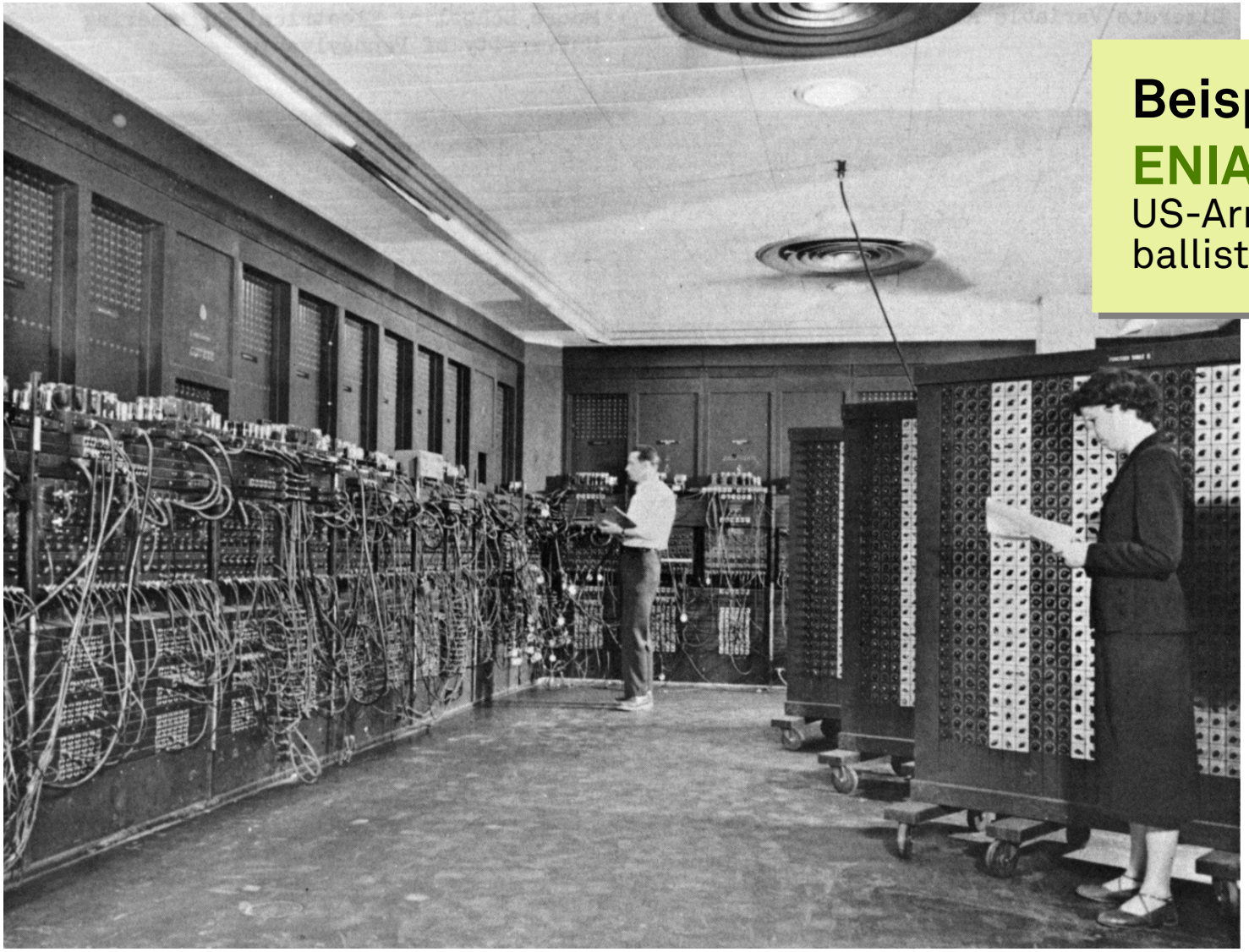
- **Ein Blick in die Geschichte**
  - Serielle Verarbeitung und Stapelbetrieb
  - Mehrprogramm- und Dialogbetrieb
- Systemabstraktionen im Überblick
  - Prozesse
    - CPU-Zuteilung
    - Synchronisation und Verklemmungen
    - Interprozesskommunikation
  - Speicherverwaltung
    - Arbeitsspeicher
    - Hintergrundspeicher

Am Anfang stand die Lochkarte

- Es gibt sie schon seit 1725 – zur Webstuhlsteuerung.
- Herman Hollerith nutzte sie 1890 für eine Volkszählung
  - aus seiner Firma und zwei weiteren ging später IBM hervor
- Sie wurde bis in die 70er Jahre als vielseitiger Speicher eingesetzt.



Erste elektronische Universalrechner



Beispiel:

ENIAC, 1946
US-Armee, Berechnung ballistischer Tabellen

Erste elektronische Universalrechner

Beispiel:

ENIAC, 1946
US-Armee, Berechnung ballistischer Tabellen

Ein Rechenmonster!

- Größe: 10m x 17m x 2,7m
- Gewicht: 27t
- Leistung: 174kW (> 17.000 Röhren!)
- Preis: $ 468.000 (heute: ~ $6,36 Mio.)
- Geschwindigkeit: **500 Additionen pro Sekunde**

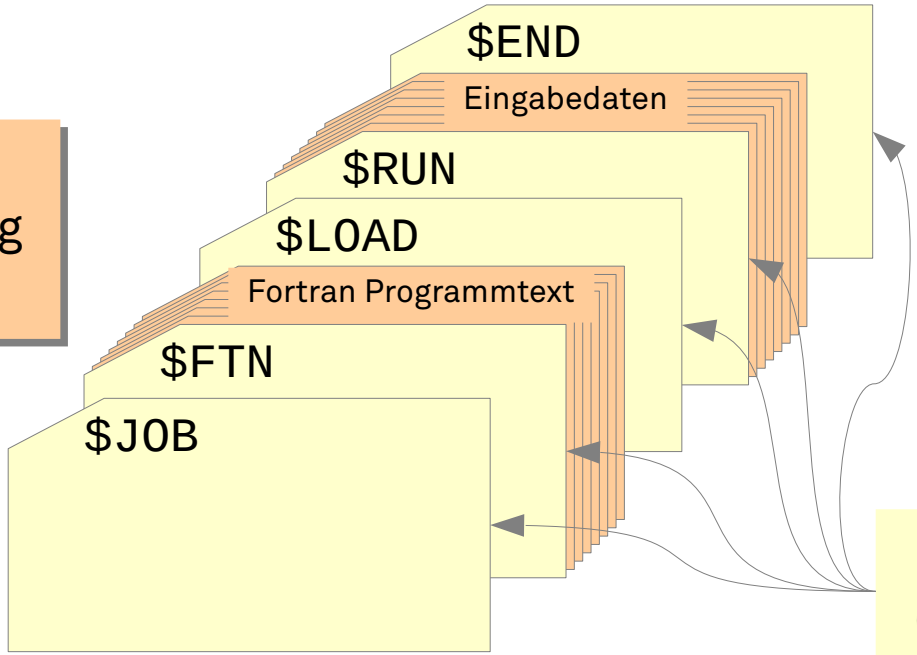
Serielle Verarbeitung (ab 1945)

- Programmierung
  - i.d.R. in Maschinencode
  - Eingabe über Lochkartenleser, Ausgaben über Drucker
  - Fehleranzeige durch Kontrolllämpchen
- Rechnerzeitzuteilung auf Papierterminkalender
  - **Rechnerzeitverschwendung** durch zu großzügige Reservierung oder Abbruch wegen Fehler
- Minimale Auslastung der CPU
  - Die meiste Zeit verbrauchten **langsame E/A-Geräte** (Lochkarten, Drucker)
- Erste Systemsoftware in Form von **Programmbibliotheken**
  - Binder, Lader, *Debugger*, Gerätetreiber, …

Einfache Stapelsysteme (ab 1955)

- Verringerten die Häufigkeit manueller Betriebseingriffe
- Die ersten Betriebssysteme: **Residente Monitore**
  - Interpretation von `Job`-Steuerbefehlen
  - Laden und Ausführen von Programmen
  - Geräteansteuerung

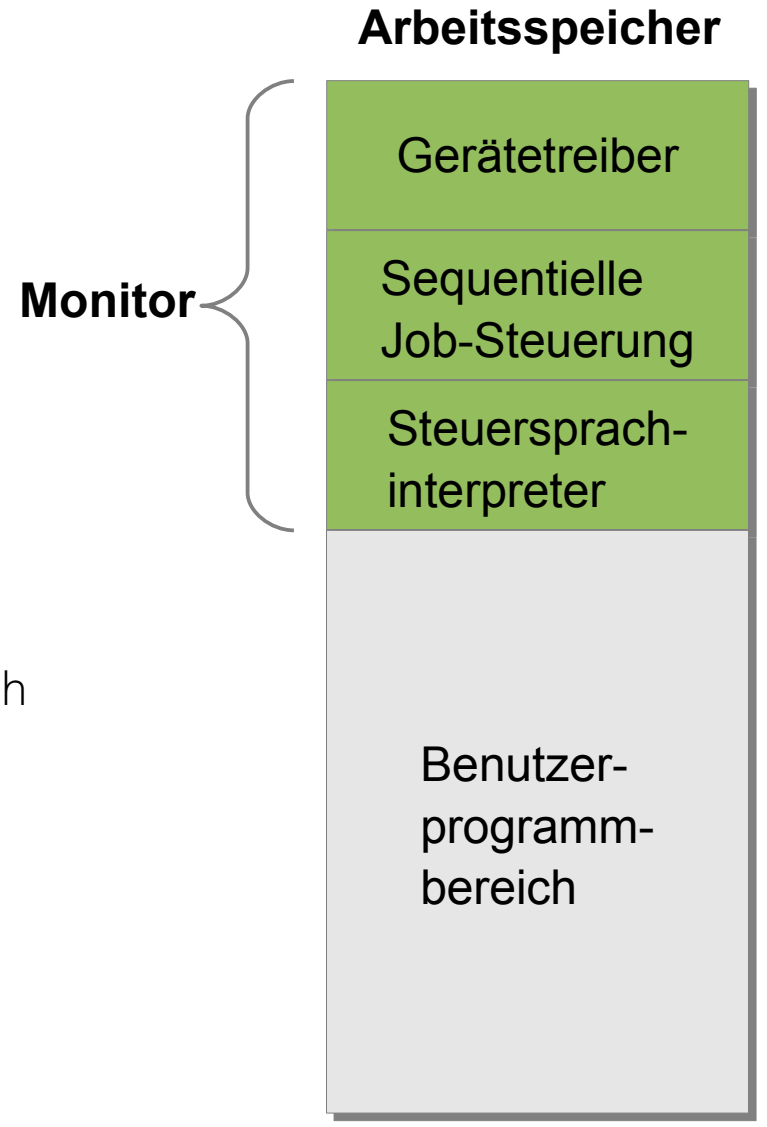
Ein Stapel Lochkarten zur Übersetzung und Ausführung eines FORTRAN-Programms



NEU: **Steuerkarten** (engl. *control cards*)

Einfache Stapelsysteme (ab 1955)

Der ***Monitor*** bliebt dauerhaft im Speicher, während er ein Anwendungsprogramm nach dem anderen ausführte.

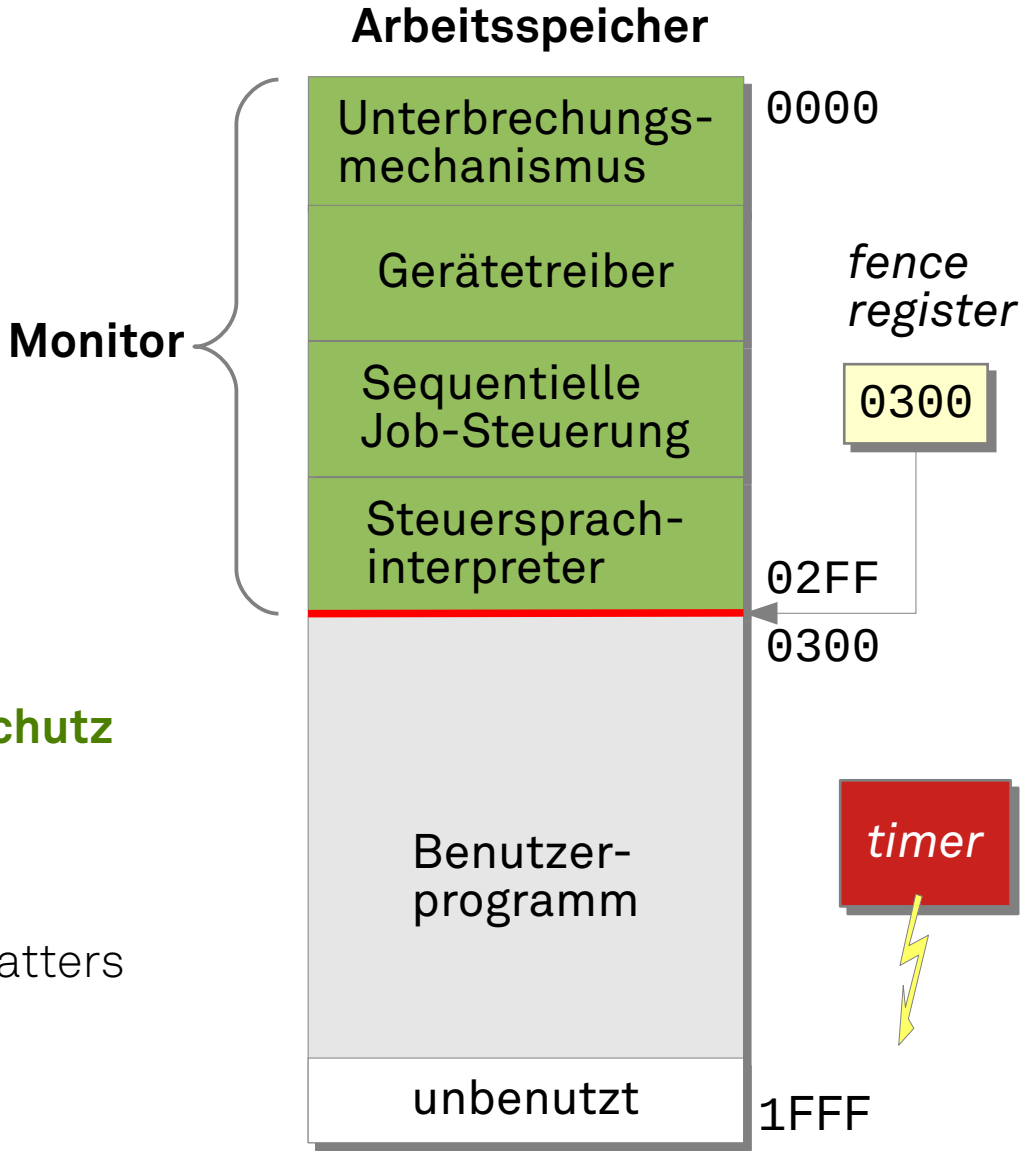


- **Probleme** durch fehlerhafte Anwendungen:
  - Programm terminiert nicht,
  - schreibt in den Speicherbereich des residenten Monitors
  - greift auf den Kartenleser direkt zu und interpretiert Steuerbefehle als Daten.

Einfache Stapelsysteme (ab 1955)

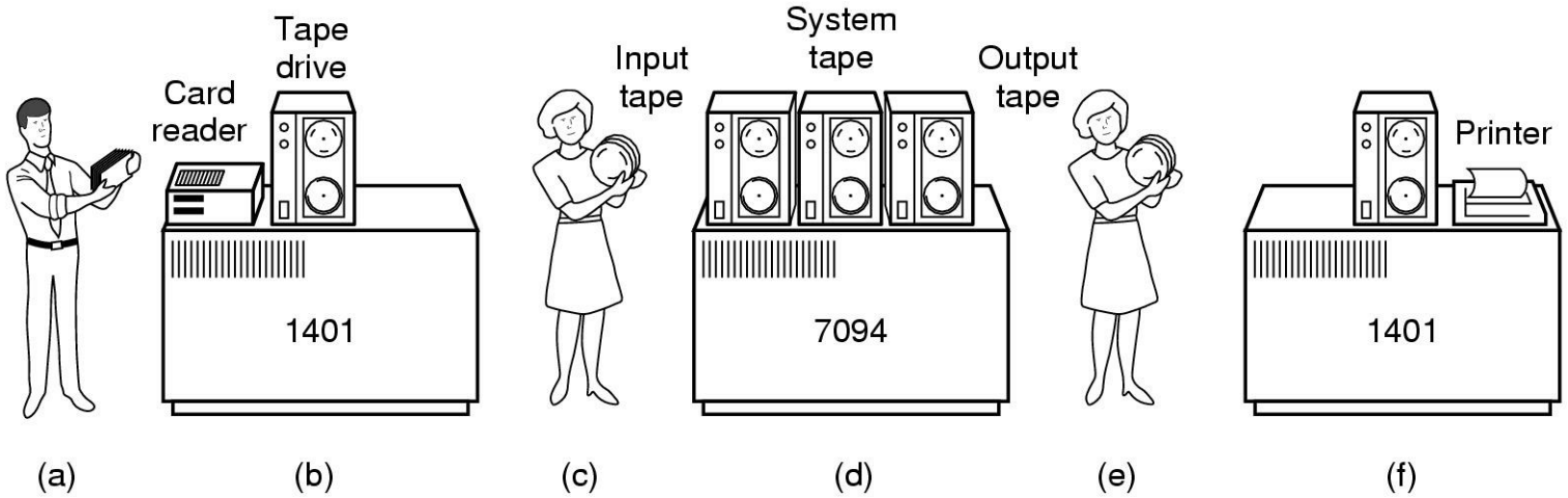
Lösungen:

- Zeitgeberbaustein (*timer*) liefert **Unterbrechungen** (*interrupts*)
- **Fallen** (*traps*) für fehlerhafte Programme
  - Schutzgatterregister (engl. *fence register*) realisiert primitiven **Speicherschutz**
  - **Privilegierter Arbeitsmodus** der CPU (supervisor mode)
    - Deaktivierung des Schutzgatters
    - Ein-/Ausgabe



Der Ein-/Ausgabe-Flaschenhals

- **Problem:** CPU ist schneller als Kartenleser und Drucker
  - kostbare Rechenzeit wird durch (aktives) Warten verschwendet
- **Lösung 1:** ***Off-line processing*** *(entkoppelte Ein-/Ausgabe)*
  - dank Bandlaufwerken
  - Parallelisierung von Ein-/Ausgaben durch mehrere Satellitenrechner



A. Tanenbaum [2]

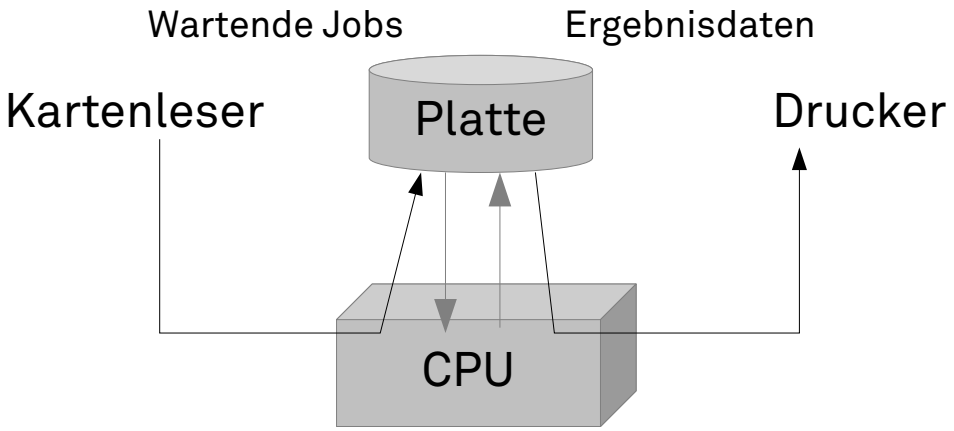
Satellitenrechner für Eingabe (ggf. mehrere)

CPU (liest/schreibt ausschließlich von/auf Band)

Satellitenrechner für Ausgabe (ggf. mehrere)

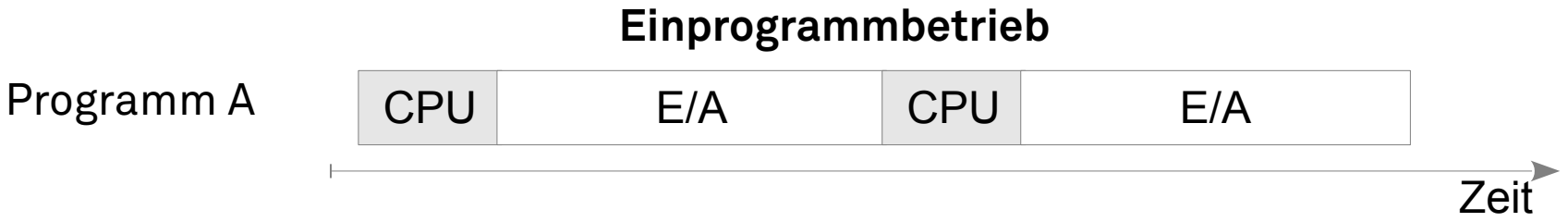
Der Ein-/Ausgabe-Flaschenhals

- **Problem:** CPU ist schneller als Kartenleser und Drucker
  - kostbare Rechenzeit wird durch (aktives) Warten verschwendet
- **Lösung 2:** ***Spooling*** *(Spulen, Puffern)*
  - dank Plattenlaufwerken (wahlfreier Zugriff), ***Direct Memory Access*** und ***Unterbrechungen***
  - Berechnungen und Ein-/Ausgaben werden dabei parallelisiert.
  - Regeln für **Prozessorzuteilung**

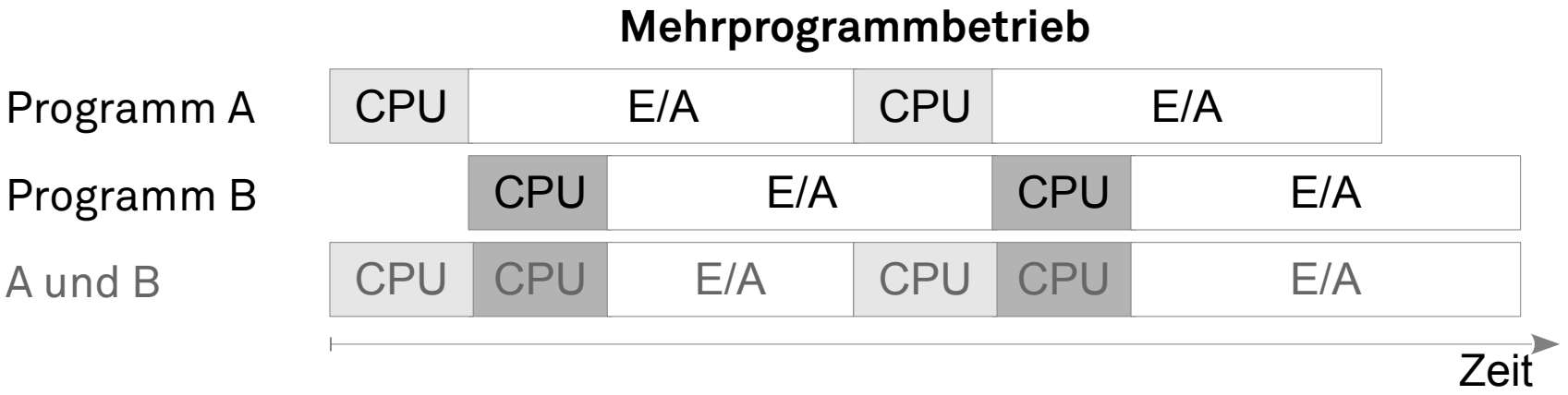


Mehrprogrammbetrieb (ab 1965)

- Trotz *spooling* nutzt ein einzelnes Programm die CPU nicht effizient
  - **CPU-Stöße** (*CPU bursts*) und **E/A-Stöße** (*I/O bursts*), bei denen die CPU warten muss, wechseln sich ab.



- Beim **Mehrprogrammbetrieb** bearbeitet die CPU mehrere Aufträge gleichzeitig:



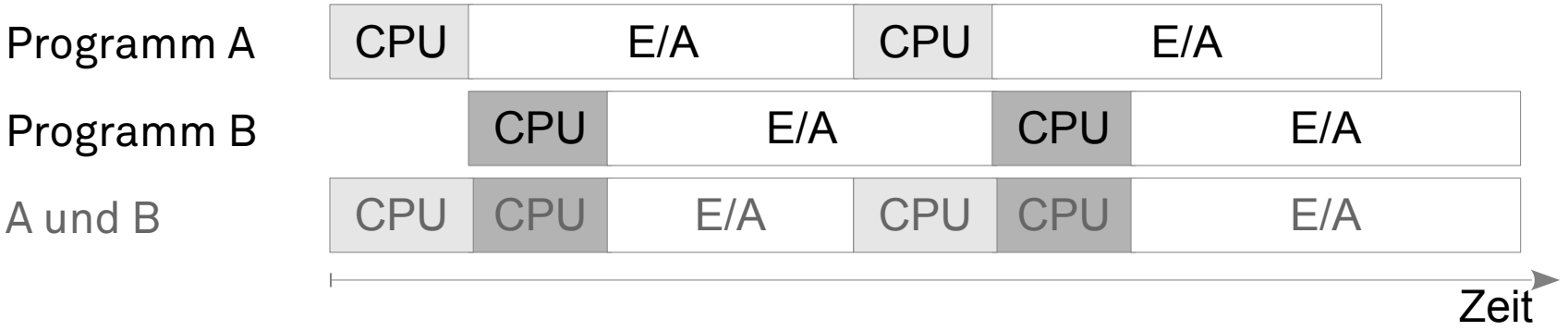
Mehrprogrammbetrieb (ab 1965)

- Trotz *spooling* nutzt ein einzelnes Programm die CPU nicht effizient
  - **CPU-Stöße** (*CPU burste*) und **E/A-Stöße** (*I/O burste*), bei denen die

Das Betriebssystem wird immer komplexer:

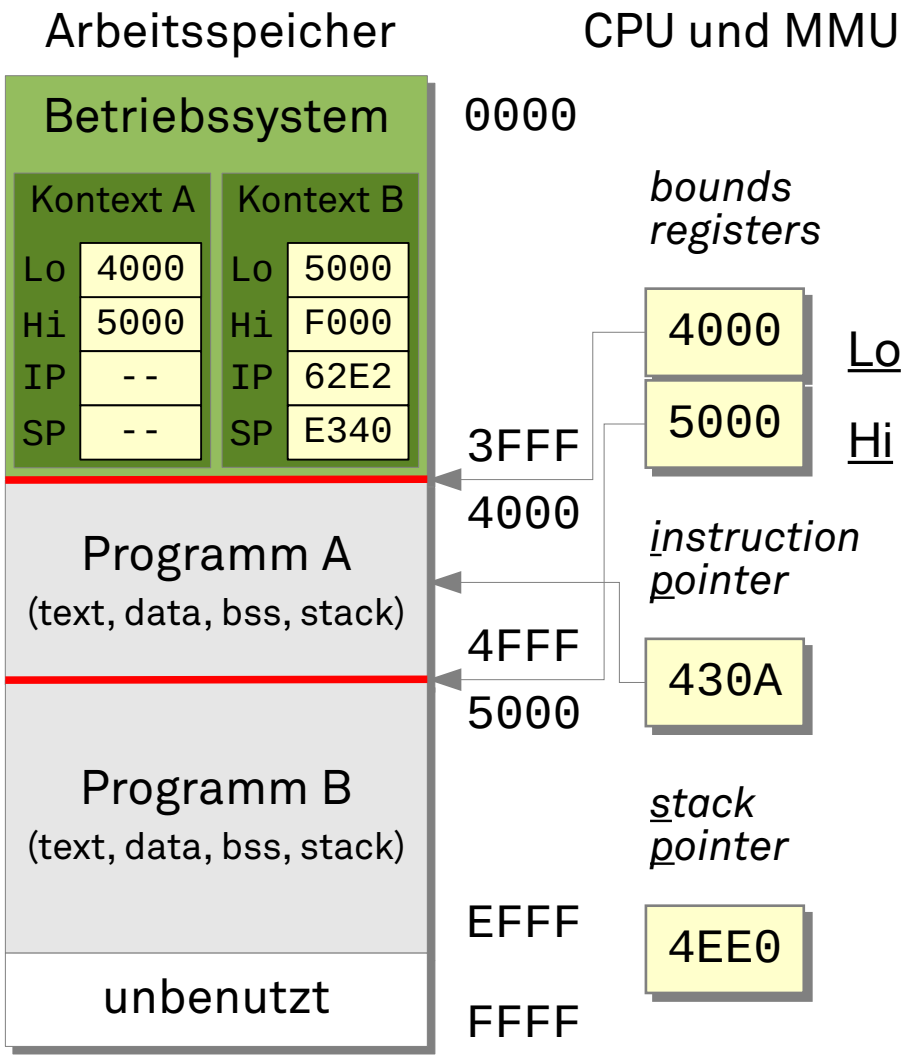
- Umgang mit nebenläufigen E/A-Aktivitäten
- **Verwaltung des Arbeitsspeichers** für mehrere Programme
- Interne Verwaltung von Programmen in Ausführung (**Prozesse**)
- **Prozessorzuteilung** (*scheduling*)
- Mehrbenutzerbetrieb: **Sicherheit** und Abrechnung (*accounting*)

Mehrprogrammbetrieb



Mehrprogrammbetrieb (ab 1965)

- **Speicherverwaltung**
  - Den zu startenden Programmen muss dynamisch freier Speicher zugewiesen werden.
- **Speicherschutz**
  - Einfaches Schutzgatter reicht nicht mehr, um einzelne Programme zu isolieren. Lösung: einfache **MMU** („*Memory Management Unit*“)
- **Prozessverwaltung**
  - Jedes „Programm in Ausführung“ besitzt einen **Kontext**. Beim Prozesswechsel muss dieser ausgetauscht werden.

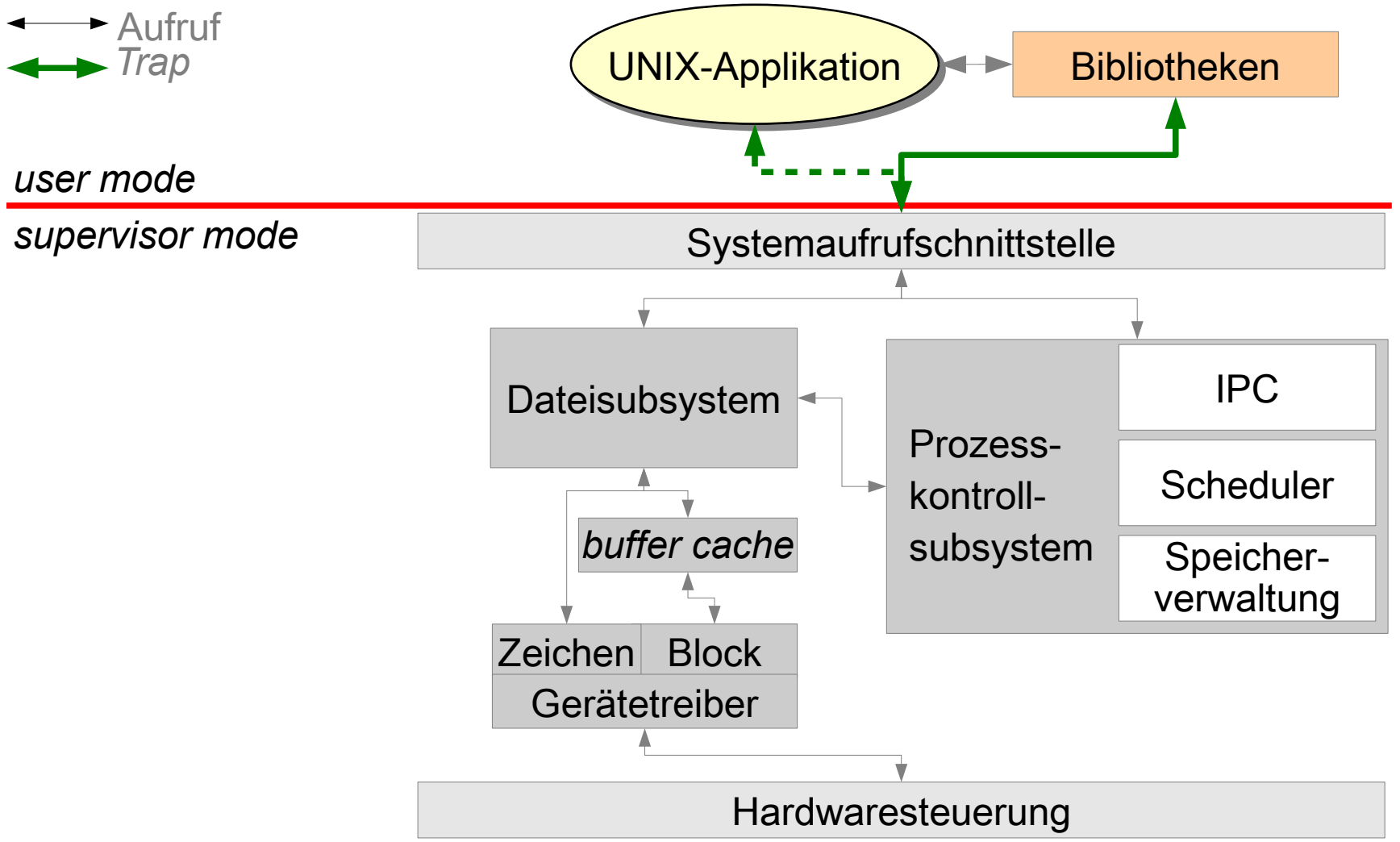


Dialogbetrieb (ab 1970)

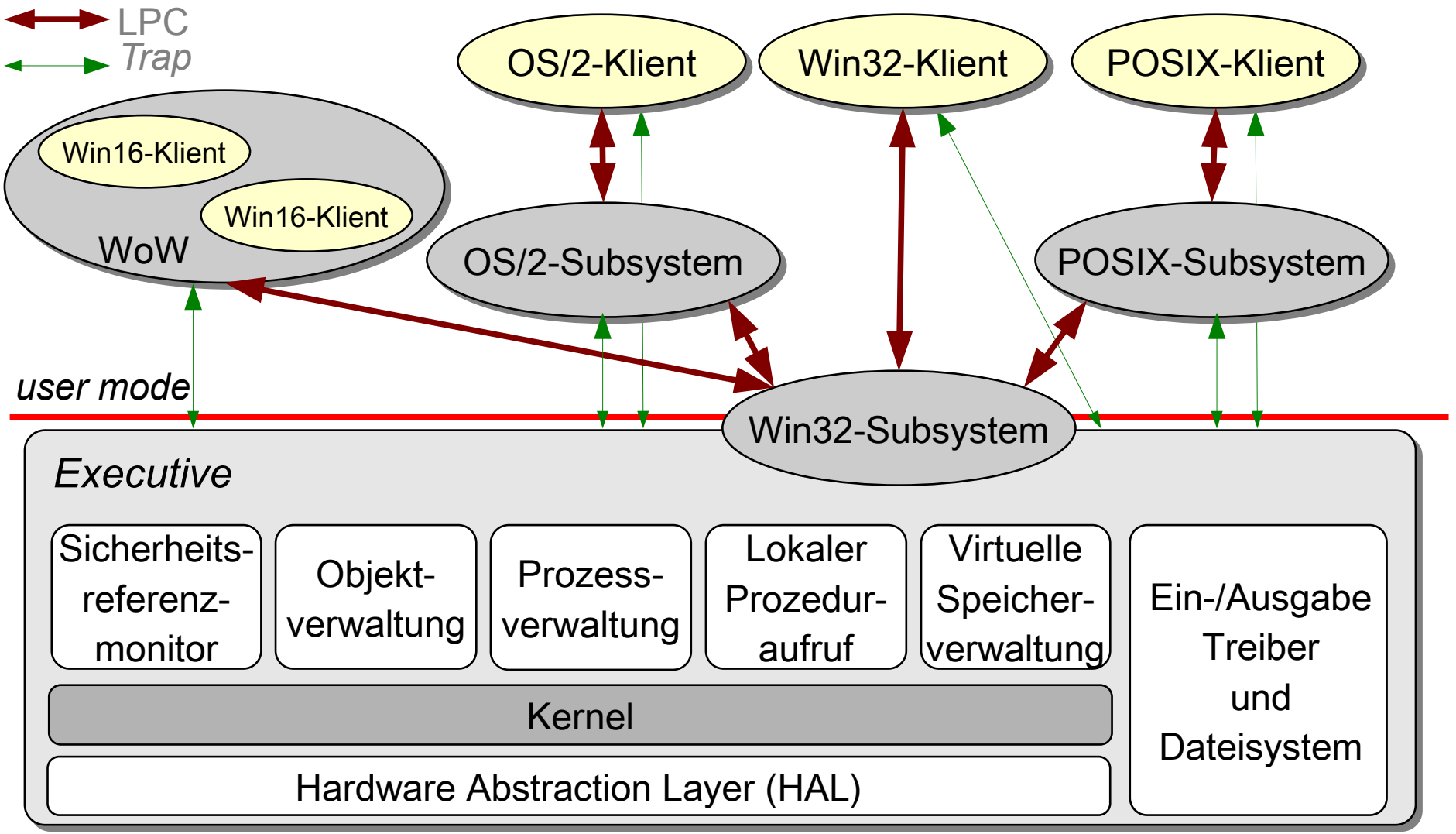
- Neue Ein- und Ausgabegeräte erlauben interaktive Software
  - Tastatur, Monitor, später Maus
- ***Time-Sharing***-Betrieb
  - ermöglicht akzeptable Antwortzeiten für interaktive Nutzer
  - Zeitgeber-Unterbrechungen sorgen für Verdrängung (zu) lang laufender Prozesse
- Systemprogramme erlauben auch interaktive SW-Entwicklung
  - *Editor*, *Shell*, Übersetzer, *Debugger*
- Platten und Dateisysteme erlauben jederzeit Zugriff auf Programme und Daten

Quelle: DIGITAL Computing Timeline

UNIX-Systemstruktur



Windows-Systemstruktur



Inhalt

- Ein Blick in die Geschichte
  - Serielle Verarbeitung und Stapelbetrieb
  - Mehrprogramm- und Dialogbetrieb
- **Systemabstraktionen im Überblick**
  - **Prozesse**
    - CPU-Zuteilung
    - Synchronisation und Verklemmungen
    - Interprozesskommunikation
  - Speicherverwaltung
    - Arbeitsspeicher
    - Hintergrundspeicher

Ein Prozess ...

- Horning/Randell, Process Structuring

> *„… P ist ein Tripel (S, f, s), wobei S einen Zustandsraum, f eine Aktionsfunktion und $s \subset S$ 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.“*

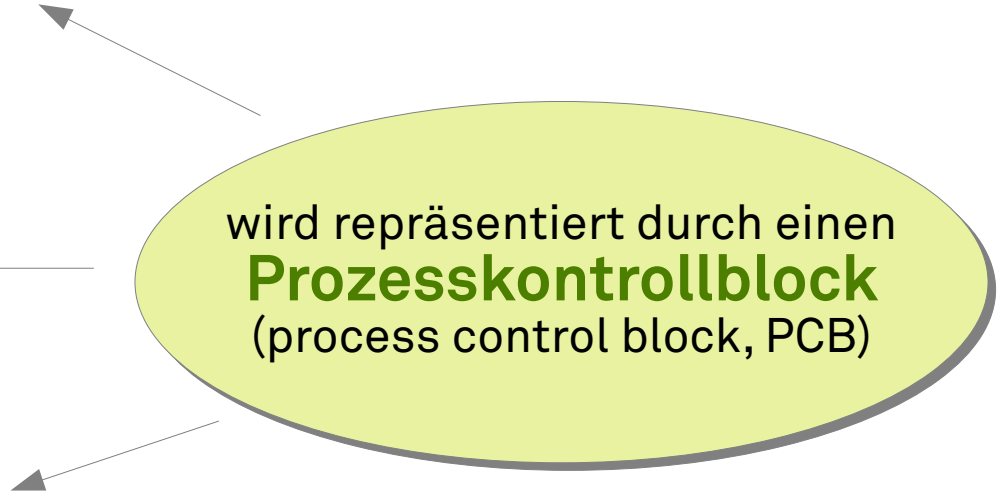
- Habermann, Introduction to Operating System Design

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

Ein Prozess …

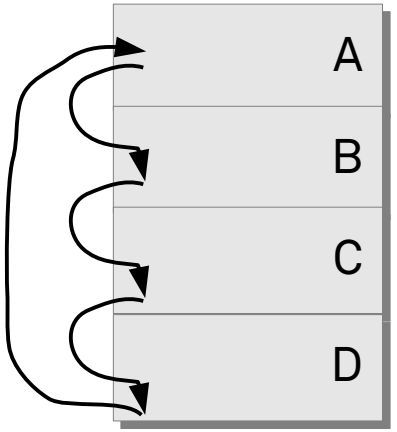
- ***…ist ein Programm in Ausführung***
  - unbekannte Referenz, Informatikfolklore

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

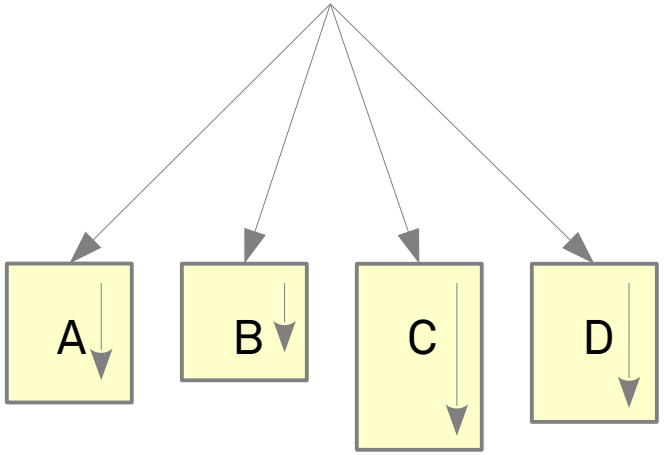


Prozessmodell

Mehrprogrammbetrieb



Nebenläufige Prozesse



Multiplexing der CPU

Technische Sicht

- 1 Instruktionszeiger
- Kontextwechsel

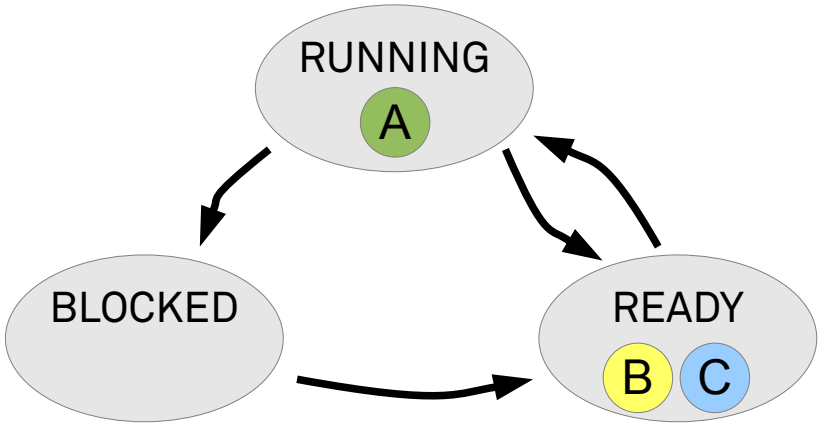
Konzeptionelle Sicht

- 4 unabhängige sequentielle Kontrollflüsse

Realzeit-Sicht

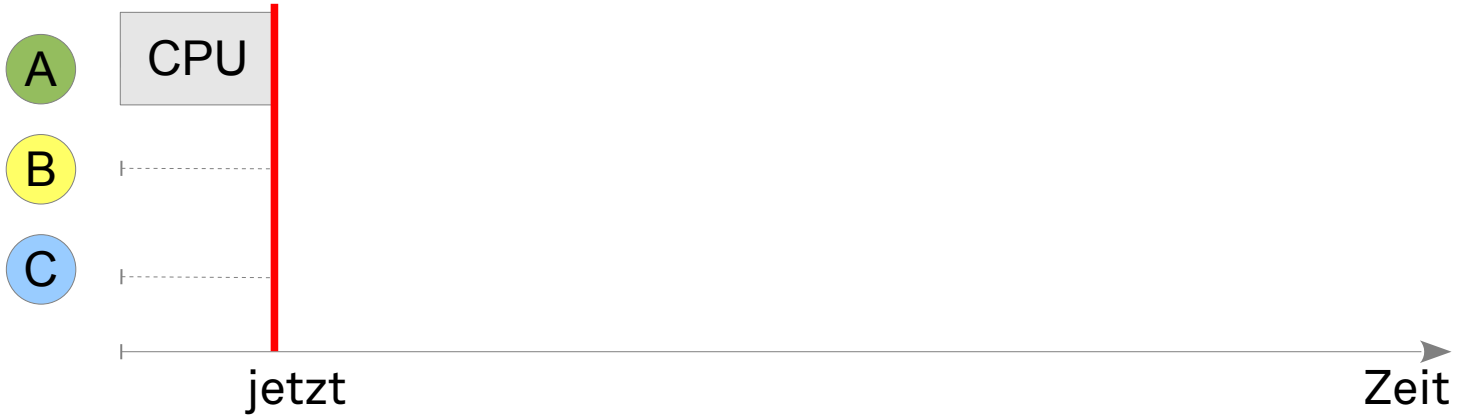
- Zu jedem Zeitpunkt ist nur ein Prozess aktiv (1 CPU)
- Gantt-Diagramm

Prozessverhalten und -zustände (3)

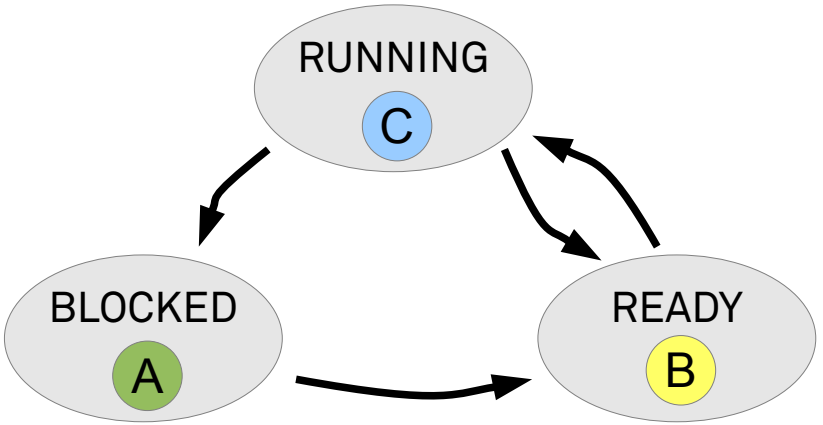


Prozesszustände

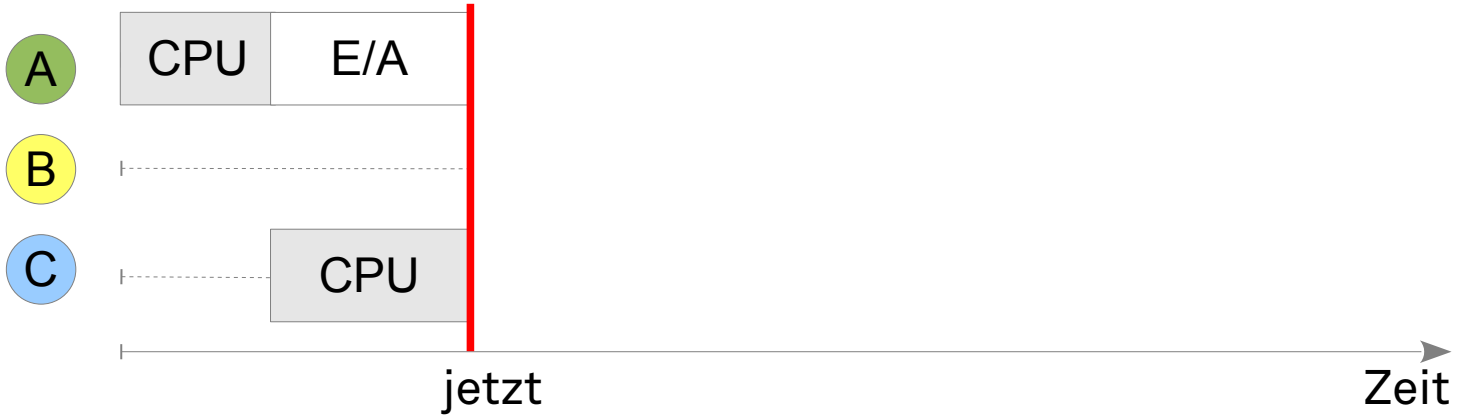
- **RUNNING**
  - Prozess wird gerade ausgeführt
- **READY**
  - Prozess ist rechenbereit, wartet auf die CPU
- **BLOCKED**
  - Prozess wartet auf die Beendigung einer E/A-Aktivität



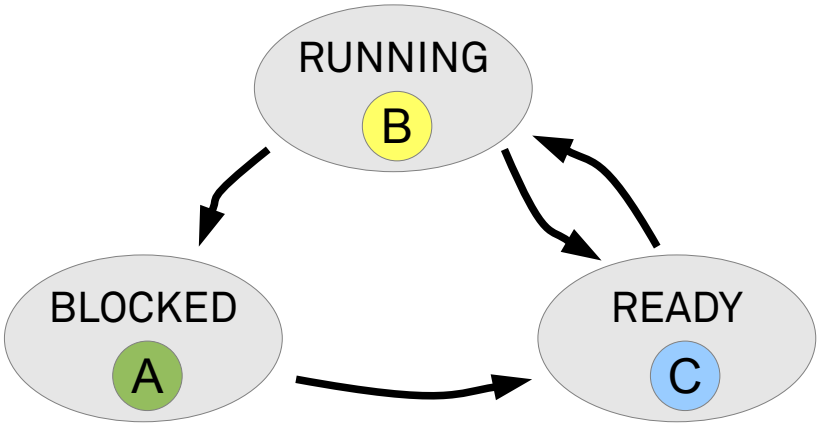
Prozessverhalten und -zustände (3)



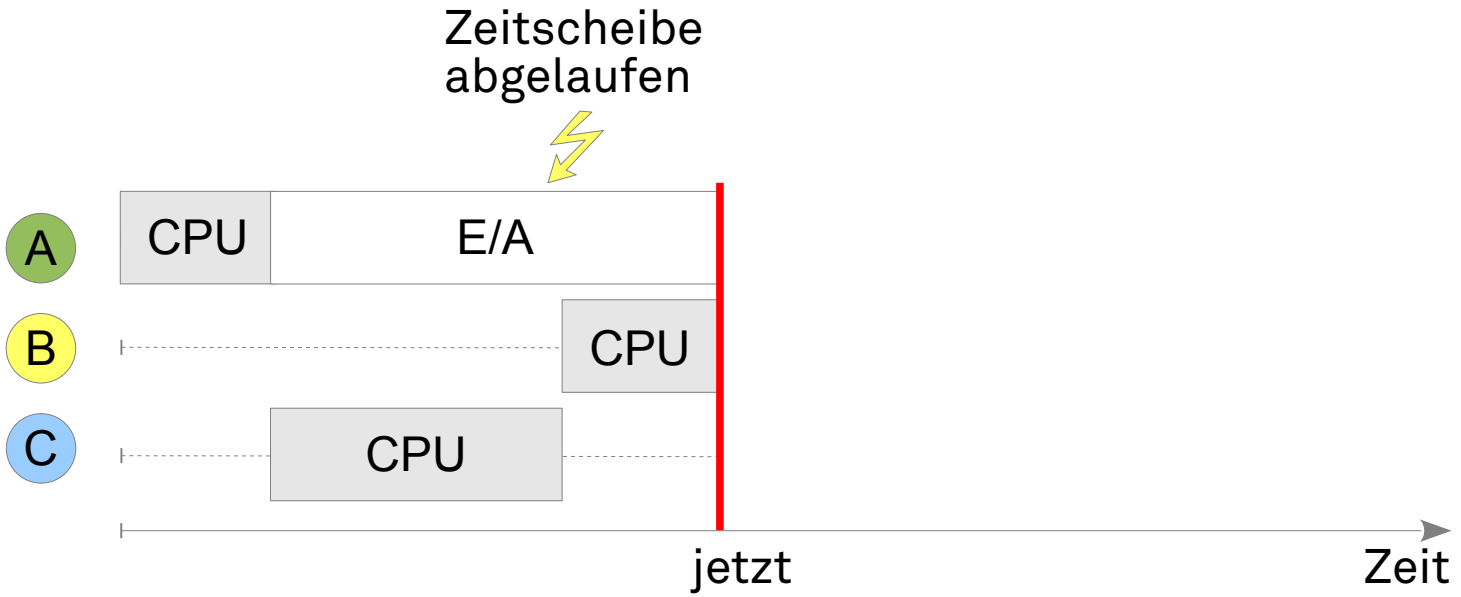
Prozess A hat einen **E/A-Vorgang** gestartet und ist in den Zustand BLOCKED übergegangen. Da A die CPU nun nicht benötigt, hat das Betriebssystem den **Prozess C** ausgewählt und von READY in RUNNING überführt. Es fand ein **Kontextwechsel** von A zu C statt.



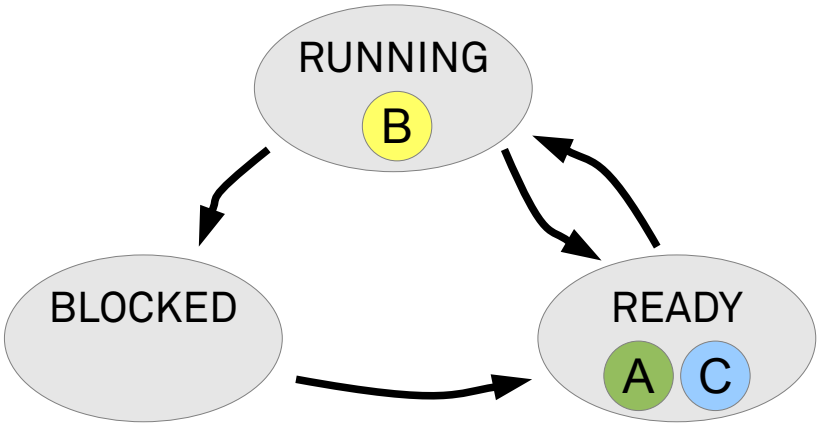
Prozessverhalten und -zustände (3)



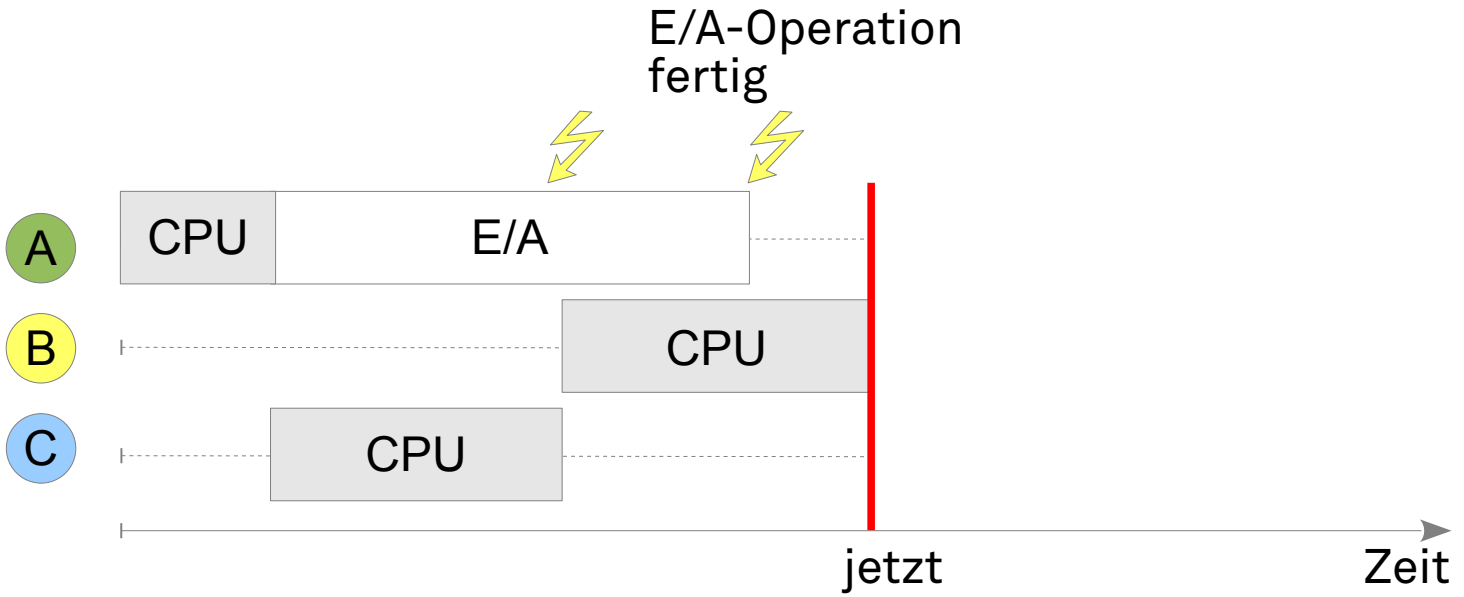
C hat die CPU zu lange besessen, wurde **verdrängt** und ist daher nun wieder im Zustand READY. Damit kann jetzt endlich auch **Prozess B** bearbeitet werden und wird RUNNING.



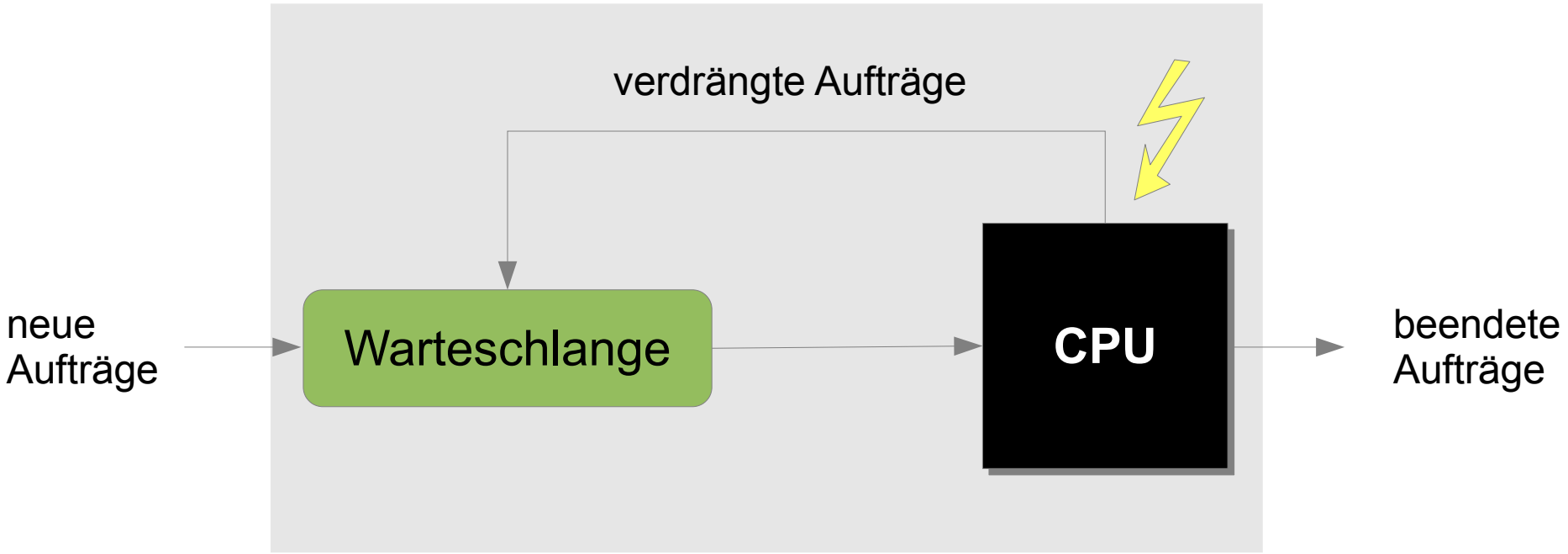
Prozessverhalten und -zustände (3)



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

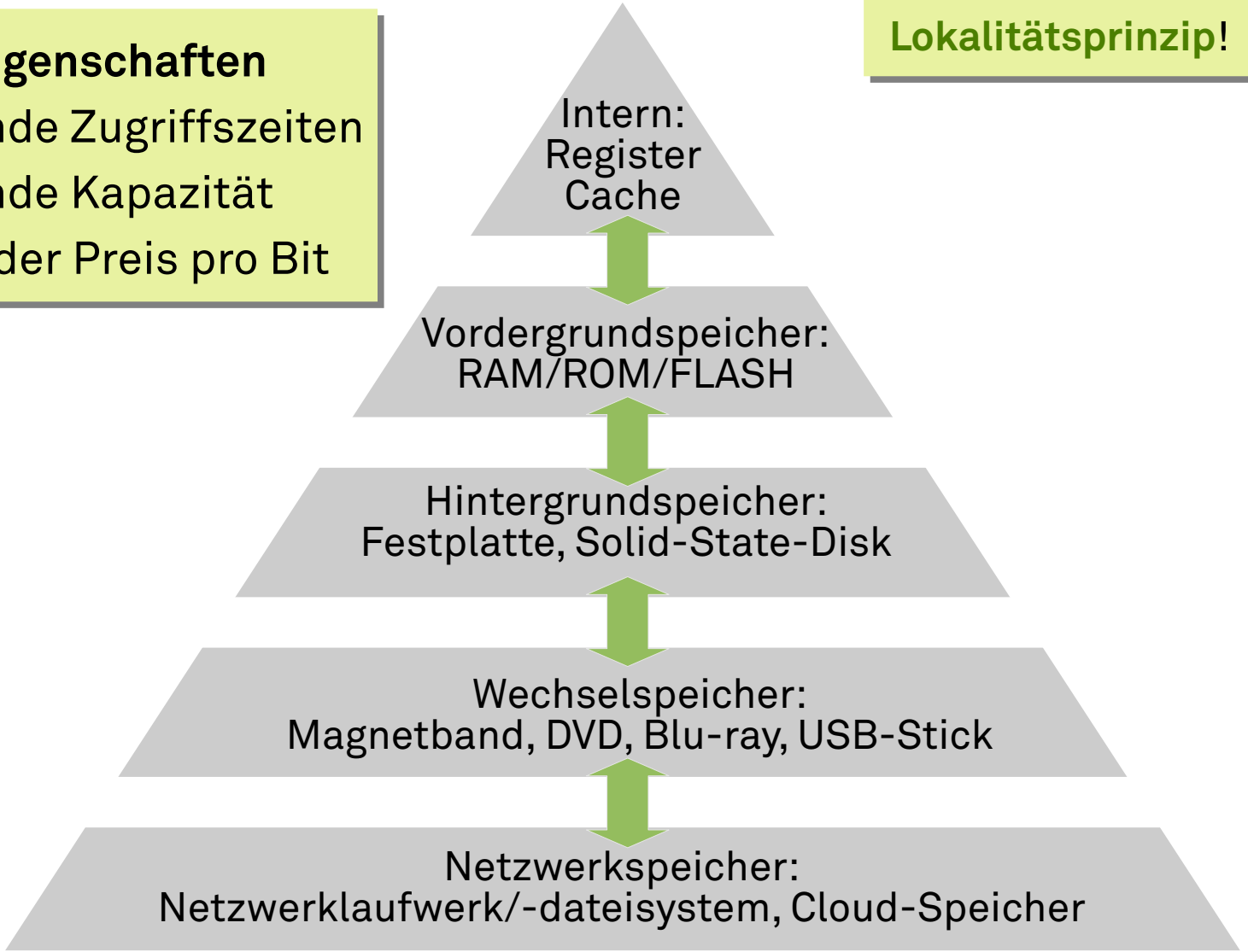
- Ein Blick in die Geschichte
  - Serielle Verarbeitung und Stapelbetrieb
  - Mehrprogramm- und Dialogbetrieb
- Systemabstraktionen im Überblick
  - Prozesse
    - CPU-Zuteilung
    - Synchronisation und Verklemmungen
    - Interprozesskommunikation
  - **Speicherverwaltung**
    - Arbeitsspeicher
    - Hintergrundspeicher

Die Speicherhierarchie

Speichereigenschaften

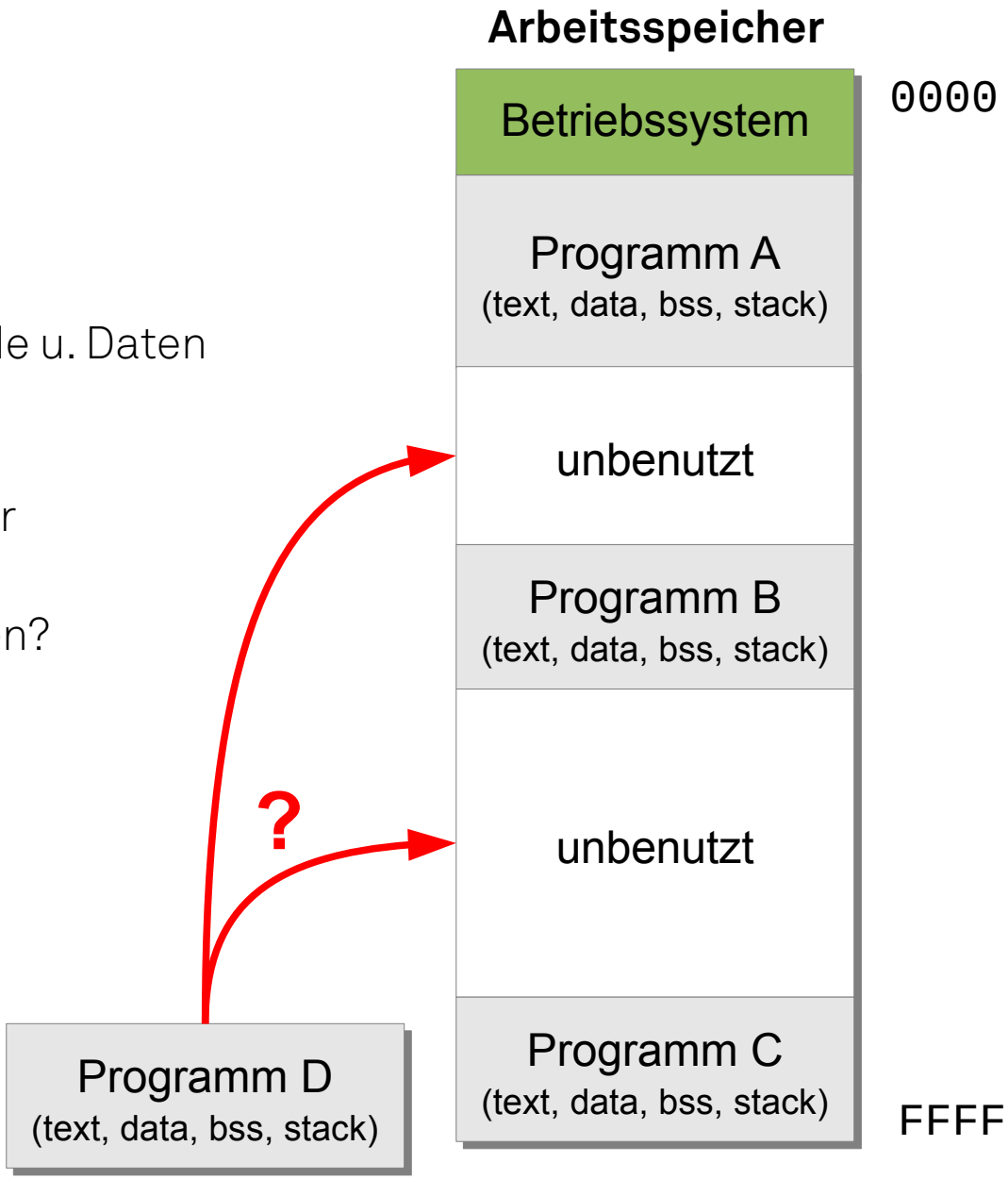
- steigende Zugriffszeiten
- steigende Kapazität
- sinkender Preis pro Bit

Sehr wirtschaftlich durch das **Lokalitätsprinzip**!

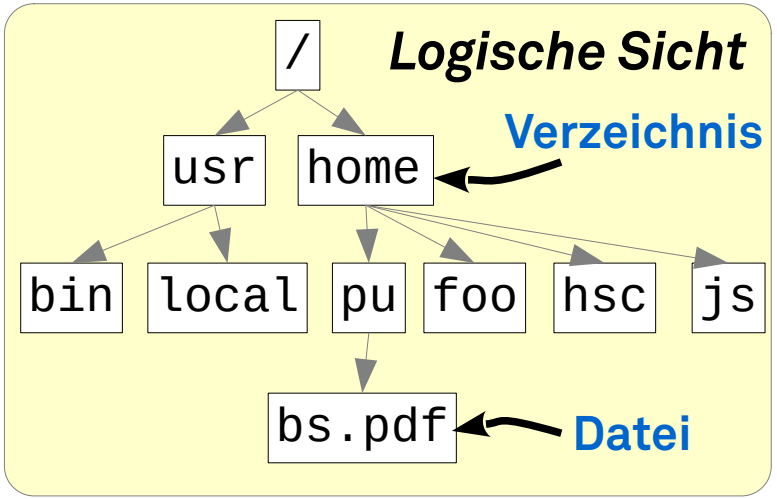


Speicherverwaltung

- **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?



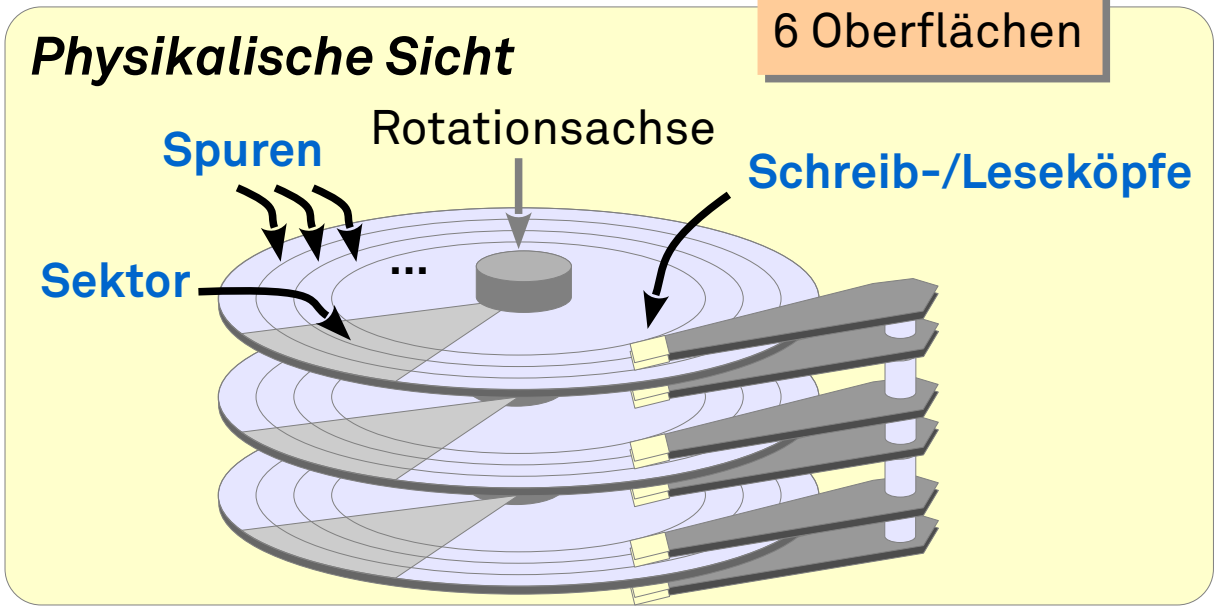
Hintergrundspeicher



Dateisysteme erlauben die dauerhafte Speicherung großer Datenmengen.

Abbildung

Festplatte mit 6 Oberflächen



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

Fazit: Das Betriebssystem …

- verwaltet Betriebsmittel, insbesondere CPU und Speicher
- stellt Abstraktionen zur Verfügung, z.B. …
  - Prozesskonzept
  - Dateien und Verzeichnisse
- 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.