

Prozesse

- BS-Abstraktionen zur Ausführung von Programmen
- Eigentümer von Ressourcen
- differenzierte Prozessmodelle: definieren konkrete Prozesseigenschaften

Prozessmanagement

- Komponente eines Betriebssystems, die Prozessmodell dieses Betriebssystems implementiert
- Aufgaben: Prozesserschöpfung u. -beendigung (u. Scheduling)
- Datenstrukturen: Prozessdeskriptor, -deskriptortabelle

Prozessdeskriptor Buchführung über sämtliche zum Management eines Prozesses notwendigen Informationen

- Prozessidentifikation
- Rechtemanagement
- Speichermanagement
- Prozessormanagement
- Kommunikationsmanagement

Prozessdeskriptortabelle enthält: Prozessdeskriptoren aller momentan existierenden Prozesse

Threads BS-Abstraktionen für sequentielle, nebenläufige Aktivitäten; sind Gegenstand des Scheduling

Multithread-Prozessmodell vollständige Beschreibung einer ablaufenden Aktivität. Dazu gehören insbesondere

1. das ablaufende Programm
2. zugeordnete Betriebsmittel (Prozessor/Speicher/Kommunikation)
3. Rechte
4. prozessorinterne parallele Aktivitäten (Threads) und deren Bearbeitungszustände

Threaddeskriptor ein TCB enthält lediglich:

1. Threadzustand (aktiv, bereit, blockiert, ...)
2. Ablaufkontext, falls nicht aktiv (Programmzähler, Stackpointer, Prozessorregister)

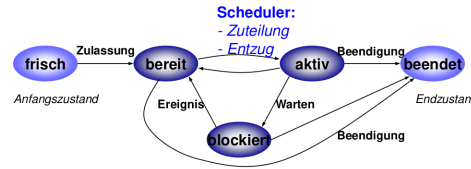
enthält nicht: Beschreibung der Ressourcen (Speicherlayout, Rechte)

Thread-Typen

- Kernel Level Threads (KLTs): Kenntnis über Threads: hat Betriebssystem, genauer: der Betriebssystem-Kern(el)
- User Level Threads (ULTs): Kenntnis über Threads: existiert nur auf Benutzer-Ebene (user level)
- der Betriebssystem-Kern(el) weiß nicht, dass Threads existieren

Scheduling Entscheidung: Welche Threads erhalten wann und wie lange einen Prozessor/Prozessorkern zugeteilt?

Zustandsmodelle Threads können verschiedene Zustände annehmen
Beispiel 3/5-Zustandsmodell)



Scheduling: Notwendigkeit u. Sinn

- allg: Anzahl Aktivitäten >> Anzahl Prozessoren
- nicht alle können gleichzeitig arbeiten
- eine Auswahl muss getroffen werden
- Auswahlstrategie: Schedulingstrategie, -Algorithmus

Scheduling-Strategien

- abhängig vom Einsatzfeld eines Betriebssystems
 - Echtzeitsysteme: Einhaltung von Fristen
 - interaktive Systeme: Reaktivität
- wichtige Strategien:
 - FCFS (First Come, First Served)
 - SRTN (Shortest Remaining Time Next)
 - Round Robin (ohne und mit Prioritäten)
 - EDF (earliest deadline first)
 - ratenmonotones Scheduling

Privilegierungsebenen

- sind typischerweise 'kernel mode' und 'user mode'
- steuern Rechte
 - zur Ausführung privilegierter Prozessorinstruktionen
 - zur Konfiguration des Arbeitsspeicherlayouts
 - zum Zugriff auf Arbeitsspeicherbereiche
 - zum Zugriff auf E/A-Geräte
- Durchsetzung von Regeln: "Nur ein im 'kernel mode' ablaufender Prozess hat Zugriff auf ..."

Kommunikation und Synchronisation

- Austausch von Daten zwischen Prozessen = Kommunikation (Inter-Prozess-Kommunikation, IPC)
- Abweichende Geschwindigkeiten von Sender und Empfänger: behandelt durch Synchronisation

kritischer Abschnitt

- in kritischen Abschnitt darf stets nur ein Thread sein
- notwendig: wechselseitiger (gegenseitiger) Ausschluss
- realisiert durch Entry- und Exit-Code z.B. die Semaphore-Operationen belegen (P) und freigeben (V)

Mechanismen zur Synchronisation

- binäre Semaphore und mehrwertige Semaphore
- (Hoar'sche) Monitore

Mechanismen zur Kommunikation

- Shared Memory (gemeinsamer Speicher)
- Botschaften
- Fernaufrufe
- Systemaufrufe

Notwendigkeit des Ereignismanagement

- in BS laufen sehr viele Aktivitäten parallel ab
- dabei entstehen immer wieder Situationen, in denen auf unterschiedlichste Ereignisse reagiert werden muss, z.B.
 - Timerablauf
 - Benutzereingaben (Maus, Tastatur)
 - Eintreffen von Daten von Netzwerken, Festplatten, ...
 - Einlegen/-stecken von Datenträgern
 - Aufruf von Systemdiensten
 - Fehlersituationen

Umgangsformen mit Ereignissen

- 'busy waiting'
- 'periodic testing'
- Unterbrechungen ('Interrupts')

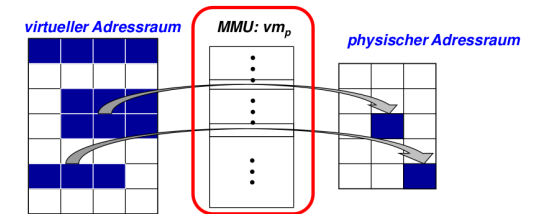
Programmiermodelle für Interrupts

- Prozeduren (→ inline Prozeduraufrufmodell)
- IPC-Operationen (→ IPC-Modell)
- Threads (→ pop-up Thread Modell)

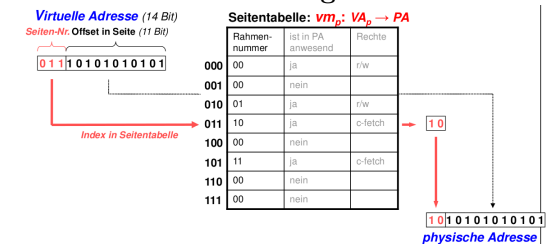
Interrupts auf Anwendungsebene

- notwendig: Event Service Routines (ESRs)
- Beispiel: UNIX/Linux-Signalbehandlung

Virtuelle Prozessadressräume und physischer Adressraum, Abbildungen



Seitenabbildungstabellen



Seitentableneinträge

Rahmennummer im Arbeitsspeicher (PA)	anwesend	benutzt	verändert	Schutz	Caching
--------------------------------------	----------	---------	-----------	--------	---------

- anwesend: liegt Seite im Arbeitsspeicher? ('present'-Bit)
- benutzt: wurde auf die Seite zugegriffen? ('used'-Bit)
- verändert: ist Seite 'schmutzig'? ('dirty/modified'-Bit)
- Schutz: erlaubte Zugriffsart je Privilegierungsebene ('access control list')
- Caching: darf Inhalt der Seite gecached werden?

Seitenaustauschalgorithmen

- Optimal: Auslagern der Arbeitsspeicherseite, deren
 - nächster Gebrauch am weitesten in der Zukunft liegt
 - Auslagerung nichts kostet
- einige Algorithmen, die sich diesem Optimum annähern:
 - First-In, First-Out (FIFO)
 - Second-Chance
 - Least Recently Used (LRU)
 - Working Set / WSClock

i-Node Metainformationen über genau eine Datei



Verzeichnis = Menge von Paaren (Name, i-Node-Index)

Superblock = Einstiegspunkt eines Dateisystems. Enthält Schlüsselparameter:

- Name des Dateisystems
- Typ (NTFS, Ext *, HFS, ...) → Layout der Metadaten

- Größe und Anzahl Sektoren
- Ort und Größe der i-Node-Tabelle
- Ort und Größe der Freiliste
- i-Node-Nummer des Wurzelverzeichnisses

Hardware-Prinzipien

- Controller-Register
 - in E/A-Adressräumen
 - im Arbeitsspeicher (Memory Mapped E/A)
 - Isolation, Robustheit, Sicherheit
- Interruptsystem: asynchrone Benachrichtigungen

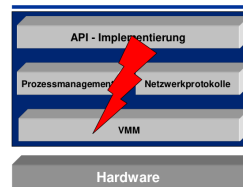
Software-Prinzipien Gerätemanager (Treiber)

- Auftragsmanagement
- ISRs

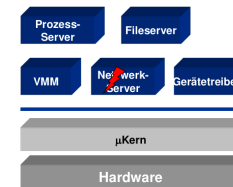
Betriebssystem-Architekturen

Betriebssystem-Architekturen

Monolithische Architektur:
Fehlerausbreitung
innerhalb des (großen) BS-Adressraums



Mikrokernarchitektur:
Fehlerausbreitung
gestoppt an Adressraumgrenze(n)!



SELinux-Ansatz neue Betriebssystem-Abstraktion

- absolute Kontrolle über kritische Funktionen des Betriebssystems
- spezifiziert durch Regelmenge
- implementiert durch die SELinux-Sicherheitsarchitektur

Robustheit Tolerierung unvorhergesehener Fehler und Ausfälle

- Mikrokernarchitekturen (Robuster als Makrokern)
- Fehlerisolation
- Möglichkeiten zur Fehlerbehebung (z.B. Micro-Reboot)

Funktionale Eigenschaften

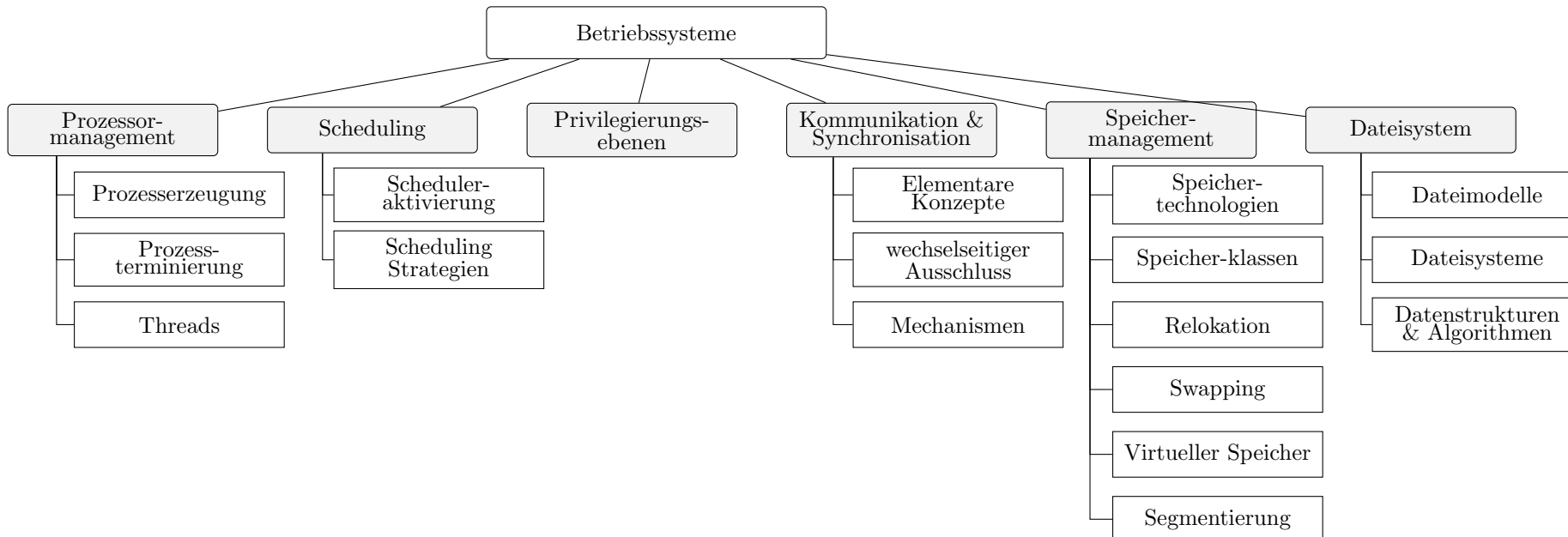
- Authentisierung, Verschlüsselung
- Informations-management
- Kommunikations-management
- Ressourcen-management

Nichtfunktionale Eigenschaften

- Sicherheit
- Korrektheit
- Echtzeitfähigkeit
- Skalierbarkeit
- Offenheit
- Sparsamkeit
- Verfügbarkeit
- Robustheit

Betriebssysteme

- Mainframe
 - performante E/A
 - Massen-daten-verarbeitung
- Server (Web Server, Fileshare)
- Parallelrechner
 - parallele Algorithmen, hoher Rechenbedarf
 - schnelle IPC
- Desktop/Laptop
- Echtzeit
- Eingebettete



Prozessmanagement

Aufgaben

- Prozess-identifikation
- Scheduling
- Ereignis-management
- Rechte-management
- Speicher-management
- Prozessor-management
- Kommunikations-management
- Virtueller Adressraum
- allg Ressourcen Management

Prozesserzeugung

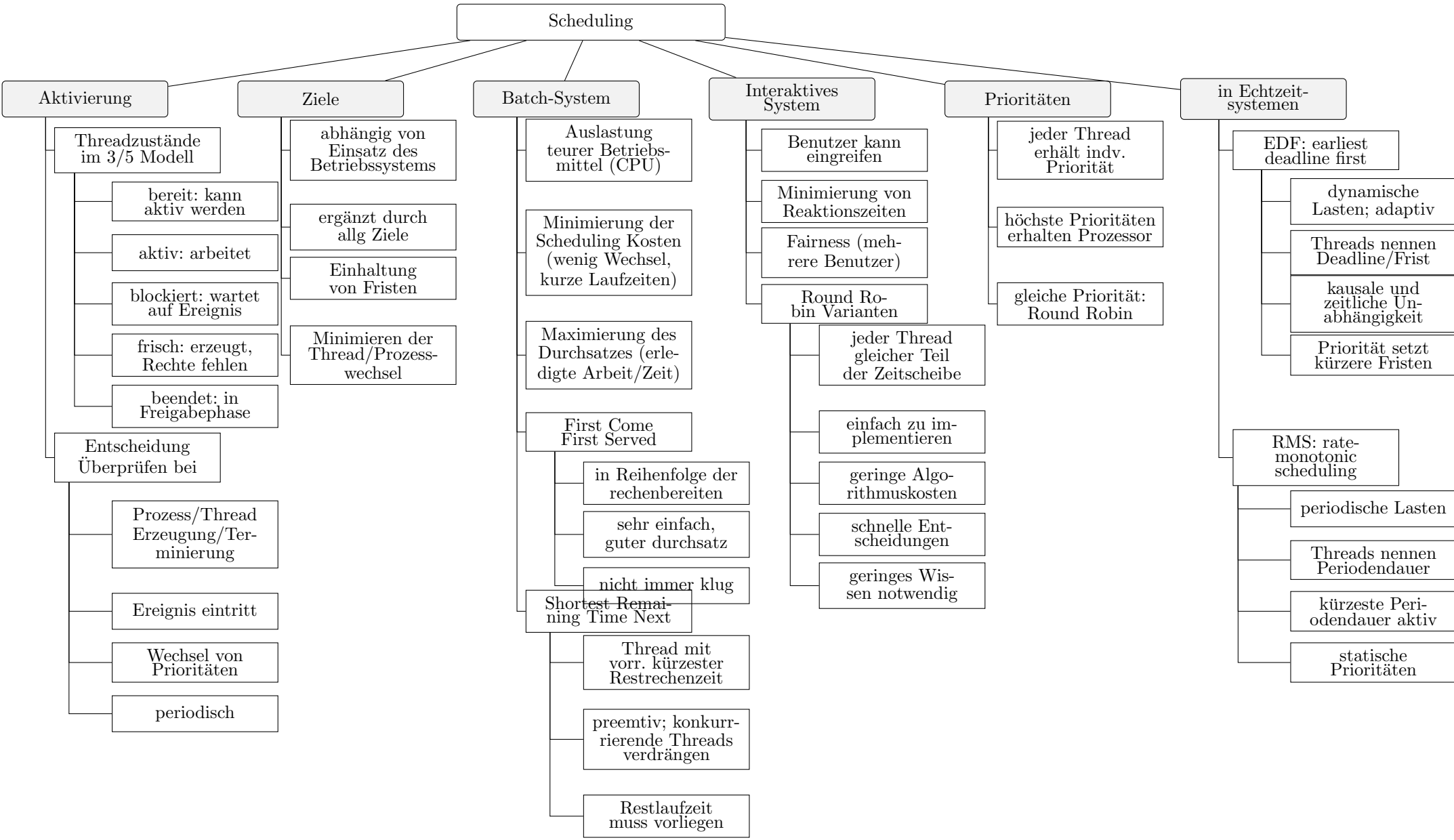
- Vorraussetzungen
 - Rechte
 - Ressourcen Verfügbar
 - Sicherheit
 - Fairness
 - Robustheit / Überlastvermeidung
- Namens-vergabe
 - eindeutig bzgl allen existierenden
 - nicht eindeutig bzgl allen
- Stammbaumpflege
 - erzeugt Kinder
 - baumartige Hierarchie
 - Verwaiste Prozesse -> Adoption
- Allokation (von Ressourcen)
 - Arbeitspeicher Größe
 - Zeitpunkt
 - Prozessorzeit
 - Format

Prozessterminierung

- durch
 - Aufgabe erledigt
 - Fehler aufgetreten
 - durch Nutzer geschlossen
- Folgen
 - Freigabe der Ressourcen
 - Benachrichtigung der 'Parents'
 - Adoption der 'Children'

Threads

- sequenziell innerhalb eines Prozesses
- Kernel Level Thread
 - Implementiert im Betriebssystem
 - Betriebssystem hat Kenntnis über Thread
 - Multi-Thread-modell
 - Performance durch Parallelität
 - Nutzung von Mehrkern-architektur
- User Level Thread
 - Implementiert auf Anwendungsebene
 - Kenntnis nur bei Endbenutzer
 - Single-Thread-Modell
 - Performance durch geringen Overhead
 - management ohne systemaufrufe
 - Individualität
 - Portabilität



Scheduling

Aktivierung

- Threadzustände im 3/5 Modell
 - bereit: kann aktiv werden
 - aktiv: arbeitet
 - blockiert: wartet auf Ereignis
 - frisch: erzeugt, Rechte fehlen
 - beendet: in Freigabephase
- Entscheidung Überprüfen bei
 - Prozess/Thread Erzeugung/Terminierung
 - Ereignis eintritt
 - Wechsel von Prioritäten
 - periodisch

Ziele

- abhängig von Einsatz des Betriebssystems
- ergänzt durch allg Ziele
- Einhaltung von Fristen
- Minimieren der Thread/Prozesswechsel

Batch-System

- Auslastung teurer Betriebsmittel (CPU)
- Minimierung der Scheduling Kosten (wenig Wechsel, kurze Laufzeiten)
- Maximierung des Durchsatzes (erledigte Arbeit/Zeit)
- First Come First Served
 - in Reihenfolge der rechenbereiten
 - sehr einfach, guter durchsatz
 - nicht immer klug
- Shortest Remaining Time Next
 - Thread mit vorr. kürzester Restrechenzeit
 - preemptiv; konkurrierende Threads verdrängen
 - Restlaufzeit muss vorliegen

Interaktives System

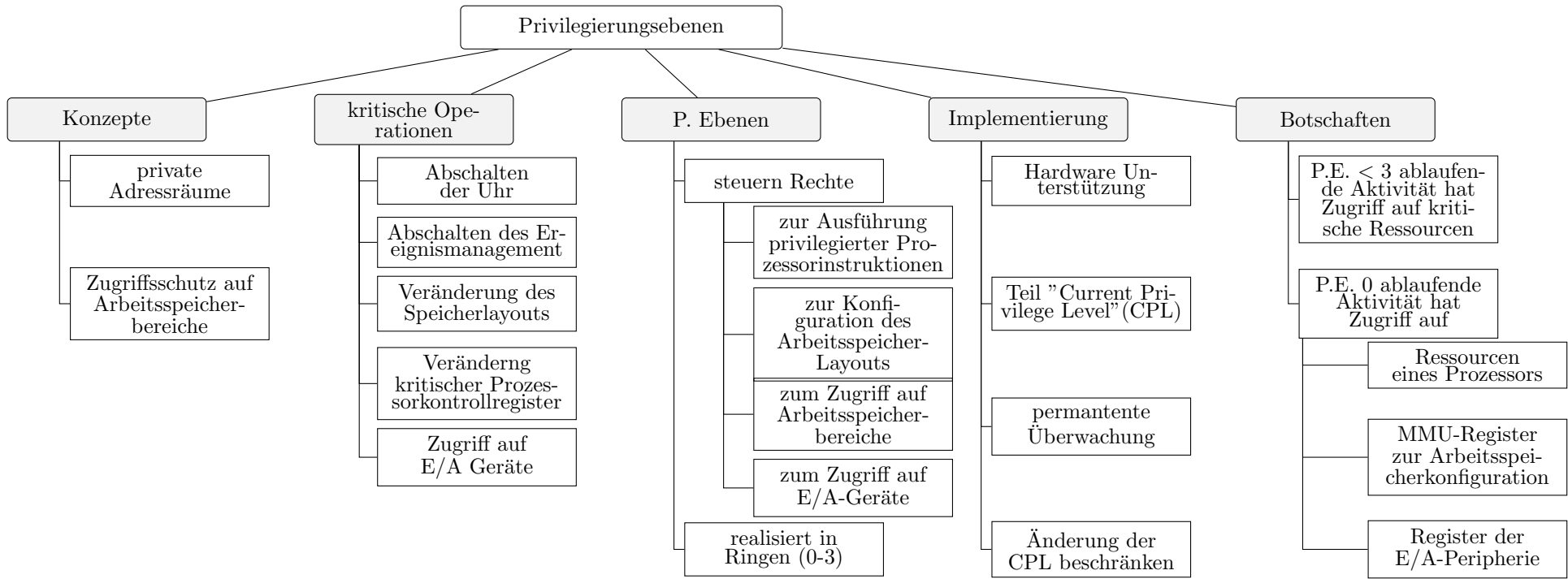
- Benutzer kann eingreifen
- Minimierung von Reaktionszeiten
- Fairness (mehrere Benutzer)
- Round Robin Varianten
 - jeder Thread gleicher Teil der Zeitscheibe
 - einfach zu implementieren
 - geringe Algorithmenkosten
 - schnelle Entscheidungen
 - geringes Wissen notwendig

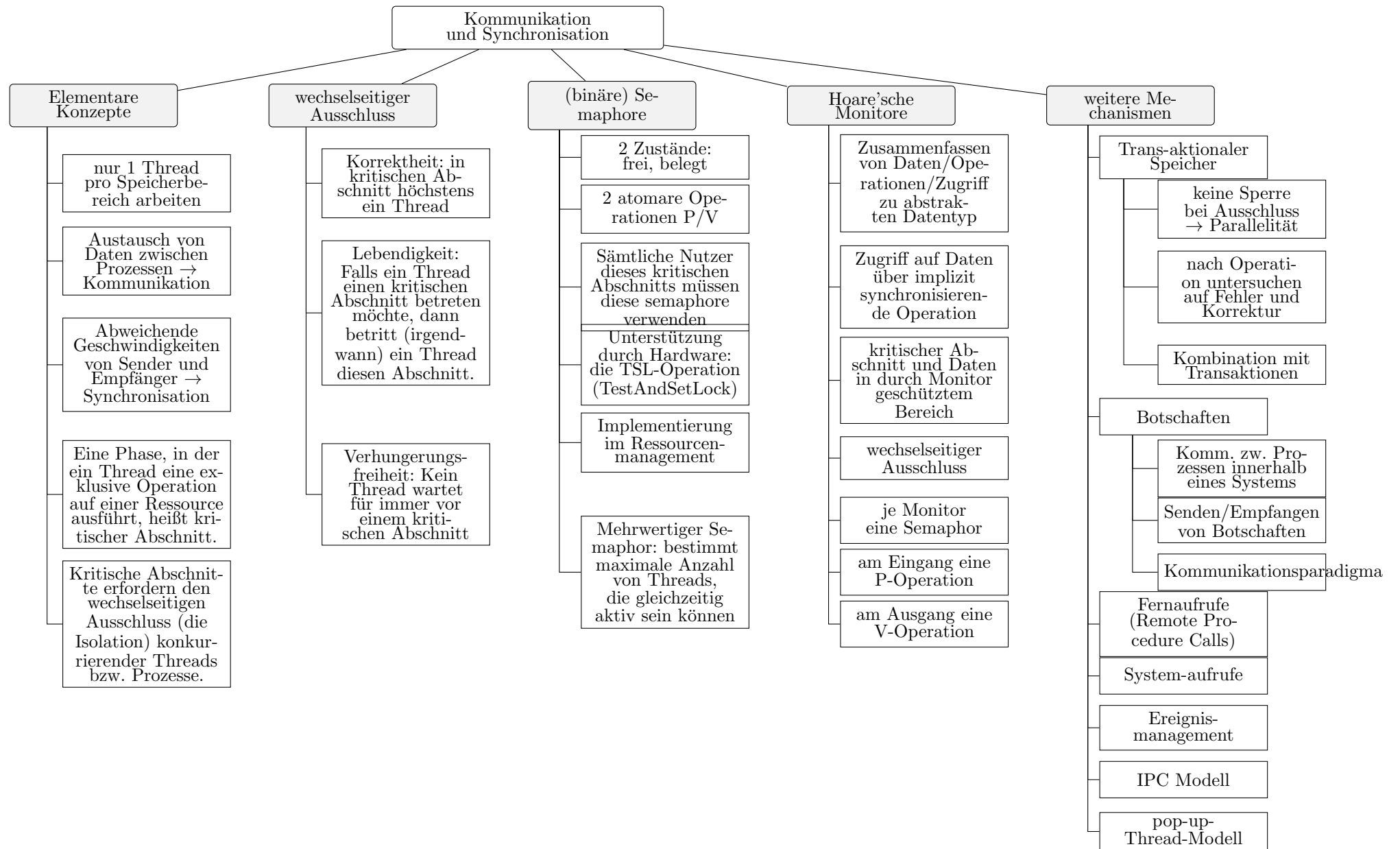
Prioritäten

- jeder Thread erhält indiv. Priorität
- höchste Prioritäten erhalten Prozessor
- gleiche Priorität: Round Robin

in Echtzeitsystemen

- EDF: earliest deadline first
 - dynamische Lasten; adaptiv
 - Threads nennen Deadline/Frist
 - kausale und zeitliche Unabhängigkeit
 - Priorität setzt kürzere Fristen
- RMS: rate-monotonic scheduling
 - periodische Lasten
 - Threads nennen Periodendauer
 - kürzeste Periodendauer aktiv
 - statische Prioritäten





Kommunikation und Synchronisation

Elementare Konzepte

- nur 1 Thread pro Speicherbereich arbeiten
- Austausch von Daten zwischen Prozessen → Kommunikation
- Abweichende Geschwindigkeiten von Sender und Empfänger → Synchronisation
- Eine Phase, in der ein Thread eine exklusive Operation auf einer Ressource ausführt, heißt kritischer Abschnitt.
- Kritische Abschnitte erfordern den wechselseitigen Ausschluss (die Isolation) konkurrierender Threads bzw. Prozesse.

wechselseitiger Ausschluss

- Korrektheit: in kritischen Abschnitt höchstens ein Thread
- Lebendigkeit: Falls ein Thread einen kritischen Abschnitt betreten möchte, dann betritt (irgendwann) ein Thread diesen Abschnitt.
- Verhungers-freiheit: Kein Thread wartet für immer vor einem kritischen Abschnitt

(binäre) Semaphore

- 2 Zustände: frei, belegt
- 2 atomare Operationen P/V
- Sämtliche Nutzer dieses kritischen Abschnitts müssen diese semaphore verwenden
- Unterstützung durch Hardware: die TSL-Operation (TestAndSetLock)
- Implementierung im Ressourcenmanagement
- Mehrwertiger Semaphore: bestimmt maximale Anzahl von Threads, die gleichzeitig aktiv sein können

Hoare'sche Monitore

- Zusammenfassen von Daten/Operationen/Zugriff zu abstraktem Datentyp
- Zugriff auf Daten über implizit synchronisierende Operation
- kritischer Abschnitt und Daten in durch Monitor geschütztem Bereich
- wechselseitiger Ausschluss
- je Monitor eine Semaphore
- am Eingang eine P-Operation
- am Ausgang eine V-Operation

weitere Mechanismen

- Trans-aktionaler Speicher
 - keine Sperre bei Ausschluss → Parallelität
 - nach Operation untersuchen auf Fehler und Korrektur
 - Kombination mit Transaktionen
- Botschaften
 - Komm. zw. Prozessen innerhalb eines Systems
 - Senden/Empfangen von Botschaften
 - Kommunikationsparadigma
- Fernaufrufe (Remote Procedure Calls)
- System-aufrufe
- Ereignismanagement
- IPC Modell
- pop-up-Thread-Modell

