Technische Informatik

7 –Prozesse und Threads

© Lothar Thiele

Computer Engineering and Networks Laboratory

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Betriebssystem


Linux


Mac OS


Windows 10


QNX
QNX SOFTWARE SYSTEMS


ANDROID


VxWorks®
WIND RIVER


eCOS

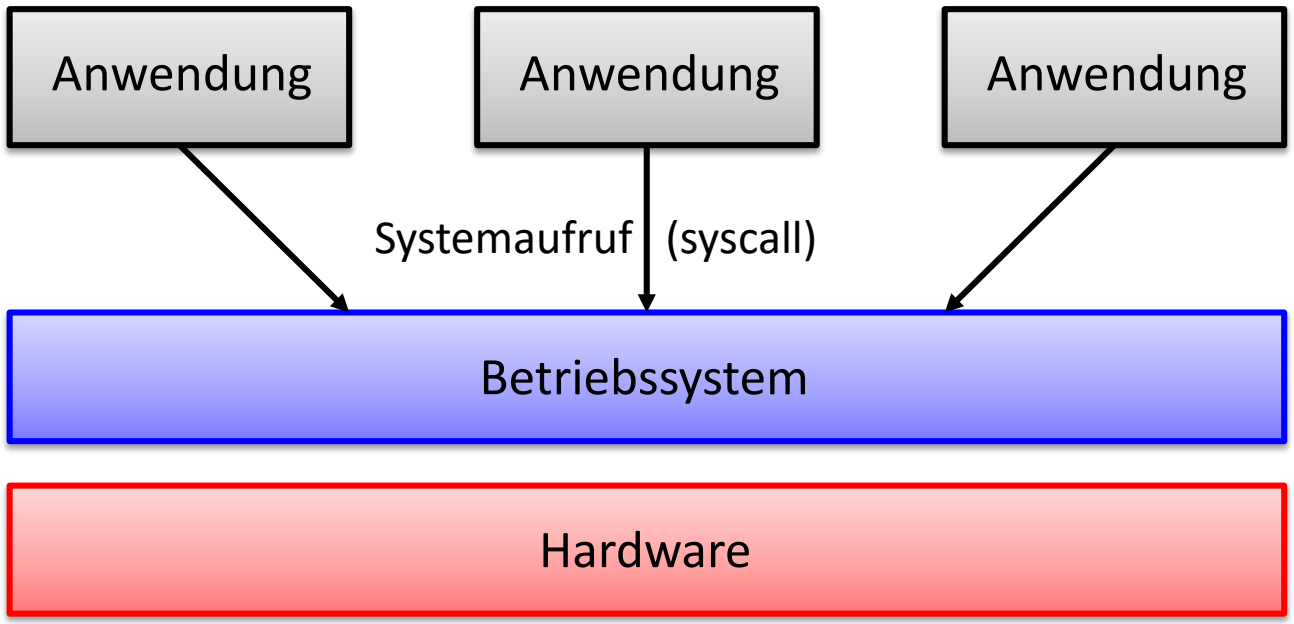

LynxOS
FROM LYNUXWORKS


Windows
Embedded


ITRON


iOS

Betriebssystem



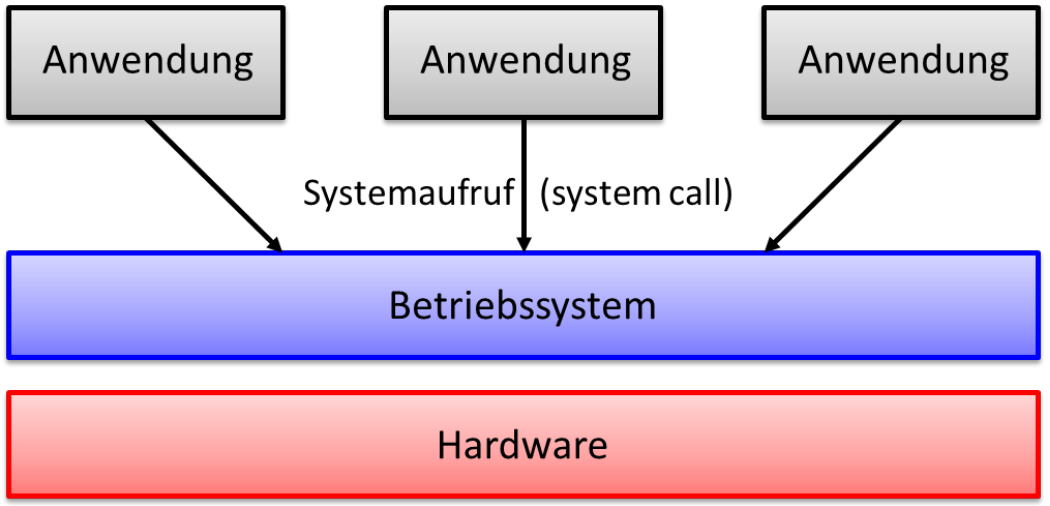
Betriebssystem-Konzepte:

- *Systemaufrufe (system call, syscall)*
- Parallele Ausführung
- *Prozesse und Threads*
- Sicherheit, *Schutzmechanismen*
- *(virtueller) Speicher*
- File System
- Ein/Ausgabe, *Unterbrechungen, DMA*
- Netzwerk, Schnittstellen, Protokolle

Aufgaben eines Betriebssystems - Illusionist

Virtualisierung:

- Betriebssystem erzeugt für die Anwendungen die Illusion eigener, unabhängiger Ressourcen (Prozessor, (virtueller) Speicher, Netzwerk, …).
- Mögliche Nutzung einer Ressource durch mehrere Anwendungen.
- Multiplexen: Aufteilung einer Ressource.
  Aggregation: Zusammenfügen mehrerer Ressourcen zu einer neuen Ressource.



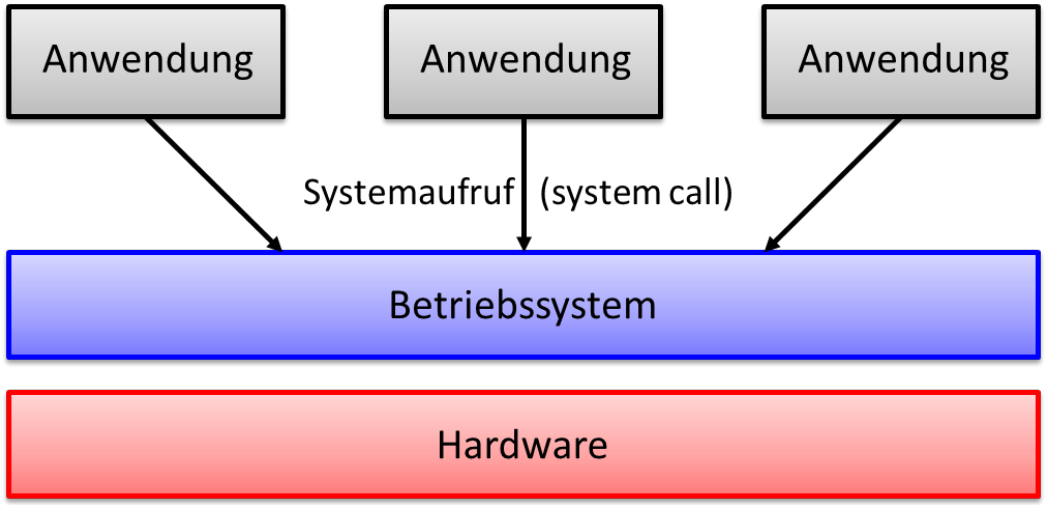
Beispiele:

- virtueller Speicher
- virtuelle Maschine
- Files und Filesystem
- Prozesse und Threads

Aufgaben eines Betriebssystems - Schiedsrichter

Anwendungen sollten sich nicht gegenseitig stören:

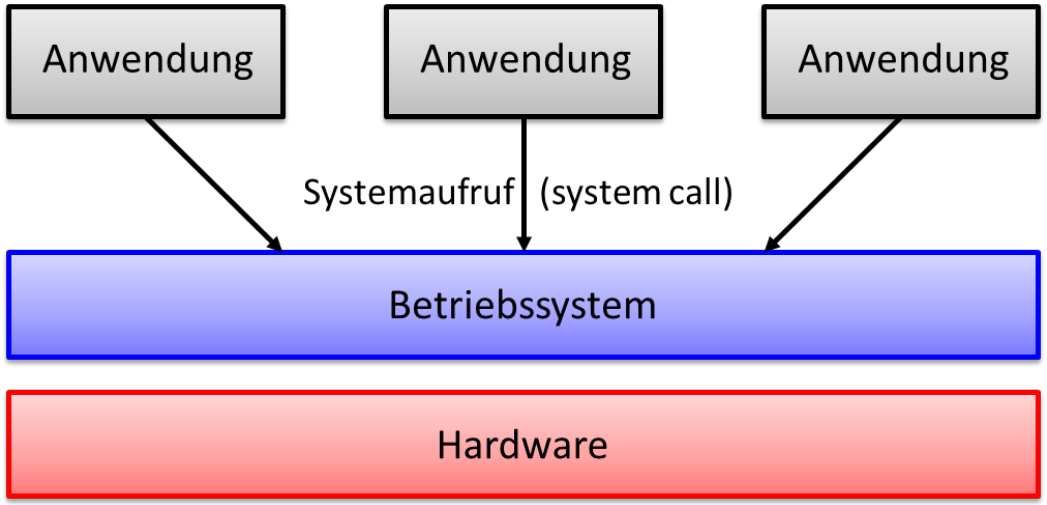
- Hardware muss zwischen den Anwendungen aufgeteilt werden (Prozessor, Speicher, Ein/Ausgabeeinheiten): Isolieren der Anwendungen voneinander.
- *Schutzmechanismen:*
  - Eine Anwendung kann die Daten einer anderen nicht schreiben und/oder lesen.
  - Eine Anwendung darf einer anderen Anwendung die Ressourcen nicht entziehen.



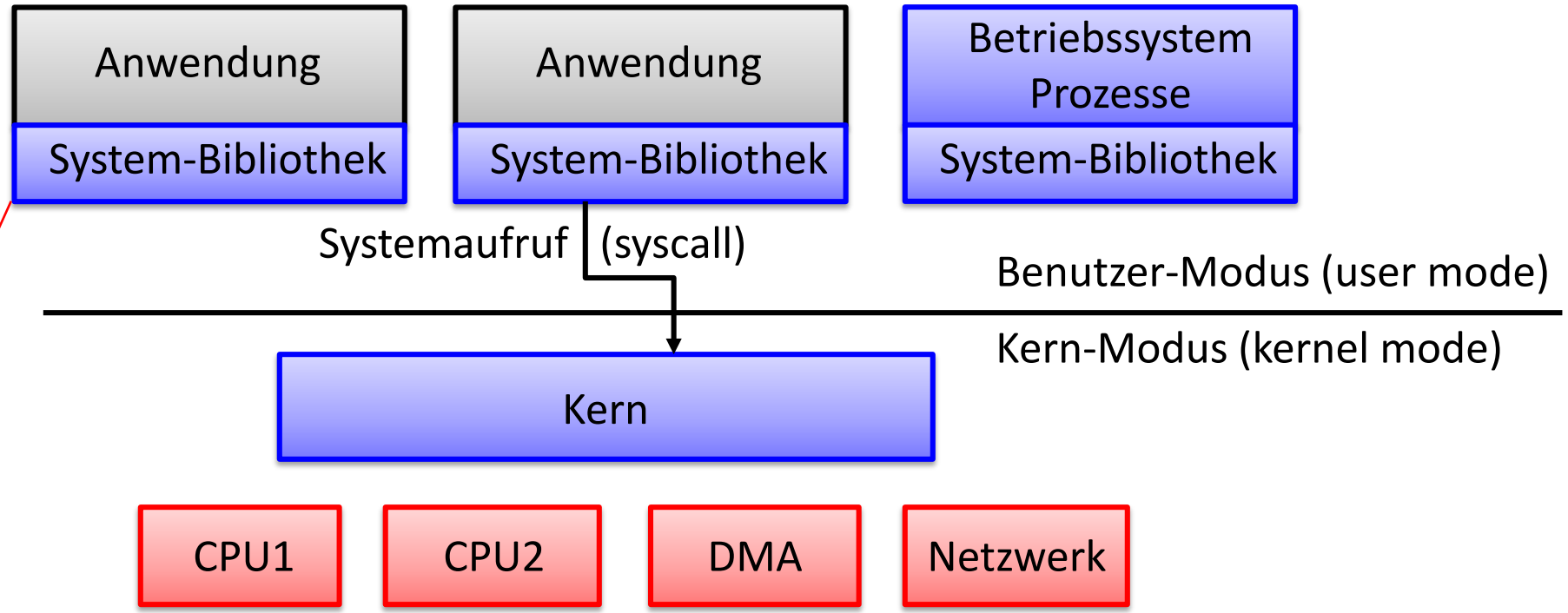
Aufgaben eines Betriebssystems - Assistent

Bereitstellen einer abstrakten Schnittstelle:

- Gemeinsame Funktionalität für alle Anwendungen (Prozesse, Threads, Ein/Ausgabe, Uhren, sekundärer Speicher, Netzwerk, Tastatur,…).
- Unabhängigkeit von der Hardwareplattform.
- Keine Notwendigkeit für eine hardwarenahe Programmierung (Assembler).



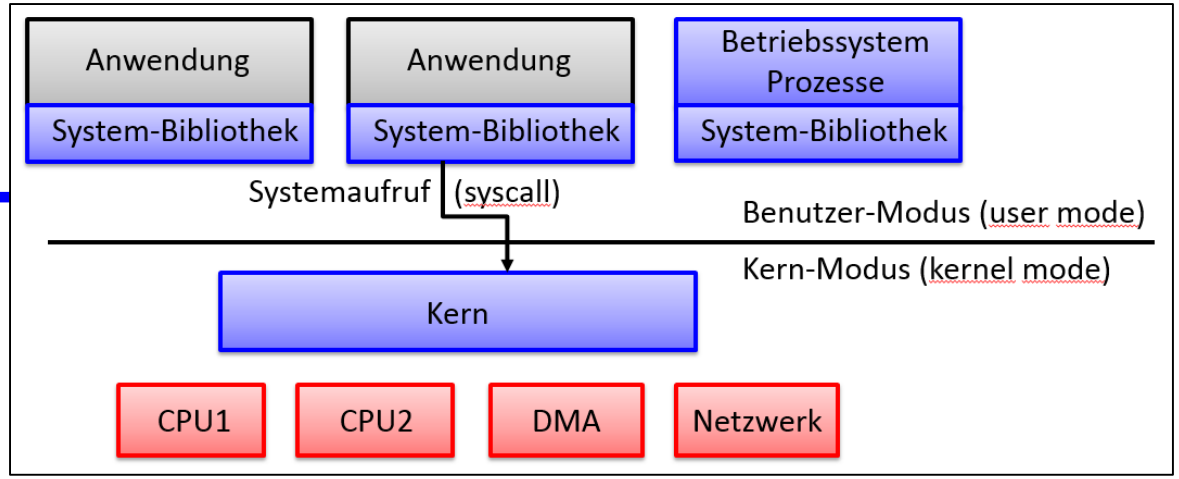
Allgemeine Struktur



Assistenz-Funktionen:

- z.B. printf(); strcmp(); …
- z.B. zum Ausführen von Systemaufrufen (syscalls) aus einer Anwendung: mkdir, mount

Allgemeine Struktur

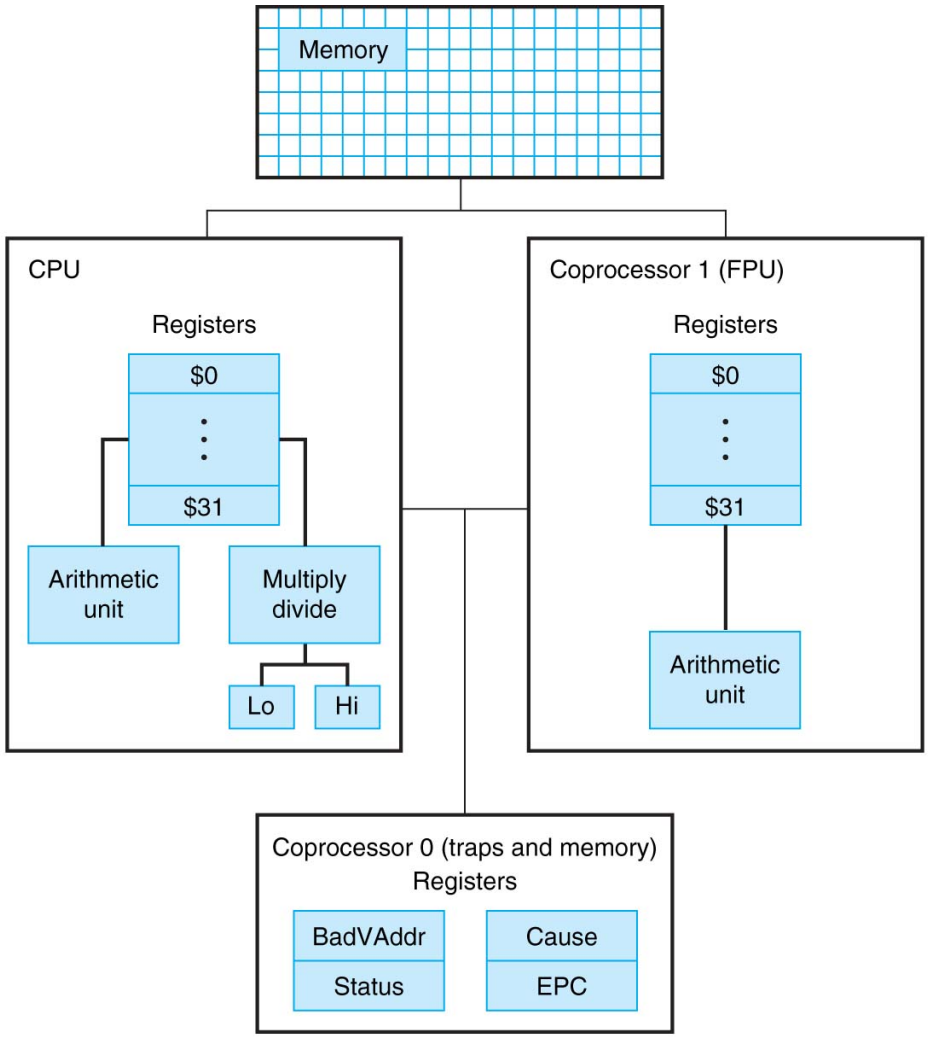


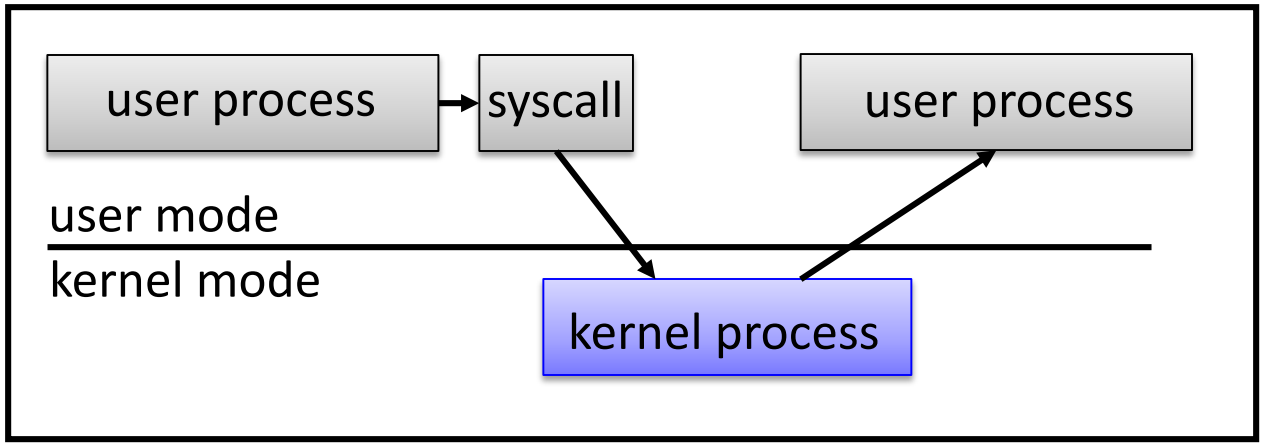
Kernel:

- *Teil des Betriebssystems*.
- *Ausführung* beim Einschalten, Systemaufruf, Unterbrechungen, Ausnahme (exception).
- *Zugriff zu privilegierten Instruktionen* und Registern (z.B. Zugriff auf speziellen privilegierten Adressbereich, Instruktionen für den TLB oder den Cache).
- Programmcode und Daten von Programmen im «kernel mode» liegen in einem speziellen, vom «user mode» getrennten und geschützten Adressbereich.
- *Aufgaben:* Management von Ressourcen, z.B. (virtueller) Speicher und Ein-Ausgabeeinheiten, Verwaltung von Prozessen und Threads, Bereitstellung von Schutzmechanismen, Synchronisation und Kommunikation zwischen Prozessen.

Allgemeine Struktur

Beispiel MIPS:





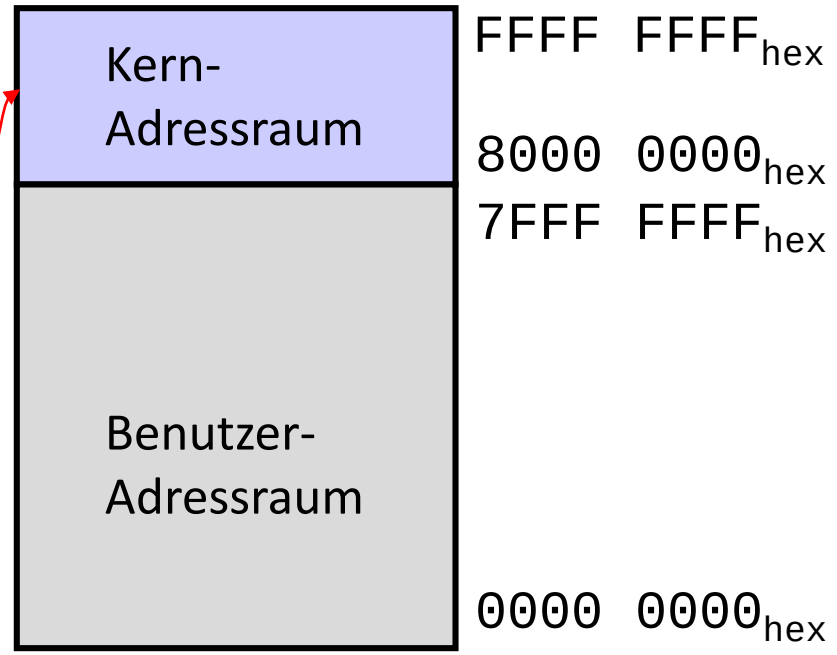
- Der Wechsel zwischen Benutzer-Modus und Kern-Modus wird unter Zuhilfenahme von Coprocessor 0 durchgeführt.
- Coprozessor 0 kann nur im Kern-Modus benutzt werden (privilegierte Instruktionen).
- Das Bit ‘1’ des Status-Registers gibt an, ob das Programm im Kern-Modus oder Benutzer-Modus arbeitet (Status[1] == 0 : Kern-Modus).

Allgemeine Struktur

Beispiel TLB Miss Ausnahme MIPS:

- Sprung zur Adresse 8000 0000$_{hex}$
- Dort steht beispielsweise folgendes Programm:

```
TLBmiss:
  mfc0 $k1, Context # copy address of PTE into $k1
  lw $k1, 0($k1)    # put PTE into $k1
  mtc0 $k1, EntryLo # put PTE into special register EntryLo
  tlbwr             # put EntryLo into TLB entry at Random
  eret              # return from TLB miss exception
```



PTE: page table entry

physikalischer Speicher

Basis-Adresse der Seitentabelle und virtuelle Seitennummer der fehlenden Seite

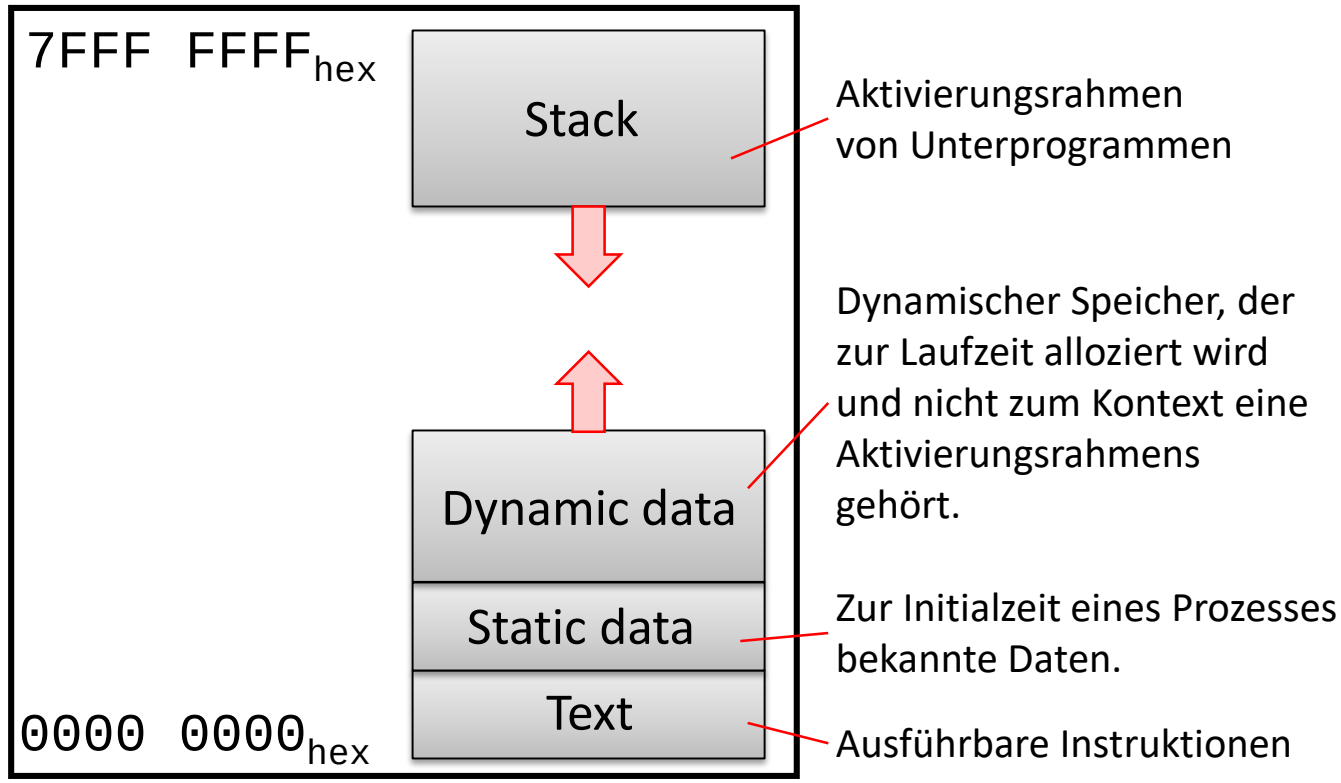
Prozesse

Prozess (Wiederholung)

«Prozess»: Die Instanz eines Programms (oder eines Teils davon), die vom Prozessor zu einem gewissen Zeitpunkt mit konkreten Eingabedaten ausgeführt wird. Ein Prozess besteht aus dem zugehörigen Programm-Code sowie dem derzeitigen Zustand.

Eigenschaften:

- Ein Prozess hat seinen eigenen privaten Adressraum.
- Zu seinem Zustand gehören zum Beispiel die Prozessor-Register, der Programmzähler und Daten.



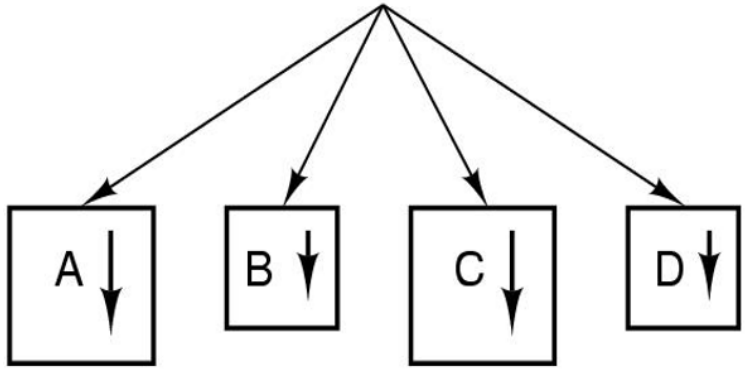
typischer Adressraum eines Prozesses

Multitasking

Multitasking bezeichnet die Fähigkeit, mehrere Aufgaben auf einer Rechenressource (scheinbar) gleichzeitig auszuführen ((quasi-)parallel).

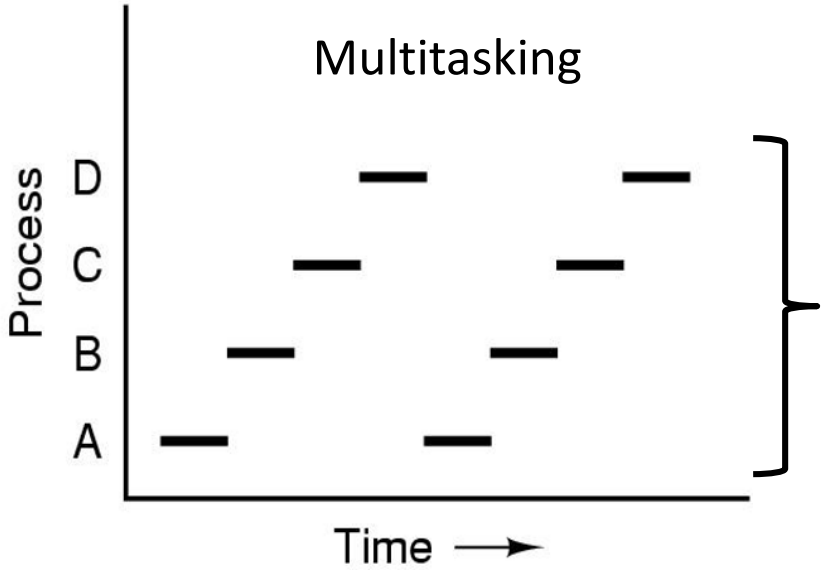
- Dabei werden die verschiedenen Prozesse in so kurzen Abständen immer abwechselnd aktiviert, dass der Eindruck der Gleichzeitigkeit entsteht.

Vier unabhängige Programmzähler



Vier Prozesse: A, B, C, D

Multitasking



Prozess-Scheduling

Prozessverwaltung

Eine Prozessverwaltung enthält die folgenden Hauptkomponenten:

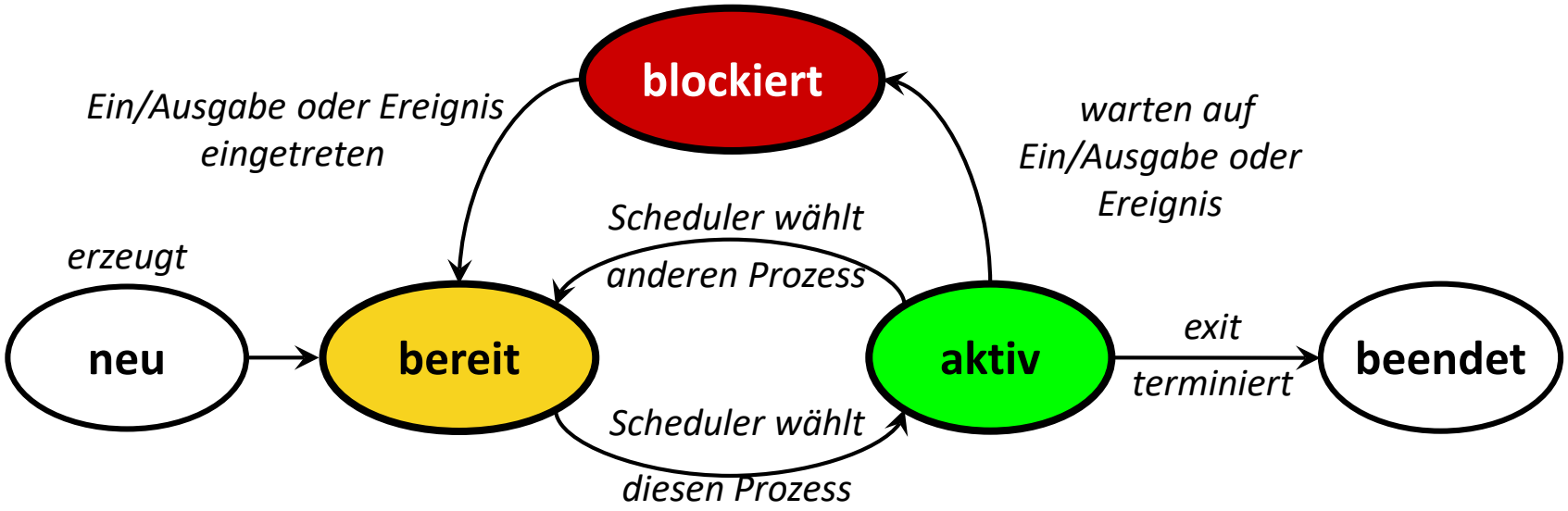
- Speicherverwaltung, z.B. virtueller Speicher mittels Seitentabellen.
- Verarbeitung der Prozesszustände.
- «Process Control Block (PCB)» zur Beschreibung des Prozesskontextes.
- «Scheduler» zur Planung der Prozesswechsel.

Das Betriebssystem setzt diese Komponenten ein um

- neue Prozesse zu generieren,
- Prozesse auszuführen,
- ihnen Betriebsmittel zukommen zu lassen (z.B. Ein- und Ausgabe),
- sie zu synchronisieren und Datenaustausch zwischen Prozessen zu ermöglichen,
- und sie zu beenden.

Prozesszustände

Jedem Prozess ist ein eigener Zustand zugeordnet:

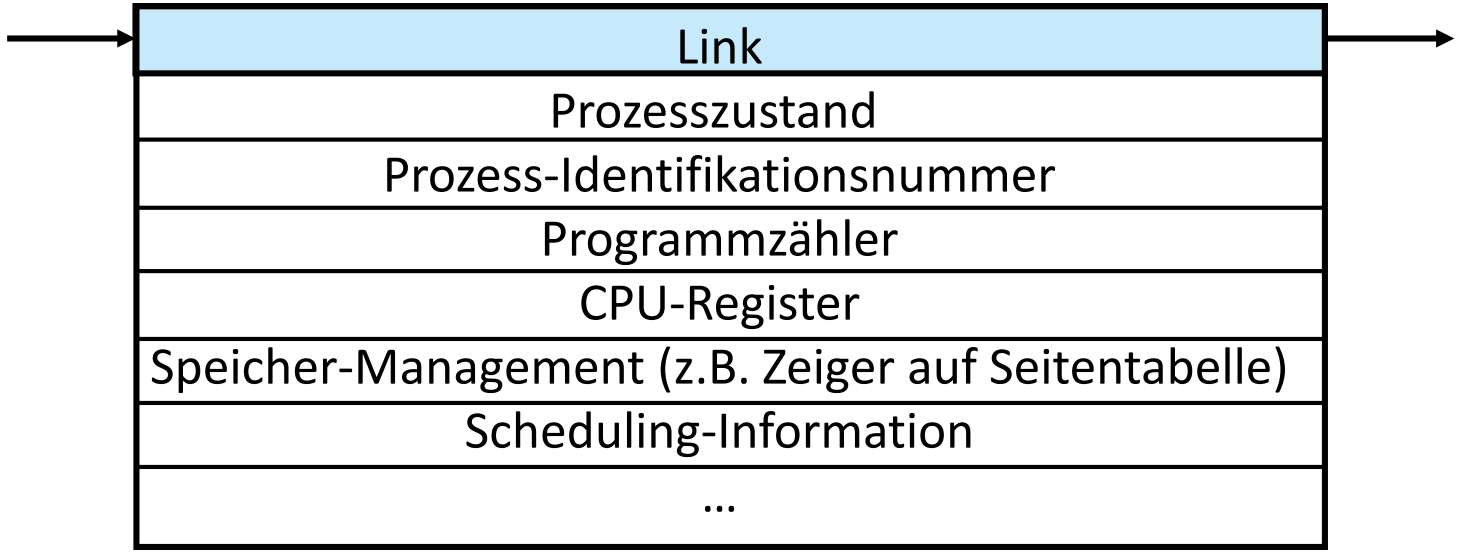


Prozesszustände

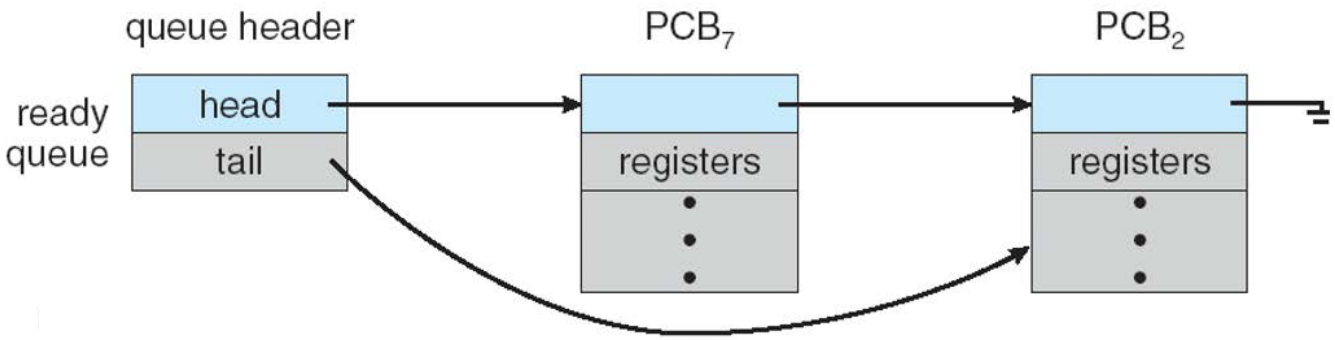
Zustand	Englisch	Bedeutung
neu	new	Die für einen Prozess notwendigen Datenstrukturen wurden erstellt und im Betriebssystemkern vermerkt.
bereit / lauffähig	ready / runnable	Der Prozess ist bereit zur Ausführung.
wartend / blockiert	waiting / blocked	Der Prozess wartet auf den Eintritt eines Ereignisses, z.B. den Abschluss einer E/A-Operation oder auf die Synchronisierung mit einem anderen Prozess.
aktiv / laufend	running	Der Prozess wird durch eine der verfügbaren CPUs ausgeführt.
beendet	terminated	Der Prozess wurde beendet. Alle wichtigen Betriebsmittel (Speicher, CPU, Ein/Ausgabe-Einheiten, …) sind freigegeben.

«Process Control Block (PCB)»

Als «Process Control Block» bezeichnet man eine Datenstruktur im Betriebssystem, die den derzeitigen Prozesskontext beschreibt:



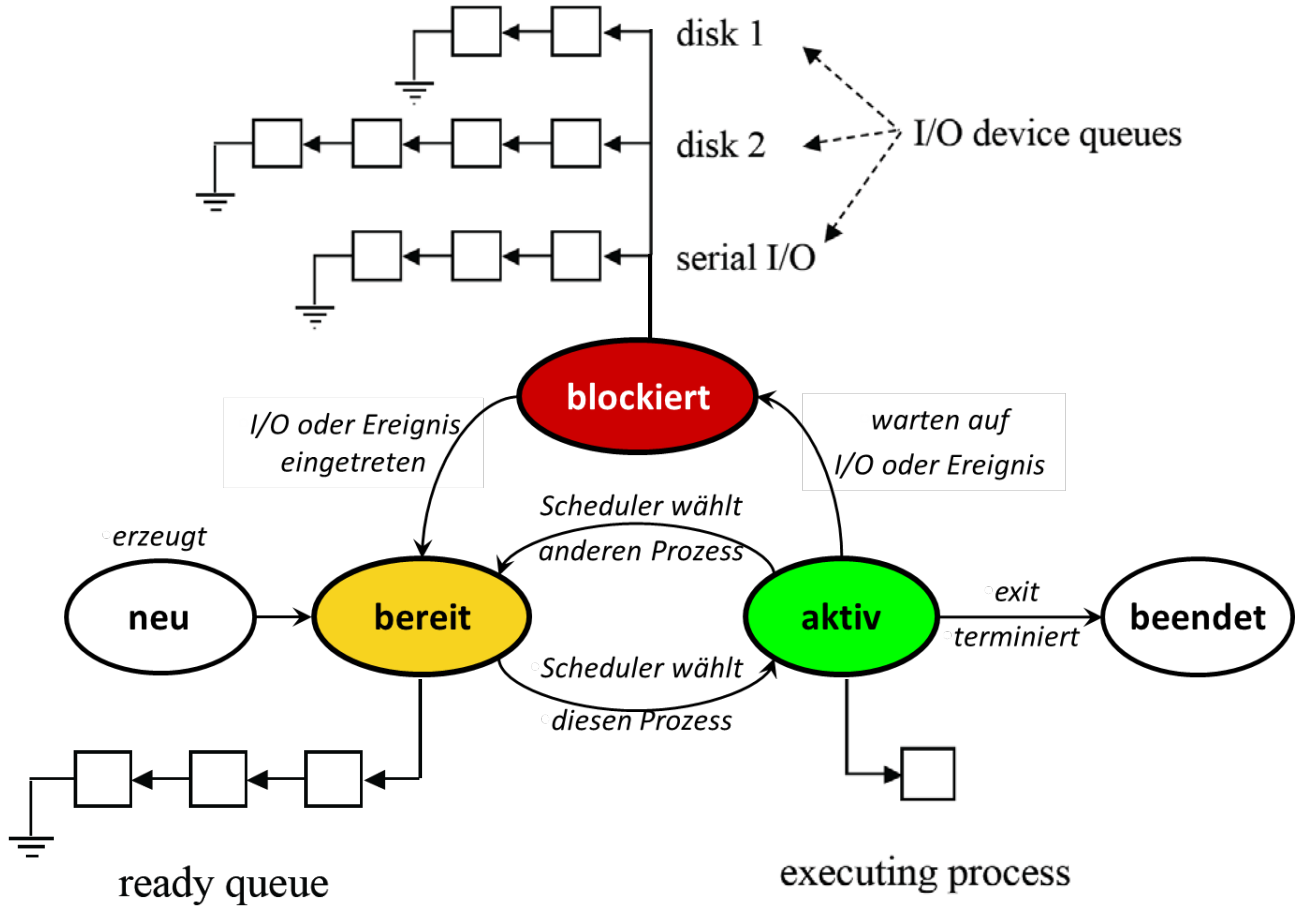
Process Control Block (PCB)



Liste von PCBs

«Process Control Block (PCB)»

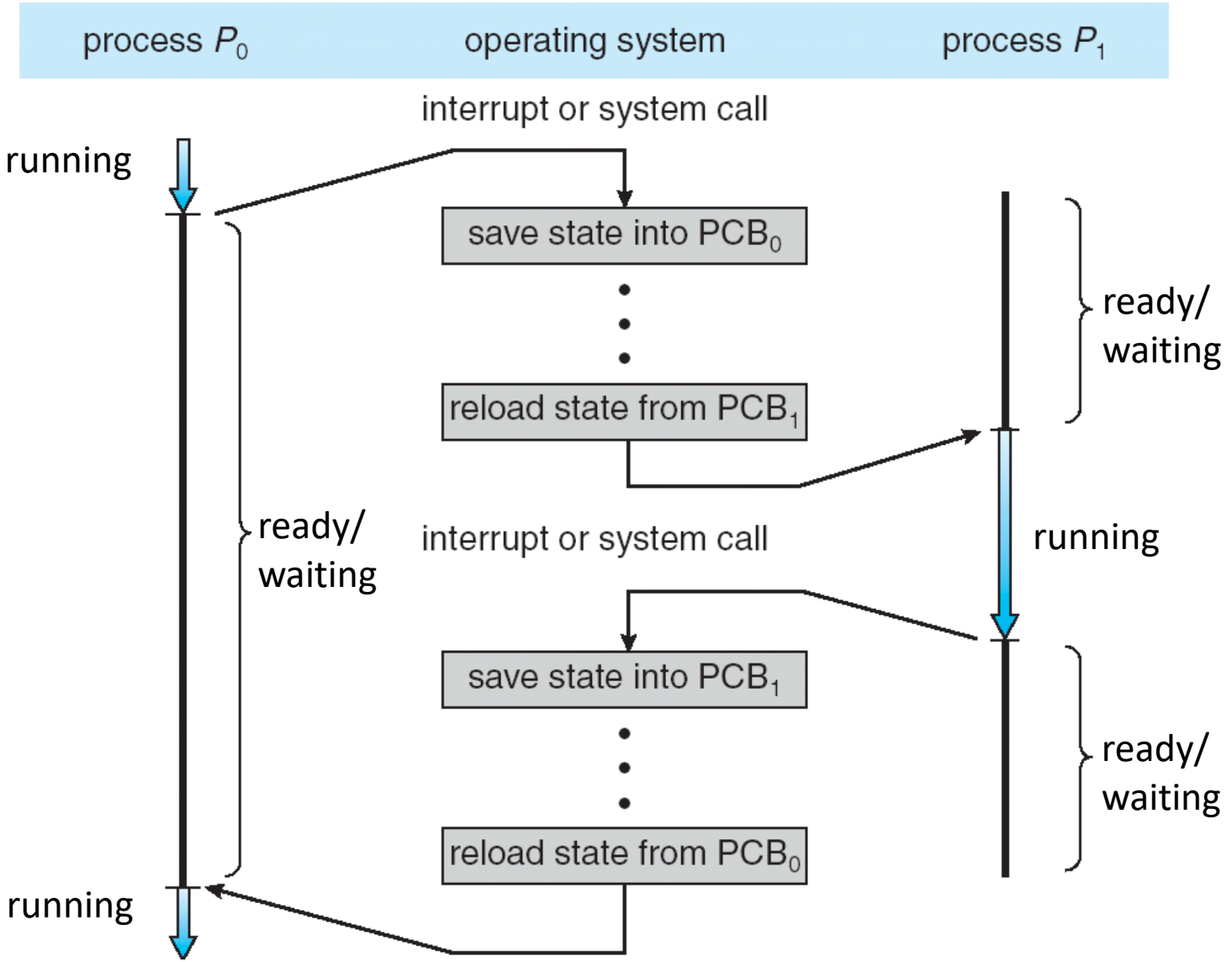
Die Kontexte aller Prozesse (PCBs) werden üblicherweise in verketteten Listen verwaltet. Alle Prozesse in einer solchen Liste haben den gleichen Prozesszustand.



Prozesswechsel

Übergang zwischen Prozesszuständen.

Speziell zu oder vom Zustand aktiv (running):



Schutzmechanismen

Schutzmechanismen

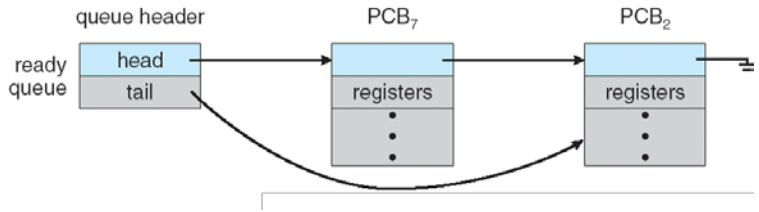
Aufgaben:

- Man sollte verbieten können, dass Prozesse die Speicherbereiche anderer Prozesse (oder gar des Betriebssystems) lesen oder sogar schreiben können.
- Benutzerprozesse dürfen keine Seitentabellen verändern.
- Benutzerprozesse sollten nur eingeschränkten Zugang zu Systemressourcen erhalten, zum Beispiel zu Ein/Ausgabe-Einheiten.

Mechanismen:

- Zusätzliche Informationen über Lese- und Schreibberechtigungen in der Seitentabelle.
- Unterstützung unterschiedlicher Prozessmodi („user“ und „kernel“): Einige Operationen und Zugriffe können nur im kernel mode ausgeführt werden, z.B. TLB-Veränderung, Änderungen der Seitentabellen oder Änderung von Zeigern auf Seitentabellen.
- Übergang der CPU zwischen den verschiedenen Modi.

Übersicht



Bearbeitung durch Kern-Prozesse (Betriebssystem).

page table
process 1

page table
process 2

page table
process 3

...

virtuelle Adressen

FFFF FFFF$_{hex}$

Kern Adressraum

8000 0000$_{hex}$

7FFF FFFF$_{hex}$

Benutzer Adressraum

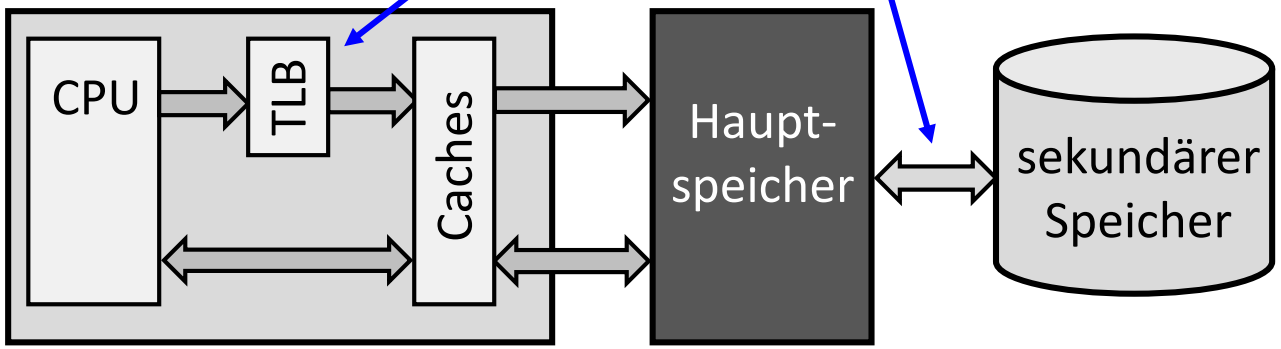
0000 0000$_{hex}$

virtueller Speicher

statische und dynamische Daten; Stack

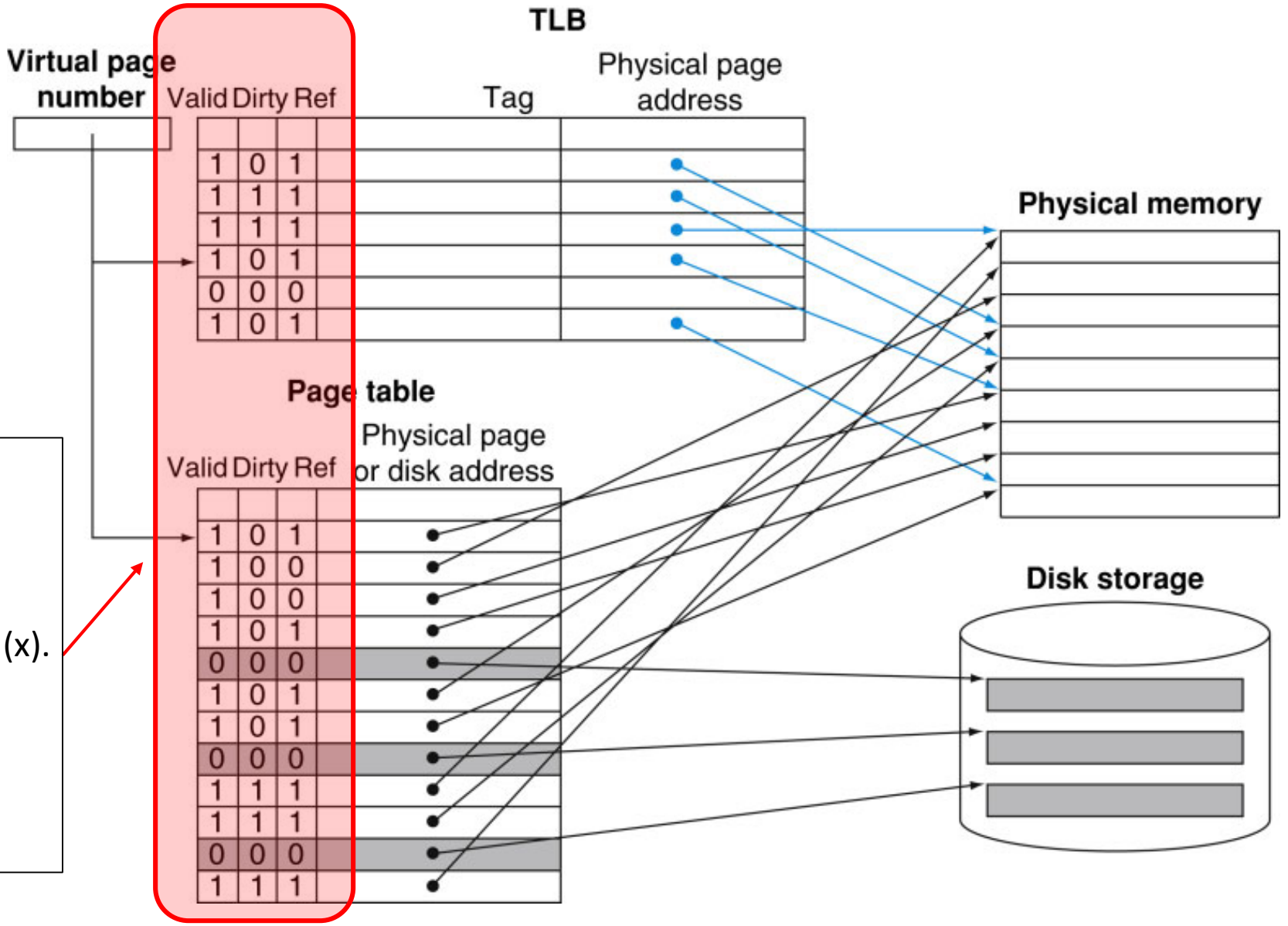
Text (Kern-Prozesse)

Miss- und Seitenfehler-Behandlung durch das Betriebssystem



physikalischer Speicher

TLB und Seitentabelle mit Zugriffsschutz



Erweiterung der Valid-Dirty-Ref Bits um Zugriffsschutz-Bits.
Sie geben die erlaubte Verwendung der Seite an:
Lesen (r), Schreiben (w), Ausführen (x).

Bei Verletzung des Zugriffs wird eine Ausnahme erzeugt, die vom Betriebssystem bearbeitet wird.

Gemeinsam genutzte Seiten («shared pages»)

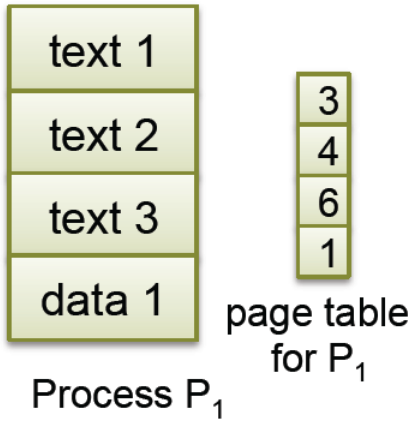
Daten und Programme können mittels Seitentabellen von mehreren Prozessen gemeinsam genutzt werden.

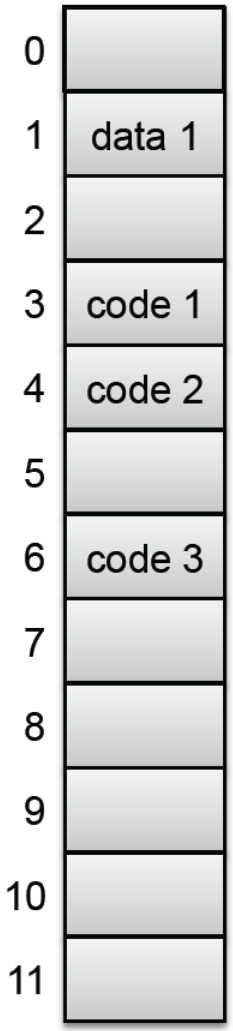
Der Zugriff kann über die Zugriffsschutz-Bits (protection bits) geregelt werden.

Beispiel:

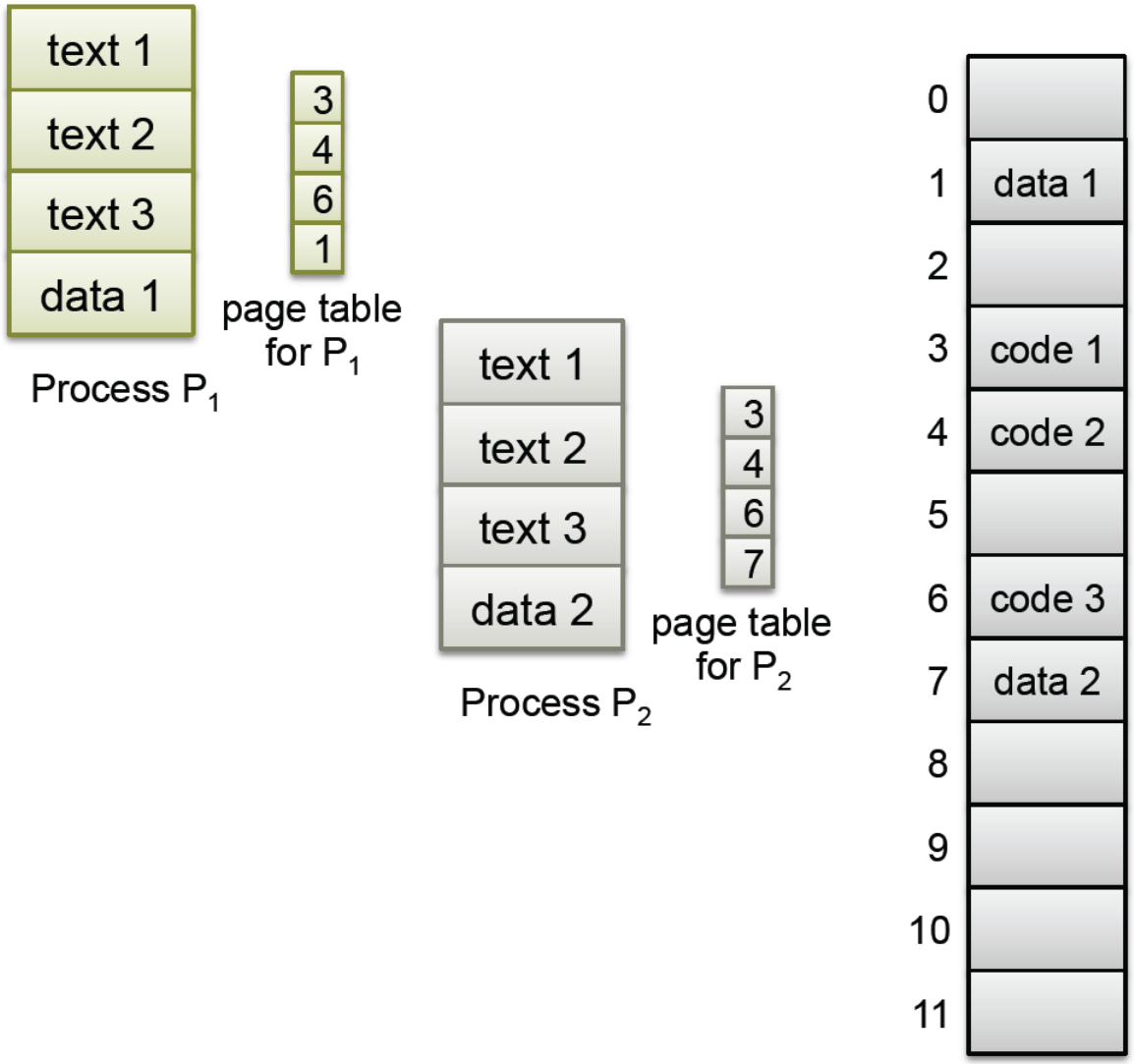
- Eine Kopie des Programms wird von 3 Prozessen gemeinsam genutzt.
- Das Programm erscheint in jedem Prozess an der gleichen virtuellen Adresse.
- Daten werden nicht geteilt, sie sind für jeden Prozess unterschiedlich.
- Daten sind aber an der gleichen virtuellen Adresse; damit kann das jeweilige (gleiche) Programm sie finden.

Beispiel geteilte Seiten

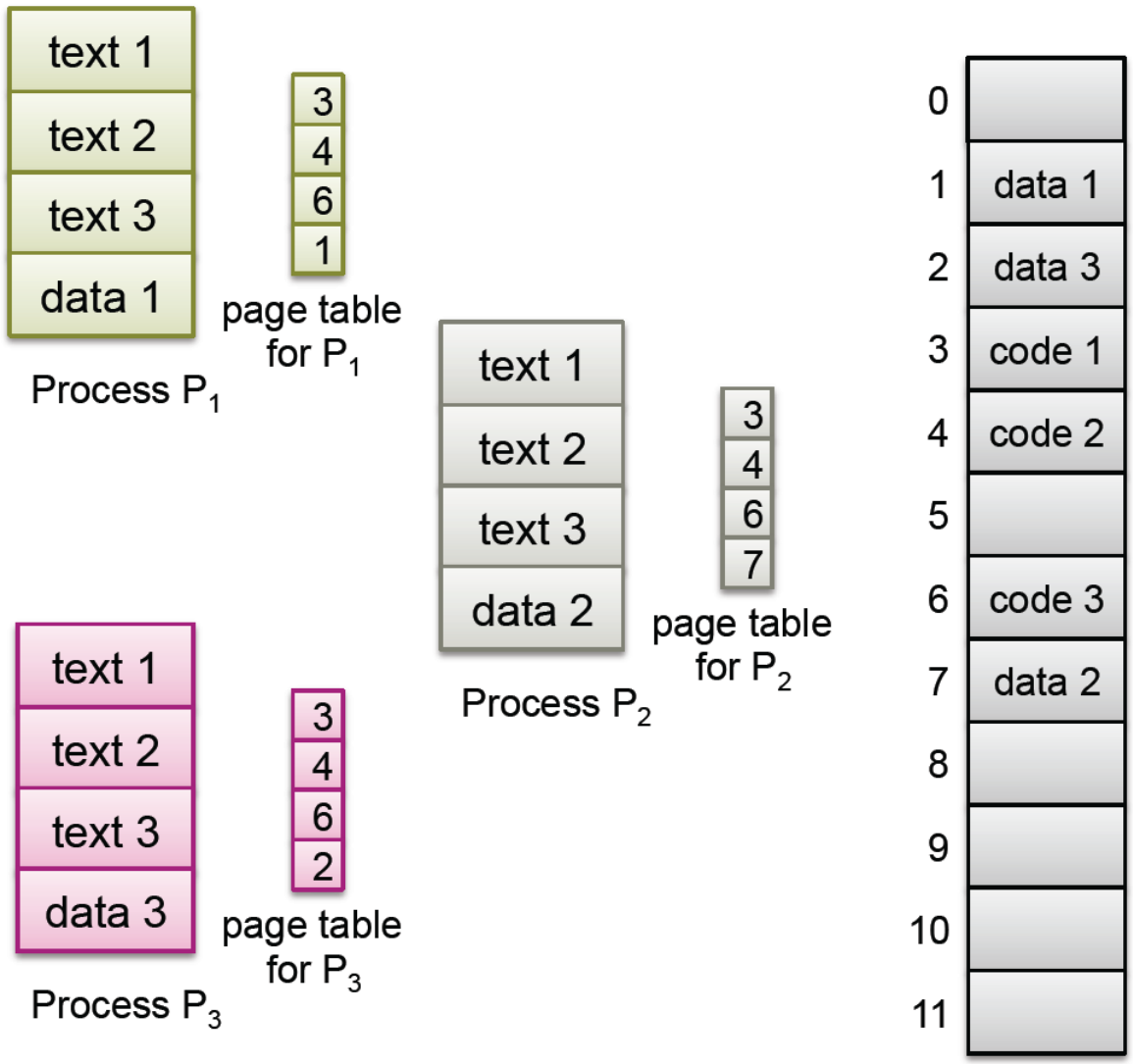




Beispiel geteilte Seiten



Beispiel geteilte Seiten



Threads

Prozesse und Threads

Prozesse:

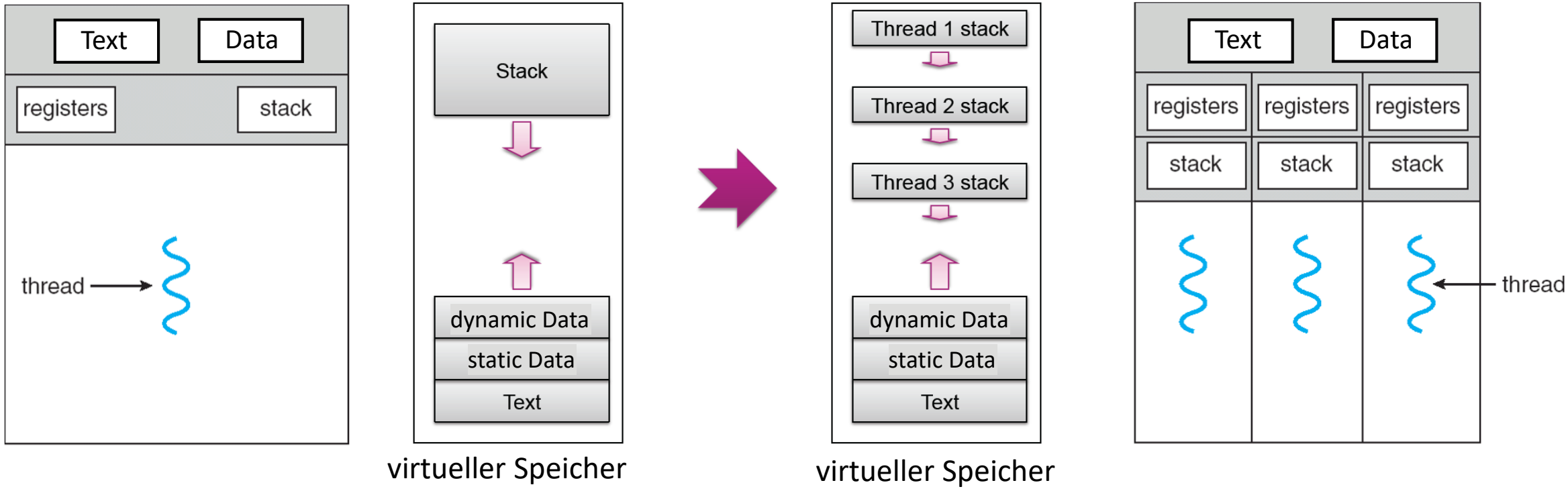
- Prozesse teilen sich die Ressourcen des Computers.
- Prozesse sind im Prinzip unabhängig voneinander (privater Speicherbereich).
- Prozesswechsel durch das Betriebssystem ist aufwändig.

Threads:

- Diese „gehören“ zu einem Prozess und werden im Kontext des Prozesses ausgeführt. Ein Thread nutzt die dem Prozess zugeordneten Ressourcen (z.B. Speicher).
- Jeder Thread führt einen Teil einer Anwendung aus, verfügt über einen eigenen PC, Stack, etc. . Globale Variablen des Prozesses sind für alle Threads sichtbar.

pro Prozess vorhanden	**pro Thread vorhanden**
virtueller Adressraum	Programmzähler, Prozessorregister
globale Variablen	Zustand
Ressourcen, z.B. geöffnete Dateien	Stack

Prozesse mit einem oder mehreren Threads



Prozess mit einem Thread

Prozess mit mehreren Threads

Kernel Space Threads gegenüber User Space Threads

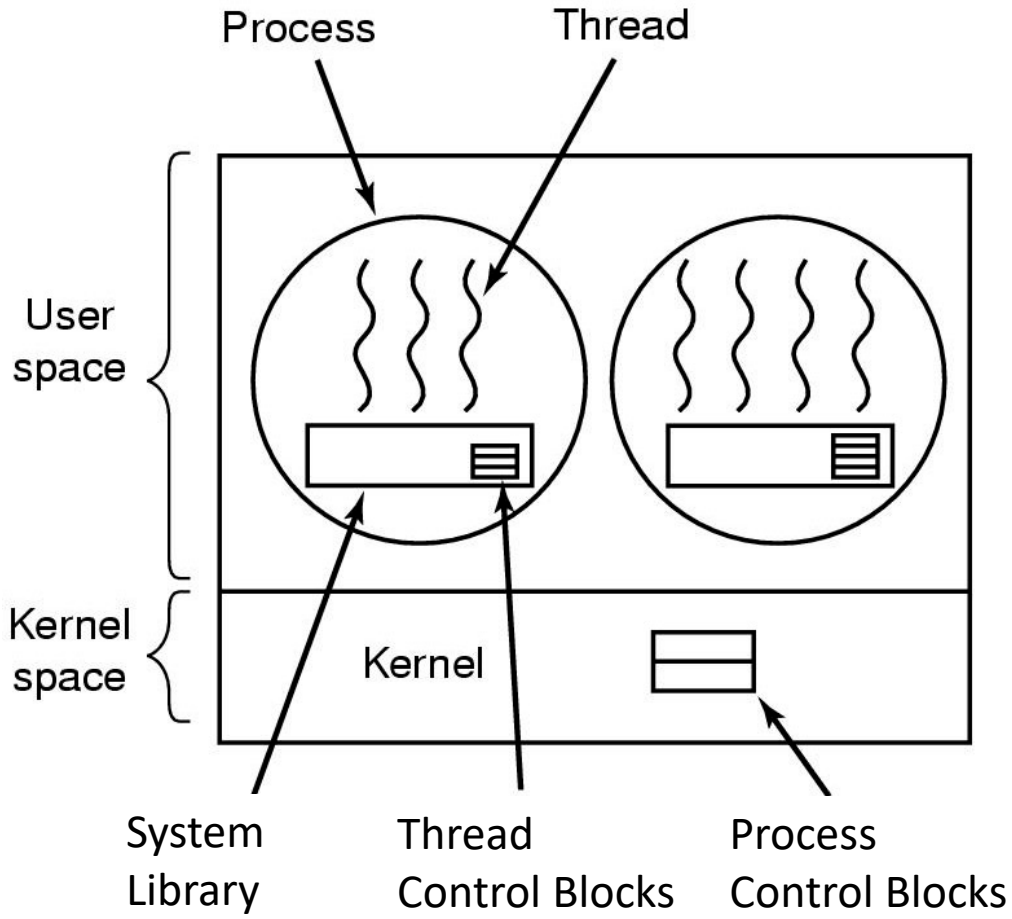
User Space Threads:

- Der Kern des Betriebssystems kennt die Threads nicht. Der Prozess ist verantwortlich für die Verwaltung der «Thread Control Blocks (TCB)», der Speicherbereiche (Stacks) der Threads und für das Thread Scheduling.
- Vorteile: schneller Wechsel zwischen Threads, schnelle Erzeugung von neuen Threads.

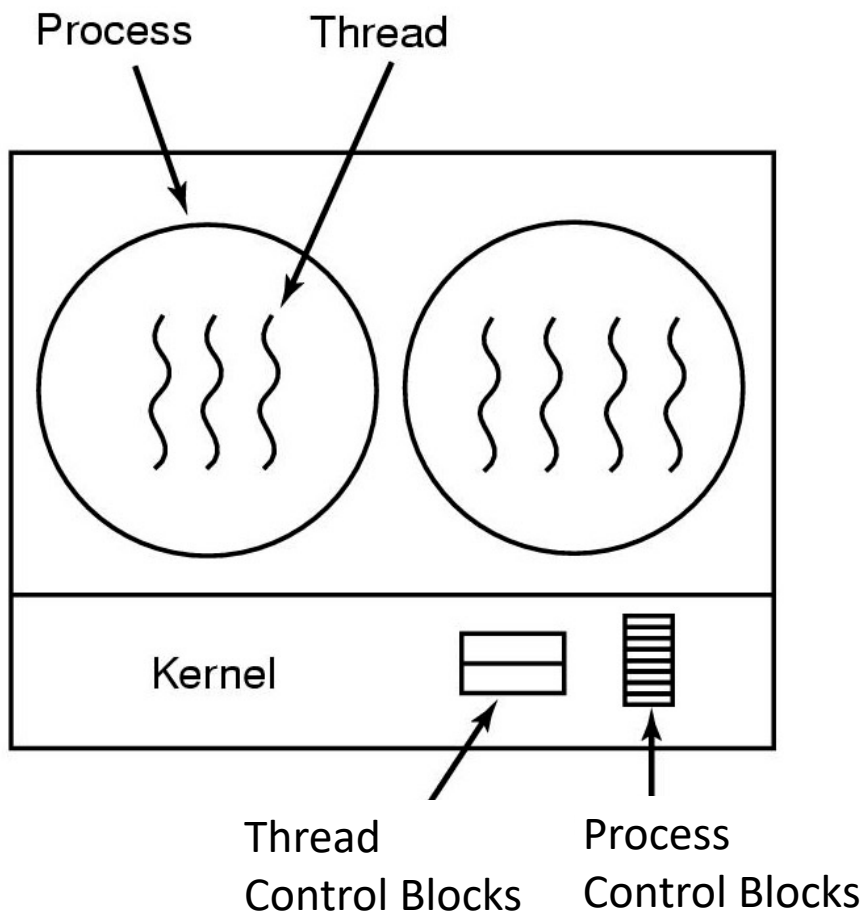
Kernel Space Threads:

- Das gesamte Management der Threads geschieht im Kernel.
- Vorteile: flexibles Scheduling, zentralisierte Speicherverwaltung, sicherere Abarbeitung

Kernel Space Threads gegenüber User Space Threads

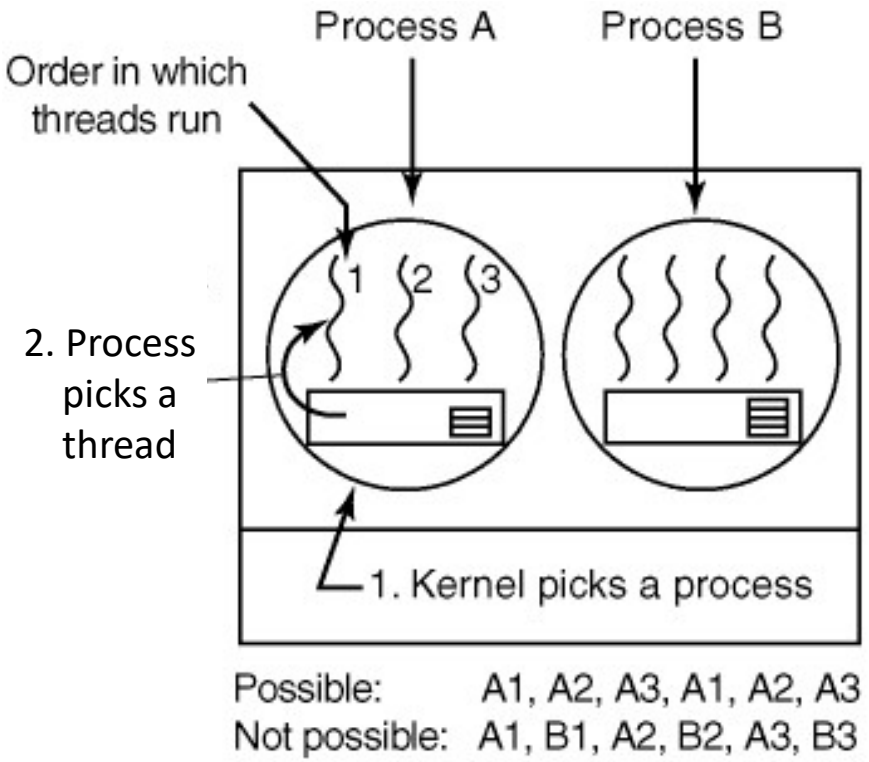


User Space Thread

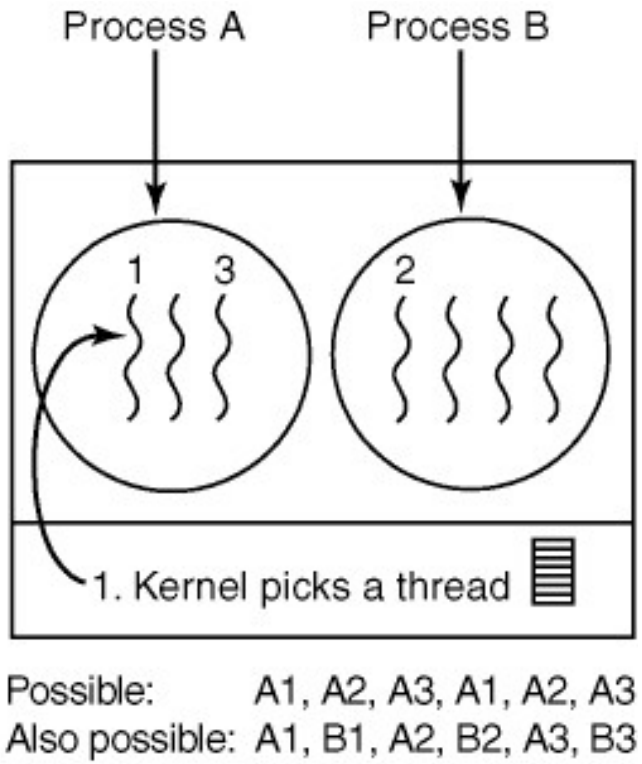


Kernel Space Thread

Scheduling



User Space Thread



Kernel Space Thread

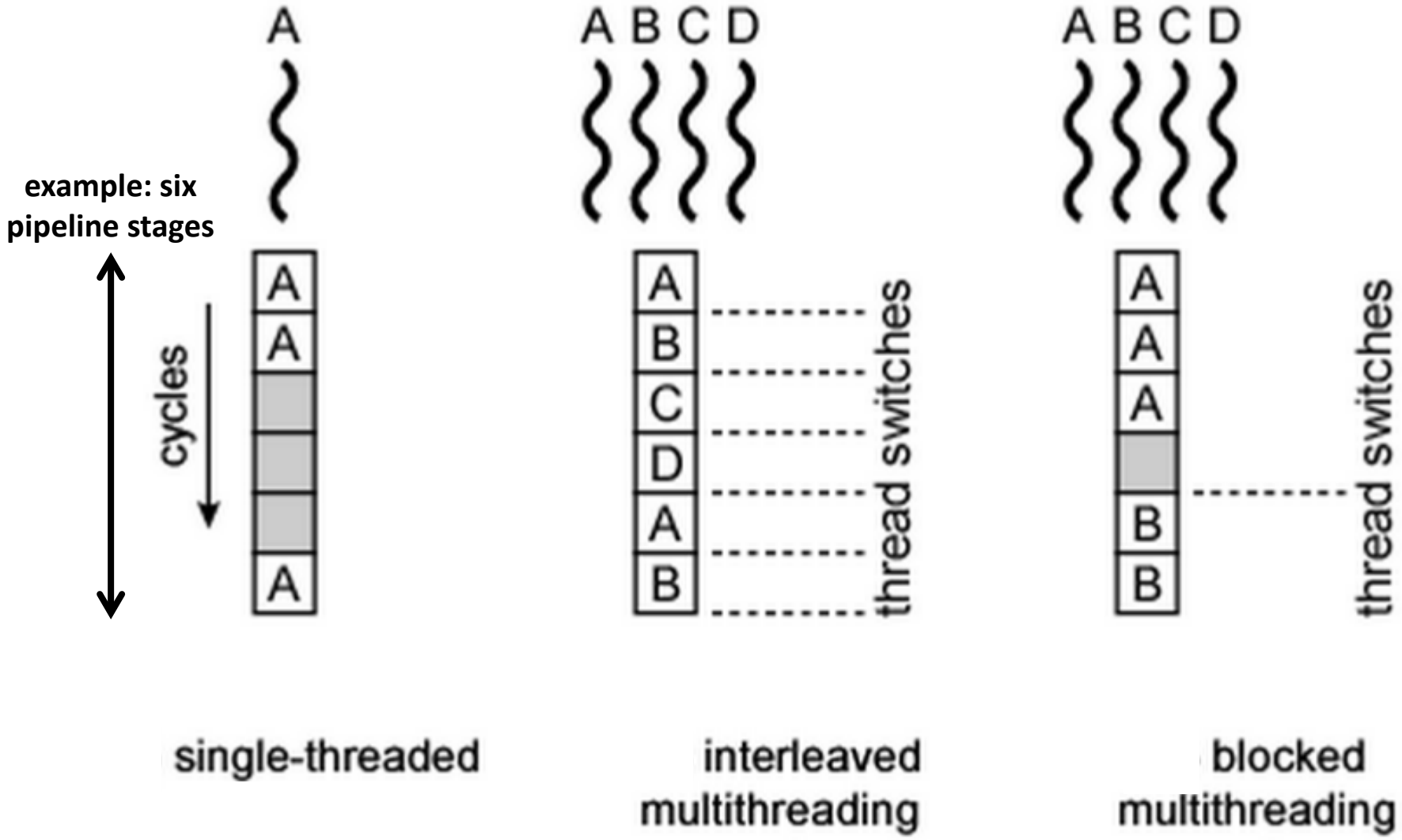
Hardware Threads

Moderne CPUs bieten Threads direkt auch auf Hardware-Ebene an.

- Zu diesem Zweck erhalten CPUs mehrere unabhängige Registersätze und Programmzähler, d.h. jeweils einen Registersatz pro möglichem Thread.

- Formen des Hardware-Multithreading
  - *Blocked multithreading*
    - Ausführen eines Threads bis die Ausführung anhält (z.B. wegen Cache Miss)
  - *Interleaved multithreading*
    - In jedem Takt wird zwischen Threads umgeschaltet. Falls ein Thread blockiert ist, wird seine Ausführung in diesem Takt übersprungen.
  - *Simultaneous multithreading*
    - Instruktionen verschiedener Threads werden gleichzeitig geladen («multiple issue»).

Single Issue Architektur



Multipe Issue Architektur