1 Betriebssystem

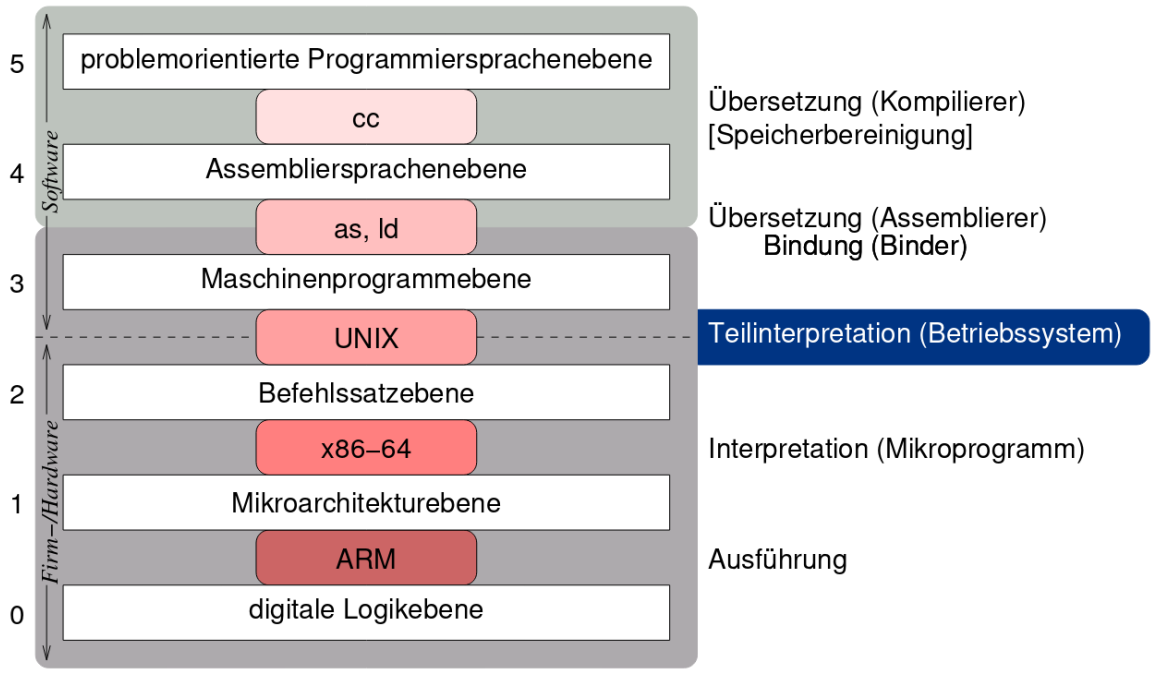
eine Menge von Programmen, die...
... Programme, Anwendungen oder BenutzerInnen assistieren sollen
... die Ausführung von Programmen überwachen und steuern
... den Rechner für eine Anwendungsklasse betreiben
... eine abstrakte Maschine implementieren
verwaltet die Betriebsmittel eines Rechensystems, kontrolliert die Vergabe der Ressourcen und verteilt diese ggf. gerecht an die mitbenutzenden Rechenprozesse
definiert sich nicht über die Architektr, sondern über Funktionen.

2 Rechnerorganisation

2.1 Virtuelle Maschinen

semantische Lücke: Hochsprache ⇔ CPU-Steuerinformation

- *Kompilator*: Softwareprozessor, transformiert in einer Quellsprache vorliegende Programme in eine semantisch äquivalente Form einer Zielsprache.
- *Interpret*: ein Hard-, Firm- oder Softwareprozessor, der die Programme direkt ausführt



Ebene ≥ 4: Übersetzung, Anwendung, symbolische Sprache
Ebene ≤ 3: Interpretation, System, numerische Sprache
Schichten [3,5] repräsentieren virt. Maschinen, die auf die eine reale Maschine ([0,2]) abzubilden sind.

- *Übersetzung*: Generierung eines Programms der Ebene j aus den Befehlen der Ebene i, $j \leq i$
- *Interpretation*:
  total: aller Befehle des Programms der Ebene i
  partiell: nur der Befehle des Programms, die der Ebene i zugeordnet sind, wobei das Programm der Ebene k, $k \geq i$, zugeordnet sein kann durch Ausführung eines Programms der Ebene j, mit $j \leq i$

Zeitpunkte der Abbildungsvorgänge:

- **vor Laufzeit**, statisch
  → preprocessing, precompilation, Übersetzung(Kompilation, Assemblieren), static linking
- **zur Laufzeit**, dynamisch
  → just in time compilation, dynamic linkung, linking/dynamic loading, (Teil)interpretation

Ausnahme (exception): Sonderfall, der Unterbrechung oder Abbruch der Ausführung des Maschinenprogramms bedeutet.

- *Wiederaufnahme*: Ausführungsfortsetzung nach erfolgter Behandlung (z.B. page fault)
- *Termination*: Ausführungsabbruch (z.B. Schutz/Zugriffsverletzung)

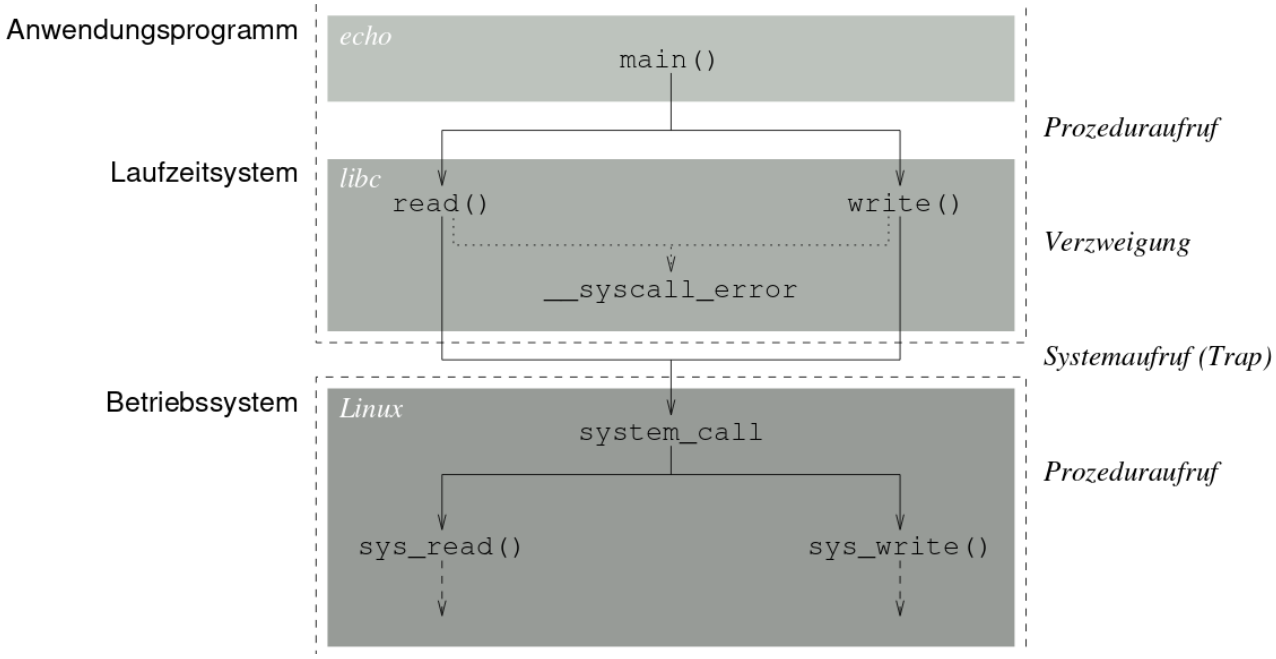
2.2 Maschinenprogramme

Maschinenprogramme enthalten zwei Sorten Befehle: Maschinenbefehle und Systemaufrufe an das Betriebssystem. Sie setzen sich aus Anweisungen zusammen, die ohne Übersetzung von einem Prozessor ausführbar sind.

Maschinenprogramm = Anwendungsprogramm + Laufzeitsystem = Benutzerebene (*user space*)
nur unpriviligierte Operationen direkt ausgeführt, privilegierte Operationen erfordern Moduswechsel (syscall)
Ausführungsplattform = Betriebssystem + CPU = Systemebene (*kernel space*)

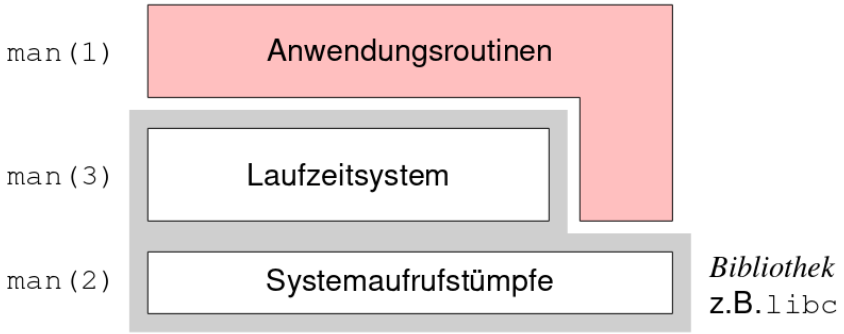
Befehlsabruf- und ausführungszyklus (fetch-execute cycle) zur Ausführung von Systemaufrufen:

1. Prozessorstatus des unterbrochenen Programms sichern
2. Systemaufruf interpretieren
3. Prozessorstatus wiederherstellen und zurückspringen



obere Domäne: Anwendungsmodus, unpriviligiert, räumlich isoliert, transient
untere Domäne: Systemmodus, privilegiert,, räumlich isoliert, resident

Grobstruktur von Maschinenprogrammen:



2.3 Betriebssystemmaschine

Ausführung von Maschinenprogrammen:
1. Ausführung eines Maschinenprogramms durch die CPU
2. Wahrnehmung einer synchronen/asynchronen Ausnahme
3. Teilinterpretation durch das Betriebssystem, Ausnahmebehandlung
4. Beendigung der Ausnahmebehandlung/Teilinterpretation
5. Wiederaufnahme der Ausführung des Maschinenprogramms

Befehle der Maschinenprogrammebene sind...
... unprivilegierte Befehle, die die CPU direkt ausführen kann
... privilegierte Befehle, die das Betriebssystem ausführt (privilegierter Arbeitsmodus)

2.3.1 Unterbrechungen

Betriebssystemlatenz: Verzögerung zw. Abfangzeitpunkt und Wiederaufnahme der Programmausführung
Unterbrechungslatenz: Verzögerung zw. Wahrnehmung durch CPU und Annahme der Unterbrechung

- **trap**: Abfangung für Ausnahmen interner Ursache
  synchron, vorhersagbar, reproduzierbar
- **interrupt**: Unterbrechung durch Ausnahmen externer Ursache
  asynchron, unvorhersagbar, nicht reproduzierbar

Unterbrechungen implizieren nicht-lokale Sprünge (von unterbrochenem zu behandelnden Prozess und umgekehrt). Der Unterbrechungshandhaber wird plötzlich aktiviert, sein exakter Aktivierungszeitpunkt ist nicht vorhersehbar ⇒ der Prozessorstatus des unterbrochenen Programms ist daher während der Unterbrechungsbehandlung invariant zu halten (Sicherung vor Sprung, Wiederherstellung vor Rücksprung).

totale oder partielle Zustandssicherung durch CPU:
... minimal: Statusregister und Befehlszähler
... maximal: den kompletten Registersatz

Betriebsystem sichert den restlichen Zustand:
alle dann noch ungesicherten / flüchtigen / im weiteren Verlauf verwendeten CPU-Register

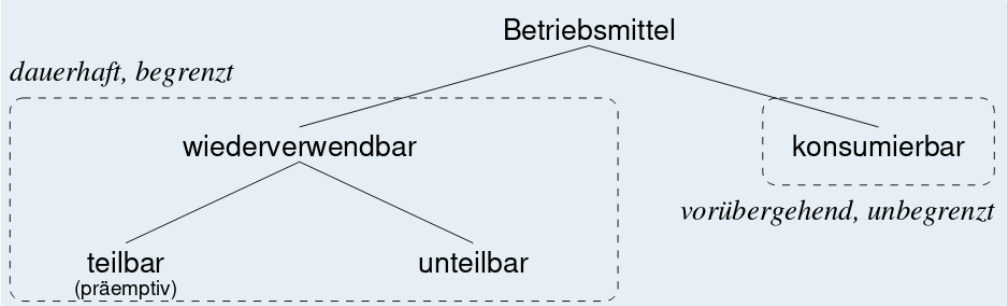
3 Prozesse

Prozess = Programmablauf, Programm in Ausführung durch einen Prozessor
Ebene 5 ⇒ Programmanweisung
Ebene 4 ⇒ Assembliermnemonik
Ebene 3 ⇒ Maschinenbefehl
Ebene 2 ⇒ Mikroprogramminstruktion

Prozess $\neq$ **Prozessinkarnation**: Prozessinkarnation ist Exemplar eines Prozesses als Bautyp
Prozesse sind das Mittel zum Zweck, (pseudo/quasi) gleichzeitige Programmabläufe stattfinden zu lassen
$\rightarrowtail$ Parallelität
1. multiprogramming: mehrere Programme
2. multitasking: mehrere Aufgaben mehrerer Programme
3. multithreading: mehrere Fäden eines oder mehrerer Programme

Ein Programmablauf ist möglich, wenn der dem Betriebssystem explizit gemacht worden ist und alle von ihm benötigten Betriebsmittel verfügbar sind. Bei gemeinsamer Benutzung oder logischer Abhängigkeit ist Synchronisation erforderlich.

3.1 Betriebsmittelklassifikation



Betriebssysteme bringen Programme zur Ausführung, in dem dazu Prozesse erzeugt, bereitgestellt und begleitet werden.

- **sequentiell**: Folge von zeitlich nicht überlappenden Aktionen, verläuft deterministisch
- **parallel**: nicht sequentiell

Programm: die für eine Maschine konkretisierte Form eines Algorithmus
Aktion: die Ausführung einer Anweisung einer (virtuellen/realen) Maschine

Parallelität in System unterschiedlich ausgeprägt:
pseudo - durch *Multiplexen* eines realen/virtuellen Prozessors
echte - durch *Vervielfachung* eines realen Prozessors

3.2 Arbitration / Verwaltung - Einplanung und Einlastung

Prozesse werden gestartet, unterbrochen, fortgesetzt und beendet.
implizite Koodinierung durch Einplanung, logische Verarbeitungszustände, Einlastung

scheduling algorithm:
beschreibt und formuliert die Strategie, nach der ein von einem Rechensystem zu leistender Ablaufplan zur Erfüllung der jeweiligen Anwendungsanforderungen entsprechend der gewählten Rechnerbetriebsart aufzustellen, abzuarbeiten und fortzuschreiben ist.
scheduling: Übergang in die Zustände bereit oder blockiert
dispatching: Übergang in den Zustand laufend

Planung innerhalb eines Betriebssystems bedeutet die koordinierte Zuteilung der Betriebsmittel Prozessor, Speicher, Datei, Nachricht... ⇒ ready list
⇒ Synchronisation: Koordination der Kooperation und Konkurrenz zwischen Prozessen

4 Prozessverwaltung

4.1 Einplanungsgrundlagen

Rechenstoß/CPU-Burst: aktive Phase eines Fadens
I/O-Burst: evtl. Wartephase auf Betriebsmittel

- **actuate:** laufender Faden stößt E/A-Vorgang an
- **schedule:** wartet passiv auf Beendigung der Ein/Ausgabe
- **dispatch:** lastet eingeplanten, laufbereiten Faden ein
- **interrupt:** Ende der Ein/Ausgabe
- **schedule:** wartender Faden wird eingeplant
- **dispatch:** Faden wird eingelastet, sobald an der Reihe

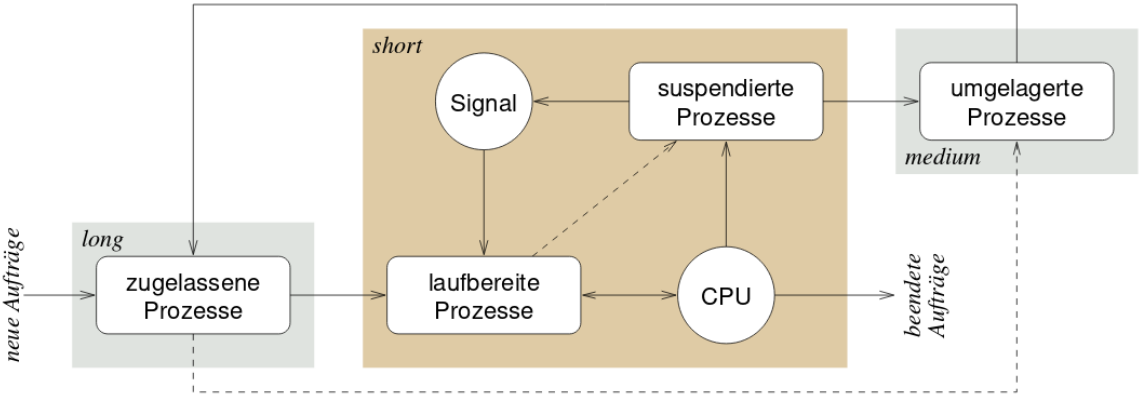
Kreislauf bis zum Leerlauf mangels laufbereiter Fäden

Mittlere Fadenverzögerung: $\frac{1}{n}\sum_{i=1}^{n}(i-1)\cdot k = \frac{n-1}{2}\cdot k$

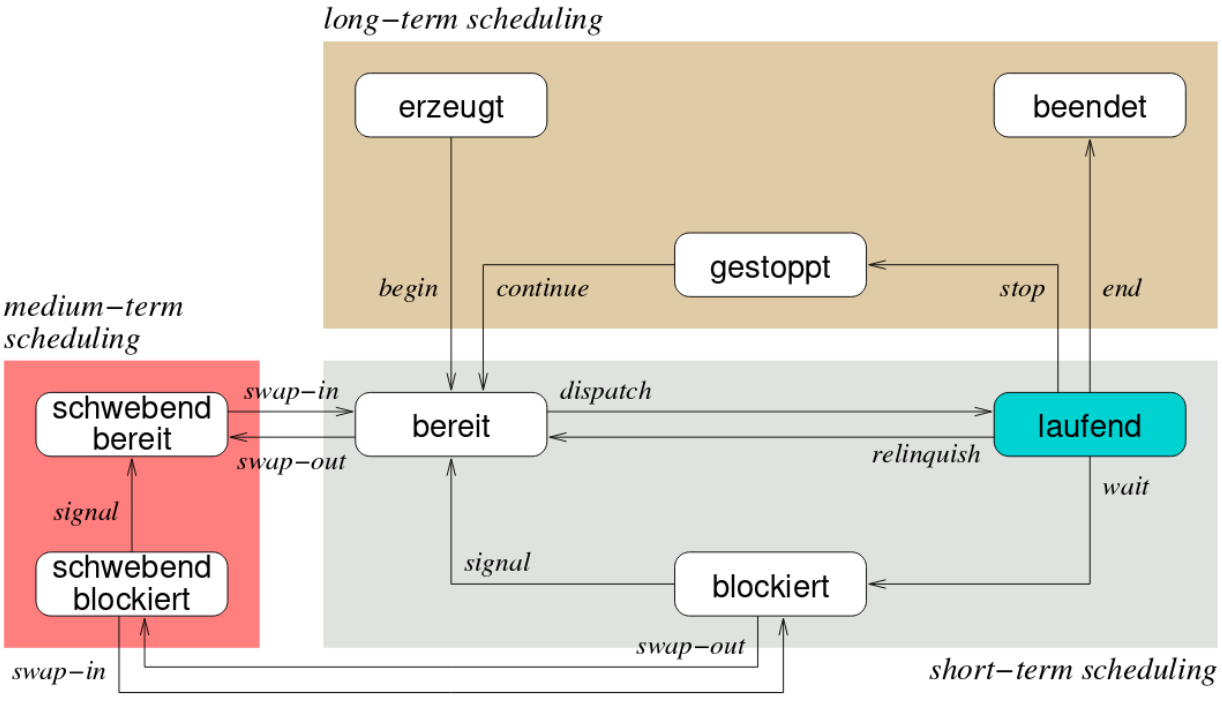
Threads / Programmfaden:
Einplanungseinheit für Prozessorvergabe (wann darf Faden wie lange laufen)
Planung: betriebsmittelorientiert, ereignis-/zeitgesteuert
Einplanung (Reihenfolgenbildung) ungleich Einlastung (Moment der Zuteilung)
Fäden sind Ausdrucksmittel von Mehrprogrammbetrieb oder nicht-sequentiellen Programmen

Überlast durch zuviele eingeplante Fäden ist zu vermeiden. Threadverzögerung wird dominiert von I/O-Wartezeiten, nicht CPU.

4.2 Arbeitsweisen



1. **long-term scheduling:** [s-min]
   Lastkontrolle d. Systems, Prozesse d. mittel-/kurzfristigen Einplanung zuführen
   - erzeugt: fertig zur Programmverarbeitung (ggf. fehlende Speicherzuteilung)
   - gestoppt: erwartet Fortsetzung
   - beendet: erwartet Entsorgung
2. **medium-term scheduling:** [ms-s]
   Umlagerung v. Programmen, Programme v. Hinter- in Vordergrundspeicher bringen
   - schwebend bereit: Exemplar ist in Hintergrundspeicher ausgelagert (swap-in erwartet)
   - schwebend blockiert: ausgelagerter ereigniserwartender Prozess
3. **short-term scheduling:** [μs - ms]
   Einlastungsreihenfolge der Prozesse festlegen, Voraussetzung für Mehrprozessbetrieb
   - bereit: Prozess auf Bereitliste, Scheduling bestimmt Listenposition
   - laufend: Prozess vollzieht CPU-Burst
   - blockiert: wartet auf Zuteilung eines Betriebsmittels



4.3 Einplanungs-/Auswahlzeitpunkt

Einplanung (scheduling) / Umplanung (rescheduling):

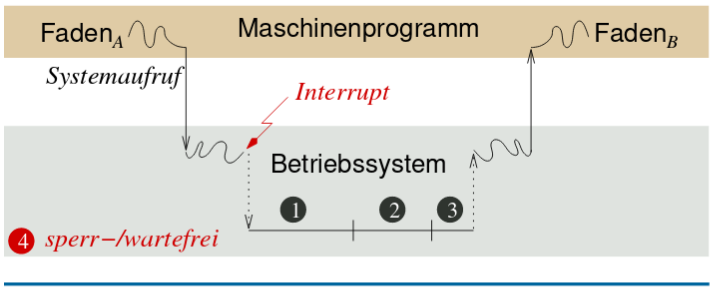
- **begin:** nachdem ein Prozess erzeugt worden ist
- **relinquish:** wenn Prozess freiwillig die CPU abgibt
- **signal:** falls das von einem Prozess erwartete Ereignis eingetreten ist
- **continue:** Prozess kann wieder aufgenommen werden

scheduling latency: Zeitdauer der Ein- oder Umplanung eines Prozesses, möglichst kurz
dispatching latency: Zeitspanne zw. Einplanung und Prozessorzuteilung eines Prozesses

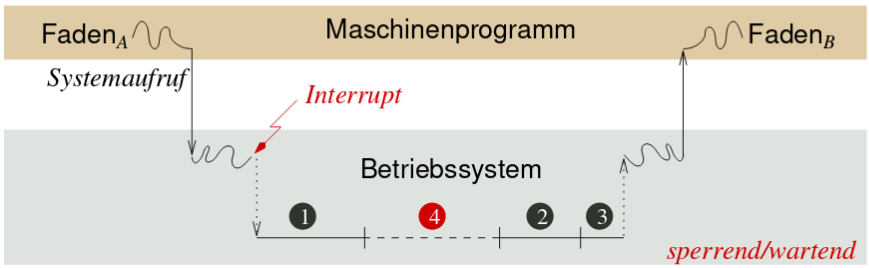
Ereignis → *Signalisierung* → *Einplanung* → *Einlastung* → *Reaktion*
Verdrängung des laufenden Prozesses:

1. Ereignis tritt ein, verdrängender Prozess von *blockiert* zu *bereit*
2. unterbrochene Prozess wird eingeplant (*laufend* zu *bereit*)
3. verdrängende Prozess wird ausgewählt

voll verdrängend



1. Behandlung
   - Interrupt
2. Einplanung
3. Einlastung
4. Synchronisation



verdrängend

4.4 Gütemerkmale

- **benutzerorientierte Kriterien:**
  Benutzerdienlichkeit → Akzeptanz des Systems
  - Antwortzeit: Zeitdauer v. Auflösung bis zur Rückantwort
  - Durchlaufzeit: effektive Prozesslaufzeit und alle anfallenden Prozesswartezeiten
  - Termineinhaltung: Starten/Beendigung zu fest vorgegebenen Zeitpunkt
  - Vorhersagbarkeit: deterministische Ausführung des Prozesses
- **systemorientierte Kriterien:**
  Systemperformanz → Rentabilität des Systems
  - Durchsatz: Anzahl vollendeter Prozesse pro vorgegebener Zeiteinheit
  - Prozessorauslastung: Prozent der Zeit in der CPU Programme ausführt
  - Gerechtigkeit: Zusicherung, Prozessen innerhalb gew. Zeiträume die CPU zuzuteilen
  - Dringlichkeiten: Vorzugsbehandlung des Prozesses mit der höchsten Priorität
  - Lastausgleich: gleichmäßige Betriebsmittelauslastung

Prozesseinplanung impliziert die Rechnerbetriebsart und umgekehrt:
Stapelbetrieb: Durchsatz, Durchlaufzeit, Prozessorauslastung
Dialogbetrieb: Antwortzeit
Echtzeitbetrieb: Dringlichkeit, Termineinhaltung, Vorhersagbarkeit

4.5 Klassifikation von Einplanungsalgorithmen

- **cooperative scheduling / kooperative Planung:**
  Ein-/Umplanung voneinander abhängiger Prozesse, Abgabe nur mittels Systemaufruf
  CPU-Monopolisierung möglich
- **preemptive scheduling / präemptive Planung:**
  Ein-/Umplanung voneinander unabhängiger Prozesse, Verdrängung ereignisbedingt von der CPU
  CPU-Schutz

- **deterministic scheduling / deterministische Planung:**
  alle Prozesse sind bekannt, genaue Vorhersage ist möglich. Zeitgarantien
- **probabilistic scheduling / probabilistische Planung:**
  Prozesse sind unbekannt, Auslastung kann nur abgeschätzt werden

- **off-line scheduling / vorlaufende Planung:**
  vor Betrieb, vollständiger Ablaufplan (für strikte Echtzeitsysteme)
- **on-line scheduling / mitlaufende Planung:**
  während Betrieb des Prozess- oder Rechensystems

- **asymmetric scheduling / asymmetrische Planung:**
  Maschinenprogrammebene, lokale Bereitliste pro CPU, ohne gegenseitige Beeinflussung
  obligatorisch in asym. Multiprozessorsystem, optional in sym. Multiprozessorsystem
- **symmetric scheduling / symmetrische Planung:**
  Befehlssatzebene, globale Bereitliste, gegenseitige Beeinflussung, Lastausgleich

4.6 Verfahrensweisen

- ***first-come, first-served*:** (kooperativ)
  Prozesse werden nach Ankunftszeit eingeplant und in dieser Reihenfolge verarbeitet ⇒ Konvoieffekt
  Prozesse mit langen/kurzen Rechenstößen werden begünstigt/benachteiligt
- ***round robin*:** (verdrängend)
  verdrängendes FCFS, CPU-Schutz, Prozesse nach Ankunftszeit ein- und periodisch umgeplant
  jeder Prozess erhält eine Zeitscheibe zugeteilt
  ⇒ Konvoieffekt: E/A-intensive Prozesse ↔ CPU-intensive Prozesse
- ***virtual round robin*:** (verdrängend, dynamisch)
  round robin mit Vorzugswarteschlange und variablen Zeitscheiben (somit keine Benachteiligung interaktiver Prozesse)
- ***shortest process next*:** (kooperativ, probalistisch, dynamisch)
  jeder Prozess wird entsprechend der für ihn erwarteten Betriebszeit eingeplant, Verhungern möglich
- ***highest response ratio next*:** (kooperativ, probalistisch, dynamisch)
  Prozesse werden nach ihrer erwarteten bedientzeit eingeplant und periodisch (Wartezeit) umgeplant
- ***shortest remaining time first*:** (verdrängend, probalistisch, dynamisch)
  verdrängendes *SPN*, Prozesse werden nach erwarteten Bedienzeit eingeplant u. spontan umgeplant
- ***multilevel queue*:** (statisch)
  Prozesse werden nach Typ in separate Listen eingeplant, jede Liste hat lokale Einplanungsstrategie
- ***multilevel feedback queue*:** (dynamisch)
  Prozesse werden nach Ankunftszeit ein- und periodisch umgeplant, penalization (lange Rechenstöße fallen nach unten durch), ageging

5 Synchronisation

- **Nebenläufigkeit:** Verhältnis v. nicht kausal abhängigen Ereignissen, die sich nicht beeinflussen
- **Kausalität:** Beziehung zw. Ursache und Wirkung, d.h. ursächliche Verbindung zweier Ereignisse
- **nebenläufige Aktion:** im Anderswo anderer Ereignisse liegendens Ereignis
  allgemein: keine benötigt das Resultat der anderen
  speziell: keine verletzt Zeitbedingungen der anderen
- **concurrent/simultaneous processes:**
  Prozesse, durch die sich mehr als eine Aktionsfolge in Raum und Zeit überlappen
  vertikale Simultanverarbeitung: Multiplexen ein und desselben Programms (pseudo Parallelität)
  horizontal: Vervielfachung des Prozessors (echte Parallelität)
- **interacting processes:** gleichzeitige Prozesse, die durch (in)direkte Nutzung einer oder mehrerer gemeinsamer Variablen bzw. Ressourcen interagieren
  interagieren schon im Moment des Zugriffs, Datenabhängigkeiten!

5.1 Sequentialisierung

Der Moment der gleichzeitigen Aktionen ist nicht vorherbestimmt, mehrere Einzelschritte, Teilbarkeit in zeitlicher Hinsicht. Wettstreitsituationen, Methode minimal invasiv wählen!

- *off-line:* statische Einplanung, erfordert Vorabwissen, implizite Synchronisation
- *on-line:* dynamische Einplanung, ohne Vorabwissen, explizite Synchronisation
- *atomare Aktion:* Aktion, deren Einzelschritte nach außen scheinbar gleichzeitig stattfinden
  $\Rightarrow$ wechselseitiger Ausschluss (mutual exclusion)
- *Unteilbarkeit:* Betriebsmittel zu einem Zeitpunkt von nur genau einem Prozess nutzbar
- *Wettstreit:* unter gleichzeitigen Prozessen auftretender Konflikt $\rightarrow$ Konkurrenzsituation
- *wechselseitiger Ausschluss:* vor Aktion Betriebsmittel sperren, nach Aktion entsperren

5.2 Synchronisationsarten-/techniken

- *unterdrückend:* verhindert Prozessauflösung anderer Prozesse $\rightarrow$ akt. Prozess nicht verzögert, betrifft konsumierbare Betriebsmittel
- *blockierend:* sperrt Betriebsmittelvergabe an Prozesse $\rightarrow$ akt. Prozess evtl. suspendiert
  betrifft wiederverwendbare/konsumierbare Betriebsmittel
- *nichtblockierend:* unterbindet Zustandsverstetigung durch Prozesse $\rightarrow$ akt. Prozess evlt. zurückgerollt
  betrifft wiederverwendbare Betriebsmittel

5.2.1 Einseitige/unilaterale Synchronisation

bei konsumierbaren Betriebsmitteln, auch logische oder bedingte Synchronisation
Benutzung eines vorübergehenden Betriebsmittels folgt einer Kausalität
wirkt auf lediglich einen der beteiligten Prozesse. Rest schreitet ungehindert fort.

- *Bedingungssynchronisation:*
  Fortschritt des einen Prozesses ist abhängig von einer Bedingung in parallelem Programm
- *logische Synchronisation:* Maßnahme resultiert aus logischer Abfolge der Aktivitäten

5.2.2 Mehrseitige/multilaterale Synchronisation

bei unteilbaren Betriebsmitteln. Wirkt auf alle in dem Moment beteiligten Prozesse
Gleichzeitige Betriebsmittelzugriffe bewirken widerstreitende Zustandsänderungen des Betriebsmittels.

- *blockierend:* pessimistisch, wechsels. Auss. gleichz. Prozesse, zeitl. begrenzte Betriebsmittelvergabe
- *nichtblockierend:* optimistisch, auf Basis einer Transaktion zw. gleichz. Prozessen

lost wake-up: zwischen Feststellung der Wartebedingung eines Prozesses und seiner daraufhin logisch korrekten Blockierung, wird diese Bedingung durch einen gleichzeitigen Prozess aufgehoben

5.2.3 Fortschrittsgarantien

- *obstruction-free:* einzelner, isolierter Prozess wird Aktion in begrenzter Anzahl von Schritten beenden
- *lock-free:* jeder Schritt trägt zum insgesamten Fortschritt bei, jedoch *starvation* möglich
- *wait-free:* garantiert systemweiten Fortschritt, frei von Aushungerung

5.3 Monitore

kritischer Bereich, control scheduling of resources among individual processes according to a certain policy
mehrseitige Synchronisation an Monitorschnittstelle, einseitige innerhalb des Monitors
In Monitorwarteschlange warten Prozessexemplare auf Eintritt in den Monitor
In Ereigniswarteschlange warten Prozessexemplare auf Aufhebung einer Wartebedingung
Bedingungsvariablen: blockierend (Vorrang Signalnehmer) oder nichtblockierend (Vorrang Signalgeber)

- *Hansen:* blockierende Bedingungsvariablen, alle Signalnehmer auf bereit
- *Hoare:* blockierende Bedingungsvariablen, ein Signalnehmer auf bereit
- *Mesa:* nichtblockierende Bedingungsvariablen, ein o. alle Signalnehmer auf bereit

Bedingungsvariablen: haben keinen vom Programm zugänglichen Wert
wait: setzt Prozess bis Anzeige eines Ereignisses aus, gibt Monitor bis zur Wiederaufnahme implizit frei
signal: zeigt Ereignis an, wirkungslos wenn niemand wartet, nimmt genau einen wartenden Prozess auf

5.4 Semaphore und Sperren

Semaphore: ganzzahlige Variable mit P (-1) und V (+1) Operationen. von jedem Prozess freigebbar
Mutex: können nur vom Besitzerprozess freigegeben werden
Atomarität der Semaphorprimitiven durch Techniken, die hierarchisch tiefer angesiedelt sind

gleichzeitige Prozesse vorbeugen durch zeitweilige außer-Kraftsetzung der Auslösung

- *Unterbrechung*: first-level interrupt handler, asynchron zum akt. Prozess und Betriebssystem
- *Forsetzung*: second-level interrupt handler, synchron zum Systemkern
- *Verdrängung*: nach SLIH, synchron zum Planer

5.5 Kreiseln und Spezialbefehle

Test And Set: write 1 to memory and return old value (as atomic operation)
Comare And Swap: CAS(Variable, Prüfwert, Neuwert) → true falls Variable = Prüfwert, false sonst
backoff: Verweilzeit bis zur Wiederaufnahme der vormals wettstreitigen Aktion
Fetch And Add: Bäckereialgorithmus

Sperren wirken einseitig (unilateral: Unterbrechungs-, Fortsetzungs-, Verdrängungssperre) oder mehrseitig (multilateral: Umlaufsperre). Wechselseitiger Ausschluss ist kein Allheilmitte, um Aktionsfolgen mit wettlaufkritischen Eigenschaften abzusichern (arbeitsloses Kreiseln, Verklemmungsgefahr!).

Nachteile blockierender Synchronisation:

- **performance:** paralleler Systeme bricht ein → reduzierte Busbandbreite, mehr Sequentialität
- **robustness:** im kritischen Abschnitt abstürzen lässt diesen gesperrt :(
- **interference:** mit dem Planer → Prioritätsverletzung
- **liveness:** Gefahr von starvation oder deadlock

Umlaufsperre: gleichzeitige Prozesse kreiseln ohne Nutzen für sich selbst, kommen in der Schleife nicht mit Berechnungen voran. ⇒ Schleifendauer bedeutet Wartezeit, Nutzarbeit startet mit atomaren Aktion
Transaktion: Konsistenzeinheit, die Aktionsfolgen eines Prozesses gruppiert
nicht atomar, das neu berechnete Datum wird jedoch nur bei Isolation übernommen
gleichzeitige Prozesse kreiseln mit Nutzen für sich selbst, kommen in der Schleife mit Berechnungen voran
⇒ Schleifendauer bedeutet Nutzarbeitszeit, Nutzarbeit endet mit atomaren Aktion

5.6 Repräsentation

Prozesse sind in einem Rechensystem verschiedenartig verankert.
- *innerhalb*: im Betriebssystem/Kernel, Prozessinkarnation als Wurzel
- *oberhalb*: im Laufzeit/Anwendungssystem, Prozessinkarnation als Blatt oder innerer Knoten

Gewichtsklasse
Arten von Prozesswechsel zur partiellen Prozessorvirtualisierung:
- feather-weight: im selben Adressraum, ebenda fortsetzend
- light-weight: im Kernadressraum, denselben Andwendungsadressraum teilend
- heavy-weight: im Kernadressraum, im anderen Anwendungsadressrraum landend

Der Prozesskontrollblock bündelt alle zur partiellen Virtualisierung relevanten Attribute eines Prozesses als die zentrale Informations- und Kontrollstruktur im Betriebssystem.
Pro Prozessor verwaltet das Betriebssystem einen Prozesszeiger, der die jeweils laufende Prozessinkarnation identifiziert. Nach außen wird eine Prozessinkarnation systemweit eindeutig durch eine Prozessidentifikation (PID) repräsentiert.

6 Stillstand

Ziele: konfliktfreie Abwicklung, korrekte Bearbeitung in endlicher Zeit, gleichmäßige Auslastung, hoher Durchsatz, kurze Durchlaufzeit, hohe Ausfallsicherheit
⇒ Betriebsmittelanforderung frei von Verhungern/Verklemmung:
- **Verhungern**: andauernde Benachteiligung von Prozessen
- **Verklemmung**: irreversible gegenseitige Blockierung von Prozessen

⇒ Aufgaben: Buchführung, Steuerung der Verarbeitung, Betriebsmittelentzug

Verfahrensweisen:

statisch: vor Laufzeit/einem Abschnitt	*dynamisch:* zur Laufzeit, in bel. Abschnitten
Anforderung aller benötigten Betriebsmittel	Anforderung des Betriebsmittels bei Bedarf
Zuteilung erfolgt ggf. lange vor Benutzung	Zuteilung erst im Moment seiner Benutzung
Freigabe aller bel. Betriebsmitten mit Laufzeitende	Freigabe, sobald kein Bedarf mehr besteht
⇒ suboptimale Auslastung der Betriebsmittel	⇒ Verklemmung von abhängigen Prozessen

6.1 Systemblockade

deadly embrace:
Situation, in der gekoppelte Prozesse gegenseitig die Aufhebung einer Wartebedingung entgegensehen, diese aber durch Prozesse eben dieses Systems selbst aufgehoben werden müsste

Warten kann hierbei inaktiv(*deadlock*) oder aktiv (*livelock*) geschehen:
- *deadlock:* Zustand, in dem die beteiligten Prozesse wechselseitig auf den Eintritt von Bedingungen warten, die nur durch andere prozesse in dieser Gruppe selbst hergestellt werden können
- *livelock:* ähnlich zu deadlock, Prozess ist nicht blockiert, jedoch kein Fortschritt bei der Programmausführung → kann nicht erkannt werden

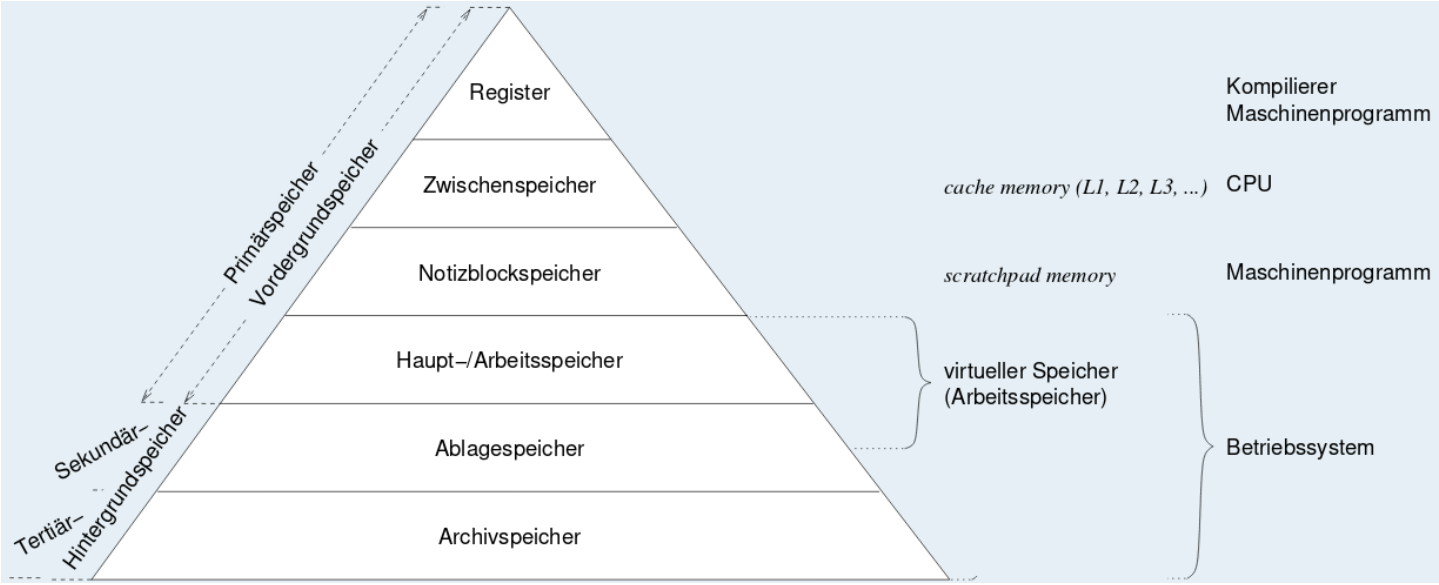
Voraussetzungen für Verklemmungen:
1. *wechselseitiger Ausschluss* in der Benutzung benötigter Betriebsmittel
2. *Nachforderung* eines oder mehrerer Betriebsmittel (*hold and wait*)
3. *Unentziehbarkeit* der zugeteilten Betriebsmittel
4. *zirkuläres Warten* muss eingetreten sein

Gegenmaßnamen: Vorbeugen, Vermeiden, Erkennen und Erholen

7 Speicher

- kurz: Primärspeicher, z.B. Register-, Zwischen-, Haupt/Arbeitsspeicher
- mittel: Sekundärspeicher, z.B. Festplatten
- lang: Tertiärspeicher, z.B. Festplattenarchive

Virtueller Speicher: Betriebssystemtechnik, die laufende Prozesse ermöglicht, obwohl ihre Maschinenprogramme samt Daten nicht komplett im Hauptspeicher liegen



7.1 Adressräume

Ein laufender Prozess generiert Folgen von Adressen auf den Haupt-/Arbeitsspeicher. Wertevorrat ist durch das Programm definiert. Dieser gibt Adressraum vor, in dem der Prozess eingeschlossen ist.

- **realer Adressraum**: lückenhafter, wirklicher Hauptspeicher (Arbeitsspeicher im Vordergrund) der durch einen Prozessor definierte Wertevorrat von Adressen.
  Nicht jede Adresse ist gültig, der Wertevorrat kann Lücken aufweisen (Ebene 2).
- **logischer Adressraum**: lückenloser, wirklicher Hauptspeicher (Arbeitsspeicher im Vordergrund) der im Programm P definierte Wertevorrat von Adressen, der einem Prozess von P zugebilligt wird.
  Jede Adresse ist gültig, das heißt der Wertevorrat enthält konzeptionell keine Lücken.
- **virtueller Adressraum**: lückenloser, scheinbarer Hauptspeicher (Arbeitsspeicher im Hintergrund) an sich logischer Adressraum, es bildet nicht jede Adresse auf im Hauptspeicher liegendes Datum ab.

Benutzung einer solchen nicht abgebildeten Adresse verursacht einen Zugriffsfehler (→ trap)
Adressraumbelegungsplan: welche Hardwareeinheiten sind über welche Adressen zugreifbar?
ohne assoziierte Hardwareeinheiten, mit spez. Verwendungszweck, für allgemeinen Verwendungszweck

Ein **logischer Adressraum** beschreibt die geradlinige Beschaffenheit des Hauptspeichers eines (schwergewichtigen) Prozesses, umfasst alle für einen Prozess gültigen Text- und Datenadressen, entsprechend des durch ihn ablaufenden Programms.

- *Text* - Maschinenanweisungen, Programmkonstanten
- *Daten* - initialisierte Daten, globale Variablen, Halde
- *Stapel* - lokale Variablen, Hilfsvariablen, aktuelle Parameter

Eindimensionaler Adressraum: Tupel (Seitennummer, Versatz)
Seitennummerierung: Unterteilung des Adressraums in gleich große Einheiten und deren lineare Aufzählung.
Seite im logischen/virtuellen Adressraum, Seitenrahmen/Kachel im realen Adressraum.
In Seitentabelle sind Seitendeskriptoren gespeichert. Diese enthalten folgende Informationen:
- Kachel-/Seitenrahmennummer: seitenausgerichtete reale Adresse
- Attribute: Schreibschutzbit, Präsenzbit, sticky-Bit, Modifikationsbit

Zweidimensionaler Adressraum: Paar (Segmentnummer, Adresse / (Seitennummer, Versatz))
Segmentierung meint die Unterteilung des Adressraums in Einheiten von möglicherweise verschiedener Größe, sogenannter Segmente, die ihrerseits gleichgroße Einheiten linear aufgezählt enthalten.
In Segmenttabelle sind Segmentdeskriptore gespeichert, Diese enthalten folgende Informationen:
- Basis: Segmentanfangsadresse im Haupt/Arbeitsspeicher mit passendem *alignment*
- Limit: Segmentlänge als Anzahl der Granulate, Zahl der gültigen Granulatadressen
- Attribute: Typ (Text, Daten, Stapel), Zugriffsrechte, ...

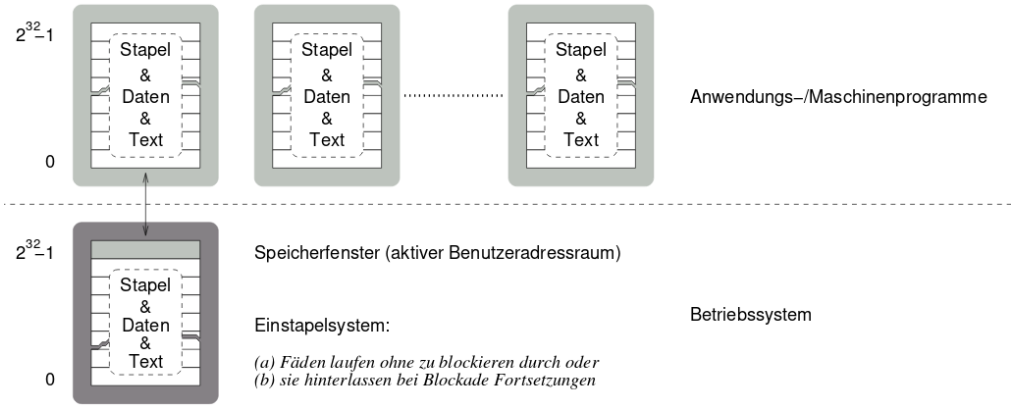
translation lookaside buffer (oder wie Wosch sagt, Übersetzungspuffer)
Zwischenspeicher für Übersetzungsergebnisse.
Umsetzungsfehler (*lookup miss*) führt zur Tabellenwanderung, *table walk* (hard- oder softwaregeführt).
- *hardwaregeführt*: die CPU läuft Tabellen ab, mittelbarer Trap bei erfolglosem *table walk*
- *softwaregeführt*: Betriebssystem läuft Tabellen ab, unmittelbarer Trap bei Umsetzungsfehler

Eine virtuelle Adresse erbt alle Eigenschaften einer logischen Adresse und erlaubt darüber hinaus ortstransparente Zugriffe auf externen Speicher. Lose Bindung zw. Adresse und adressierten Entität:
- *logische Adresse*: entkoppelt von Lokalität im Hauptspeicher, ermöglicht dynamisches Binden aktiver Prozesse und Tauschen inaktiver Prozesse
- *virtuelle Adresse*: logische Adresse, von der Lokalität im Arbeitsspeicher entkoppelt, erlaubt *paging*

Adressbildung impliziert partielle Interpretation der Zugriffe
Präsenzbit: **0** - abwesend, verursacht Zugriffsfehler/Trap, **1** - anwesend, unterbricht Dereferenzierung nicht

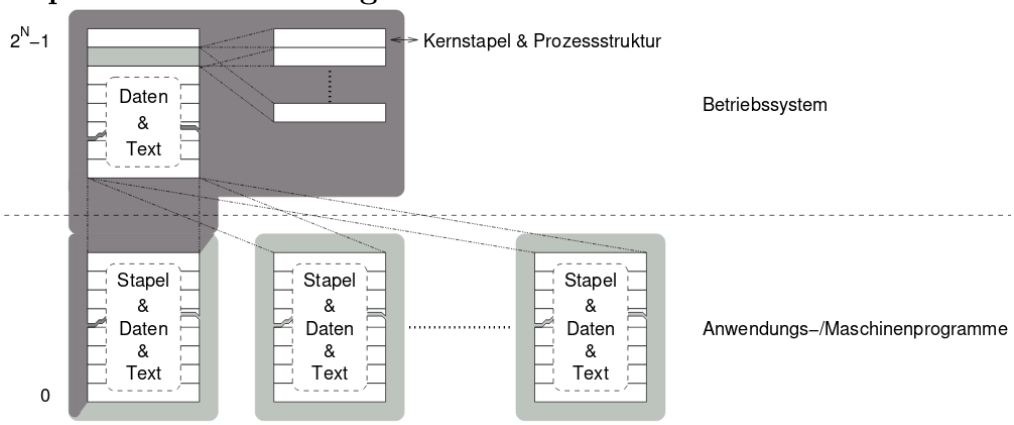
Mehradressraumsysteme
Virtualität: Eigenschaft einer Sache, nicht in der Form zu existieren in der sie zu existieren scheint, aber in ihrem Wesen oder ihrer Wirkung einer in dieser Form existierenden Sache zu gleichen.
Private Adressräume: Illusion von einem eigenen physischen Adressraum für Betriebssystem und Maschinenprogramme → Vervielfachung des Adressbereichs.
Im Vordergrund: strikte Isolation von ganzen Adressräumen
MMU/MPU verhindert Ausbrechen von Prozessen. Datenaustausch erfordert Spezialbefehle des Betriebssystems (Systemaufrufe) und der CPU für die Betriebssystemprogramme.

Programmierte Mitbenutzung:



horizontal: Interprozesskommunikation + seitenbasierte Mitbenutzung
vertikal: Zugriffe auf den Benutzeradressraum durch Speicherfenster
Partiell private Adressräume: Illusion von einem eigenen physischen Adressraum für Maschinenprogramme

Implizite Mitbenutzung:



horizontal: Interprozesskommunikation, Segment/Seitenmitbenutzung
vertikal: speicherabgebildeter Zugriff auf den Benutzeradressraum

segmentierte ↔ gekachelte Speicherverwaltung:
Segmente können verschieden groß sein, ihre jew. Attribute sind Granularität, Adresse und Länge. Kacheln dagegen sind alle gleich groß, sie unterscheiden sich lediglich durch ihre Adresse.

7.2 Speicherverwaltung

- **Platzierungsstrategie**/*placement policy*: wo im Hauptspeicher ist noch Platz?
- **Ladestrategie**/*fetch policy*: wann muss ein Datum im Hauptspeicher liegen?
- **Ersetzungsstrategie**/*replacement policy*: welches Datum im Hauptspeicher ist ersetzbar?

Verantwortlichkeiten:
Speicherzuteilung (placement): Maschinenprogramm und Betriebssystem, Halden-/Hauptspeicher
Speichervirtualisierung (fetch, replacement): allein Betriebssystem, Haupt-/Arbeits-/Ablagespeicher

Das Maschinenprogramm verwaltet den seinem Prozess zugeteilten Speicher lokal eigenständig.
Das Betriebssystem verwaltet den gesamten Haupt-/Arbeitsspeicher global für alle Prozesse/adressräume
⇒ Symbiose / Arbeitsteilung

Zerlegungsgrad:
- **Wort**: Platzierungseinheit für Hauptspeicher
- **Zelle**: Platzierungseinheit für Halden- und Hauptspeicher
- **Kachel**: Platzierungs-, Lade- und Ersetzungseinheit für Arbeitsspeicher

7.2.1 Platzierungsstrategie

Loch: Hohlraum im Innern des Haupt/Arbeitsspeichers, mehrere davon im realen Adressraum
Bitkarte: für Hohlräume fester Größe
Lochliste: für Hohlräume variabler Größe, dynamische Datenstruktur

Abspeicherung: jedes Listenelement beschreibt ein Stück freien Speicher.
- Löcher in anderem Adressraum wie Liste: wirklich leer, Listenelemente belegen Speicher im Adressraum des Betriebssystems, Listenoperationen wirken in der Schutzdomäne des Betriebssystems
- Löcher in gleichem Adressraum wie Liste: scheinbar leer, beinhalten Listenelemente, Mindestgröße

Gebrauchsstück: allgemein zur Ablage von Programmtext, - daten und Stapeln im Hauptspeicher von Prozessexemplaren in Gebrauch. Wird bei Zuteilung ausgeblendet, bei Zurücknahme wieder in den Betriebssystemadressraum eingeblendet.

Lochliste der Größe nach auf-/absteigend sortiert.
Restloch in Liste sortiert, wenn größer als Mindestgröße

- *best-fit*: aufsteigende Lochgrößen, das kleinste passende Loch suchen
  beste Zuteilung, minimaler Verschnitt, aber eher langsam
  → hinterlässt kleine Löcher bei steigendem Suchaufwand
- *worst-fit*: absteigende Lochgrößen, das größte passende Loch suchen
  sehr schnelle Zuteilungsentscheidung, begünstigt Zerstückelung
  → hinterlässt eher große Löcher bei konstantem Suchaufwand

Lochliste der Größe des Adresswerts nach aufsteigend sortiert.
keins der Verfahren erzeugt Restloch, das im zweiten Listendurchlauf einsortiert werden müsste

- *first-fit*: schnell Zuteilung, begünstigt aber Verschwendung
  erzeugt kleine Löcher von vorn, erhält große Löcher hinten
  → hinterlässt eher kleine Löcher bei steigendem Suchaufwand
- *next-fit*: reihum Variante von first-fit, Suche beginnt beim zuletzt zugeteiltem Loch
  → hinterläst eher gleichgroße Löcher, Konsequenz ist ein im Mittel eher abnehmender Suchaufwand

Lochliste ist der Zweierpotenzgröße nach aufsteigend sortiert.

- *buddy*: sucht das kleinste passende Loch $buddy_i$ der Größe 2^i
  - i ist Index in eine Tabelle von Adressen auf Löcher der Größe 2^i
  - i ist so zu bestimmen, dass $2^{i-1} < size < 2^i, i > 1$
  $buddy_i$ entsteht durch sukzessive Splittung von $buddy_j, j > i$:
  - 2 gleichgroße Stücke, die Kumpel des jew. anderen sind
  i wird fortgesetzt dekremntiert, solange $2^{i-1} > size, i > 1$

7.2.2 Ladestrategie

bestimmt, wann ein Datum im Hauptspeicher liegt und wie es dort hingebracht wird.
Seitenfehler (*page fault*) oder Segmentfehler (*segment fault*): *present bit* = 0

Einzelanforderung: on demand
gesteuert durch das Präsenzbit
anwesend → Zugriff möglich
abwesend → page/segment fault
bei Abwesenheit partielle Interpretation

Vorausladen: anticipatory
der Einzelanforderung zuvorkommen (*prefetch*)
Heuristiken: Hinweise über zukünftige Zugriffe
→ ggf Verdrängung

7.2.3 Ersetzungsstrategie

bestimmt, welches Datum seinen Platz im Hauptspeicher für ein anderes Datum freimachen muss
Globale Verfahren: untersuchen Mengen aller residenten Seiten aller Prozesse im System, unabhängig vom Verursacher des Zugriffsfehlers (Seitenflattern möglich!), FIFO, LRU
Lokale Verfahren: untersuchen Menge der residenten Seiten des Prozesses, der Zugriffsfehler verursacht (hoher Overhead bei Arbeitsmenge), Freiseitenpuffer, Arbeitsmenge

Ersetzungsstrategien:

- *FIFO*: first-in, first-out
- *LFU*: least frequently used - am seltesten referenzierte Seite, oder *MFU* (most frequently used)
- *LRU*: least recently used - am längsten nicht mehr referenzierte Seite
  → aging register - regelmäßiger Check, ob Referenzbit (*use*) gesetzt (*second-chance*)
  bei *third-chance replacement* wird zusätzlich das *dirty bit* geprüft (use, dirty):

	Bedeutung	Entscheidung
(0,0)	ungenutzt	beste Wahl
(0,1)	beschrieben	keine schlechte Wahl
(1,0)	kürzlich gelesen	keine gute Wahl
(1,1)	kürzlich beschrieben	schlechteste Wahl

Speichervirtualisierung: Abbildung der logisch unendlichen Mengen von Seiten eines oder mehrerer virtueller Adressräume auf beschränkte Menge von Seitenrahmen des realen Adressraums
resident set: endliche Menge der Seiten des virtuellen Adressraums eines Prozessexemplars, die gegenwärtig auf Seitenrahmen abgebildet ist und damit im Hauptspeicher platziert vorliegt
working set:
thrashing/Seitenflattern: E/A durch Seitenein-/auslagerungen dominiert die gesamten Systemaktivitäten
page buffering/Freiseitenpuffer: FIFO-Prinzip, verwaltet zusätzl. *cache* potentiell zu ersetzender Seiten
Standpunkt eines Prozesses: kleinste Sammlung von Programmtext und -daten, die im Hauptspeicher vorliegen muss, damit effiziente Programmausführung zugesichert werden kann

7.3 Speicherverschnitt

Fragmentierung:

- *intern*: angeforderte Größe ist kleiner als das zugeteilte Stück
  Seitennummerierung, Buddy-Verfahren
  → Verschwendung, ist unvermeidbar
- *extern*: angeforderte Größe ist zu groß für jedes einzelne Loch
  Segmentierung, Buddy-Verfahren
  → Verlust, nur aufwändig vermeidbar

Verschmelzung von Löchern erzeugt größere Hohlräume und bringt damit positive Eigenschaften:

- weniger Löcher → geringere externe Fragmentierung
- weniger Lochdeskriptoren → kürzere Listen und Suchzeiten

Verschmelzungsaufwände:

- *buddy*: Stück wird mit seinem Buddy-Stück verschmolzen, Adressen gleichen sich bis auf ein Bit
- *first/next-fit*: beim Einsortieren in Lochliste Nachbarschaft prüfen
- *best/worst-fit*: beim Einsortieren prüfen
  Listennachfolger müssen keine Nachbarn sein → Liste durchlaufen und Nachbarn finden

Vereinigung des globalen Verschnitts
um externe Fragmentierung aufzulösen, sind Gebrauchsstücke im Hauptspeicher durch Kopiervorgänge umzulagern. Umlagerung zieht Relokation der betroffenen Segmente oder Seiten nach sich, wenn sie ihre neue Position im realen Adressraum haben.

8 Namen

Adresse von Programmierer benutzt heißt Name/virtuelle Adresse, Set von solchen ist address/name space.
Adresse vom *memory* benutzt heißt location/memory address, Set von solchen ist memory space.
address space ist Sammlung von möglicherweise benutzbaren Namen.
There is no requirement that every virtual address represent or contain any information.

Adressen können...
- **realer**: muss exakt dort liegen, intern
- **logischer**: kann woanders liegen, intern
- **virtueller**: kann woanders liegen, intern oder extern

...Natur sein, wobei ein interner Ort ein Platz im Hauptspeicher ist.

paging: jede Adresse wird interpretiert als Tupel einer Oktett- und Seitennummer → eindim. Adressraum
segmentation: jede Adresse ist repräsentiert als Paar von Segmentname und Adresse
→ zweidim. Adressraum, d.h. Segmente in der ersten und Segmentinhalte in der zweiten Dimension
paged segmentation: jedes Segment ist seitennummeriert ausgelegt.

8.1 Namensraum

Ein Kontext (Verzeichnis) repräsentiert einen Namensraum flacher Struktur (eindeutige Namen).
In einem Namensraum hierarchischer Struktur kann derselbe Namen in versch. Kontexten definiert sein.
Das **Verzeichnis** (*directory*) ist fundamental für die Hierarchiebildung.
Ein Verzeichniseintrag (*directory entry*) speichert Abbildung eines Names auf Informationsstruktur(*inode*).

inode: aggregiert wesentliche Attribute eines benannten Gegenstandes:
- user ID (Eigentümer)
- group ID (Gruppenzugehörigkeit)
- Rechte (lesen, schreiben, ausführen)
- Zeitstempel
- Anzahl der Verweise (*hard link*-Zähler)
- Typ: Verzeichnis, sym-link, pipe, socket, Gerätedatei, reguläre Datei

und besitzt in einem Namensraum eine eindeutige numerische Adresse, die *inode number*.
inode table: array von Indexknoten
directory: Abbildungstabelle, übersetzt symbolisch repräsentierte Namen in Indexknotennummern
file: abgeschl. Einheit zusammenhängender Daten beliebiger Repräsentation, Struktur und Bedeutung

8.2 Bindung und Auflösung

name binding kommt zuerst, also die Abbildung der symbolischen Adresse in eine numerische Adresse.
name resolution kommt später, die Umsetzung der symbolischen Adresse in eine numerische Adresse.

Statische Namensauflösung:
- **Kompilierer**: verteilt Programmtext und -daten auf Programmsegmente
- **Assemblierer**: ordnet Programmsymbolen Werte und Attribute zu
  generiert Symbol- und Relokationstabellen für den Binder
- **Binder**: platziert das gebundene Programm im Adressraum
  produziert das Lademodul dazu für das Betriebssystem
- **Lader**: legt den gefüllten Adressraum im Arbeitsspeicher ab

Ergebnis der Assemblierung ist das Binde- oder Objektmodul, das u.a. zwei Tabellen enthält.
⇒ Symboltabelle: listet alle def. Symbole und assoziiert jedes Symbol mit Werten/Attributen
⇒ Relokationstabelle: listet alle undef. Stellen (an denen def. Adresswerte noch fehlen)
jedes Text- u. Datenelement besitzt eine vorläufige Adresse, die relativ zur Basisadresse 0 ausgerichtet ist.

9 Betriebsarten

9.1 Stapelverarbeitung

Stapelbetrieb: seq. Abwicklung von Aufgaben, Aufgabenverteilung auch auf systeminterne Abläufe:
- abgesetzter Betrieb, überlappte Ein-/Ausgabe
- überlappte Auftragsverarbeitung, abgesetzte Ein-/Ausgabe

Mehrprogrammbetrieb: hält mehrere Programme im Hauptspeicher.
Hauptspeicherknappheit wird durch dynamisches Laden begegnet, Prozessor wird im Multiplexverfahren betrieben. Schutzvorkehrungen schotten Prozessadressräume voneinander ab für sicheren Simultanbetrieb. Leistungssteigerung mittels *multi-programming* und Simultanverarbeitung (*multiprocessing*).

- Multiplexen der CPU: Nebenläufigkeit (*concurrency*)
  Passives Warten: Prozess, der warten muss, gibt die CPU zugunsten eines anderen Prozesses ab.
- *Abschottung* der sich in Ausführung bef. Programme: Adressraumschutz (*address space protection*)
  Schutz durch Eingrenzung und Segmentierung. Begrenzungsregister legen Grenzen im realen Adressraum fest. Zugriffsfehler führt zum Abbruch d. Programmausführung → Trap
- Überlagerung unabhängiger Programmteile: Segmentierung (*segmentation*)
  Programm in hinreichend kleine Teile, die nicht ständig im Hauptspeicher vorhanden sein müssen.

Multistapelbetrieb:
Simultanverarbeitung mehrerer Auftragsströme (Betriebssystem als *multi-stream batch monitor*)
parallele Verarbeitung von Auftragsströmen sequentieller Programme gleichzeitig im Multiplexverfahren
Stapelwechsel kommen kooperativ, verdrängend oder als Kombination von beiden zustande

9.2 Dialog- und Echtzeitverarbeitung

9.2.1 Mehrzugangsbetrieb

Dialogbetrieb: Benutzereingaben u. Verarbeitung wechseln sich anhaltend ab → dynamische Einplanung
Prozesse im Vordergrund und im Hintergrund vollziehen → Mischbetrieb:

- *Vordergrund*: echtzeitabhängige Prozesse ⇀ Echtzeitbetrieb
- *Mittelgrund*: E/A-intensive Prozesse ⇀ Dialogbetrieb
- *Hintergrund*: CPU-intensive Prozesse ⇀ Stapelbetrieb

Dialog mit teilhabenden Prozessen: *client/server system*

9.2.2 Echtzeitbetrieb

Echtzeitbetrieb: Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten werden ereignisgesteuert oder zeitgesteuert verwendet.

Terminvorgaben: was passiert bei nicht termingerecht geleisteten Berechnung?

- *weich* (soft), auch schwach
  Ergebnis weiterhin von Nutzen, verliert mit weiterem Zeitverzug zunehmend an Wert, tolerierbar
- *fest* (firm), auch stark
  Ergebnis wird verworfen, Prozess abgebrochen und neu gestartet. Terminverletzung ist tolerierbar.
- *hart* (hard), auch strikt
  Verspätung kann zur Katastrophe führen. Terminverletzung ist keinesfalls tolerierbar, aber möglich.

9.2.3 Systemmerkmale

symmetric multiprocessing: ≥ 2 gleiche Prozessoren, eng gekoppelt über gemeinsames Verbindungssystem
jeder Prozessor hat gleichberechtigten Zugriff auf Hauptspeicher (*shared-memory access*) u. Peripherie.
Zugriff auf Hauptspeicher ist für alle Prozessoren gleichförmig (*uniform memory access*).

shared-memory processor: Parallelrechnersystem, in dem alle Prozessoren den Haupspeicher mitbenutzen, ohne jedoch einen gleichberechtigen/-förmigen Zugriff darauf haben zu müssen
architektonische Merkmale der Befehlssatzebene:

- *asymmetrisch*: hardwarebedingter, zwingender asym. Betrieb, Programme ggf. prozessorgebunden
- *symmetrisch*: anwendungsorientierter Betrieb mögl., BS legt Multiprozessorbetriebsart fest

Sicherheit: Schutz vor unautorisierten Zugriffen durch Prozesse.
Lösungen:

- jeden Prozessadressraum in Isolation betreiben
- Prozessen eine Befähigung zum Zugriff erteilen
- Objekten eine Zugriffskontrollliste geben

Multics: segmentbasierter Ringschutz, Ringwechsel kontrolliert durch die Hardware, Ringfaults

Grad an Mehrprogrammbetrieb:
selektive Überlagerung des Hauptspeichers durch programmiertes dynamisches Laden hat seine Grenzen. Umlagerung der Speicherbereiche gegenwärtig nicht ausführbereiter Programme verschiebt die Grenze nach hinten.
Granularität der Umlagerungseinheiten:

- *Seitennummerierung*: feste Größe, interne Fragmentierung
- *Segmentierung*: variable Größe, externe Fragmentierung
- *seitennummerierte Segmentierung*: Segmente in Seiten untergliedert, interne Fragmentierung

10 Dateisysteme

Festplatten: blockorientierter, wahlfreier Zugriff. Blockgröße zw. 32 und 4096 Bytes. Zugriff erfordert Positionierung des Schwenkarms auf den richtigen Zylinder und Warten auf den entsprechenden Sektor. Blöcke sind nummeriert. Es gibt zudem verschiedene Zylinder (Satz von übereinanderliegenden Spuren).

10.1 Speicherung von Dateien

Kontinuierliche Speicherung:
Datei wird in Blöcken mit aufsteigenden Blocknummern gespeichert
→ Nummer d. ersten Blocks und Folgeblöcke speichern
Vorteile: Zugriff mit minimaler Positionierzeit des Schenkarms
Probleme: Finden des freien Platzes auf der Festplatte, Fragmentierung, Größe nicht im Voraus bekannt
Variation: Unterteilen in Folgen von Blöcken → Verschnitt

Verkettete Speicherung:
Blöcke einer Datei sind verkettet
Probleme: Speicher für Verzeigerung geht von den Nutzdaten im Block ab, schlechter direkter Zugriff
Alternative: Verkettung wird in speziellen Plattenblocks (*File Allocation Table*) gespeichert

- *Vorteile*: kompletter Inhalt des Blocks ist nutzbar, mehrfache Speicherung der FAT
- *Probleme*: mind. ein Block muss zusätzl. geladen werden, aufwändige Suche nach zugehörigem Datenblock, häufiges Positionieren des Kopfes bei verstreuten Datenblöcken

Indiziertes Speichern:
Spezieller Plattenblock enthält Blocknummern der Datenblocks einer Datei
Problem: feste Anzahl von Blöcken im Indexblock
Variante: Einsatz von mehreren Stufen der Indizierung → mehrere Blöcke müssen geladen werden

Freispeicherverwaltung:
Bitvektoren zeigen für jeden Block Belegung an, verkettete Listen repräsentieren freie Blöcke

10.2 UNIX Dateisysteme

System V File System: Blockorganisation
Boot-Block enthält Informationen zum Laden eines initialen Programms
Super-Block enthält Verwaltungsinformation für ein Dateisystem:
Blöcke, # Inodes, # und Liste freier Blöcke und Inodes, Attribute (z.B. Modified-flag)

BSD 4.2 (Berkeley Fast File System): Blockorganisation in Zylindergruppen
Kopie des Super Blocks in jeder Zylindergruppe
freie Inodes u. Datenblöcke im Cylinder Group Block
Datei möglichst innerhalb einer Zylindergruppe gespeichert

Linux EXT2/3/4: Blockorganisation
EXT2: ähnliches Layout wie BSD FFS, Blockgruppen jedoch unabhängig von Zylindern
EXT3: Erweiterung als Journaling-File-System
EXT4: Einführung von Extents (siehe NTFS) und viele neue Details

10.3 Windows NTFS

Datei: bel. Inhalt, Rechte verknüpft mit NT-Benutzern und - Gruppen
kann automatisch komprimiert oder verschlüsselt gespeichert werden, große Dateien bis zu 2^{64} Bytes lang
Hard links: mehrere Einträge derselben Datei in verschiedenen Katalogen möglich

Dateiinhalt: Sammlung von Streams (einfache, unstrukturierte Folge von Bytes)
dynamisch erweiterbar, Syntax: dateiname:streamname
Basiseinheit "Cluster": wird auf Menge von hintereinanderfolgenden Blöcken abgebildet
Basiseinheit "Strom": jede Datei kann mehrere Datenströme speichern, einer für eigentlichen Daten
(MS-DOS) Dateiname, Rechte, Attribute, Zeitstempel werden in eigenen Datenströmen gespeichert
File-Reference: bezeichnet eindeutig eine Datei oder einen Katalog

Master-File-Table: große Tabelle mit gleich langen Elementen, Rückgrat des gesamten Systems
entsprechender Eintrag für eine File-Reference enthält Informationen über Streams der Datei
Eintrag für kurze Datei: Standardinfo, Dateiname, Zugriffsrechte, Daten
Streams:

- Standard-Information (immer in der MFT):
  Länge, Standard-Attribute, Zeitstempel, # Hard links, Sequenznummer der gültigen File-Reference
- Dateinamne (immer in der MFT), kann mehrfach vorkommen (Hard links)
- Zugriffsrechte, eigentliche Daten
- mögliche weitere Streams:
  Index
  Indexbelegungstabelle: Belegung der Struktur eines Index
  Attributliste (immer in der MFT): benötigt, falls nicht alle Streams in einen MFT Eintrag passen
  Streams mit beliebigen Daten

Metadaten: alle Metadaten werden in Dateien gehalten

- MFT
- MFT Kopie
- Log File: protokollierte Änderungen am Dateisystem
- Volume Information: Name, Größe und ähnliche Attribute des Volumes
- Attributtabelle: definiert mögliche Ströme in den Einträgen
- Wurzelkatalog
- Clusterbelegungstabelle: Bitmap für jeden Cluster des Volumes
- Boot File: enthält initiales Programm zum Laden sowie ersten Cluster der MFT
- Bad Cluster File: alle nicht lesbaren Cluster der Platte

10.4 Dateisysteme mit Fehlererholung

Metadaten und geöffnete Dateien werden im Hauptspeicher gehalten (Dateisystem-Cache)
→ effizienter Zugriff
Konsistenz zwischen Cache und Platte muss regelmäßig hergestellt werden

- *synchrone Änderungen*: Operation kehrt erst zurück, wenn Änderungen auf Platte
- *asynchrone Änderungen*: Änderungen erfolgen nur im Cache, Operation kehrt sofort zurück
  Synchronisation mit der Platte erfolgt später

Konsistenzprobleme: Cache-Inhalte und akt. E/A-Operationen gehen verloren, inkonsistente Metadaten

Journaling-File-Systems:
Zusätzl. zum Schreiben der Daten und Meta-Daten Protokoll über Änderungen → Log-based Recovery
Protokollierung: jeder Einzelvorgang einer Transaktion → Logeintrag, danach Änderung am Dateisystem
Fehlererholung: beim Booten überprüfen, ob protokollierte Änderungen vorhanden
→ können wiederholt/abgeschlossen werden (*Redo*), angef. Transakt. werden rückgängig gemacht (*Undo*)
→ Log vollständig: *Redo* wenn nötig. Log unvollständig: *Undo*
Checkpoints: gelegentlich wird für konsistenten Zustand auf Platte gesorgt und dieser protokolliert

Copy-on-Write-/Log-Structured-File-Systems:
Alle Änderungen erfolgen auf Kopien, der Inhalt veränderter Blöcke wird in einen neuen Block geschrieben
Vorteile: gute Schreibeffizienz, Datenkonsistens bei Systemausfällen, Schnappschüsse einfach realisierbar
Nachteile: starke Fragmentierung

Fehlerhafte Plattenblöcke: Blöcke, die beim Lesen Fehlermeldungen erzeugen
Hardwarelösung: Platte und Plattencontroller bemerken selbst fehlerhafte Blöcke und maskieren diese aus, Zugriff auf den Block wird vom Controller automatisch auf einen gesunden Block umgeleitet
Softwarelösung: File-System bemerkt fehlerhafte Blöcke und markiert diese auch als belegt

10.5 Datensicherung

Ziel: Schutz vor Totalausfall von Platten

Sichern auf Tertiärspeicher:
braucht viel Zeit, inkrementelle Backups sichern nur Änderungen ab einem bestimmten Zeitpunkt

Einsatz mehrerer (redundanter) Platten:

- *RAID 0*: Daten werden über mehrere Platten gespeichert
  keinerlei Datensicherung: Ausfall einer Platte lässt Gesamtsystem ausfallen
- *RAID 1*: Daten werden auf zwei Platten gleichzeitig gespeichert
  eine Platte kann ausfallen, aber doppelter Speicherbedarf
- *RAID 4*: Daten werden über mehrere Platten gespeichert, eine enthält Parität
  Paritätsblock aus byteweisen XOR-Verknüpfungen,
  eine Platte kann ausfallen, prinzipiell beliebige Plattenanzahl (ab 3)
  Nachteil: Paritätsplatte hoch belastet
- *RAID 5*: Paritätsblock wird über alle Platten verstreut
- *RAID 6*: doppelte Paritätsblöcke