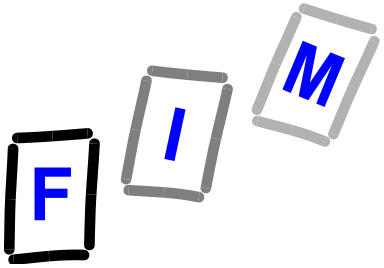
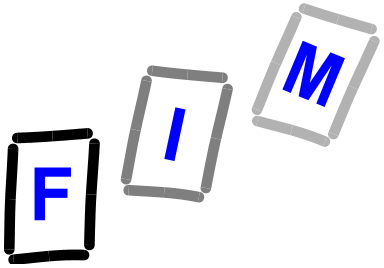


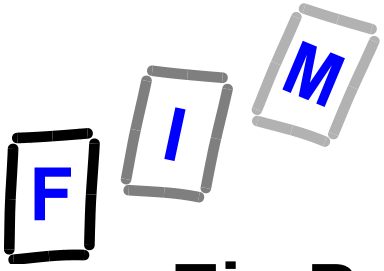
Betriebssysteme

Kapitel E : Prozesse



Inhalt

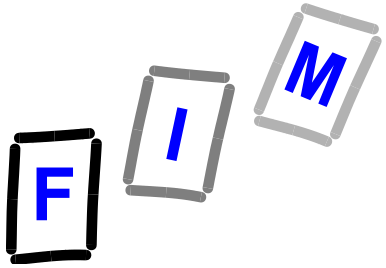
- Prozesse
  - ➔ Zustand eines Prozesses
    - » Kontext
    - » Kontextswitch
  - ➔ Prozessbeschreibungsblock PCB
  - ➔ Zustandsübergänge
    - » Zustandsdiagramm



Hinweis

- **Ein Programm(code) kann zur Laufzeit mit mehreren Prozessen assoziiert werden**
  - ➔ **Z.B. weil Quellcode eine Folge von Prozessen vorsieht, oder ein Programm mehrfach gestartet wird**
  - ➔ **Zur Vereinfachung: Beschränkung auf Fall 1:1**

- **Prozesse und / versus *leichtgewichtige Prozesse* = Threads**
  - ➔ **Details, warum *„light weight“,* später**
  - ➔ **Vernachlässigen vorerst, dass Prozesse in Threads aufgespaltet gedacht sein können**

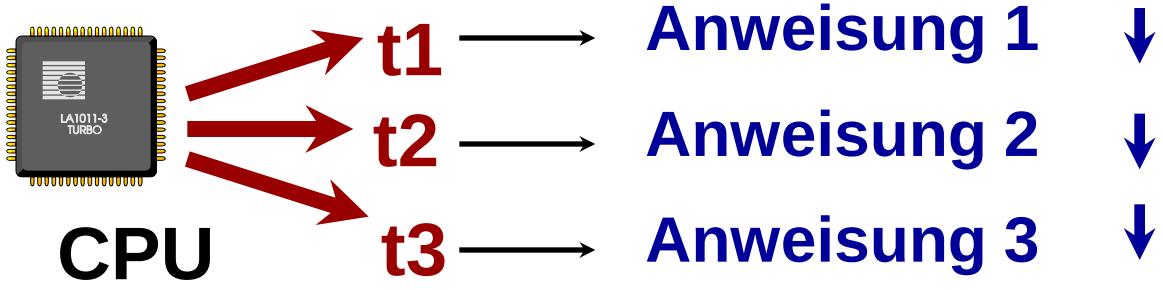


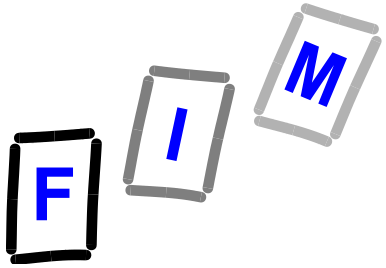
Prozess
Bündel von Threads

- Die Ausführung eines Programms erzeugt einen Prozess
  - ➔Generell: Ein Bündel von Prozessen
    - » Meist gesehen als: Threads
  - ➔Hier vereinfacht: Nur ein Prozess
    - » Entspricht: Prozessbündel enthält nur 1 Thread
- 1 Programm = 1 … N Prozesse
- 1 Prozess = 1 … N Threads

Sequentieller Prozess

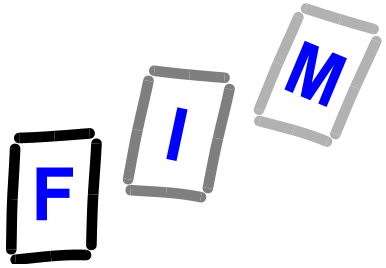
- ***Sequentielle* Prozesse:**
  Zu jedem Zeitpunkt t wird genau eine Anweisung (Instruktion) ausgeführt
  - ➔ **Was eine Anweisung ist, hängt vom Grad der Abstraktion ab**





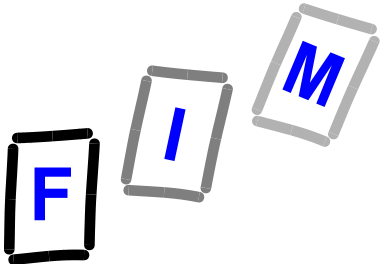
Prozess - Task

- **Manchmal wird statt „Prozess“ auch „Task“ verwendet.**
  - **➔Englisch: task::= job of work, mission**
  - **➔insofern: Task eher saloppe Wortwahl**
- **Ist in der Literatur nicht exakt definiert:**
  - **➔Oft: Task = Prozess =**
    = Bündel von Threads
    + beteiligte Ressourcen



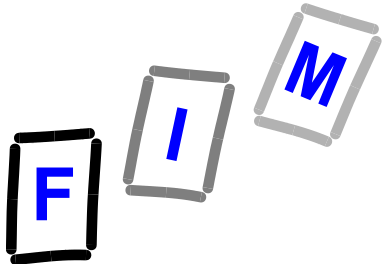
Kontext für einen Prozess P

- **Alle von P zum Zeitpunkt t_i belegten Betriebsmittel bestimmen seinen *Zustand,***
- **und bilden den *Kontext von P* (zum Zeitpunkt t_i)**
  - ➔ **CPU mit Registerinhalten (insb. PC!)**
  - ➔ **Zugewiesener Speicher (samt Inhalt)**
  - ➔ **Zeiger auf Warteschlangen, Files (+Speicher)**
  - ➔ **Betriebssystem-Ressourcen (zB Fenster)**
  - ➔ **Prozessnummer**
  - ➔ **Priorität und sonstige Schedulinginformation**
  - ➔ **……**



Kontextswitch

- **Umschalten von P1 auf P2 bewirkt Kontextswitch**
  - **➔Entzug der CPU, Retten der Registerinhalte**
    - **» Überlegen Sie: Warum nicht auch CPU-Cache?**
  - **➔Zuteilung der CPU an P2**
- **Offensichtlich:**
  - **➔Zustand eines Prozesses muss gerettet werden**
  - **➔Vgl. Situation beim Interrupt**



Prozess-Beschreibungsblock

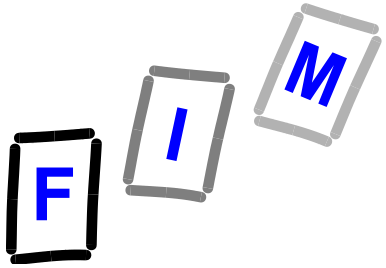
Jeder Prozess wird im Betriebssystem durch einen

Prozessbeschreibungsblock (process control block, PCB)

auch genannt

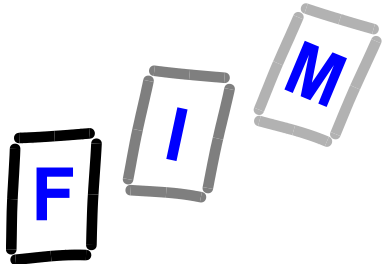
task control block, TCB

dargestellt.



Zweck des PCB

- **PCB und die enthaltene Prozessnummer (proc_ID) identifizieren einen Prozess P**
- **Der PCB (TCB) dient zum Speichern aller veränderlichen Informationen über einen Prozess**
  - **➔Beschreibt wesentliche Teile des Kontexts von P**
- **Pro Prozess existiert genau ein PCB**



Inhalt Prozess-Beschreibungsblock

- **Der Aufbau des Prozessbeschreibungs-blockes (PCB) ist i.A. von Betriebssystem zu Betriebssystem unterschiedlich**
- **Auf der nächsten Folie werden daher nur beispielhaft eine Struktur und Inhalte gezeigt, die typisch sind**

Beispiel für Struktur des PCB

Zeiger auf nächsten / vorigen PCB
Prozessnummer / Prozess-ID
Prozesszustand
Programmzähler + Registerinhalte
Priorität und Schedulinginformation
Zeiger für Warteschlangen
Speichergrenzen
Sonstige prozessspezifische Informationen wie Liste der offenen Dateien, Accounting-Infos, ...

Beispiel Linux

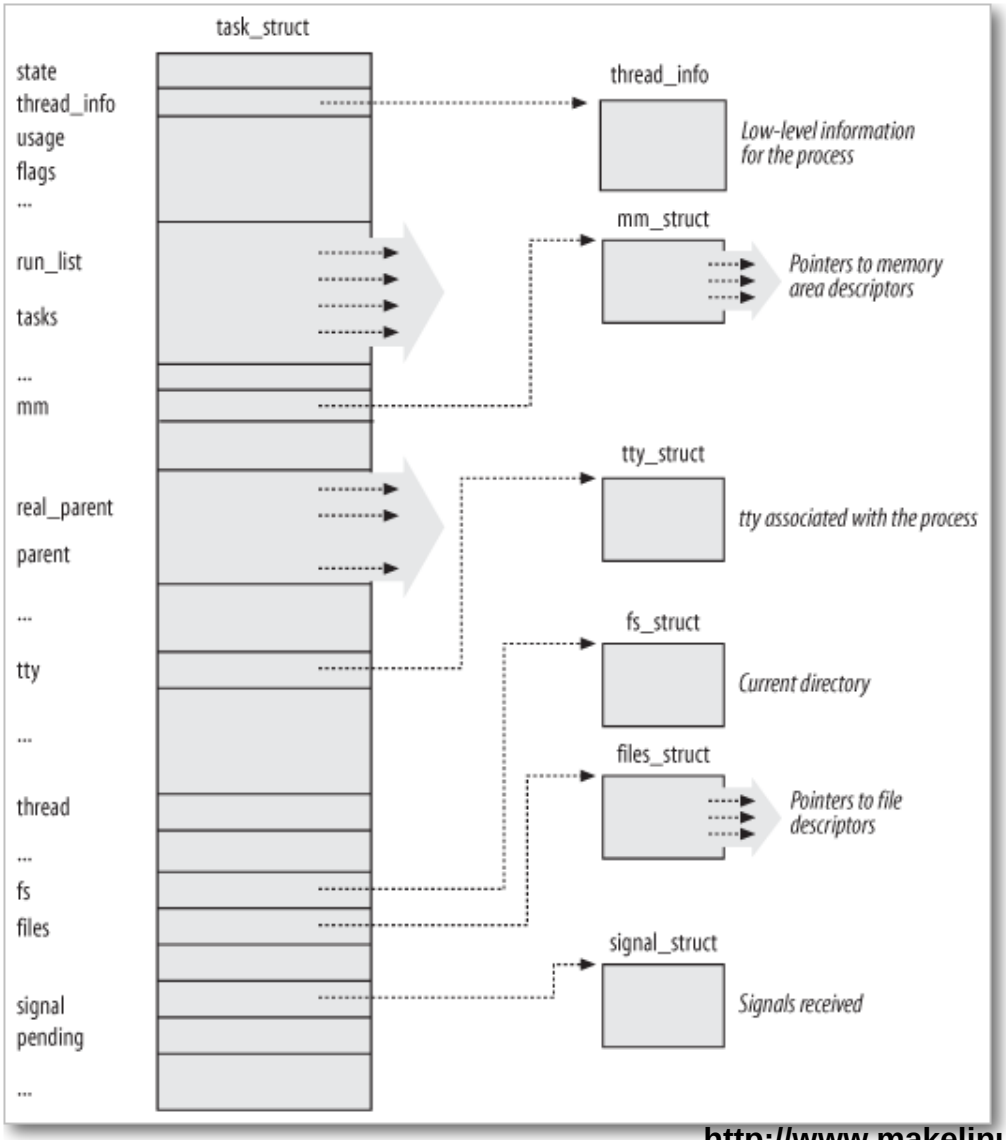
Die Prozessdaten bestehen aus zwei Strukturen

- ➔ task_struct (include/linux/sched.h); ca. 400 Zeilen
  - » Allgemeine und Architektur-unabhängige Daten
  - » Zustand, Flags, Priorität, Scheduling-Daten, CPU-Daten (Welche/wieviele CPUs sind erlaubt), Exit-Code, PID, Stack canary, Parent/Child-Prozesse, offene Dateien, Signale, Spinlocks/Mutexes, Virtual-Memory-Daten, …
- ➔ thread_info (arch/x86/include/asm/thread_info.h)
  - » Daten spezifisch für die konkrete Hardware-Architektur; x86: ca. 20 Zeilen
  - » Zeiger auf task_struct (siehe oben), execution domain, flags, Status, aktuelle CPU, unterbrechbar (preemption), Adress-Grenze, …
  - » Steht „adressmäßig unterhalb“ des Kernel-Stacks

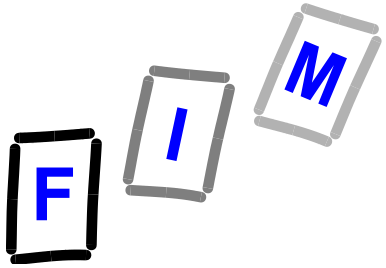
    Kernel-Stack darf daher nicht zu groß werden!

    Zugriff auf „aktuellen Prozess“ ist daher sehr einfach

Beispiel Linux



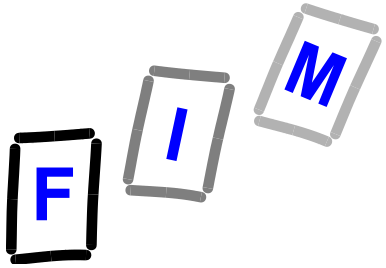
Bovet/Cesati: Understanding the Linux Kernel
http://www.makelinux.net/books/ulk3/understandlk-CHP-3-SECT-2



Typische Operationen auf PCB

Operationen auf Prozesse sind meist Operationen auf die zugehörigen PCBs

- **create (erzeugen), terminate (beenden), fork (aufspalten in einen parallelen Prozess), set priority (Priorität ändern)**
- **set state (in einen Zustand bringen)**
- **get state (den Zustand auslesen)**
- **. . .**



Prozessverwaltung

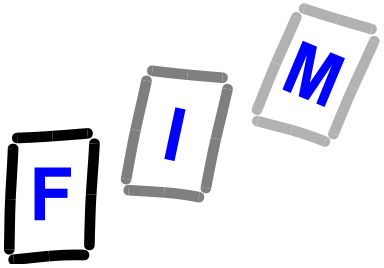
Die Anzahl der Prozesse, die im System ablaufen, ändert sich in der Zeit:

- ➔ **Prozesse werden erzeugt**
- ➔ **Prozesse werden beendet**

→ Anzahl PCBs variabel

→ Verkettete Listen von PCBs

- ➔ **Doppelt-verkettete Listen (auf Basis dynamischer Speicherverwaltung) für die Prozessbeschreibungsblöcke**
- ➔ **Erlaubt ein Durchlaufen in beide Richtungen**

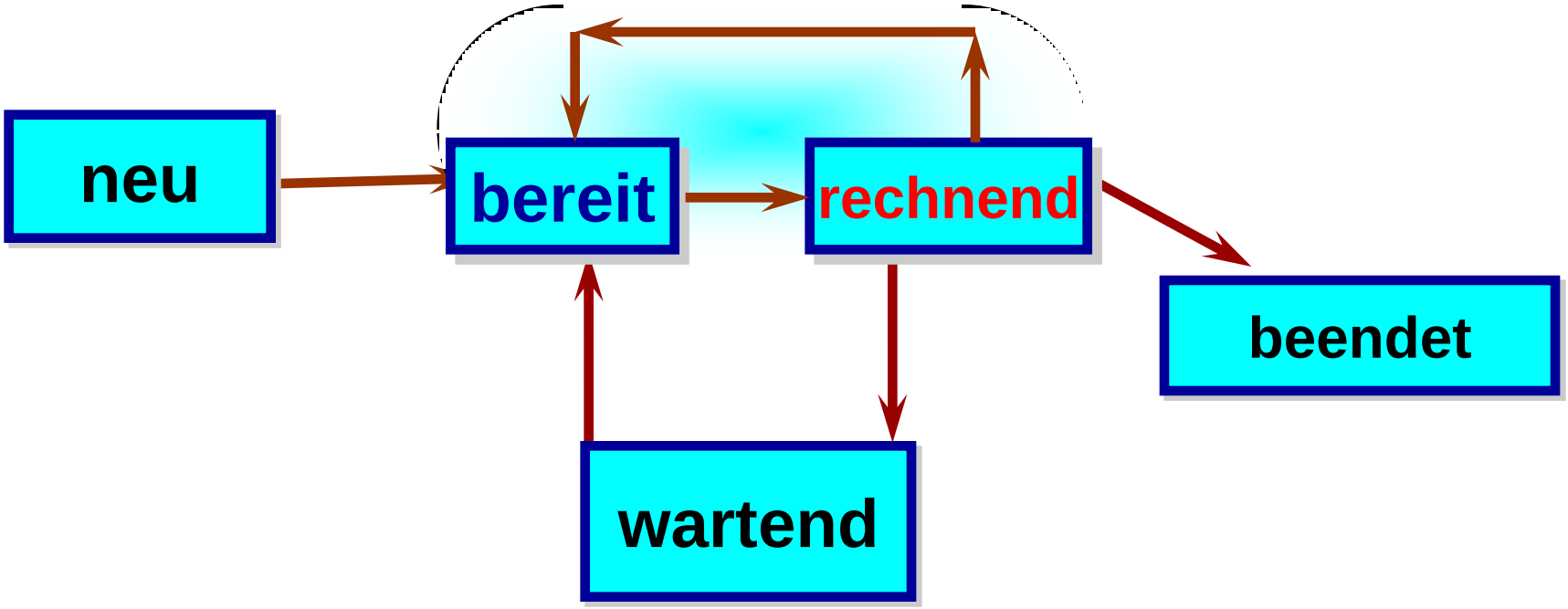


Besondere Zustände: Zustandsübergänge

- **Im Prinzip unterscheidet man**
  - **➔Laufend (running, active, rechnend)**
  - **➔Wartend (waiting, blocked)**
  - **➔Bereit (ready)**
  - **➔ (trivial): finished, beendet**
- **Herstellerabhängig gibt es Varianten oder Verfeinerungen**
  - **➔Unterscheidung, ob im User/Kernel Mode**
  - **➔Unterschiedliche Ebenen für Warten**
  - **➔……**

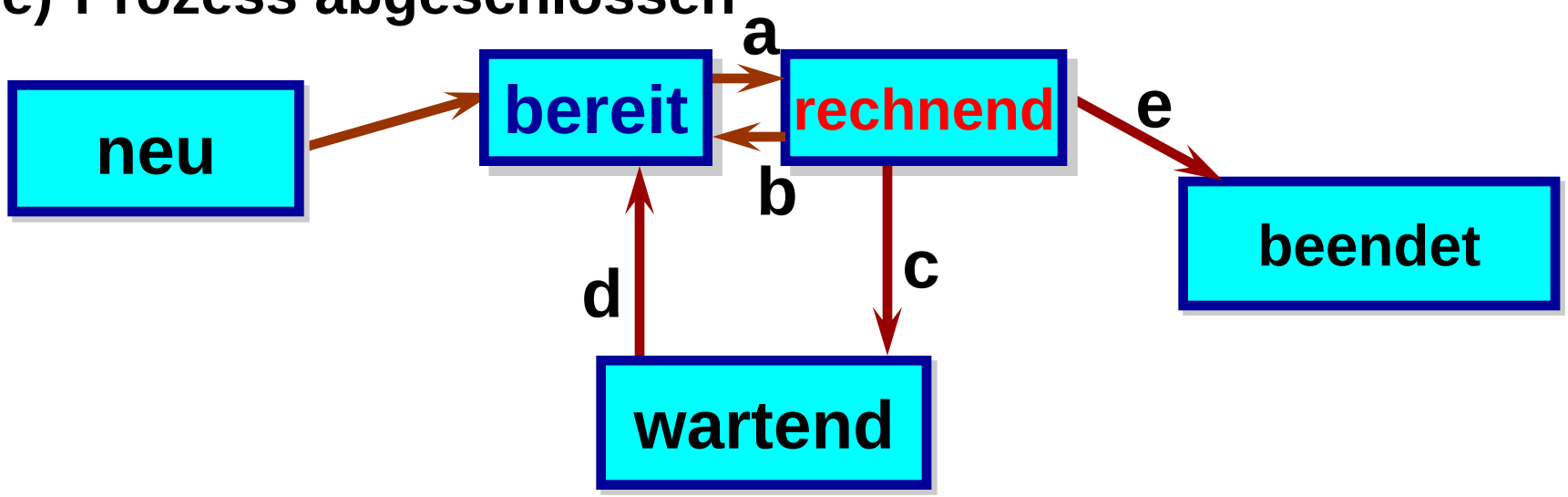
Prozesszustände
Zustandsdiagramm

Das Zustandsdiagramm beschreibt die Übergänge



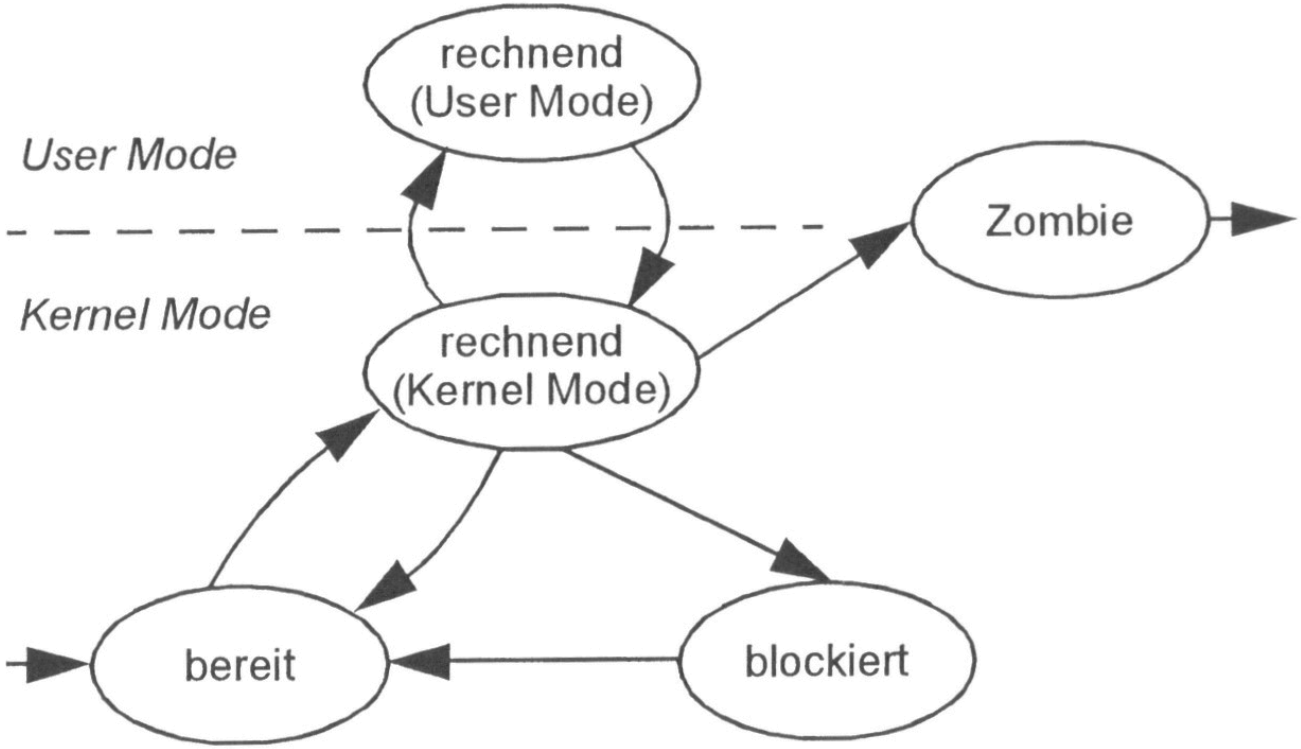
Auslöser für Zustandsüberführung

a) Prozess erhält vom Scheduler die CPU
b) Interrupt entzieht dem Prozess die CPU
c) Prozess löst I/O etc. aus und muss warten
d) Wartebedingung beendet (z.B. I/O fertig)
e) Prozess abgeschlossen



(vergleiche Silberschatz/Galvin Bild 4.1)

Zustandsdiagramm Unix/Linux



Zombie: **Terminierter Kindprozess (= Speicher/… freigegeben), steht jedoch noch in der Prozesstabelle, Elternprozess (=erzeugender Prozess) kann (muss) davon noch z.B. Rückgabewert abfragen**

Zustandsdiagramm für Windows XP
Verfeinert auf Thread-Ebene

Quelle: Microsoft