Prozesse und Threads

Betriebssysteme I
WS 2006/2006

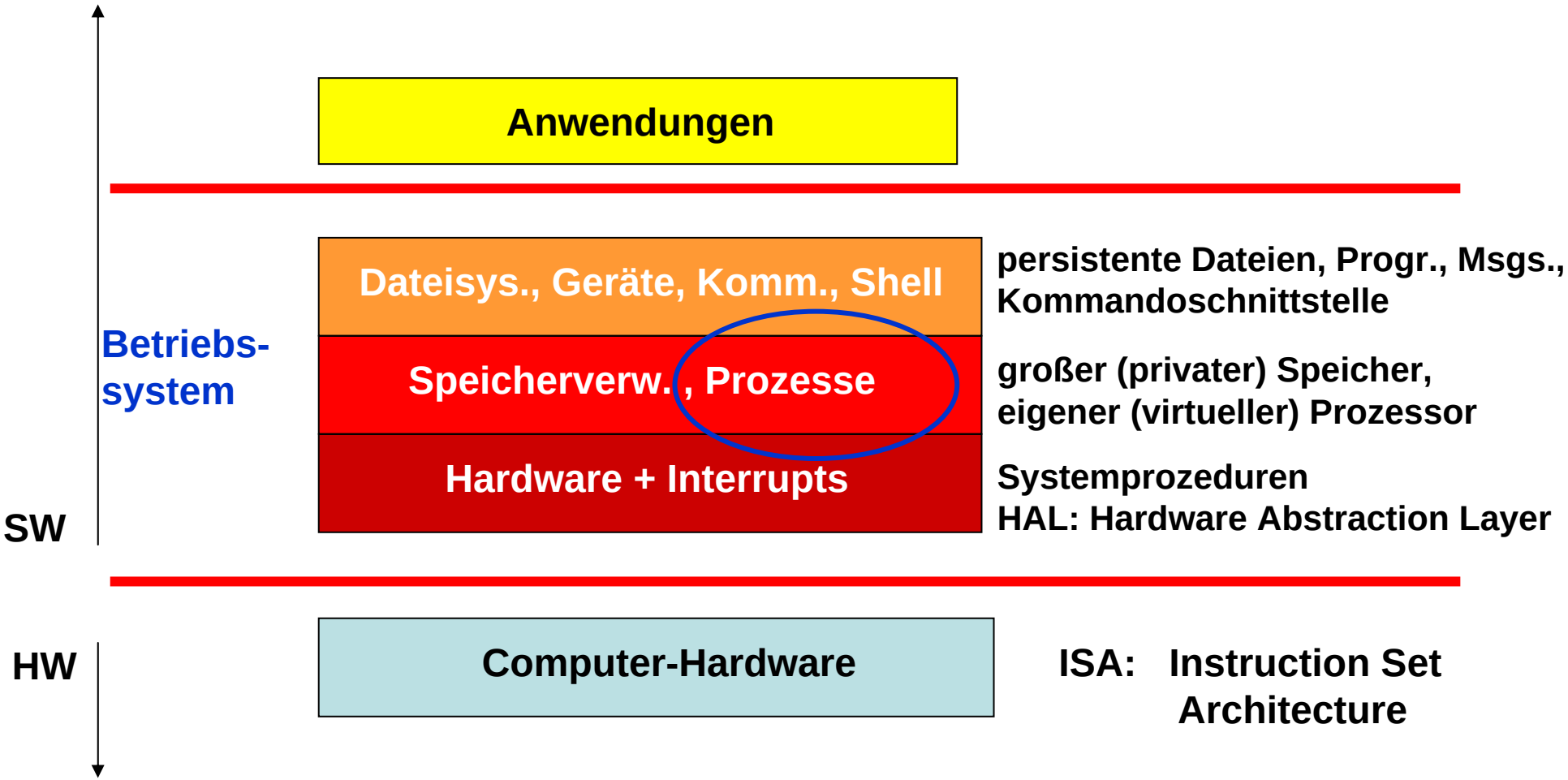
Jörg Kaiser
IVS – EOS

Otto-von-Guericke-Universität Magdeburg

Die wichtigste Aufgabe moderner Betriebssysteme ist die Prozessverwaltung.

W. Stallings

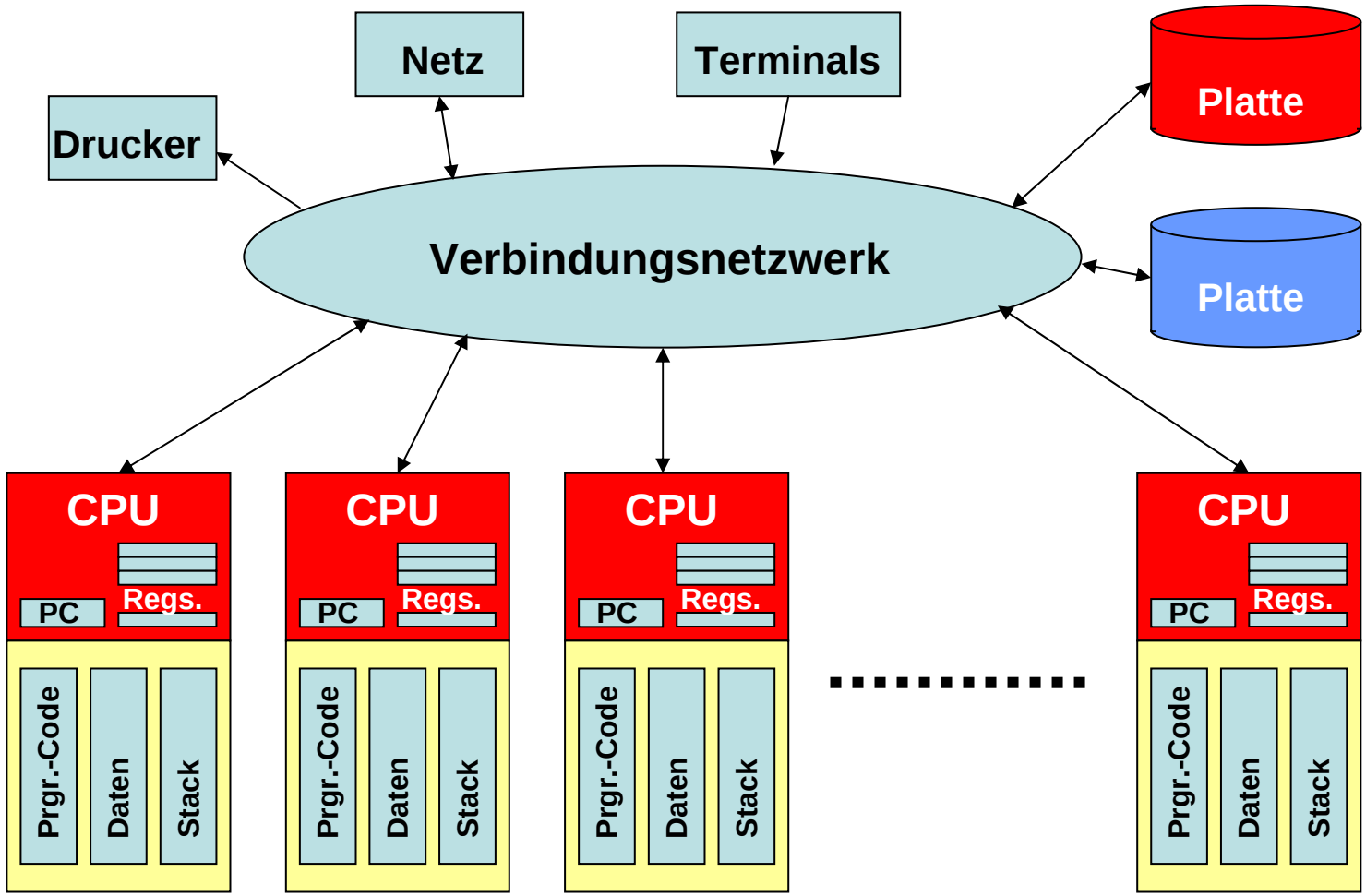
Schichtenmodell



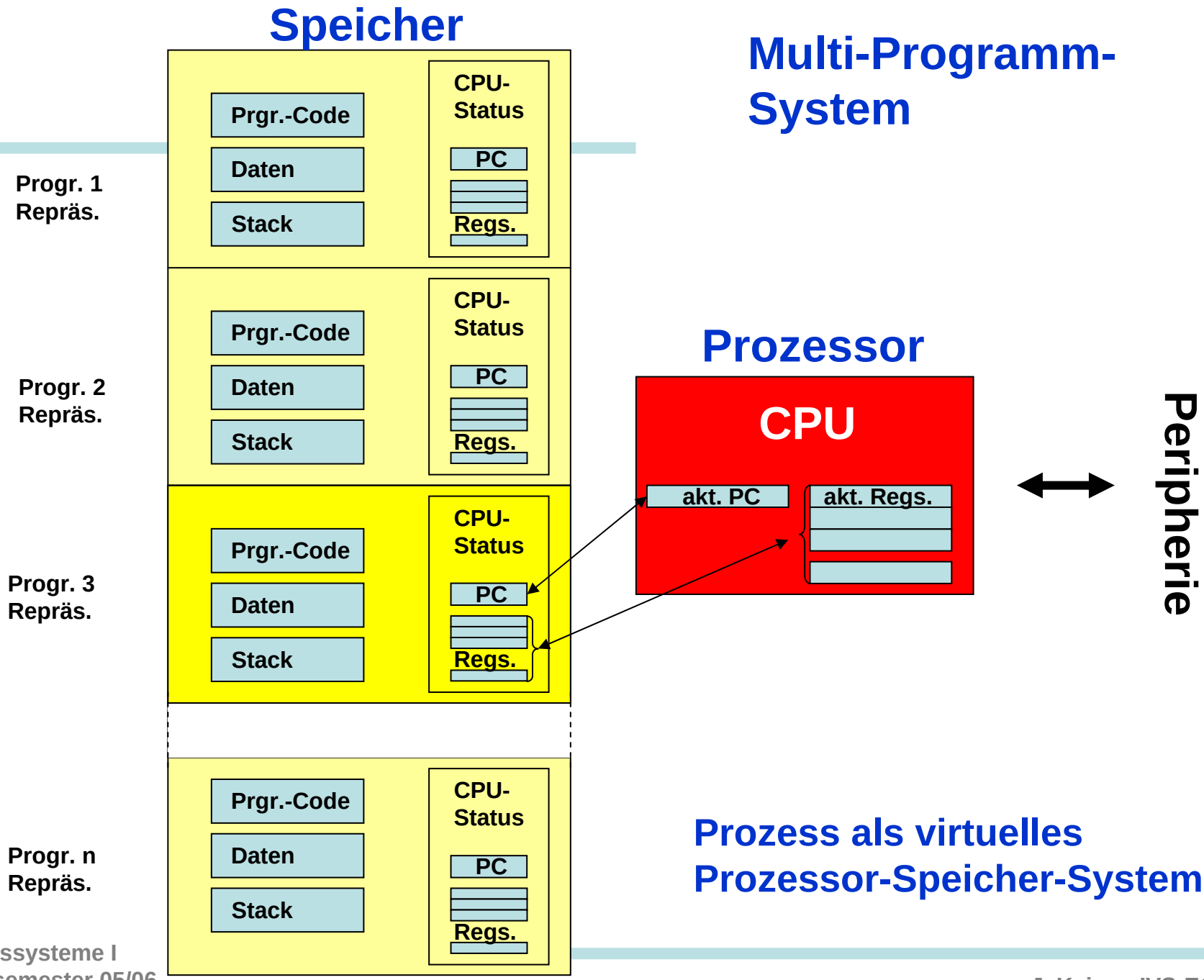
Ressourcen: Aktivitäten und Zustand

	Aktivitätsträger	Zustandsträger
physische HW-Komponenten	Prozessor(en)	flüchtiger Speicher, persistenter Speicher, Geräte
BS- Abstraktionen	Prozesse, Threads (Ausführungsfäden)	Adreßräume, Dateien

Multi-Computer-System mit gemeinsamer Peripherie



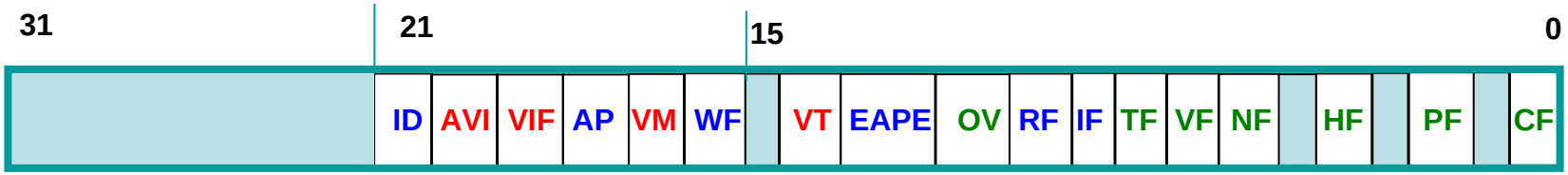
Multi-Programm-System



Prozess als virtuelles Prozessor-Speicher-System

Beschreibung von Prozessen: Unterstützung durch die CPU

Repräsentation des CPU-Status im Pentium EFLAGS-Register:



ID: Identifikation, Information über CPUID-Befehl
AVI: Anliegender "virtueller Interrupt" im Zusammenhang mitden 8086 Modus
VIF: Virteulles Interrupt-Flag (8086 Modus)
AP: Alignment: zur Erkennung /Kontrolle von nicht auf Wortgrenzen liegenden Informationen.
VM: Virtueller 8086 Modus
WF: Wiederaufnahme-Flag (Resume), erlaubt die Sperrung einer Exception nach Fehlerbehandlung
VT: Nested-Task-Flag, eine Task ist mit einer anderen Task in einem geschützten Bereich verschachtelt.
EAPE: Ein/Ausgabe Privilegebene. Während des geschützten Modus werden Ausnamen bei E/A generiert.
OV: Overflow Flag
RF: Zur Verarbeitung von Zeichenketten benutzt.
IF: Interrupt Freischaltungs-Flag
TF: Trap-Flag
VF: Vorzeichen Flag
NF: Null-Flag
HF: Übertrag zwischen Halbbytes
PF: Paritäts-Flag (gerade oder ungerade)
CF: Carry-Flag

Steuerbits
Betriebsmodus-Bits
Zustandscodes

Beschreibung und Initialisierung von Prozessen

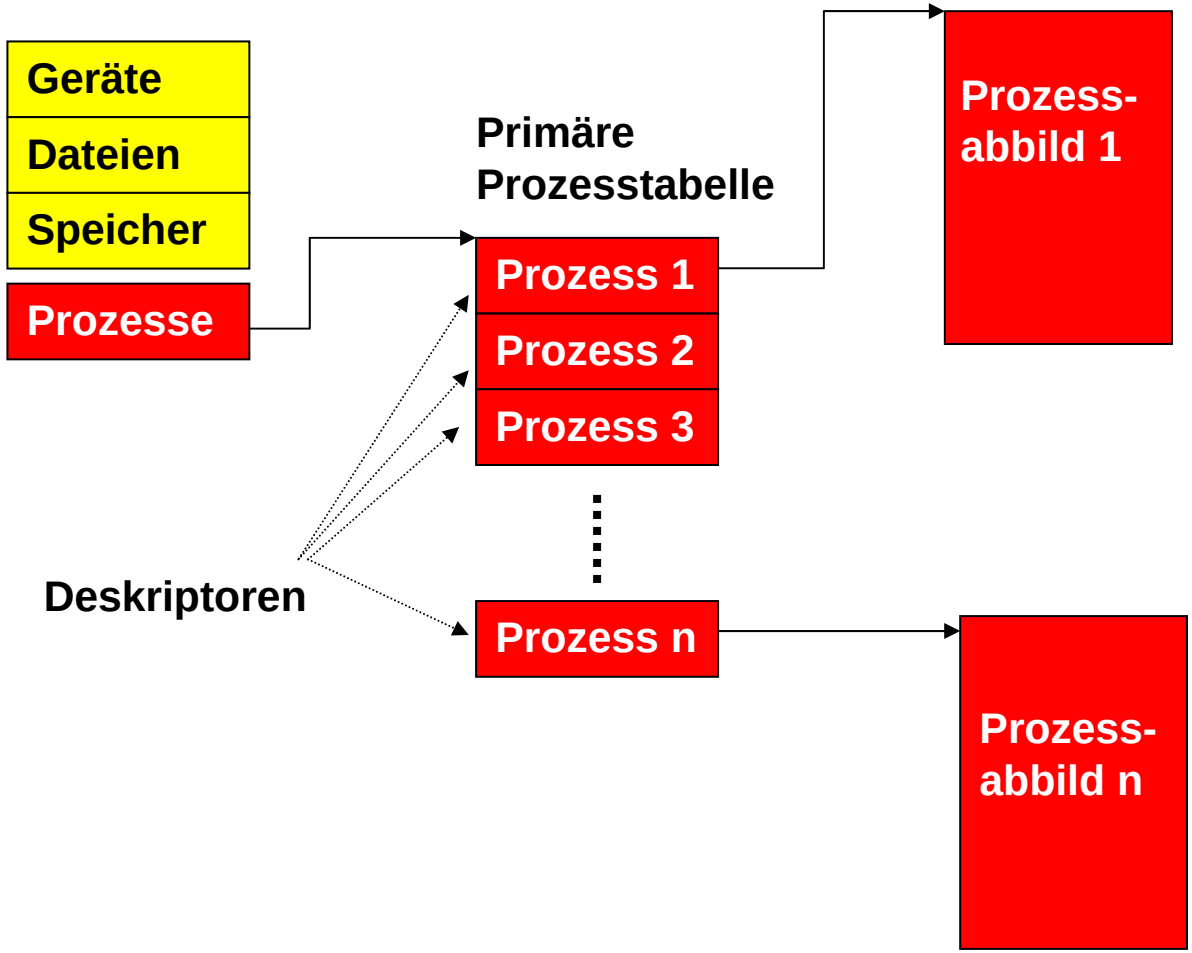
Verwaltung von Prozessen

Zustände von Prozessen

Wechsel zwischen Prozessen

Beschreibung von Prozessen

Systemkontrolltabellen in einem Betriebssystem



Prozessverwaltung:

Prozessidentifikation:
Prozessname, Programmname.

Zustandsinformation:
Programmzähler,
Register, Stackpointer.

Verwaltungsdaten:
Priorität. Rechte, Profilingdaten.

Speicherverwaltung:

Prozess-Segmente:
(Programm-) Textsegment
Datensegment
Stacksegment

Dateiverwaltung:

Verzeichnisse,
Deskriptoren,
Zugriffsschutzinfo.

Beschreibung von Prozessen

Prozessidentifikation: Numerische Kennung des Prozesses

- Kennung des Prozesses
- Kennung des "Eltern-" Prozesses
- Benutzerkennung

Prozess-Zustandsinformation:

- Benutzer-Register
- Steuer- und Statusregister
  - Programmzähler
  - Zustands- und Bedingungscodes, z.B. VZ, Carry, Overflow, ..
  - Statusinformation, z.B. Ausführungsmodus, Interruptsperren, ..
  - Stackpointer

Prozessverwaltungsinformation:

- Scheduling- und Zustandsinformation
  - Prozesszustand
  - Priorität
  - Schedulingparameter, z.B. Wartezeit, Zeitquantum, Deadline,...
  - ID des Ereignisses, auf das der Prozess wartet

Beschreibung von Prozessen

Zeiger zur Beschreibung der Prozessbeziehungen:
- **z.B. Eltern-Kind-Beziehungen und Warteschlangen.**

Interprozesskommunikation:
- **z.B. Flags, Signale oder Nachrichten.**

Prozessprivilegien:
- **erlaubt Zugriff auf Speicher, Ausführung privilegierter Operationen, Diensten und Systemprogrammen.**

Speicherverwaltung:
- **Zeiger auf Segment- und Seitentabellen für den zugeteilten Speicher.**

Ressourcenbesitz und Ressourcennutzung:
- **z.B. geöffnete Dateien oder genutzte Geräte**
- **Profiling-Informationen**

Prozesstabelleneintrag in UNIX

Allgemeine Prozessinformation

Prozesszustand
Prozesszeiger
Prozessgröße
Benutzerkennungen
Prozesskennungen
Ereignisbeschreibung
Priorität
Signal
Timer
P_Link
Speicherstatus

User Area

Prozesstabellenzeiger
Benutzerkennungen
Timer
Signalbearbeitungsarray
Kontrollterminals
Fehlerfeld
Rückgabewert
E/A-Parameter
Dateiparameter
Beschreibungstabelle für B-Dateien
Beschränkungen
Erlaubnismodusfelder

Prozessabbild in UNIX

Benutzerkontext

- Prozesstext (ausführbares Maschinenprogramm)
- Prozessdaten
- User Stack
- Gemeinsam genutzter Adressraum

Registerkontext

- Programmzähler
- Prozessor Status Register
- Stackpointer
- Allgemeine Register

Systemebenenkontext

- Prozesstabelleneintrag
- Benutzerbereich (Prozesskontrollinformation für den Kern)
- Prozessbereichstabelle
- Kernel Stack

Typische Funktionen des Betriebssystemkerns

im Hinblick auf Prozessverwaltung

- Prozesserzeugung und Prozessterminierung
- Prozesswechsel
- Verwaltung der Prozesskontrollblöcke
- Prozessablaufplanung und Zuteilung (Scheduling und Dispatching)
- Prozesssynchronisation und Interprozesskommunikation
- Zuteilung von Adressraum an Prozesse
- Interrupt- und Trapbehandlung
- Buchführung

Prozesserzeugung

Zuteilung von Speicherplatz

Initialisierung des Prozesskontrollblocks

Integration in die dynamischen Datenstrukturen zur Verwaltung

- **Zuweisung eines Abschnitts in der Prozesstabelle,**
- **Zuweisung einer eindeutigen Prozesskennung,**
- **Kopie des Elternprozessabbildes wird erstellt (ohne gem. genutzt. Speicher),**
- **Aktualisieren der Zählerwerte für Dateien (Kind "erbt" alle Ressourcen),**
- **Zuweisung des Zustands "bereit",**
- **Kennzahl des Kindprozesses wird an den Elternprozess übergeben.**

Prozesswechsel

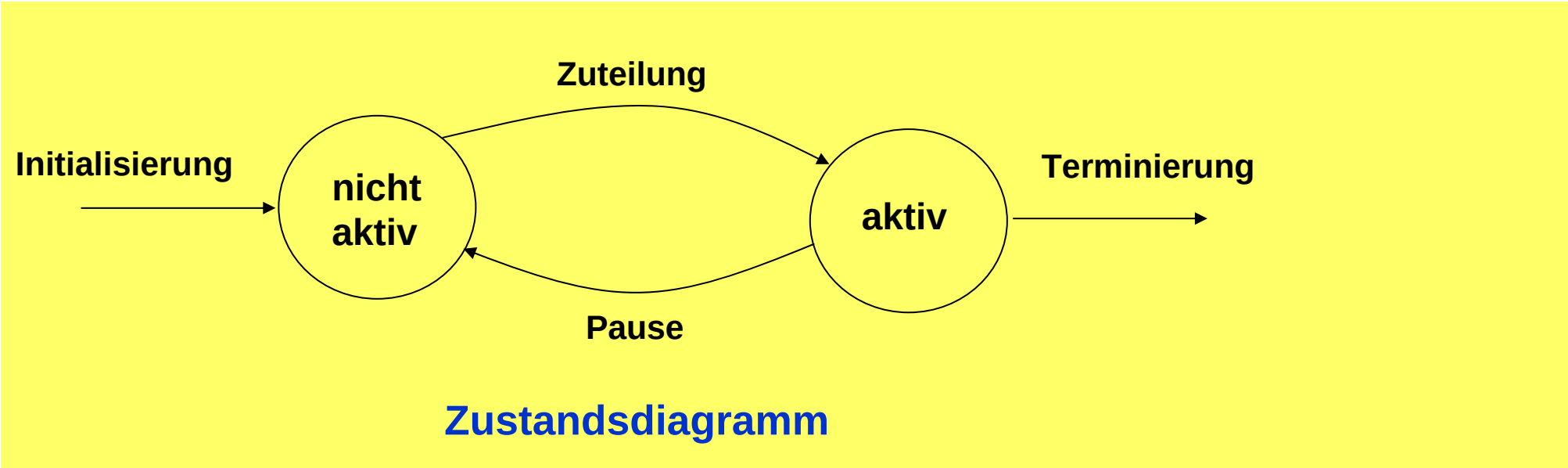
Ursachen für einen Prozesswechsel:

- Interrupt ➡ Zeit, E/A, Speicherfehler
- Trap ➡ ill. Op., ALU, Befehl, ...
- Supervisor Aufruf ➡ spez. Befehl

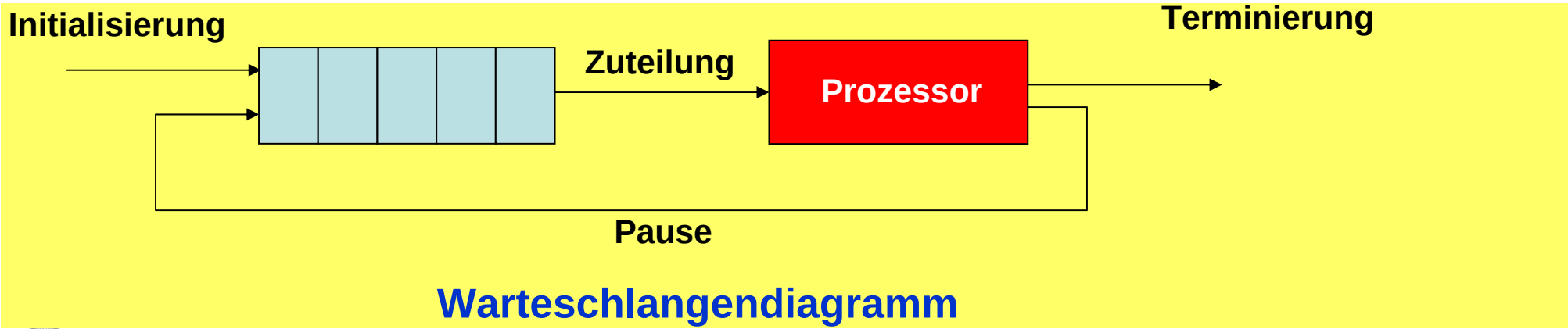
Aktionen bei einem Prozesswechsel:

- Abspeichern des CPU-Zustands
- Aktualisierung des Prozesskontrollblocks (PCB)
- Verschieben des PCB in eine entsprechende Warteschlange
- Auswahl eines anderen Prozesses für die Ausführung
- Aktualisierung des PCB für den neuen Prozess
- Aktualisierung der Speicherverwaltungsstrukturen
- Herstellung des CPU-Zustands des neuen Prozesses

Prozessmodell und Prozesszustände

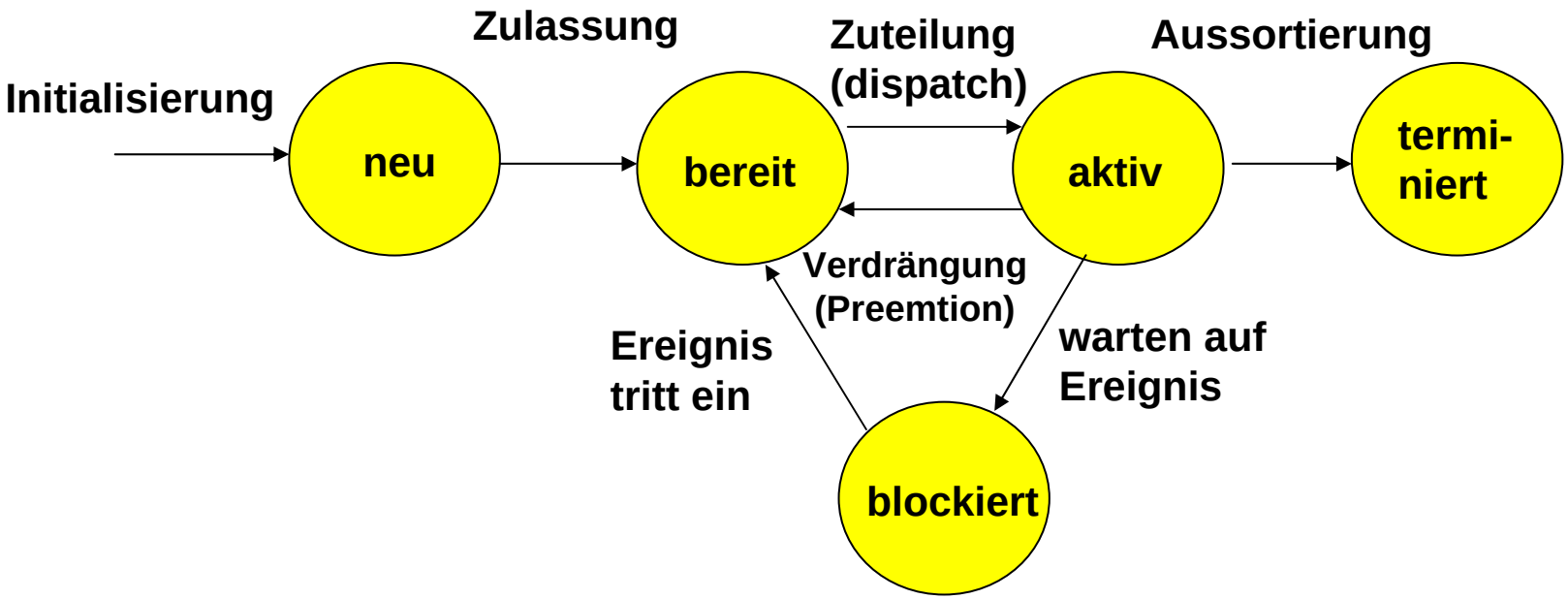


Zustandsdiagramm

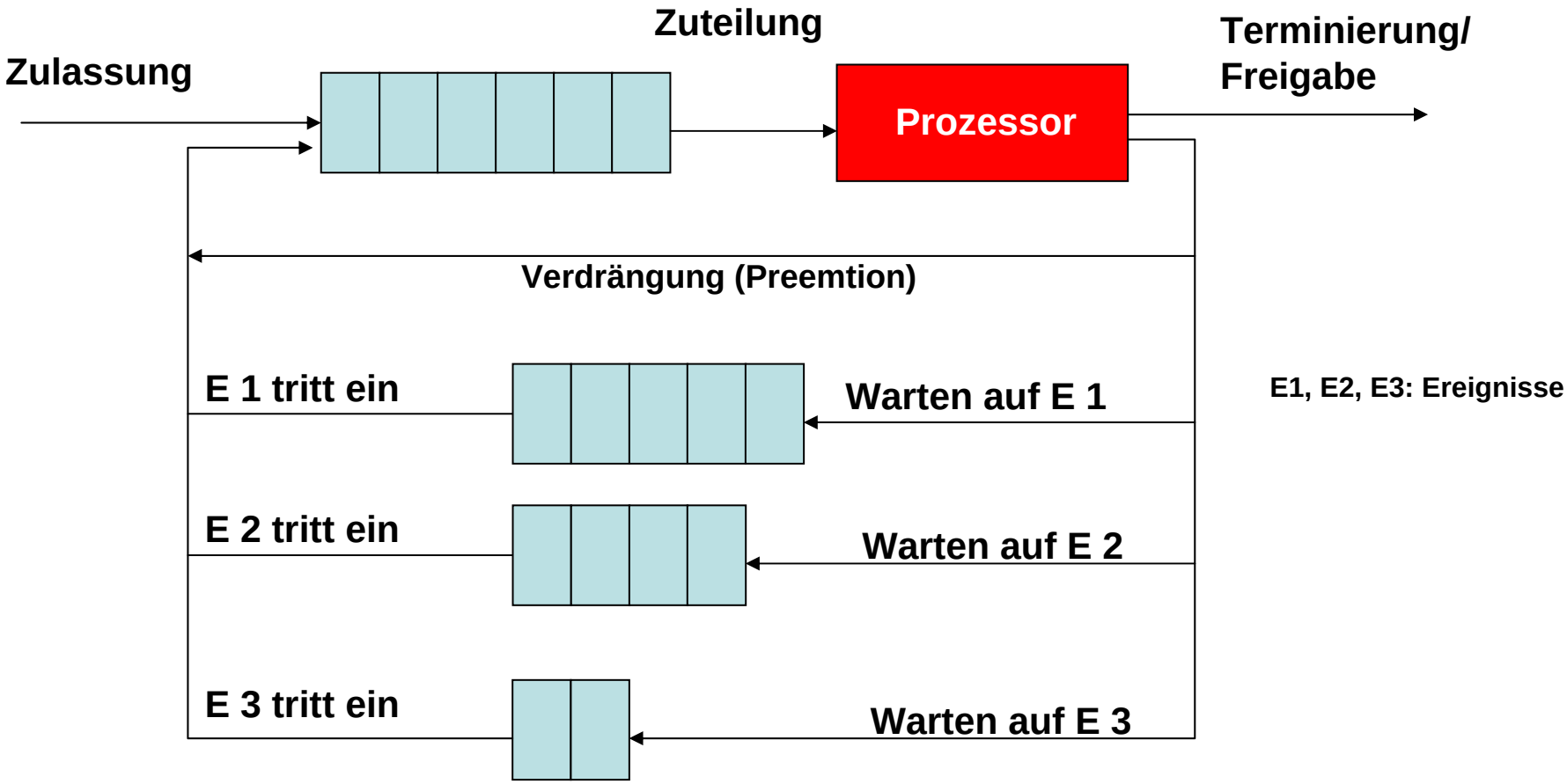


Warteschlangendiagramm

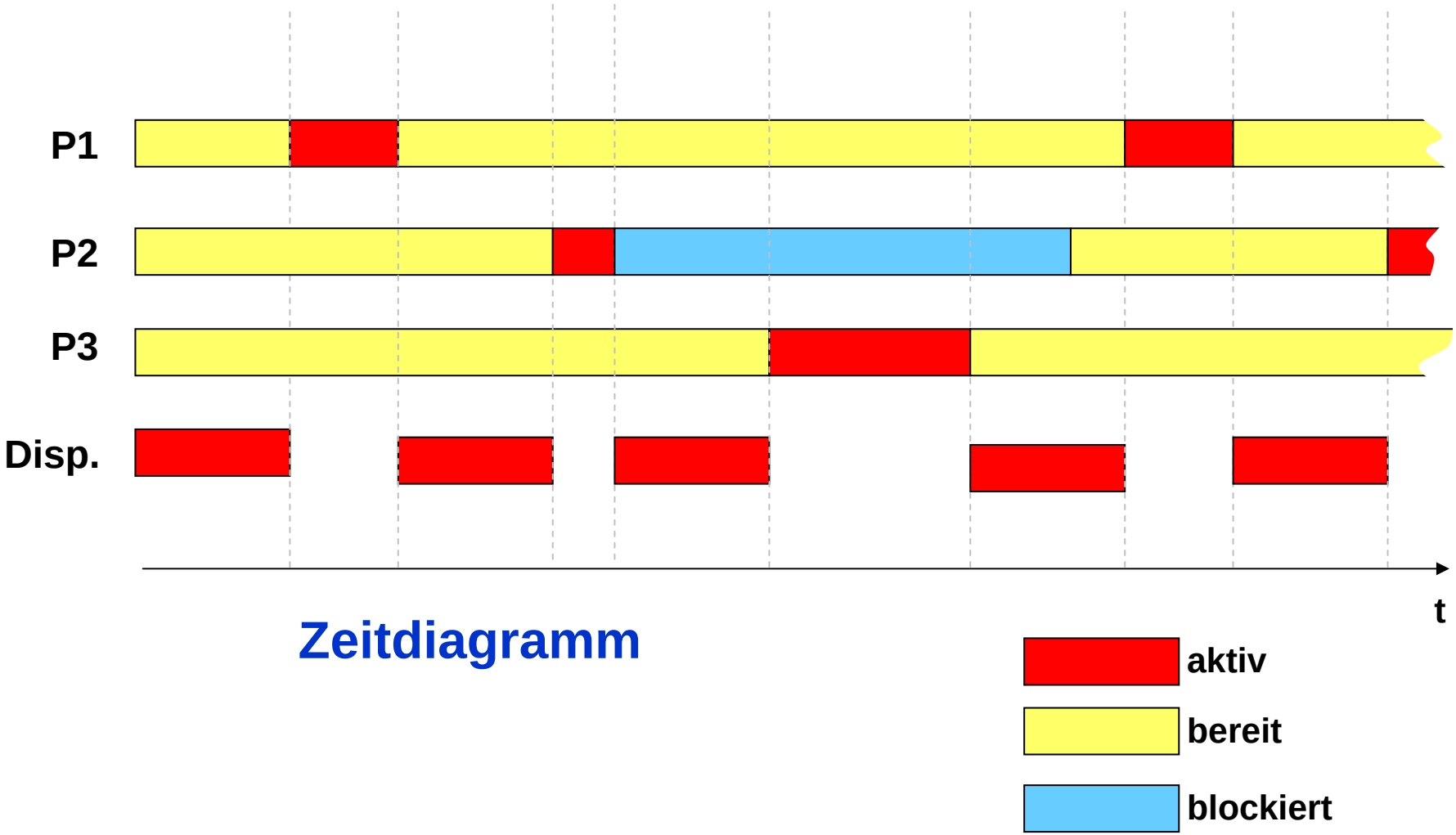
Prozessmodell und Prozesszustände



Prozessmodell und Prozesszustände



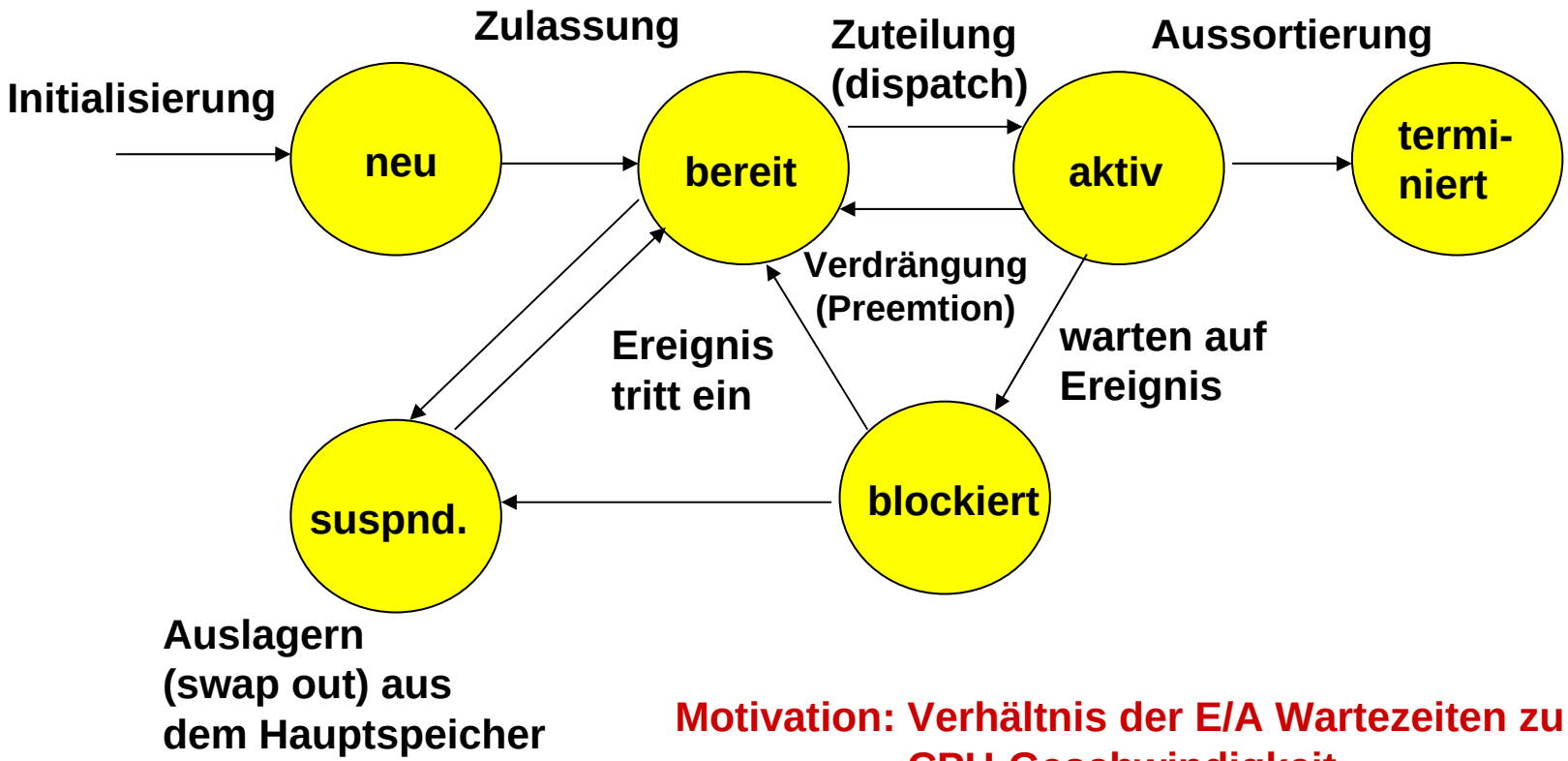
Prozessmodell und Prozesszustände



Zeitdiagramm

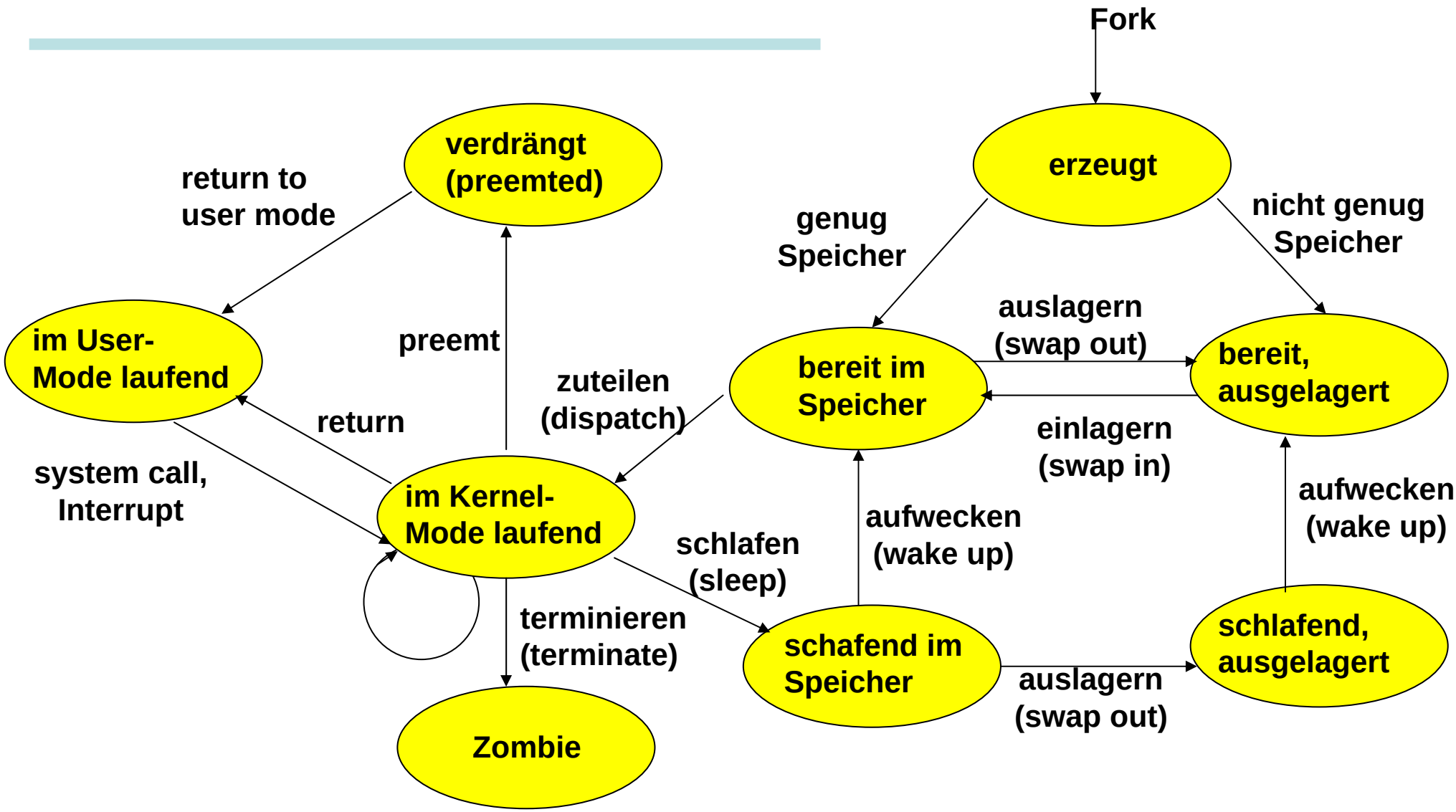
Prozessmodell und Prozesszustände

Weitere Zustände: Suspendierte Prozesse



Motivation: Verhältnis der E/A Wartezeiten zu CPU-Geschwindigkeit.

Unix Prozesszustände



Prozesse und Threads

Der Prozessbegriff umfasst zwei Aspekte:

1. Ein Prozess besitzt Ressourcen
2. Ein Prozess ist Träger der Aktivität

Die Aspekte sind orthogonal und können unabhängig vom Betriebssystem behandelt werden!

Threads

Trennung von Ressourcenzuteilung und Prozessorzuteilung

Motivation: Aufwand beim Umschalten von Prozessen.

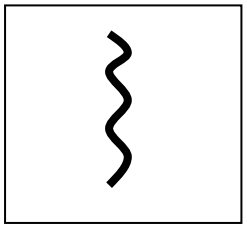
Einführung des "Thread of Control"

- **Threads sind sequentielle Befehlsausführungen.**
- **Threads sind die Einheit für die Prozessorzuteilung.**
- **Threads laufen in einem Prozessadressraum ab.**
- **Threads werden auch als "leichtgewichtige Prozesse" bezeichnet.**

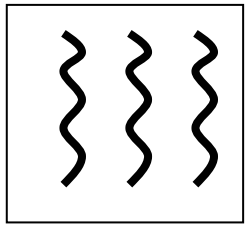
Ziele:
1. Strukturierung unabhängiger Programme und Programmkomponenten
2. Leistungssteigerung durch effiziente Parallelarbeit.

Threads und Prozesse

Ein Prozess

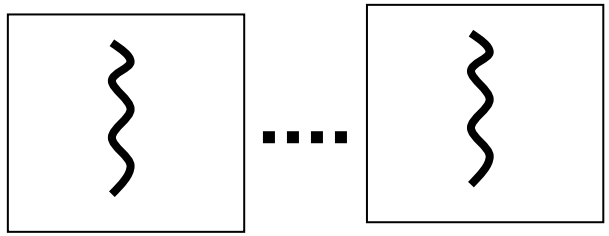


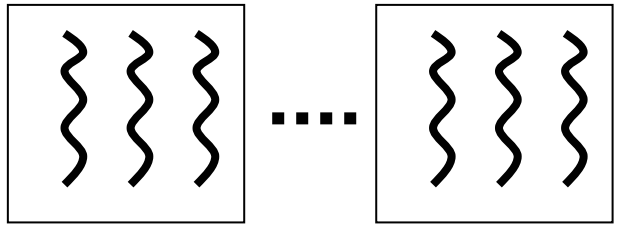
ein Thread/Prozess



mehrere Threads/Prozess

Mehrere Prozesse





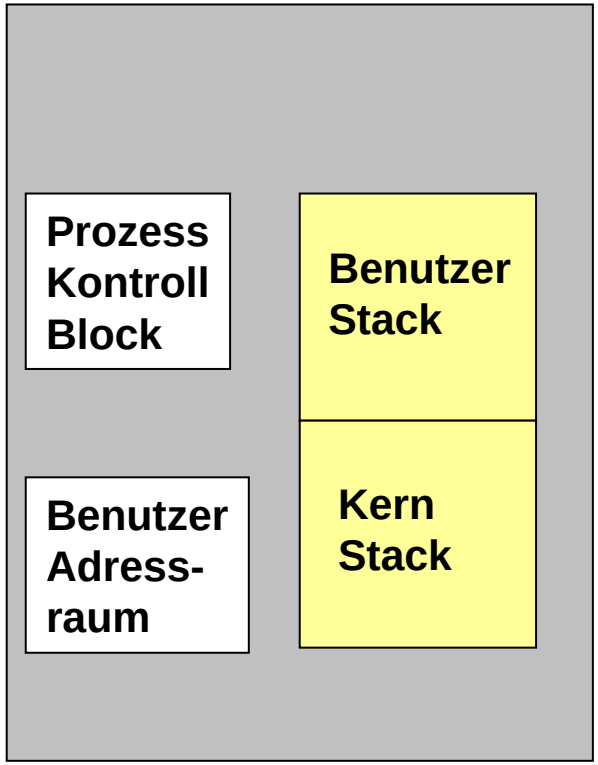
Threads und Prozesse

Thread besteht aus:

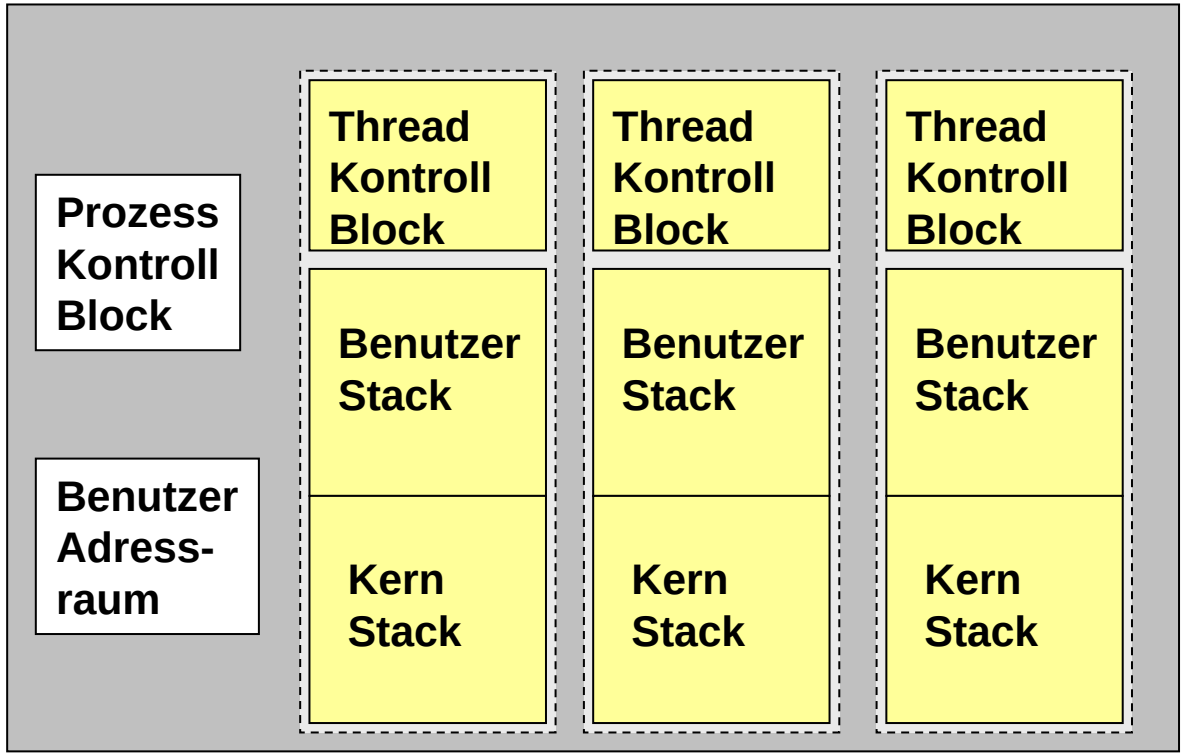
- **Ausführungszustand (bereit, aktiv, ...)**
- **Kontext**
- **Ausführungs-Stack**
- **Speicherplatz für Thread-lokale Variablen**

Ein Thread hat Zugriff auf alle Ressourcen des Prozesses in dem er abläuft!

Threads und Prozesse



Single-Thread Prozessmodell



Multi-Thread Prozessmodell

Threads und Prozesse

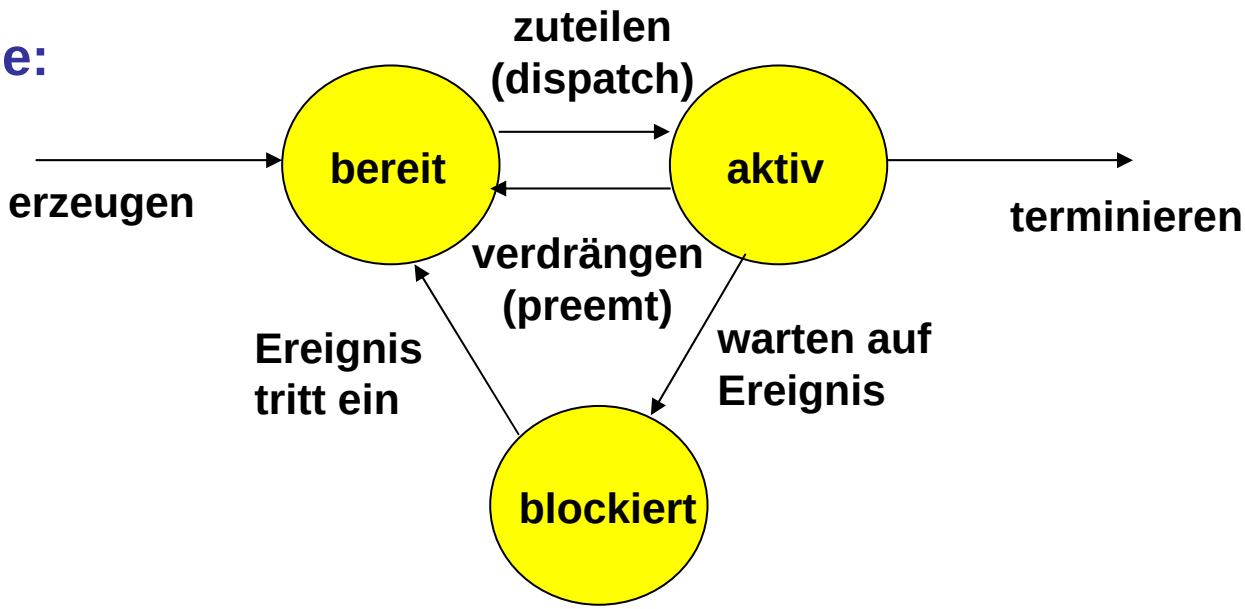
Vorteile:

- **Einfache und schnelle Erzeugung (~Faktor 10 gegenüber Prozessen)**
- **Schelleres Terminieren**
- **Schnelleres Umschalten**
- **Schnellere Kommunikation**

	Operation	User Thread	Kernel Thread	Prozess
Erzeugungsaufwand	Null Fork	34	948	11300
Synchronisation	Signal/wait	37	441	1840

Threads

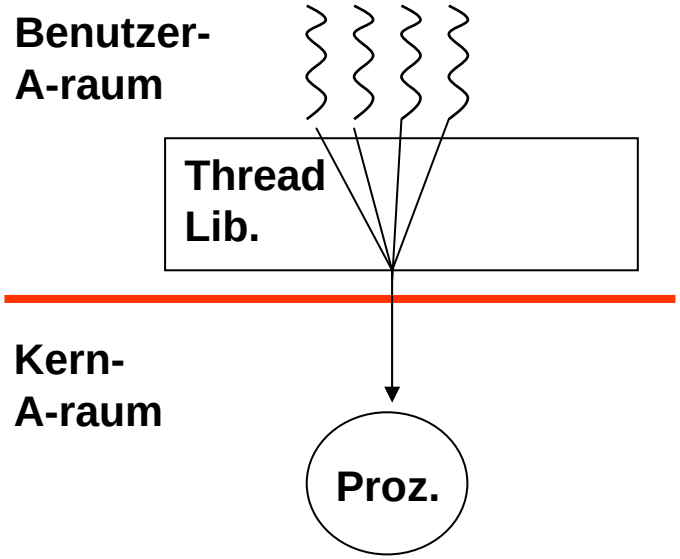
Thread Zustände:

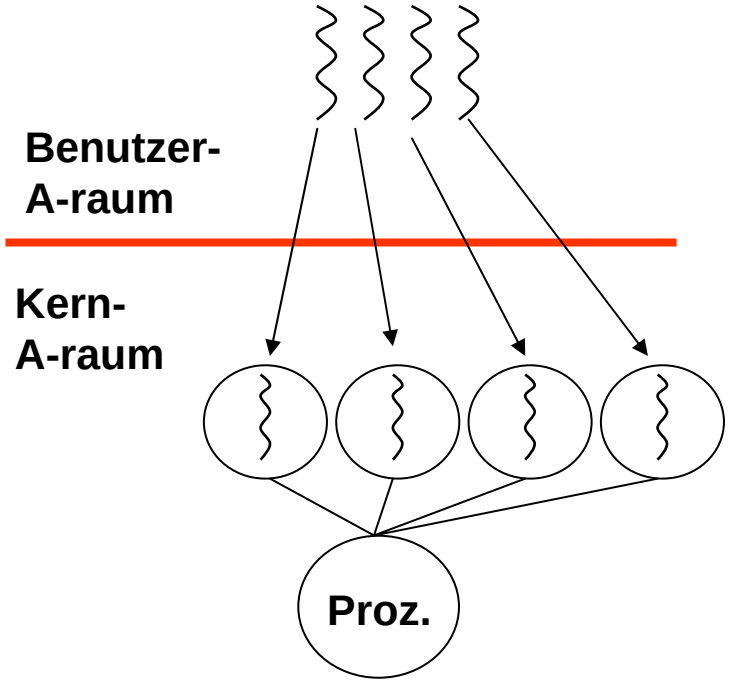


Thread Operationen:

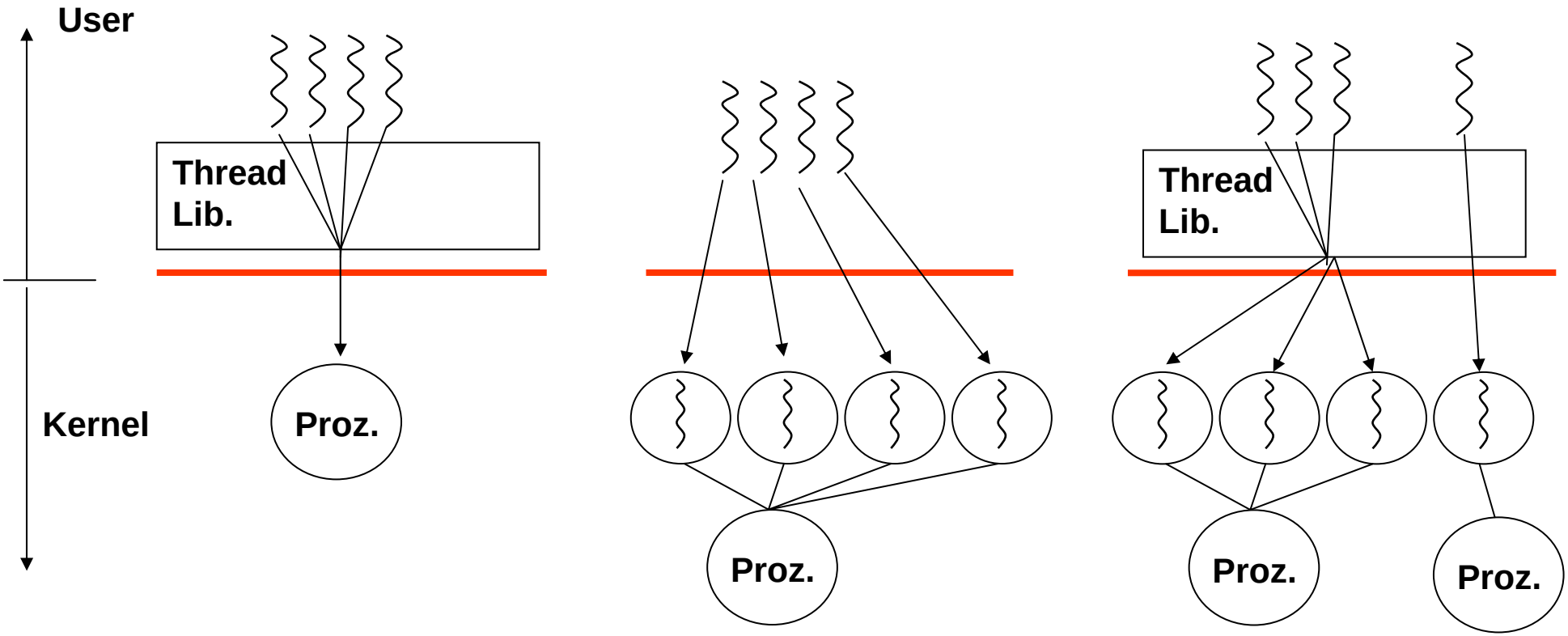
- Erzeugen (Spawn)
- Blockieren
- Aufhebung der Blockierung
- Terminierung

Kernel-Threads





Kernel-Threads



Threads

Benutzer-Thread vs. Kernel-Threads

Pro Benutzer-Threads:

1. Für Thread Wechsel werden keine Kernel Modus Privilegien benötigt. Alle Datenstrukturen zur Verwaltung der Threads befinden sich in einem einzigen Prozess-Adreßraum.
2. Anwendungsspezifisches Scheduling der Threads.
3. Unabhängig vom Betriebssystem.

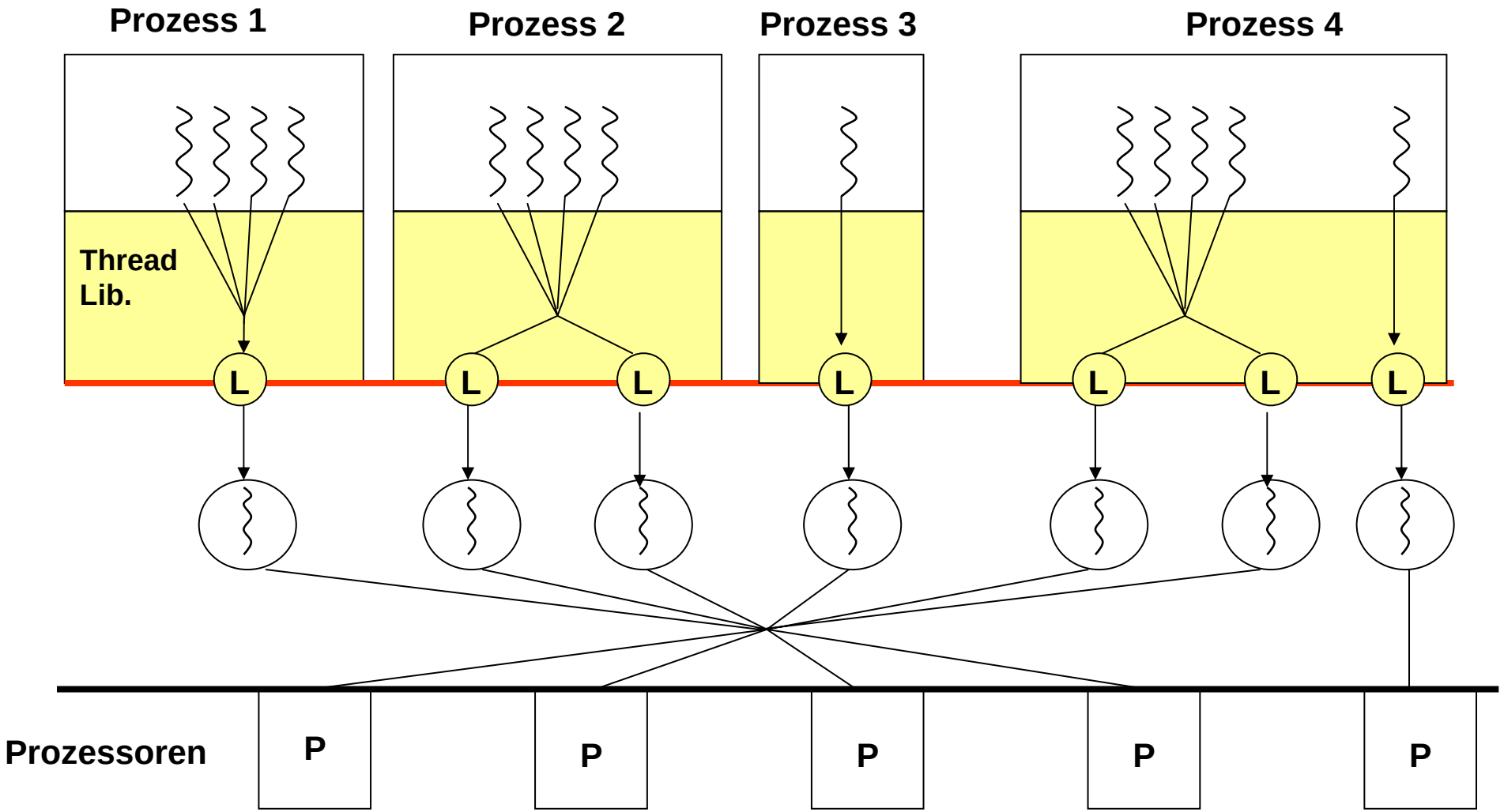
Con Benutzer-Threads:

1. Führt ein Benutzer-Thread einen blockierenden Systemaufruf durch, sind alle Threads blockiert.
2. Keine Parallelarbeit.

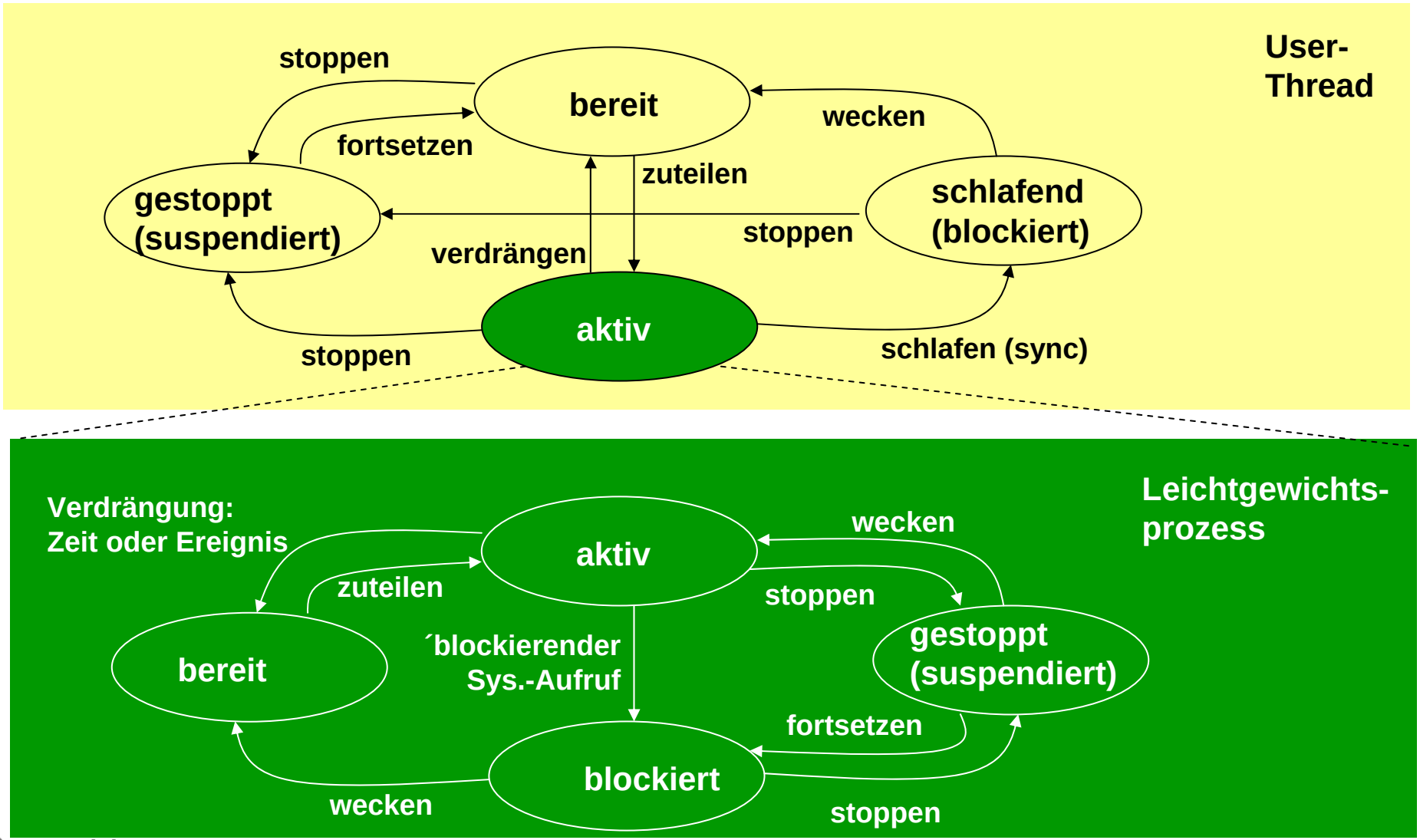
Verhältnis zw. Threads und Prozessen

1:1	Jeder Thread Ausführung ist ein eigener Prozess zugeordnet mit Adreßraum und Ressourcen	Traditionelle UNIX-Implementierung
n:1	Innerhalb eines Prozesses werden mehrere Threads ausgeführt.	Windows NT, Solaris, Linux, OS/2, Mach.
1:n	Ein Thread wandert von einer Prozessumgebung zur anderen.	Ra (Clouds), Emerald, MONADS, Grasshopper.

Threads in Solaris



Threads und Leichtgewichtsprozesse



Threads in Solaris

Prozess:	**wie bisher**
Benutzer Thread:	**wie bisher**
Leichtgewichtige Prozesse (L):	**Bewirken die Zuordnung von Benutzer-Thread zu Kernel-Thread. Leichtgewichtsprozesse werden unbhängig vom Kernel eingeplant.**
Kernel-Threads:	**Elemetare Einheiten des Scheduling: Sie werden vom Dispatcher einem Prozessor zugeteilt.**

Besonderheiten: Interrupts werden in Threads bearbeitet.

- **Eine Reihe von Kernel Threads werden als Interrupt-Threads mit eigener Kennung, Priorität, Kontext und Stack vorgesehen.**
- **Kernel steuert Zugriff auf Datenstrukturen und synchronisiert Interrupt-Threads.**
- **Interrupt-Threads wird eine höhere Priorität als anderen Kernel-Threads zugewiesen.**

Prozeduren, Koroutinen und Threads

Prozedur: Synchroner Aufruf von Unterprogrammen.

Aufruf: Speicherung des Kontexts (Aktivierungsblock) der aufrufenden Routinen auf dem Stack.
Rückkehr: Kontext wird wiederhergestellt, Kontextinformation auf dem Stack geht verloren, Rückkehr in das aufrufende Programm
Asymmetrie (Hierarchie) zwischen aufrufenden und aufgerufenen Programm.
Stack global für alle Routinen.

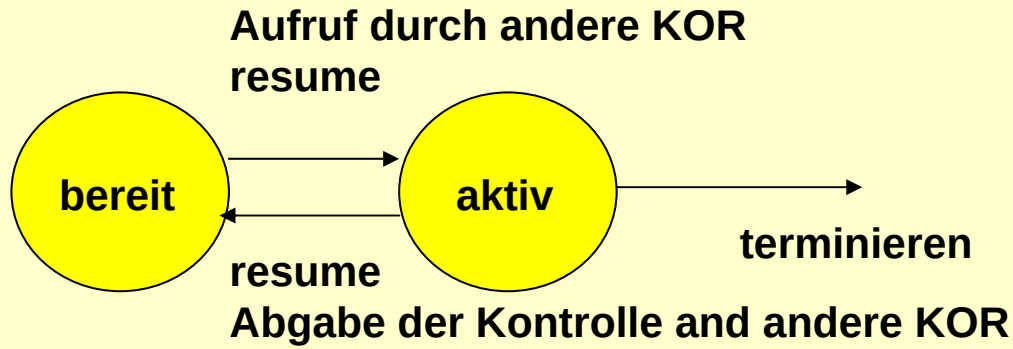
Koroutine: Synchroner Aufruf von Unterprogrammen.

Aufruf: Speicherung des Kontexts der aufrufenden Koroutine in einem Aktivierungsblock (Kontrollblock), der der Koroutine zugeordnet ist.
Bei erneutem Aufruf, Einsprung an die Stelle, an der die Koroutine die Kontrolle explizit durch ein Resume an eine andere Koroutine abgegeben hat.
Rückkehr: nicht vorhanden.
Symmetrie zwischen aufrufenden und aufgerufenem Programm.
Kooperative Ablaufsteuerung.

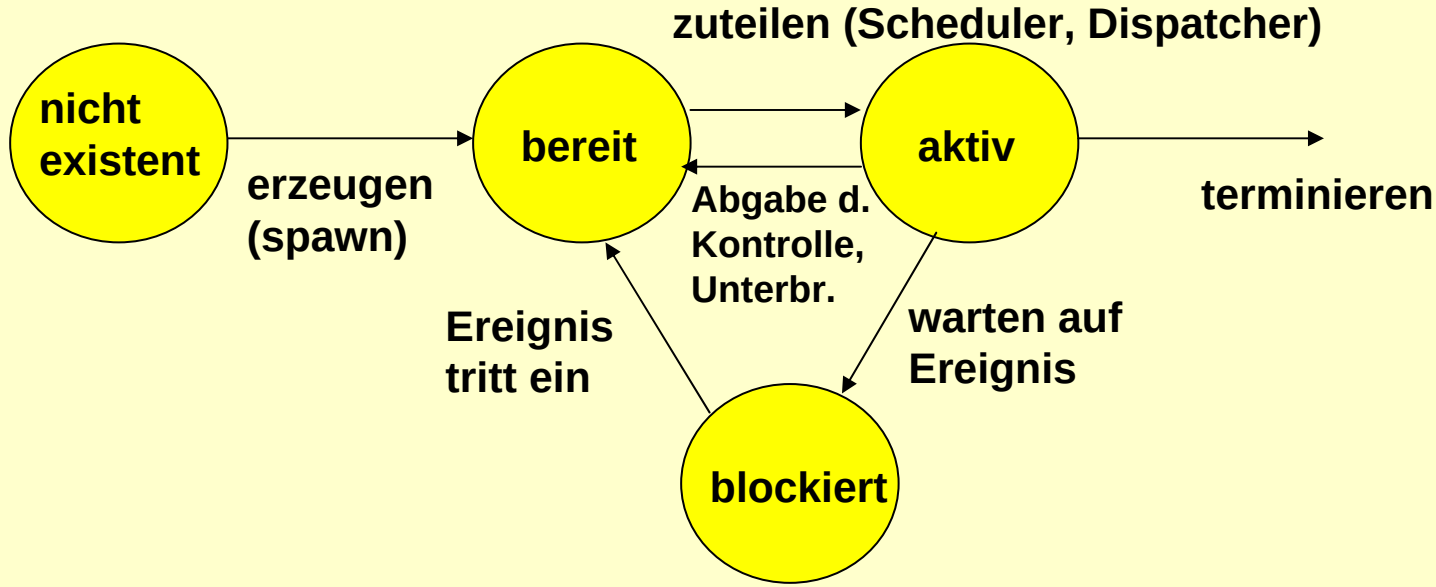
Threads: Asynchroner Aufruf. Unterschiede zur Koroutine:

1. Die Auswahl des Threads an dén die Kontrolle transferiert wird, übernimmt ein Scheduler, d.h. kein Einfluss auf Aufruffolge.
2. Ein Thread kann blockieren und auf ein Signal warten.
3. Ein Tread kann durch ein externes Ereignis oder einen Zeitgeber unterbrochen werden.

Kooperative Koroutinen



Threads



Zusammenfassung Prozesse:

Prozesse sind die Einheiten der Ressourcenverwaltung. Ressourcen sind Speicher und Prozessor.

Prozesse werden durch Datenstrukturen repräsentiert, welche die vollständige Beschreibung dieser Ressourcen zu jedem Zeitpunkt ihrer Ausführung enthalten.

Prozesse haben Zustände. Zustandsübergänge werden durch innere oder äussere Ereignisse angestossen.

Prozesse stehen möglicherweise in einer hierarchischen Beziehung zueinander. Die Kind-Prozesse erben die Ressourcen des Eltern-prozesses.

Zusammenfassung Threads:

Threads sind lineare Codesequenzen, die logisch nebenläufig ausgeführt werden können.

Motiviert durch den Aufwand einer Prozessumschaltung, die eine neue Schutzumgebung herstellt, werden Threads nebenläufig mit den vom Prozess zur Verfügung gestellten Ressourcen und Schutzumgebung ausgeführt.

Benutzer-Threads können besonders effizient verwaltet werden, da sie ausschließlich im Benutzer-Modus ablaufen. Sie erlauben aber keine Ausnutzung echter Hardwareparallelität.

Kernel-Threads laufen im Kerneladressraum eines Prozesses ab und erlauben die Synchronisation mit externen Ereigneissen und Ausnutzung von Hardwaremechanismen zur Unterstützung der Nebenläufigkeit.

Das Zusammenspiel von Benutzer- und Kernel-Threads kann z.B. unter Solaris sehr flexibel und anwendungsorientiert gestaltet werden.