Prozesse und Threads

Betriebssysteme
WS 2010/2011

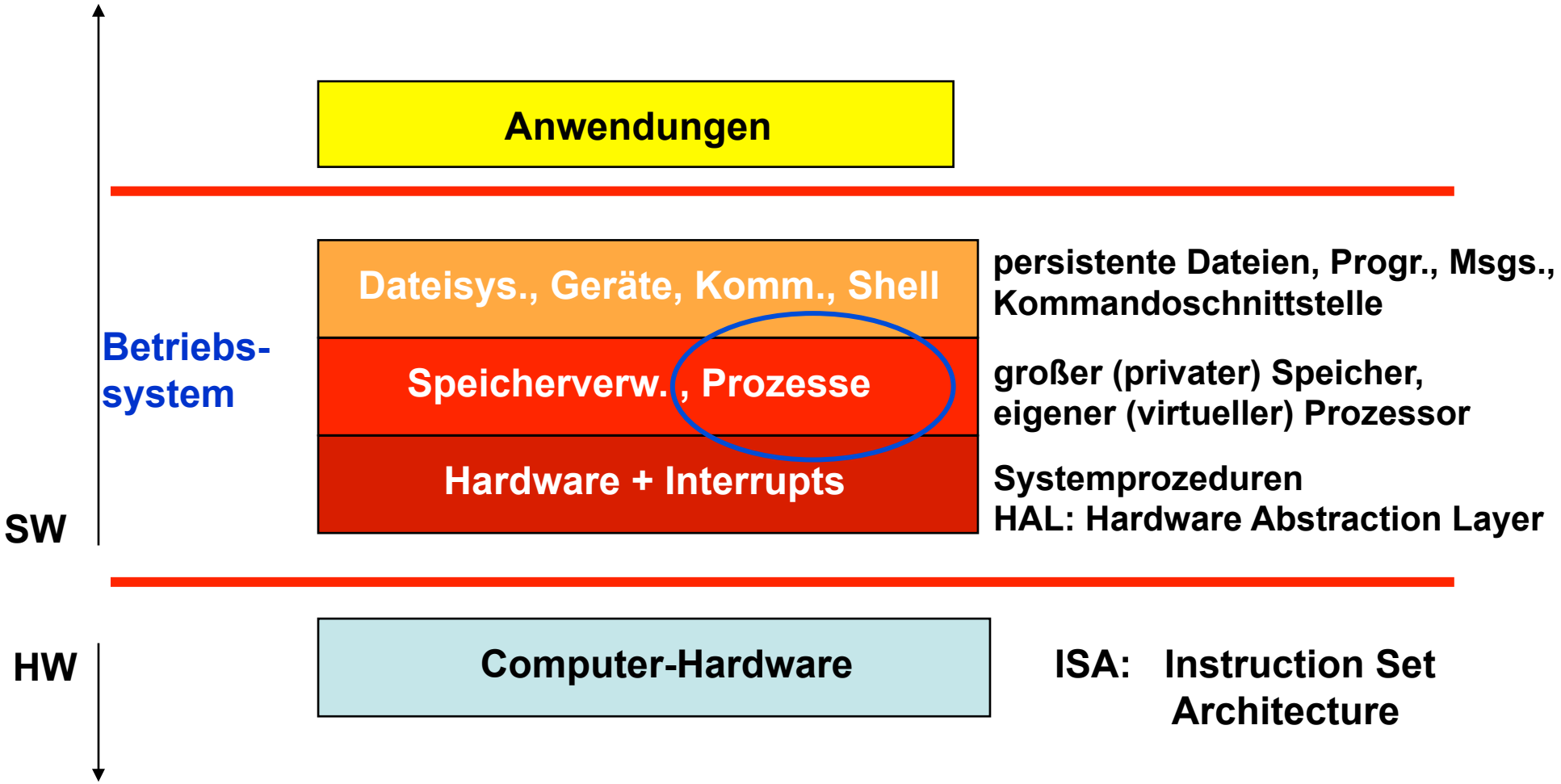
Jörg Kaiser
IVS – EOS

Otto-von-Guericke-Universität Magdeburg

Die wichtigste Aufgabe moderner Betriebssysteme ist die Prozessverwaltung.

W. Stallings

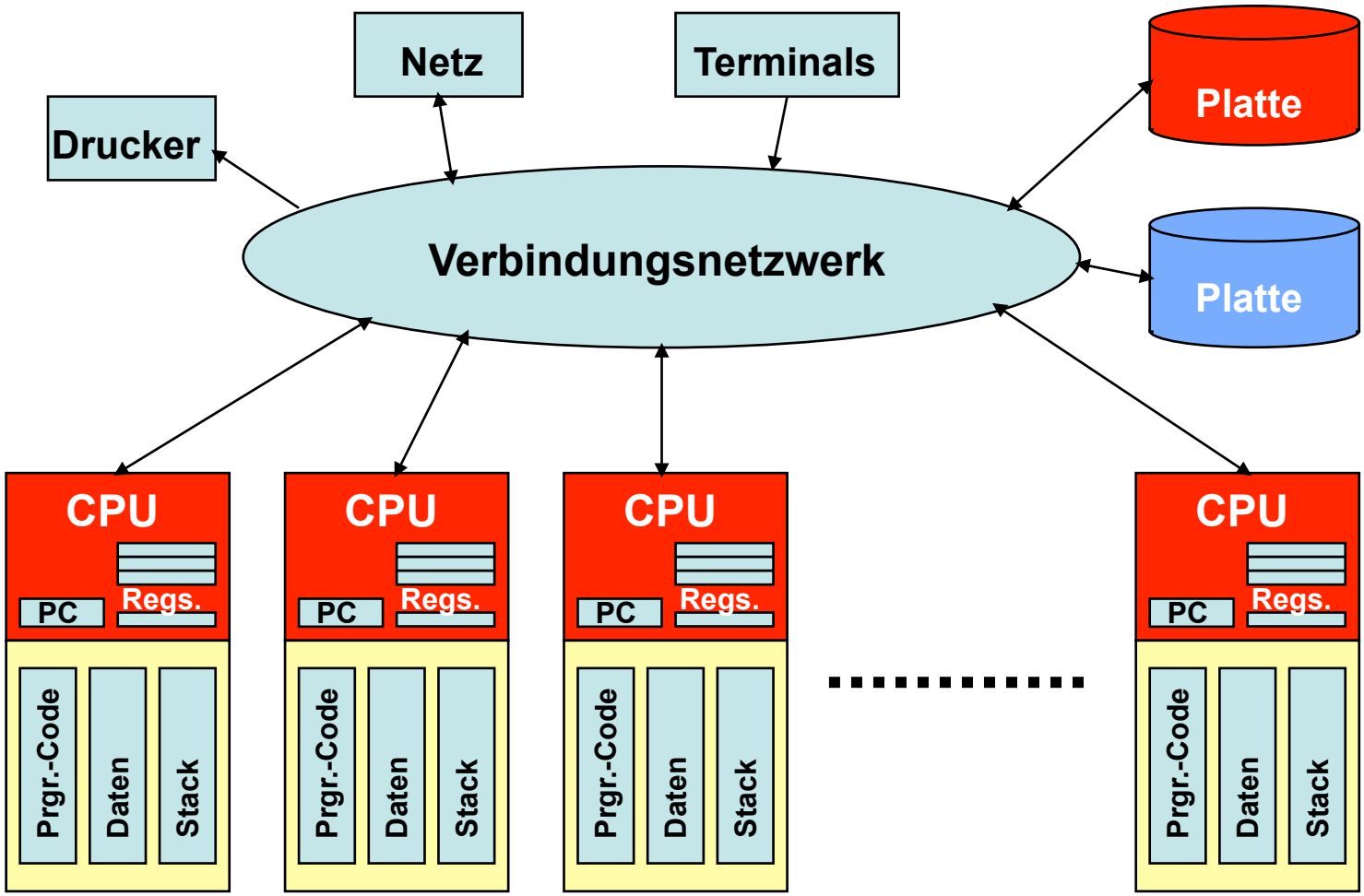
Schichtenmodell



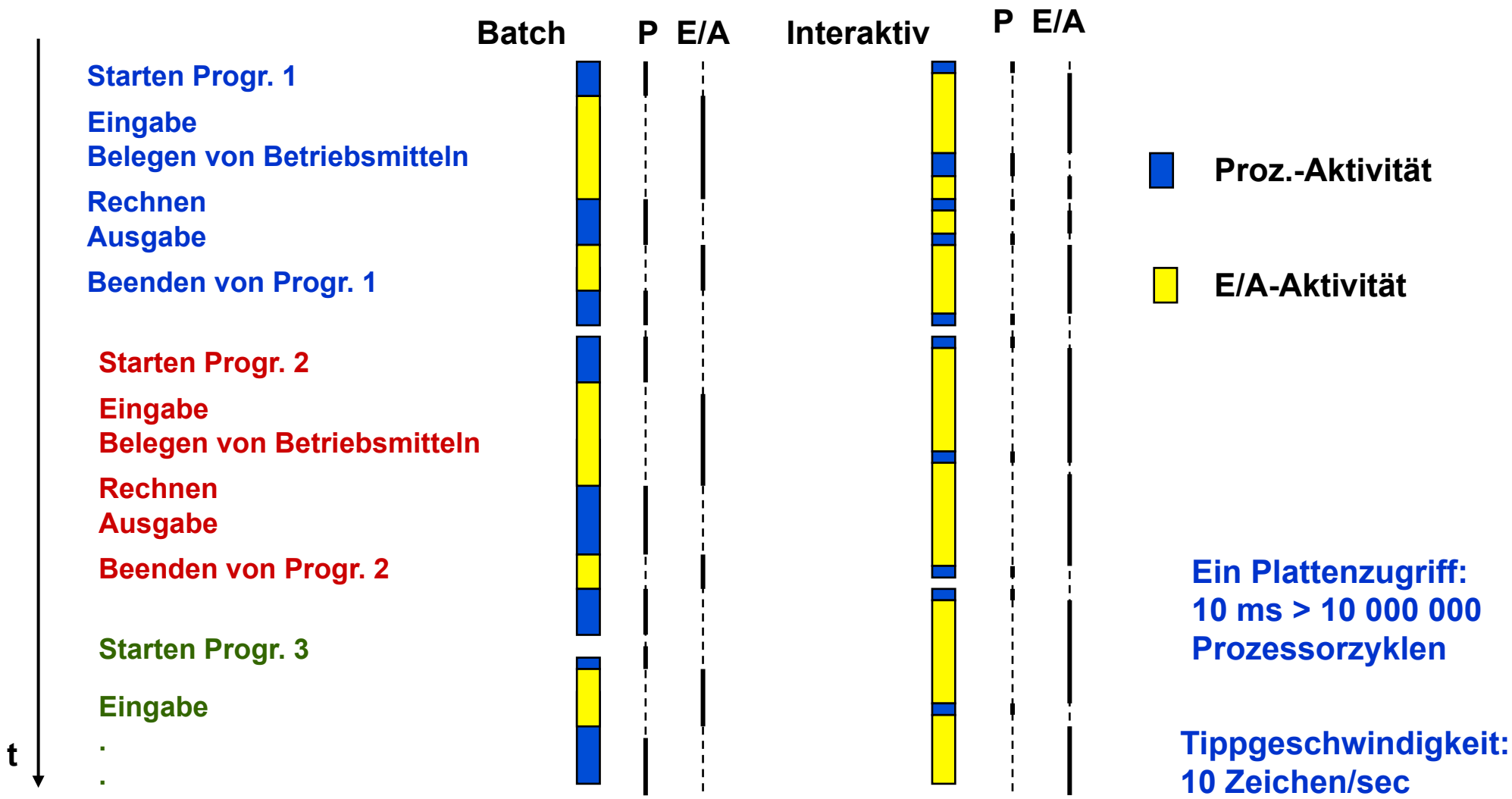
Ressourcen: Aktivitäten und Zustand

	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



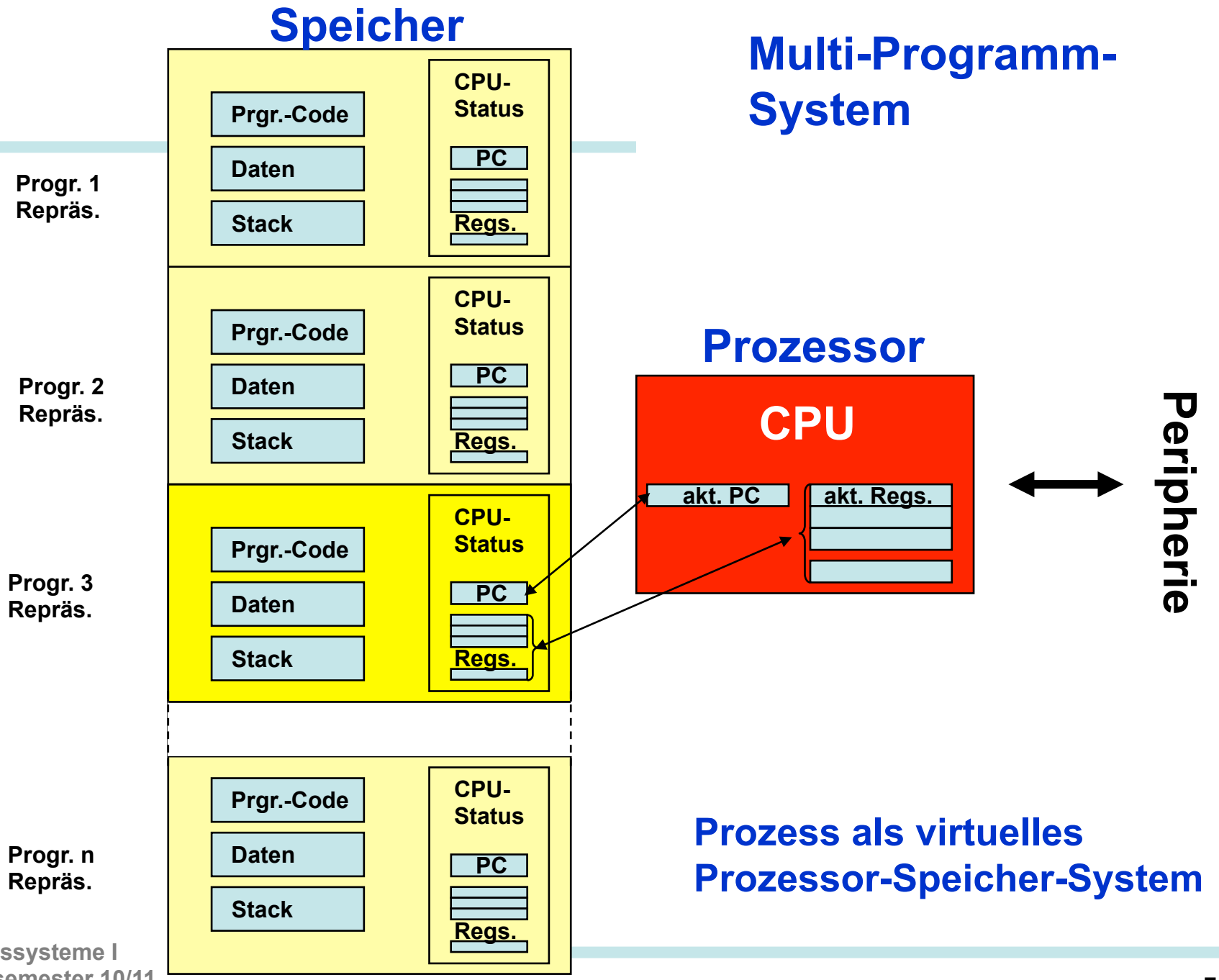
"One program at a time"



Ein Plattenzugriff:
10 ms > 10 000 000
Prozessorzyklen

Tippgeschwindigkeit:
10 Zeichen/sec

Multi-Programm-System



Prozess als virtuelles Prozessor-Speicher-System

Themen im Hinblick auf Prozesse und deren Verwaltung

Beschreibung und Initialisierung von Prozessen

Zustände von Prozessen

Verwaltung von Prozessen

Wechsel zwischen Prozessen

Beschreibung von Prozessen (Prozesskontrollblock)

Prozess-Identifikation

Prozess-Status

Prozess-Kontrolle

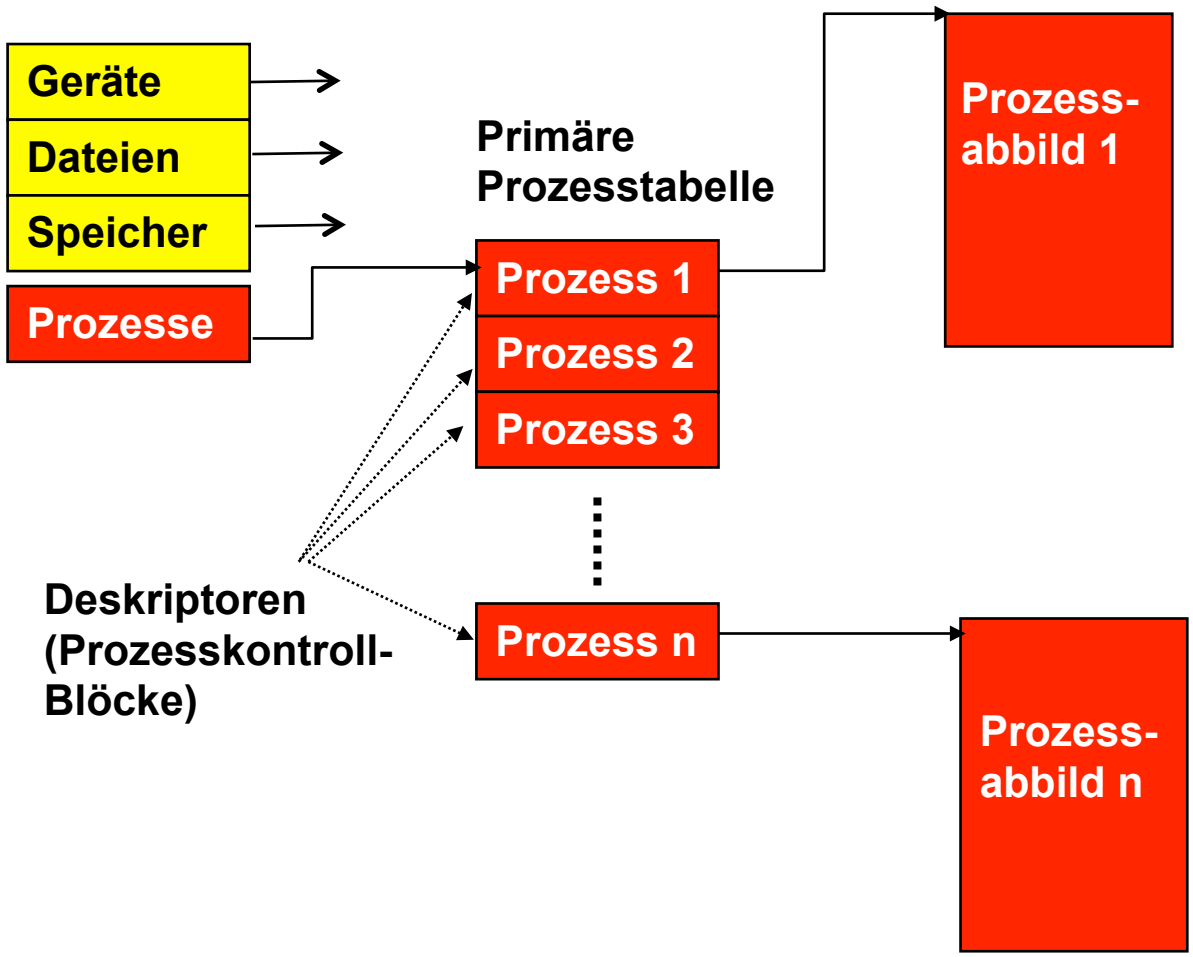
Prozessstruktur wird im Prozess-Kontrollblock beschrieben

Prozess-Abbild

Speicherabbild wird dynamisch angelegt und enthält zusätzl. Progr.-, Daten-, und Stackbereiche

Beschreibung von Prozessen

Systemkontrolltabellen in einem Betriebssystem



Prozessverwaltung:

Prozessidentifikation:
Prozessname, Programmname.

Zustandsinformation:
Programmzähler, Cond.-Code-R, allg. Register, Stackpointer.

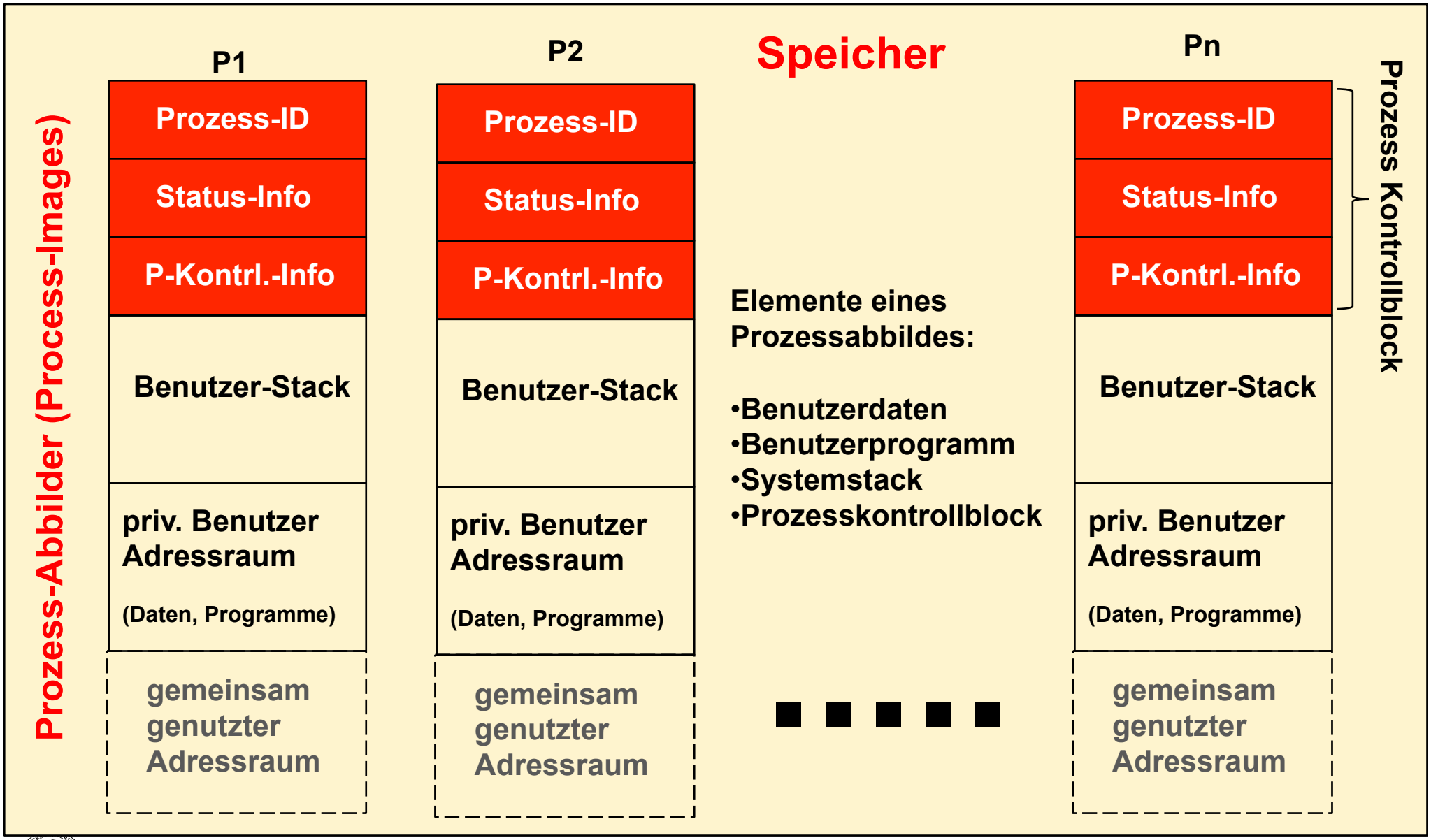
Verwaltungsdaten:
Priorität. Rechte, Profilingdaten.

Speicherverwaltung:

Prozess-Segmente:
(Programm-) Textsegment
Datensegment
Stacksegment

Dateiverwaltung:

Verzeichnisse,
Deskriptoren,
Zugriffsschutzinfo.


Prozess-Abbilder (Process-Images)
P1
P2
Speicher
Pn
Prozess-ID
Status-Info
P-Kontrl.-Info
Benutzer-Stack
priv. Benutzer Adressraum
(Daten, Programme)
gemeinsam genutzter Adressraum
Prozess Kontrollblock
Elemente eines Prozessabbildes:
•Benutzerdaten
•Benutzerprogramm
•Systemstack
•Prozesskontrollblock

Beschreibung von Prozessen (Prozesskontrollblock)

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 (PSW)*, z.B. Ausführungsmodus, Interruptsperren, ..
- Stackpointer (einer oder mehrere)

Prozessverwaltungsinformation:

- Scheduling- und Zustandsinformation
  - *Prozesszustand* (ready, active, waiting, suspended,...)
  - *Priorität*
  - *Schedulingparameter*, z.B. Wartezeit, Zeitquantum, Deadline,...
  - *Ereignis:* ID des Ereignisses, auf das der Prozess wartet

Beschreibung von Prozessen

Strukturierung der Prozessbeziehungen (Warteschlangen):

(grundsätzlich beliebige Strukturen möglich durch Verzeigerung)

z.B.:

- Verkettung wartender Prozesse einer Prioritätsebene
- Verkettung von Eltern-Kind-Beziehungen,

Interprozesskommunikation:

- z.B. Flags, Signale oder Nachrichten.

Prozessprivilegien:

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

Speicherverwaltung:

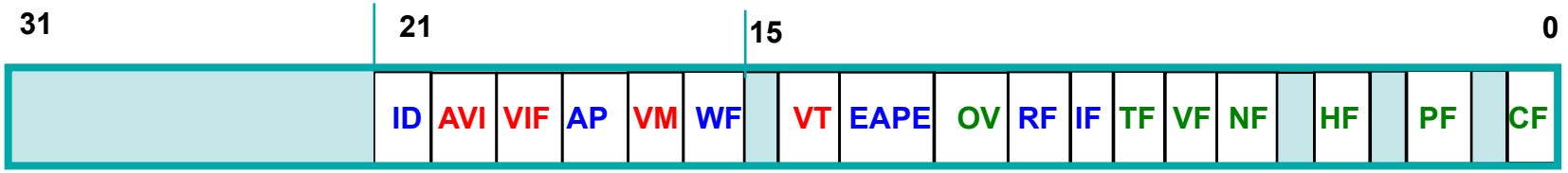
- Zeiger auf Segment- und Seitentabellen für den zugeteilten Speicher.

Ressourcenbesitz und Ressourcennutzung:

- z.B. geöffnete Dateien oder genutzte Geräte
- Profiling-Informationen

Beschreibung von Prozessen (PSW): Unterstützung durch die CPU

Repräsentation des CPU-Status im Pentium EFLAGS-Register:



ID: Identifikation, Information über CPUID-Befehl
AVI: Anliegender "virtueller Interrupt" im Zusammenhang mit dem 8086 Modus
VIF: Virtuelles 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 Ausnahmen 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

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

Benutzerebenenkontext

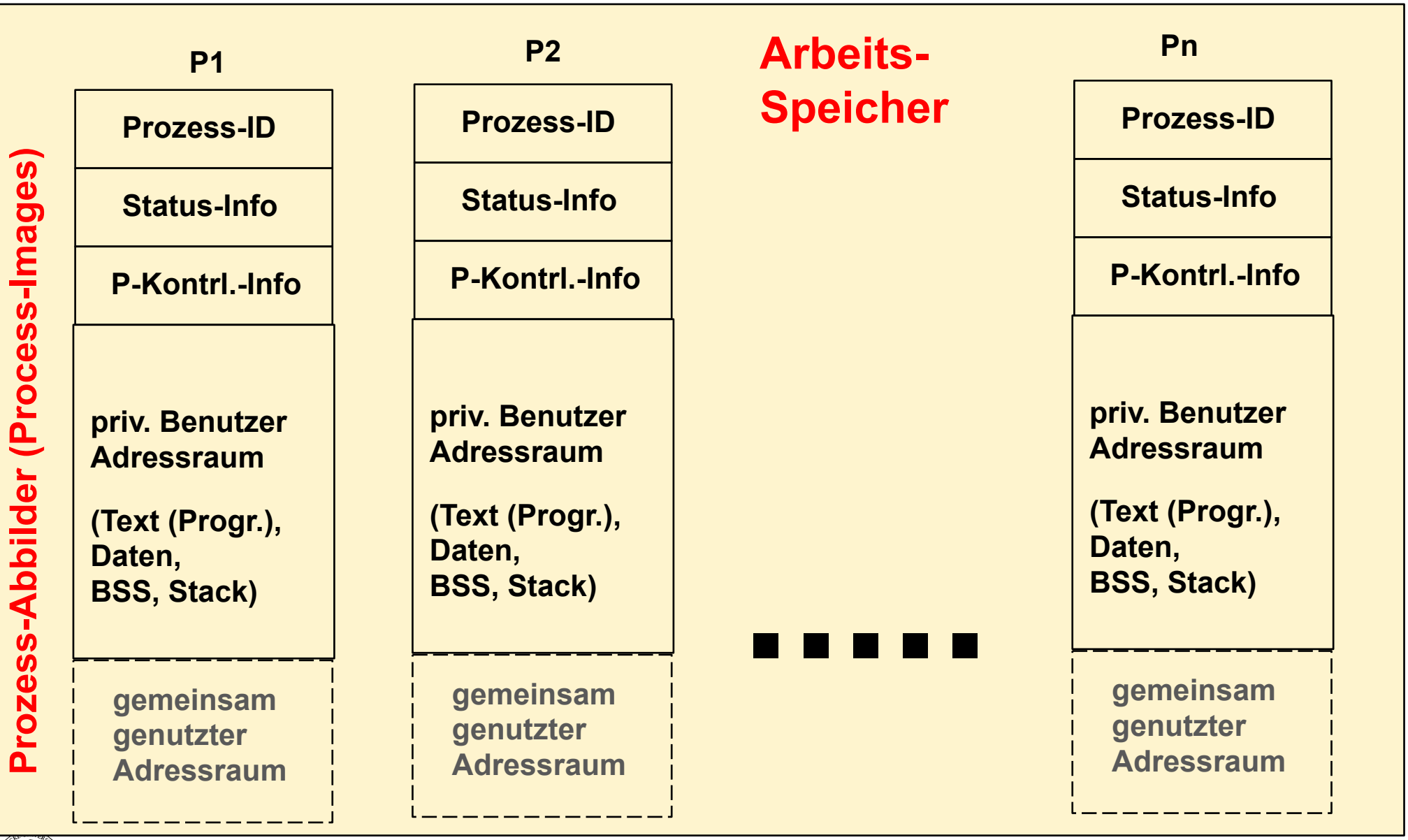
- Prozesstext (ausführbares Maschinenprogramm)
- Prozessdaten
- User Stack
- Gemeinsam genutzter Adressraum

Registerkontext

- Programmzähler
- Prozessor Status Register
- Stackpointer
- Allgemeine Register

Systemebenenkontext

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


Prozess-Abbilder (Process-Images)
P1
Prozess-ID
Status-Info
P-Kontrl.-Info
priv. Benutzer Adressraum
(Text (Progr.), Daten, BSS, Stack)
gemeinsam genutzter Adressraum
P2
Prozess-ID
Status-Info
P-Kontrl.-Info
priv. Benutzer Adressraum
(Text (Progr.), Daten, BSS, Stack)
gemeinsam genutzter Adressraum
Arbeits-Speicher
Pn
Prozess-ID
Status-Info
P-Kontrl.-Info
priv. Benutzer Adressraum
(Text (Progr.), Daten, BSS, Stack)
gemeinsam genutzter Adressraum

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

Wann?

Welche Ereignisse führen zu einem Prozesswechsel?

- **Expliziter Aufruf: fork() (Erzeugung eines Kind-Prozesses)**
- **Terminierung des Prozesses (Abgabe an Scheduler)**
- **Supervisor Aufruf z.B. bei E/A: - Prozess blockiert und wartet**
- **Interrupt: Zeit, E/A, Speicherfehler**
- **Trap: Fehler oder Ausnahmezustand**

Wie?

Prozess- oder Moduswechsel?

Moduswechsel bei Interrupt: Benutzer → Kern

- **nur Prozessor + Programm-Zustand muss abgespeichert werden!**

Prozesswechsel:

- **kompletter Prozess-Zustand wird abgesp.**

Prozesswechsel

Ursachen für einen Prozesswechsel:

- Terminierung
- Fork

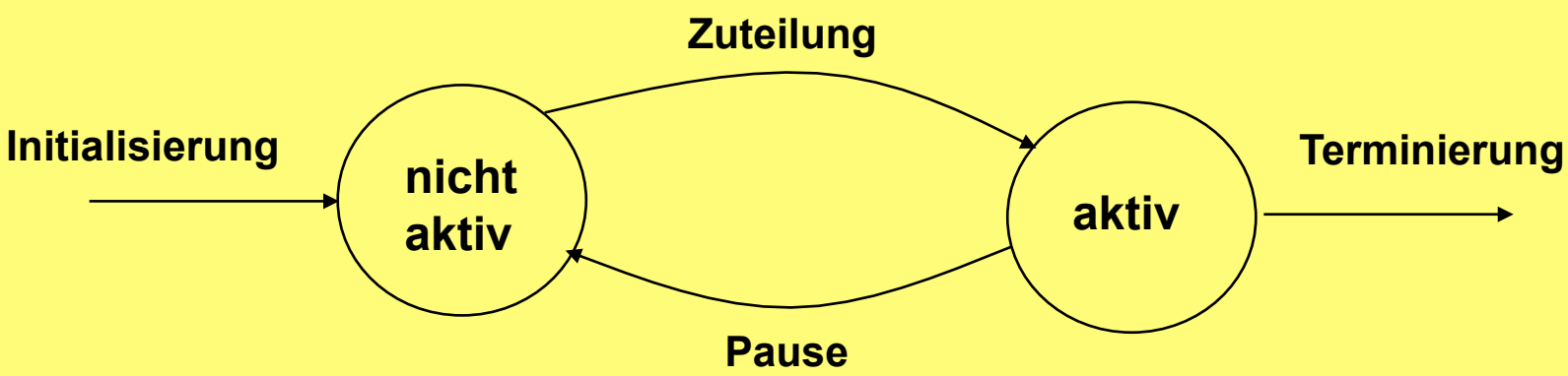
- Interrupt ➡ Zeit, E/A, Speicherfehler
- Trap ➡ ill. Op., ALU, Befehl, ...
- Supervisor Aufruf ➡ spez. Befehl

} Modus-wechsel

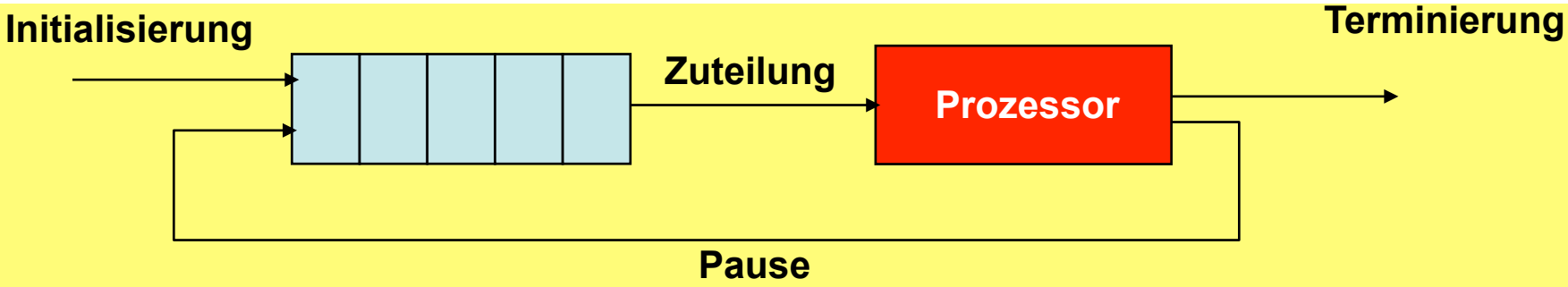
Aktionen bei einem Prozesswechsel:

- 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

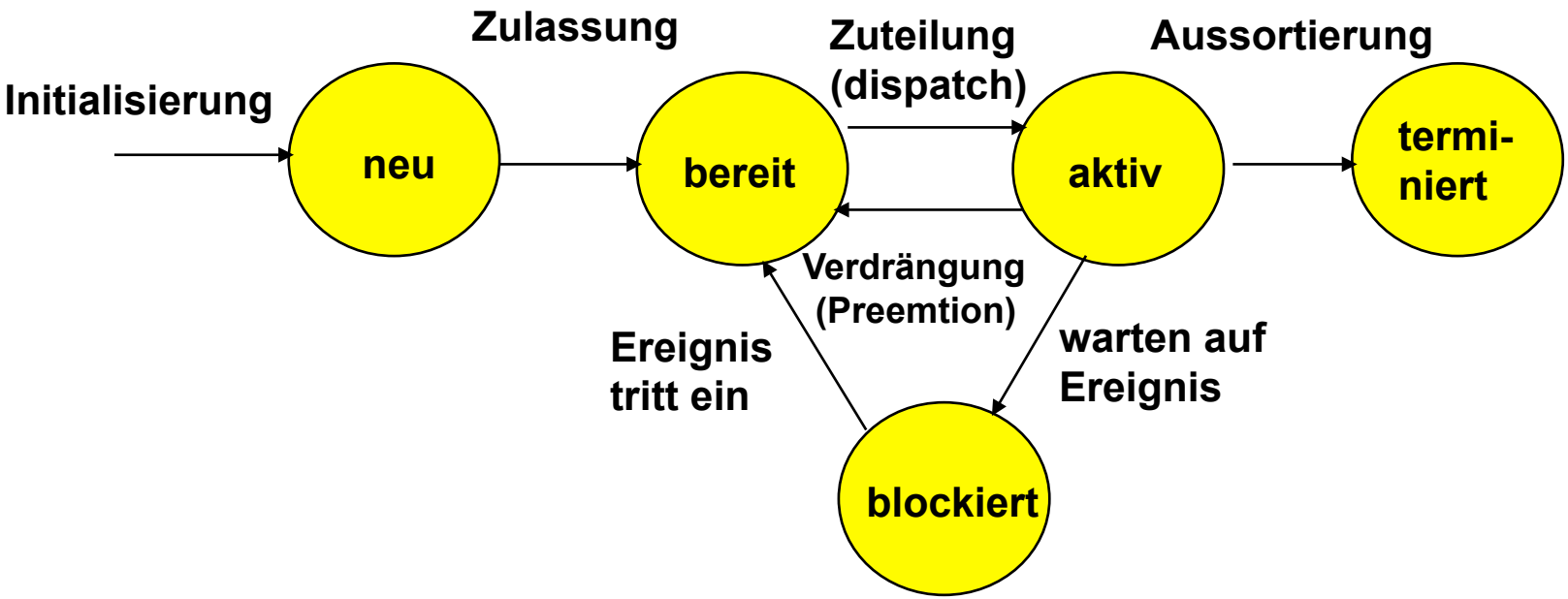


Zustandsdiagramm

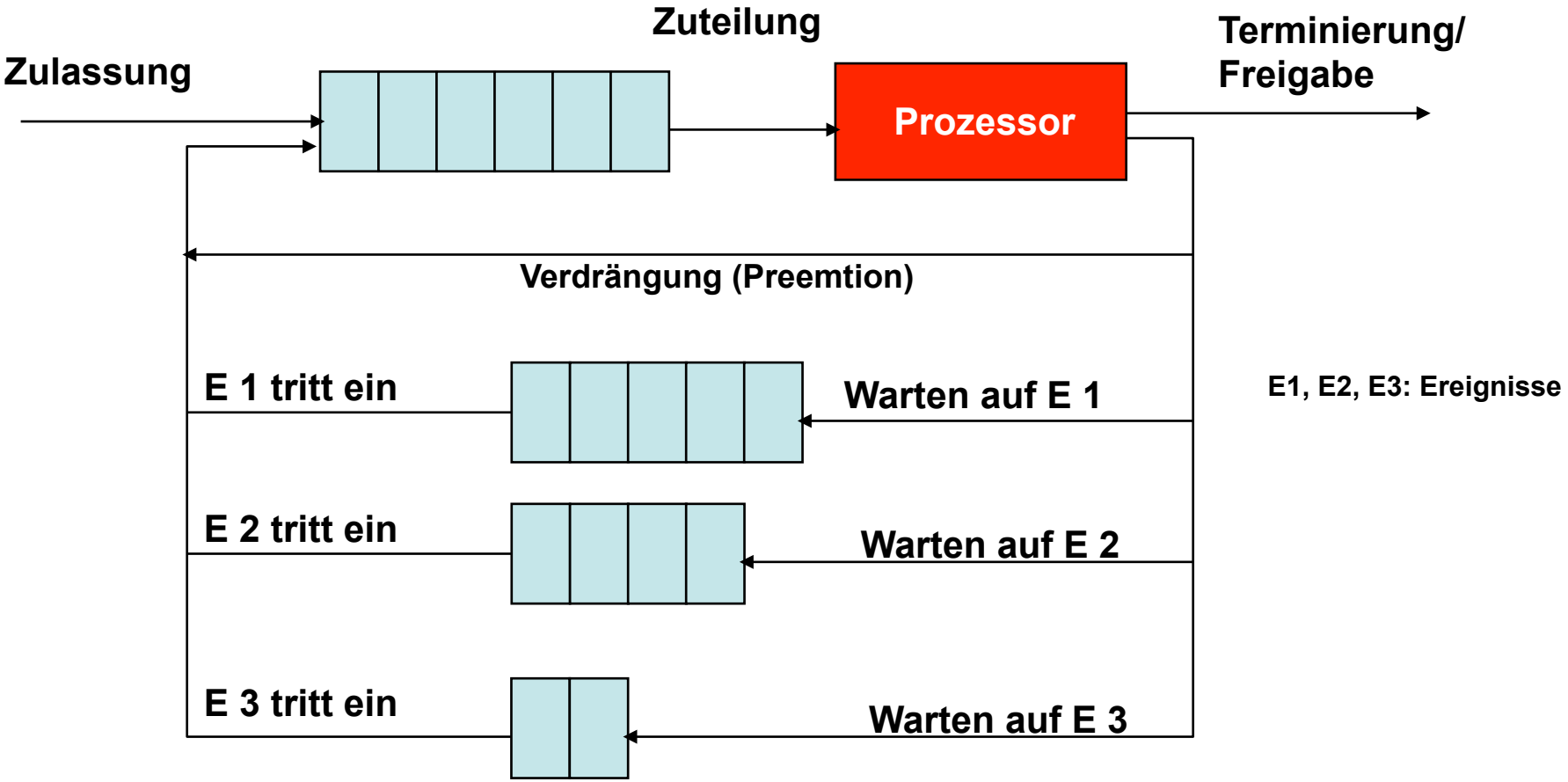


Warteschlangendiagramm

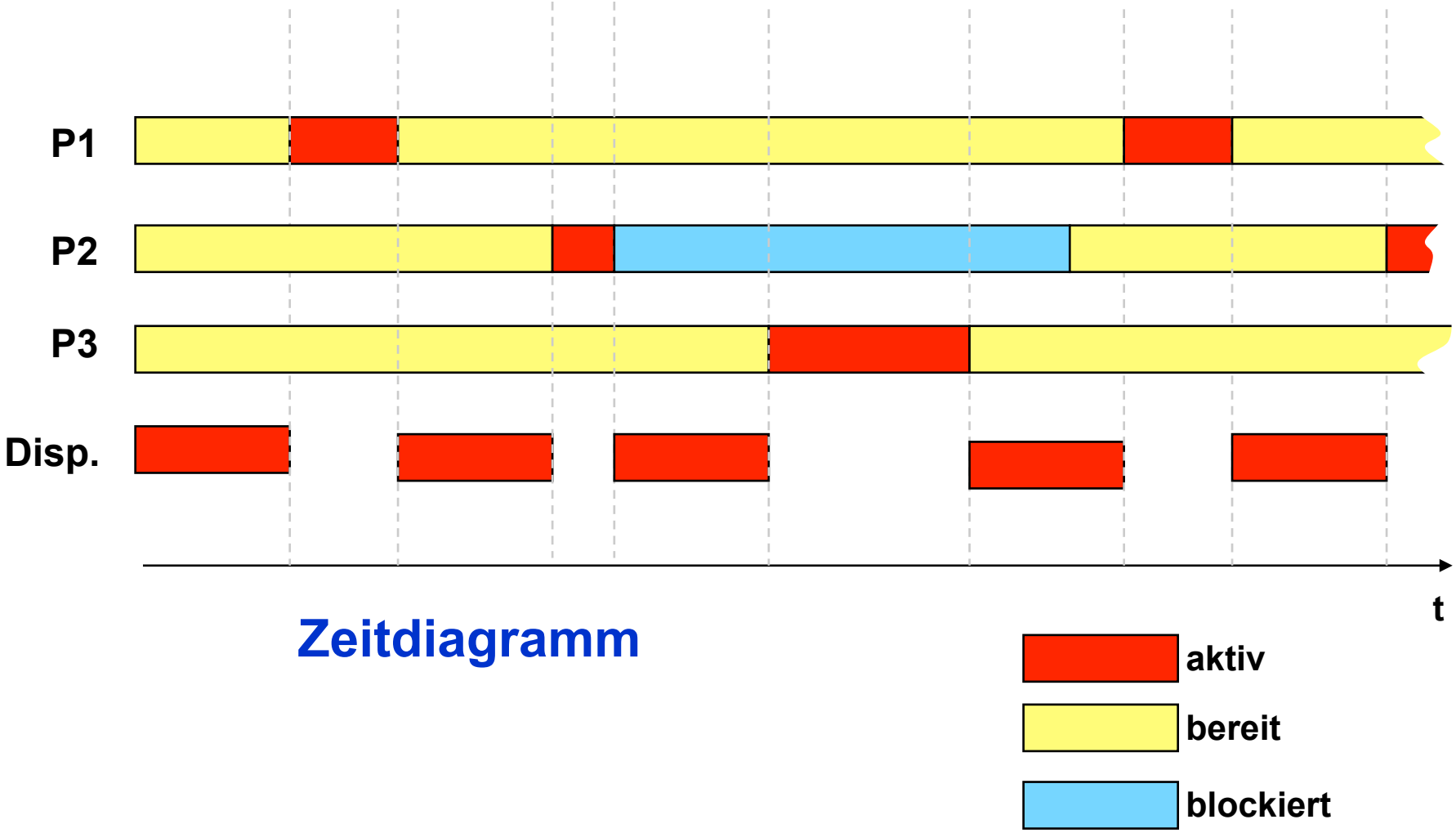
Prozessmodell und Prozesszustände



Prozessmodell und Prozesszustände



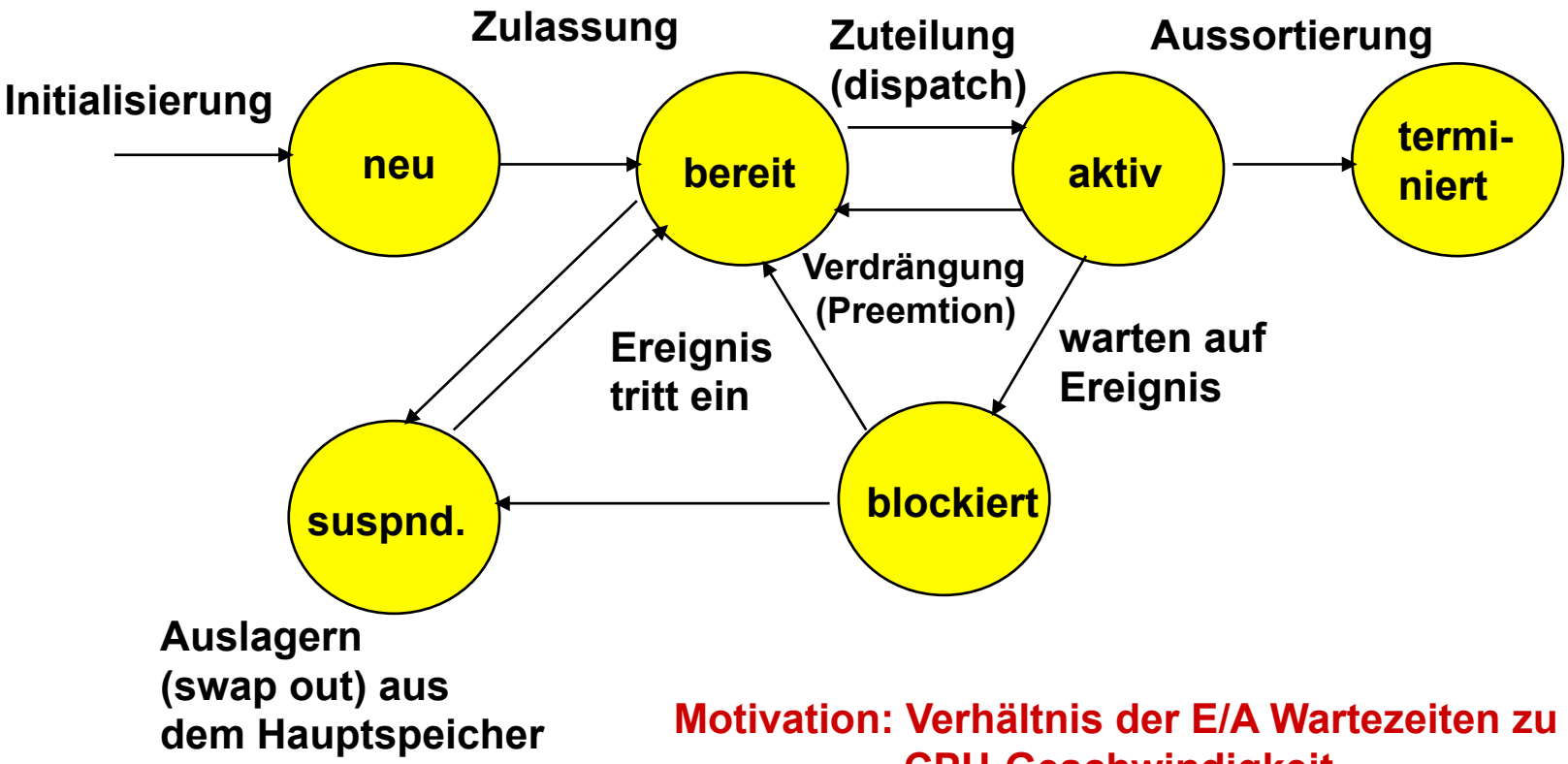
Prozessmodell und Prozesszustände



Zeitdiagramm

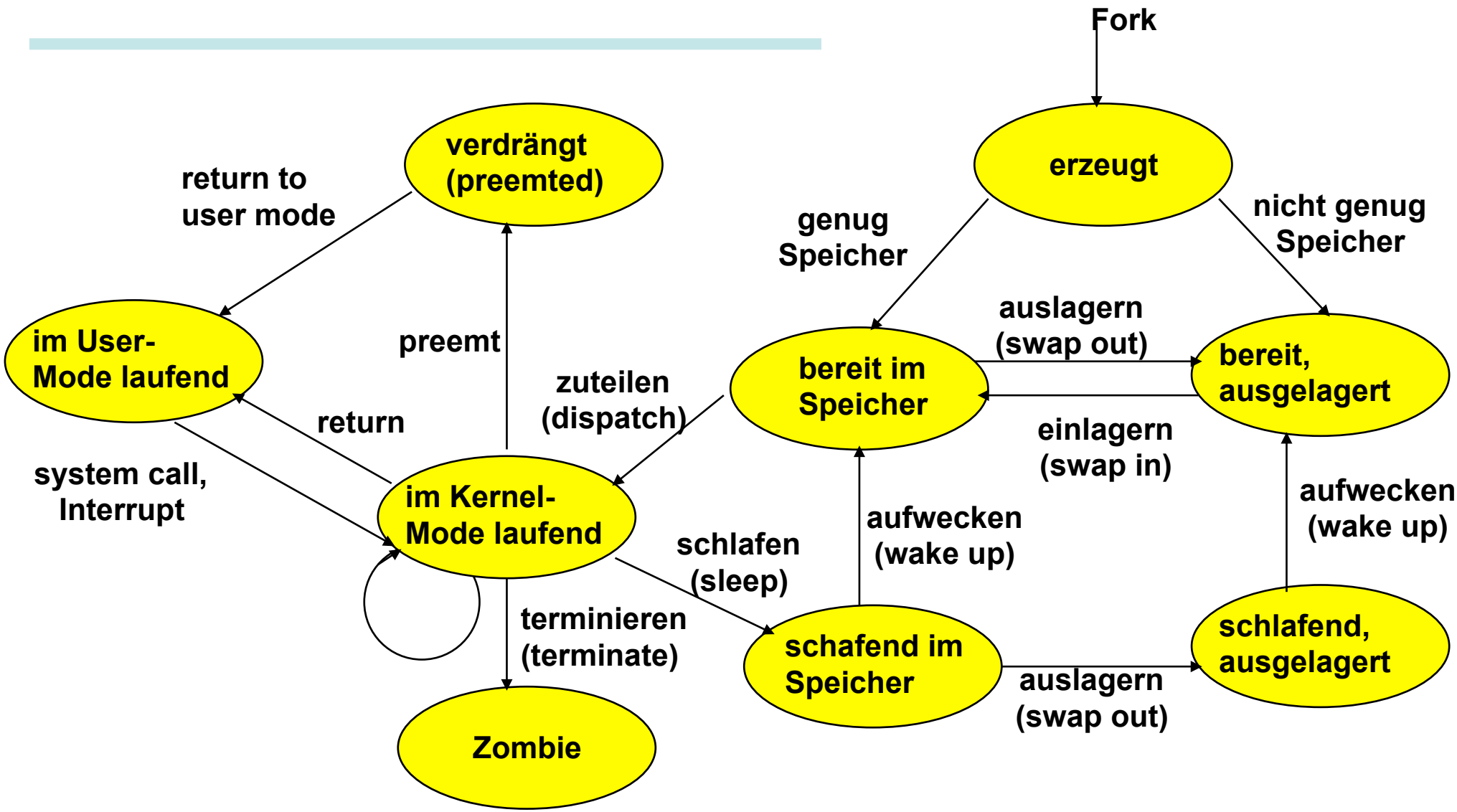
Prozessmodell und Prozesszustände

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"

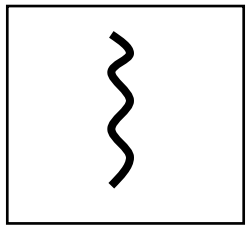
- Threads sind sequentielle Befehlsausführungen.
- Threads sind die Einheit für die Prozessorzuteilung.
- Threads laufen in einem Prozessadressraum ab.
- Threads werden auch als "leichtgewichtige Prozesse" bezeichnet.

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

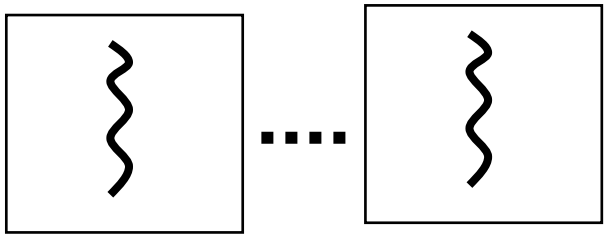
Threads und Prozesse

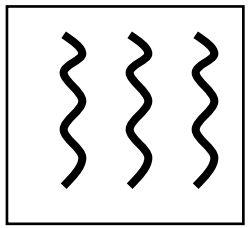
Ein Prozess

Mehrere Prozesse

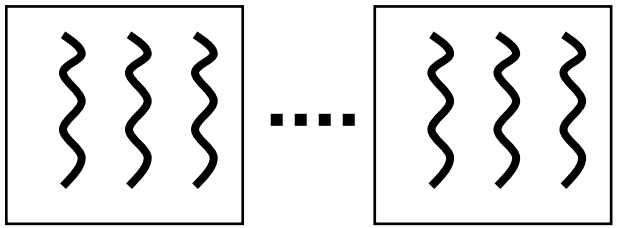


ein Thread/Prozess





mehrere Threads/Prozess



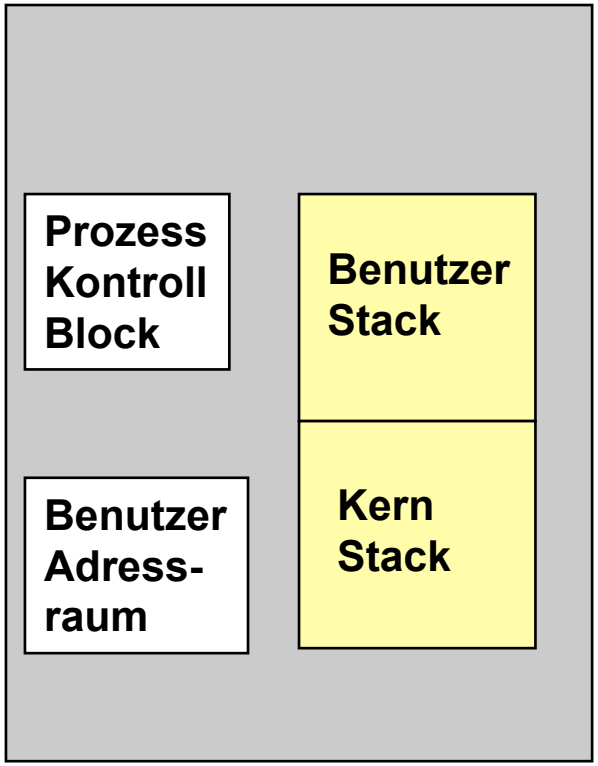
Threads und Prozesse

Thread besteht aus:

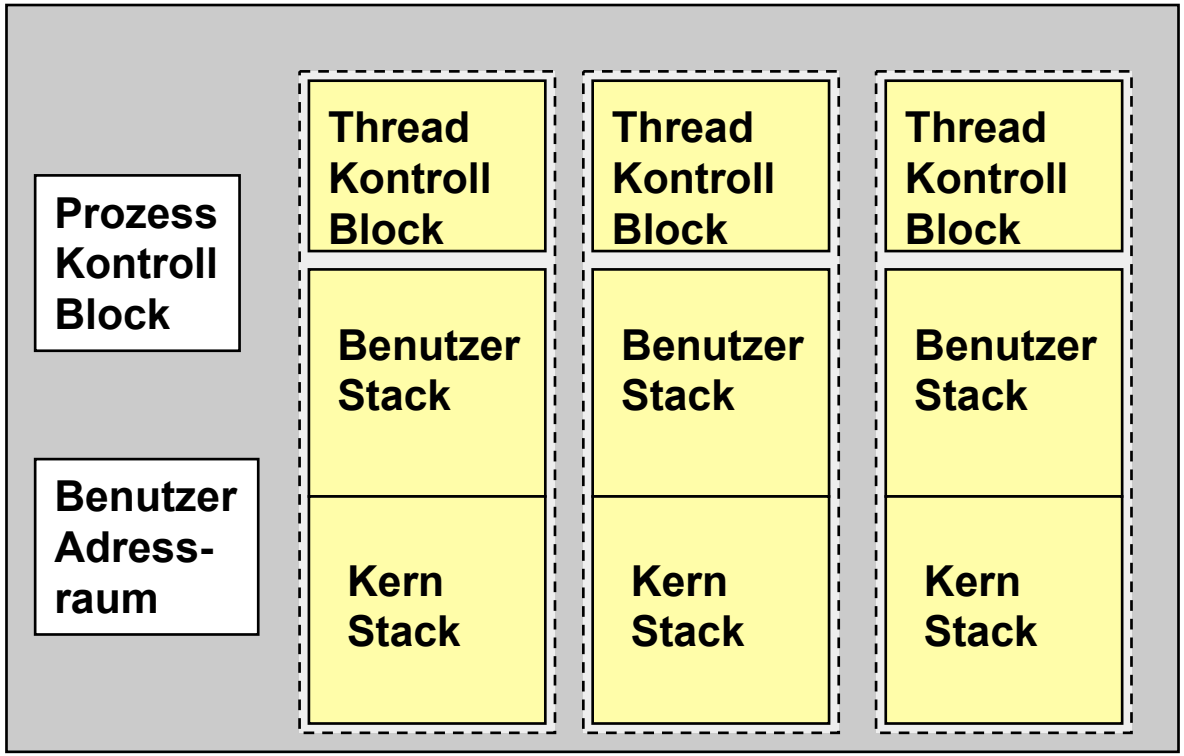
- **Ausführungszustand (bereit, aktiv, ...)**
- **Kontext**
- **Ausführungs-Stack**
- **Speicherplatz für Thread-lokale Variablen**

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

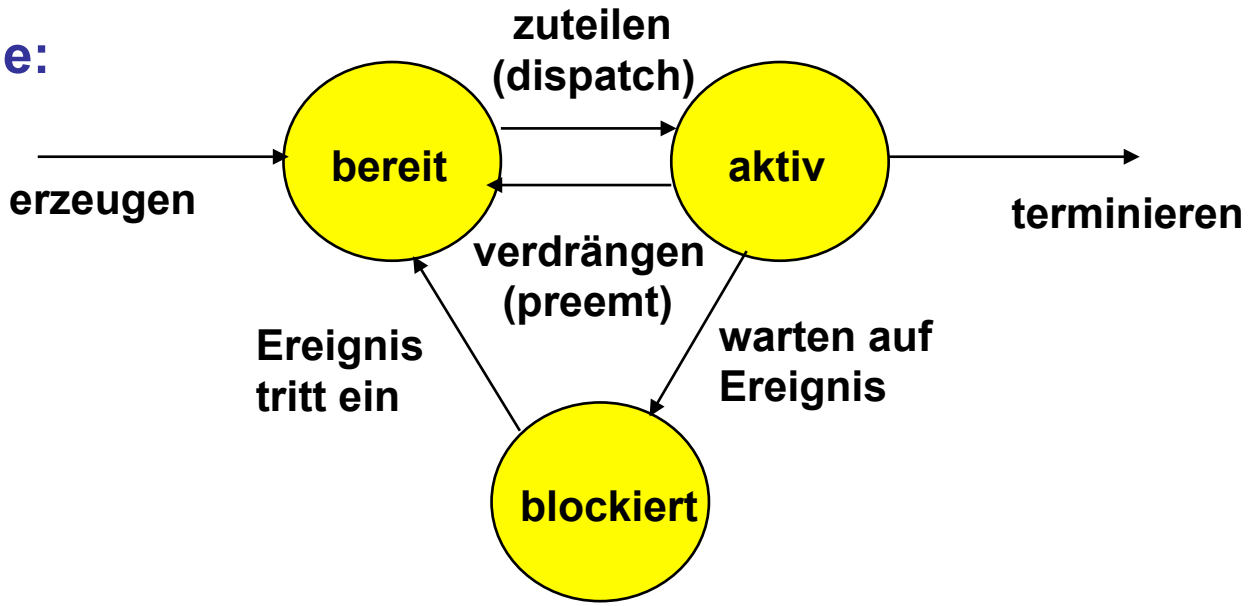
Vorteile:

- 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

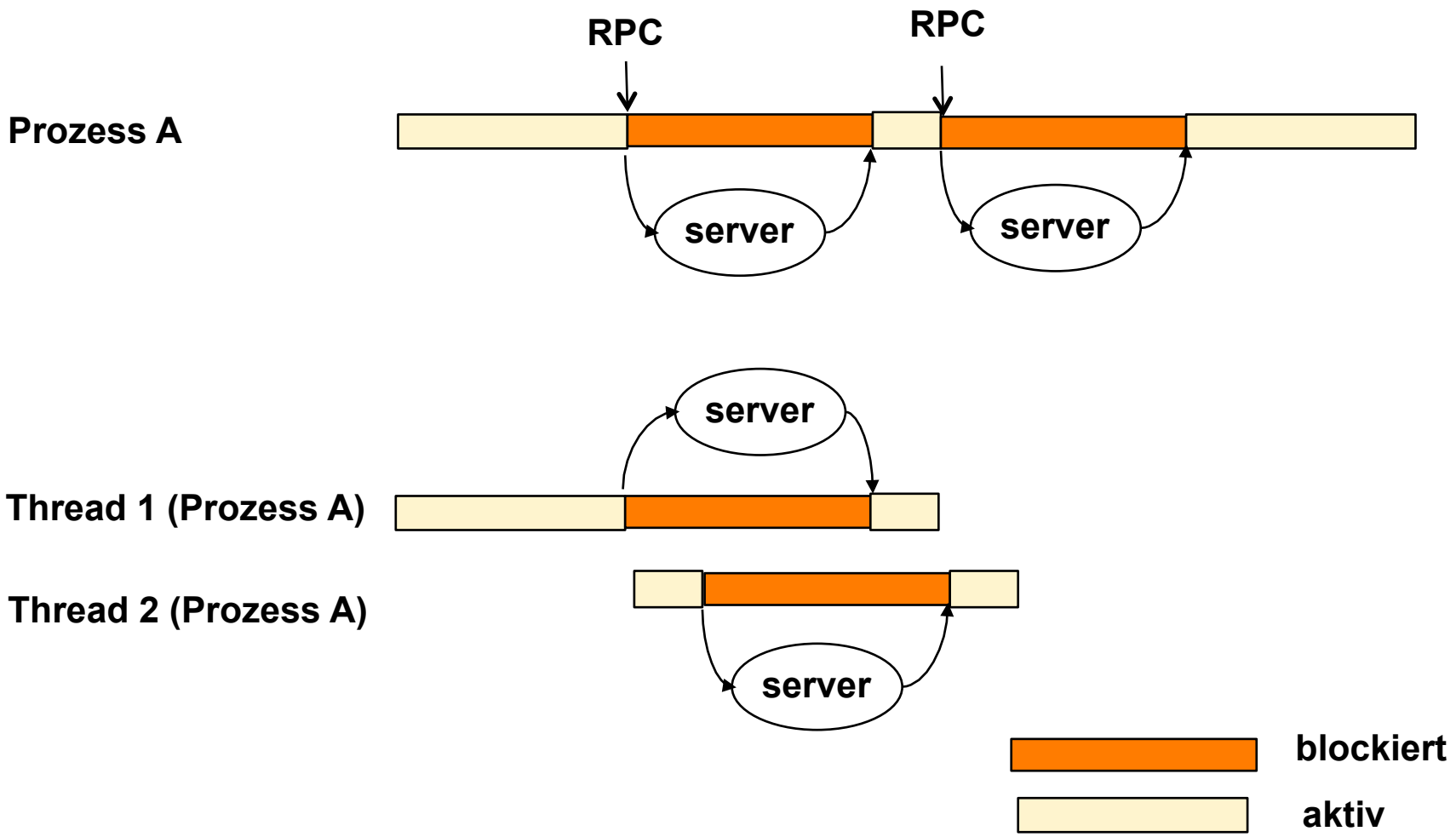
Thread Zustände:



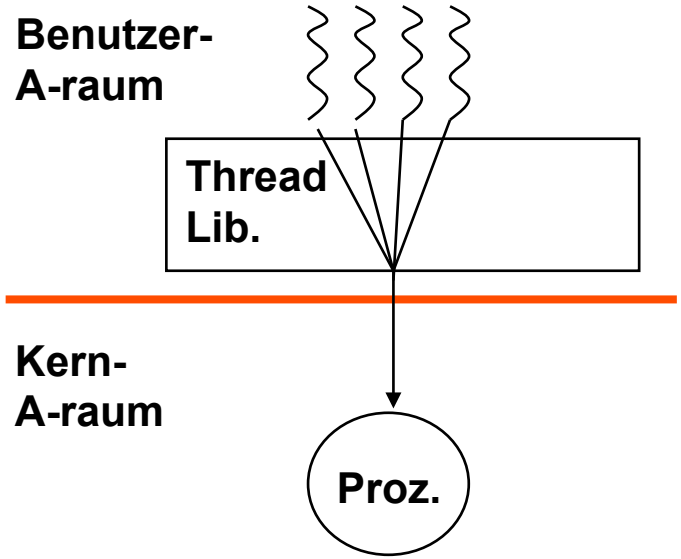
Thread Operationen:

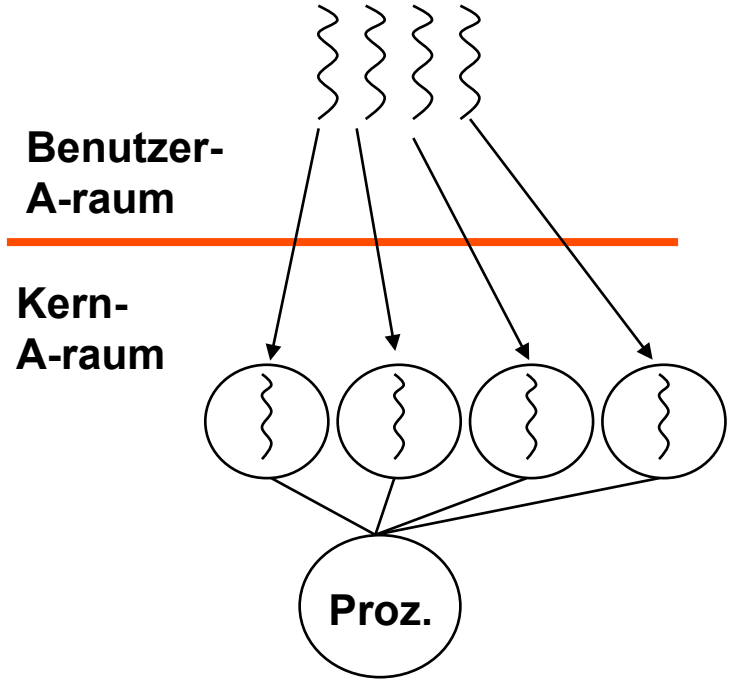
Erzeugen (Spawn)
Blockieren
Aufhebung der Blockierung
Terminierung

Geschwindigkeitssteigerung durch Treads



Kernel-Threads

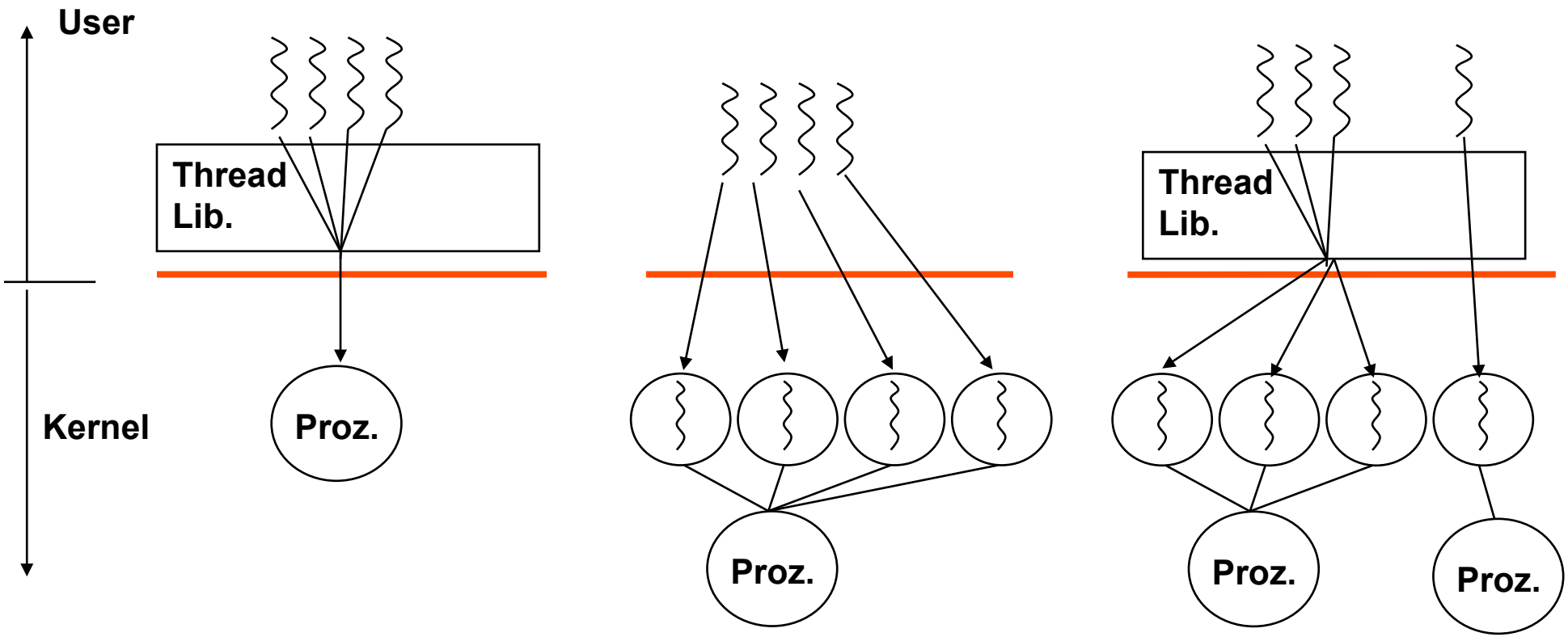




Threads und Prozesse

Thread:	**Einheit der Aktivität**
	Einheit des Schedulings
Prozess:	**Einheit der Ressourcenverwaltung**
Ressourcen:	**CPU-Zeit, Speicher, Dateien**

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.

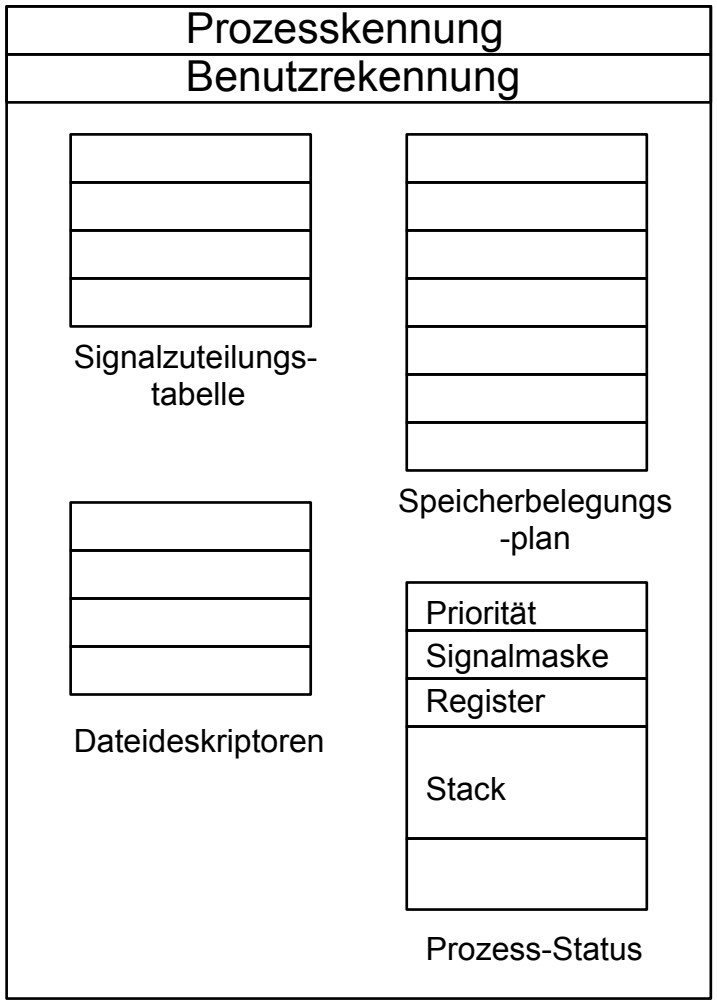
Con Benutzer-Threads:

1. Führt ein Benutzer-Thread einen blockierenden Systemaufruf durch, sind alle Threads blockiert.
2. Eingeschränkte Parallelarbeit.
3. Thread kann Prozessor monopolisieren.

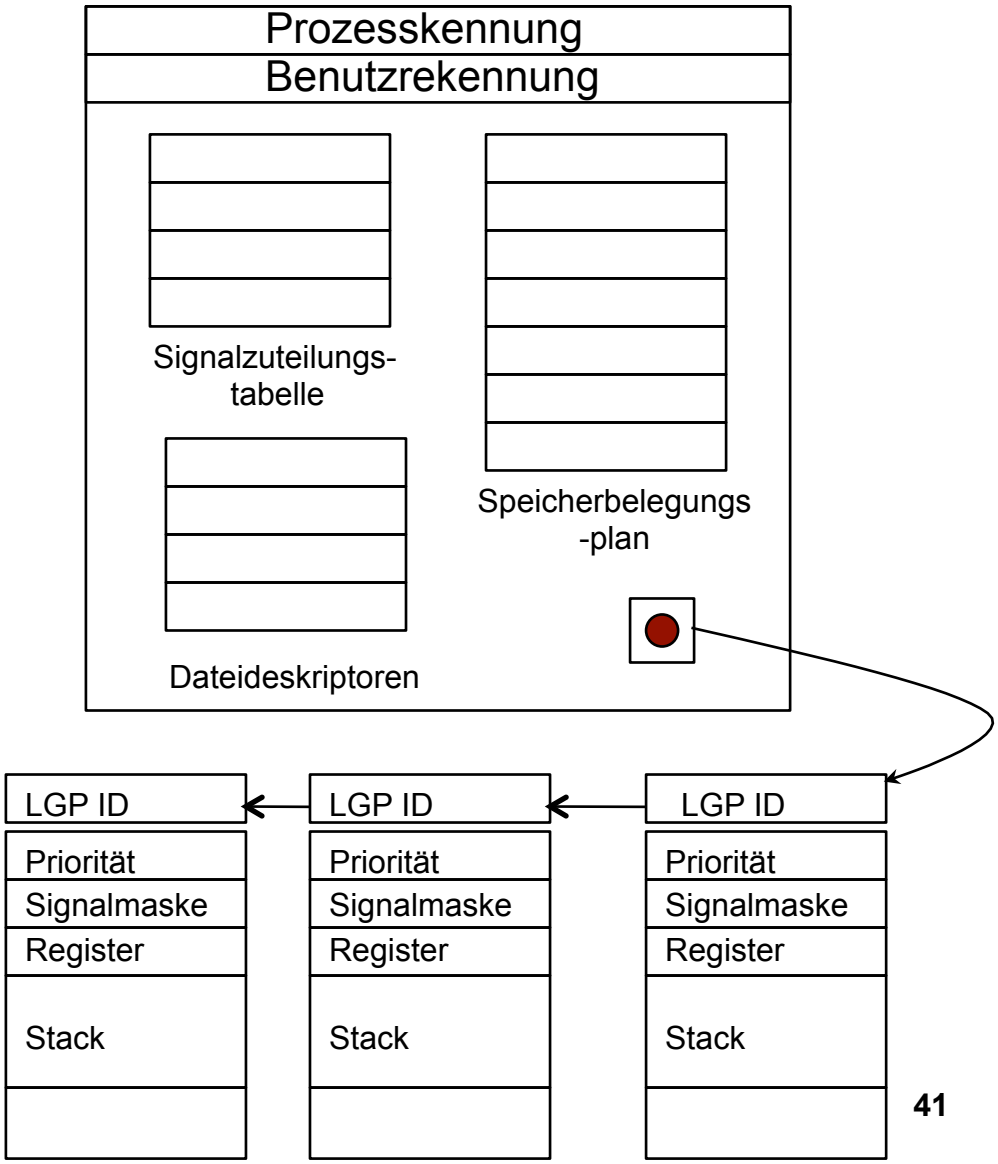
Verhältnis zw. Threads und Prozessen

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.**

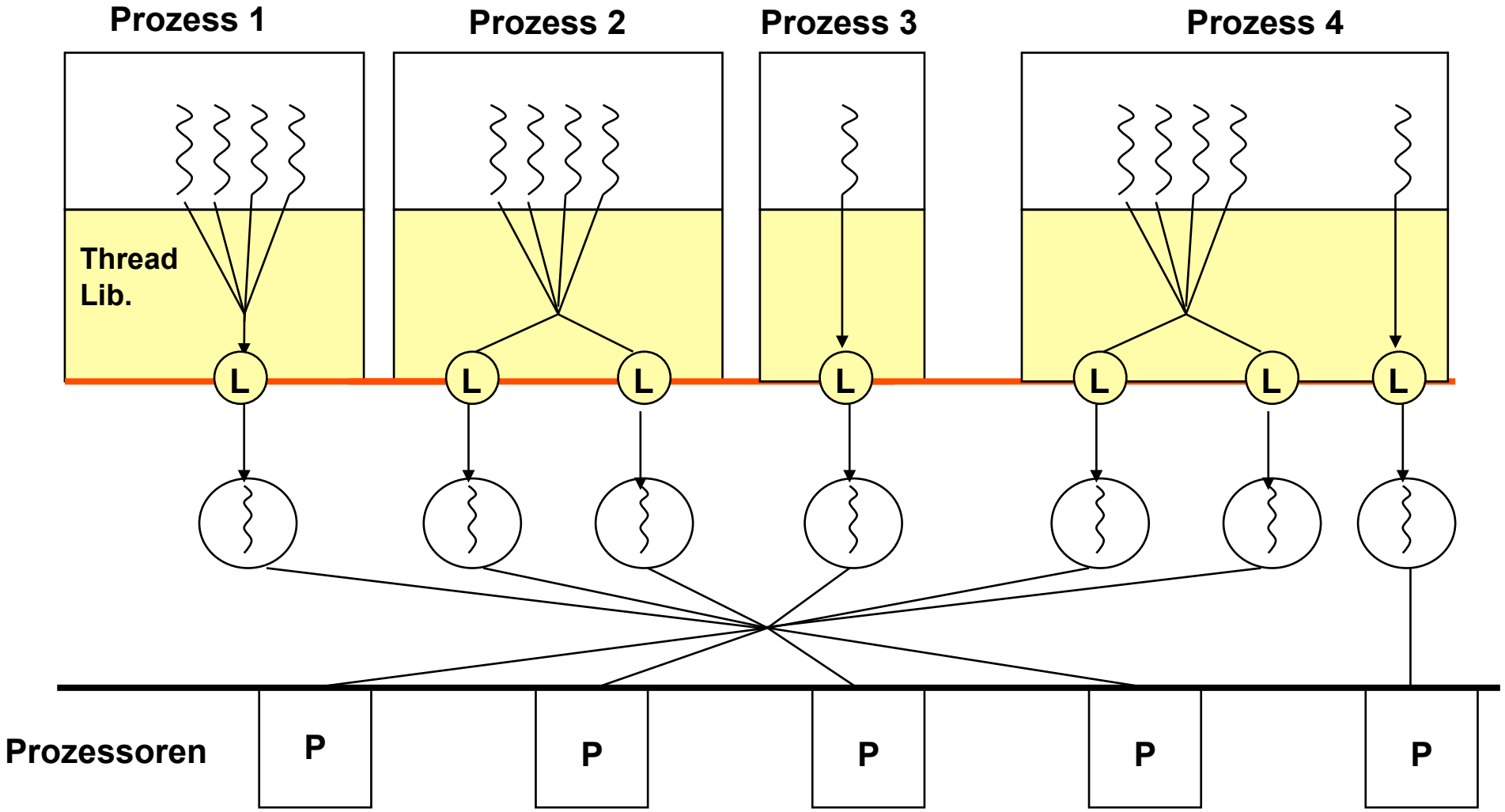
UNIX Prozessstruktur



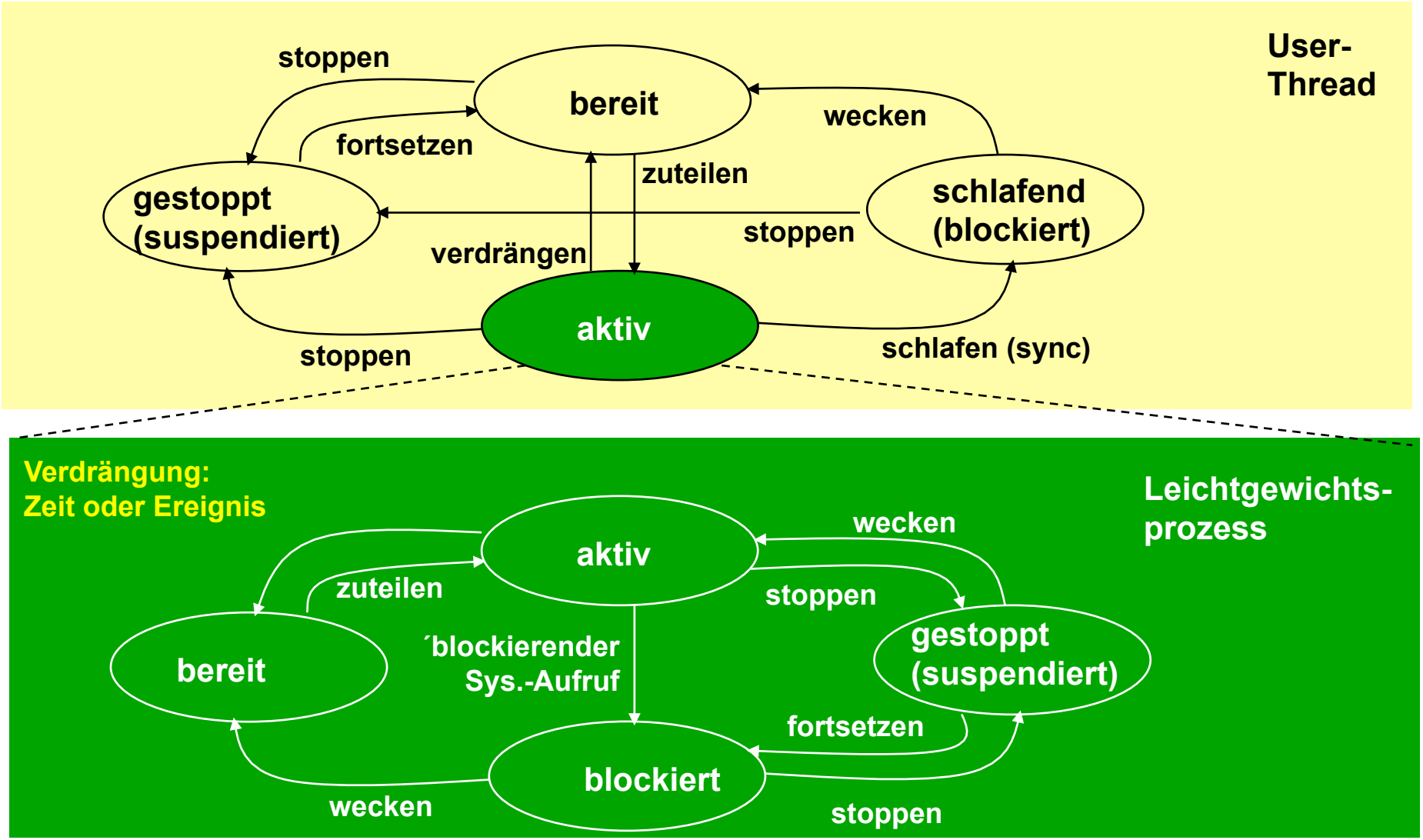
Solaris 2.x Prozessstruktur



Threads in Solaris



Threads und Leichtgewichtsprozesse



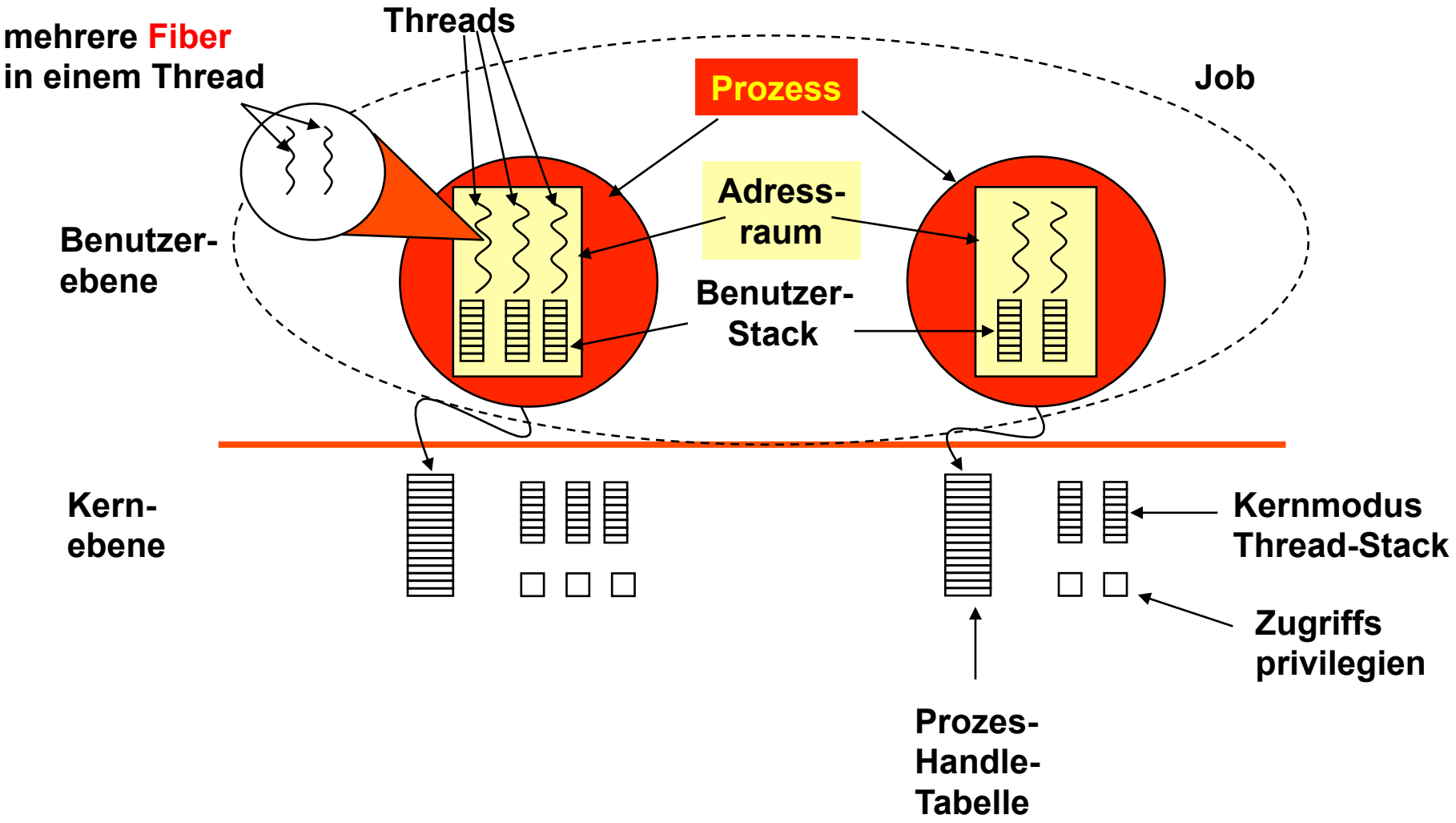
Threads in Solaris

Prozess:	**wie bisher, aber: mehrere Statusbeschreibungen, die auf Threads verweisen.**
Benutzer Thread:	**wie bisher**
Leichtgewichtige Prozesse (L):	**Bewirken die Zuordnung von Benutzer-Thread zu Kernel-Thread. Leichtgewichtsprozesse werden unabhängig 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.**

Threads in Windows 2K



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.

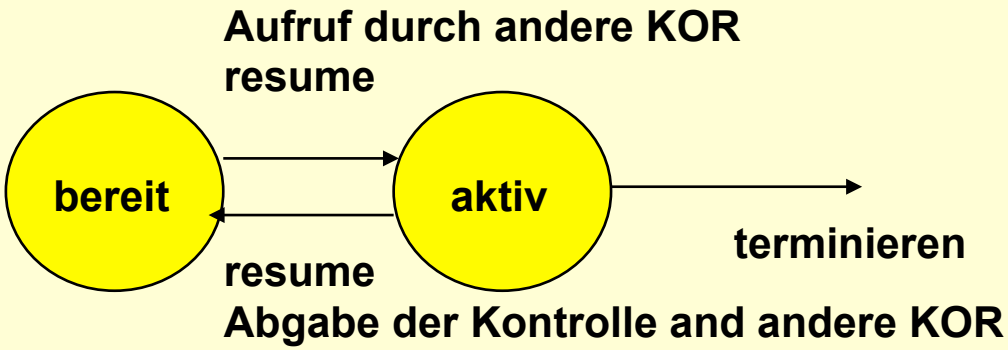
Koroutine: Synchroner Aufruf von Unterprogrammen.

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.

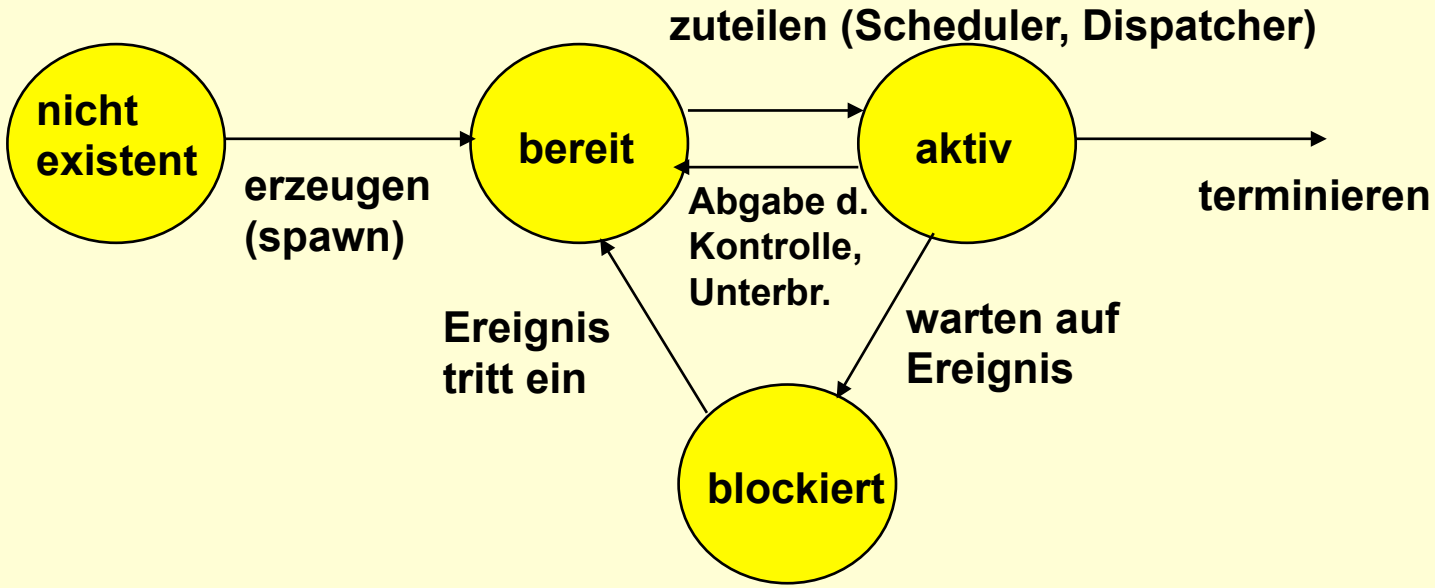
Threads: Asynchroner Aufruf. Unterschiede zur Koroutine:

1. Die Auswahl des Threads, an den die Kontrolle transferiert wird, übernimmt ein Scheduler, d.h. Thread hat eingeschränkten Einfluss auf Aufruffolge.
2. Ein Thread kann blockieren und auf ein Signal warten.
3. Ein Thread 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 (-zeit).

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 Ereignissen 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.

Anhang:

Unix Prozess Tabelleneinträge
Unix Prozess-Abbild

aus:
William Stallings: *Betriebssysteme, Prinzipien und Umsetzung*, 2. Auflage, Pearson Studium, 2003

Tabelle 3.11 UNIX-Prozesstabelleneintrag

Prozesszustand	Aktueller Zustand des Prozesses.
Zeiger	Mehrere Zeiger auf U-Area und den Prozessspeicherbereich (Text, Daten, Stapel).
Prozessgröße	Ermöglicht es dem Betriebssystem, zu erkennen, wie viel Platz dem Prozess zugewiesen werden muss.
Benutzer-kennungen	Die **reale Benutzerkennung** identifiziert den Benutzer, der für den aktiven Prozess verantwortlich ist. Die **effektive Benutzerkennung** kann von einem Prozess verwendet werden, um temporäre Privilegien, die mit einem bestimmten Programm in Zusammenhang stehen, zu erlangen. Während das Programm als Teil des Prozesses ausgeführt wird, arbeitet der Prozess mit der effektiven Benutzerkennung.
Prozesskennun-gen	Kennung (ID) des Prozesses. Kennung (ID) des Elternprozesses. Die Kennungen werden erstellt, wenn der Prozess während des Fork-Systemaufrufs in den Zustand »erzeugt« wechselt.
Ereignis-beschreibung	Gilt, wenn sich ein Prozess im schlafenden Zustand befindet. Wenn das Ereignis eintritt, wird der Prozess in einen Zustand der Laufbereitschaft versetzt.
Priorität	Wird für die Prozessablaufplanung verwendet.
Signal	Aufzählung von Signalen, die an einen Prozess geschickt, aber noch nicht bearbeitet wurden.
Timer	Beinhaltet die Prozessausführungszeit, die Kernel-Ressourcennutzung und benutzerdefinierte Timer, die dazu verwendet werden, Alarmsignale an einen Prozess zu schicken.
P_link	Zeiger auf das Folgeelement in der Warteschlange für bereite Prozesse (gültig, wenn der Prozess bereit für die Ausführung ist).
Speicherstatus	Zeigt an, ob sich das Prozessabbild im Hauptspeicher befindet oder ausgelagert wurde. Wenn es sich im Speicher befindet, zeigt dieses Feld außerdem an, ob es ausgelagert werden kann oder temporär im Hauptspeicher bleiben muss, weil ein Lock gesetzt ist.

Unix Process Table Entry

Tabelle 3.11 UNIX-Prozesstabelleneintrag

Prozesstabellenzeiger	Zeigt den Eintrag an, der der U-Area entspricht.
Benutzerkennungen	Reale und effektive Benutzerkennungen. Werden verwendet, um Benutzerprivilegien festzulegen.
Timer	Zeichnen die Zeit auf, die der Prozess (und seine Kindprozesse) mit der Ausführung im Benutzermodus und im Kernel-Modus verbracht hat.
Signalverarbeitungs-Array	Für jeden im System festgelegten Signaltyp. Zeigt an, wie der Prozess auf den Erhalt des entsprechenden Signals reagieren wird (Terminieren, Ignorieren, Ausführen einer bestimmten Benutzerfunktion).
Kontrollterminals	Zeigt das Login-Terminal für den Prozess an, falls ein solches existiert.
Fehlerfeld	Zeichnet Fehler auf, die während eines Systemaufrufs auftreten.
Rückgabewert	Enthält das Ergebnis von Systemaufrufen.
E/A-Parameter	Beschreiben den Umfang der zu übertragenden Daten, die Adresse des Quell- oder Zieldaten-Arrays im Benutzerraum und Datei-Offsets für die E/A.
Dateiparameter	Das aktuelle Verzeichnis und die aktuelle Wurzel beschreiben die Dateisystemumgebung des Prozesses.
Benutzerdatei-beschreibungstabelle	In dieser Tabelle sind die vom Prozess geöffneten Dateien aufgezeichnet.
Begrenzungsfelder	Schränken die Größe des Prozesses und die Größe einer Datei, in die er schreiben kann, ein.
Erlaubnismodusfelder	Maskieren Moduseinstellungen für Dateien, die der Prozess erzeugt.

Unix U-Area

aus: William Stallings:*Betriebssysteme, Prinzipien und Umsetzung*, 2. Auflage, Pearson Studium, 2003

aus: William Stallings:
Betriebssysteme, Prinzipien und Umsetzung,
2. Auflage, Pearson Studium, 2003

Unix Process Image

Tabelle 3.10 UNIX-Prozessabbild

	Benutzerebenenkontext
Prozesstext	Ausführbare Maschinenbefehle des Programms.
Prozessdaten	Daten, auf die durch das Programm dieses Prozesses zugegriffen werden kann.
Benutzerstapel	Enthält Argumente, lokale Variablen und Zeiger für Funktionen, die im Benutzermodus ausgeführt werden.
Gemeinsam genutzter Speicher	Speicher, der gemeinsam mit anderen Prozessen genutzt wird. Wird für die Interprozesskommunikation verwendet.
	Registerkontext
Programmzähler	Adresse des als Nächstes auszuführenden Befehls. Kann sich im Kernel- oder Benutzerspeicherraum des Prozesses befinden.
Prozessorstatusregister	Enthält den Hardwarestatus zum Zeitpunkt der Verdrängung. Inhalte und Format sind abhängig von der Hardware.
Stapelzeiger	Zeigt abhängig vom Betriebsmodus zum Zeitpunkt der Verdrängung auf das obere Ende des Kernel- oder Benutzerstapels.
Register für allgemeine Zwecke	Abhängig von der Hardware.
	Systemebenenkontext
Prozesstabelleneintrag	Legt den Status eines Prozesses fest. Das Betriebssystem kann jederzeit auf diese Informationen zugreifen.
Benutzerbereich	Prozesskontrollinformationen, auf die nur im Kontext des Prozesses zugegriffen werden muss.
Prozessbereichstabelle	Legt die Zuordnung von virtuellen zu physikalischen Adressen fest. Enthält außerdem ein Erlaubnisfeld, das den für den Prozess zulässigen Zugriffstyp anzeigt: nur Lesen, Lesen und Schreiben oder Lesen und Ausführen.
Kernel-Stapel	Enthält den Stapelrahmen von Kernel-Prozeduren, wenn der Prozess im Kernel-Modus läuft.

Beziehung zwischen Threads und Prozess