Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Übung V
Speicherverwaltung/Seitenersetzung

Michael Schöbel

Betriebssystemarchitektur I – WS 2007/08

24. Januar 2008

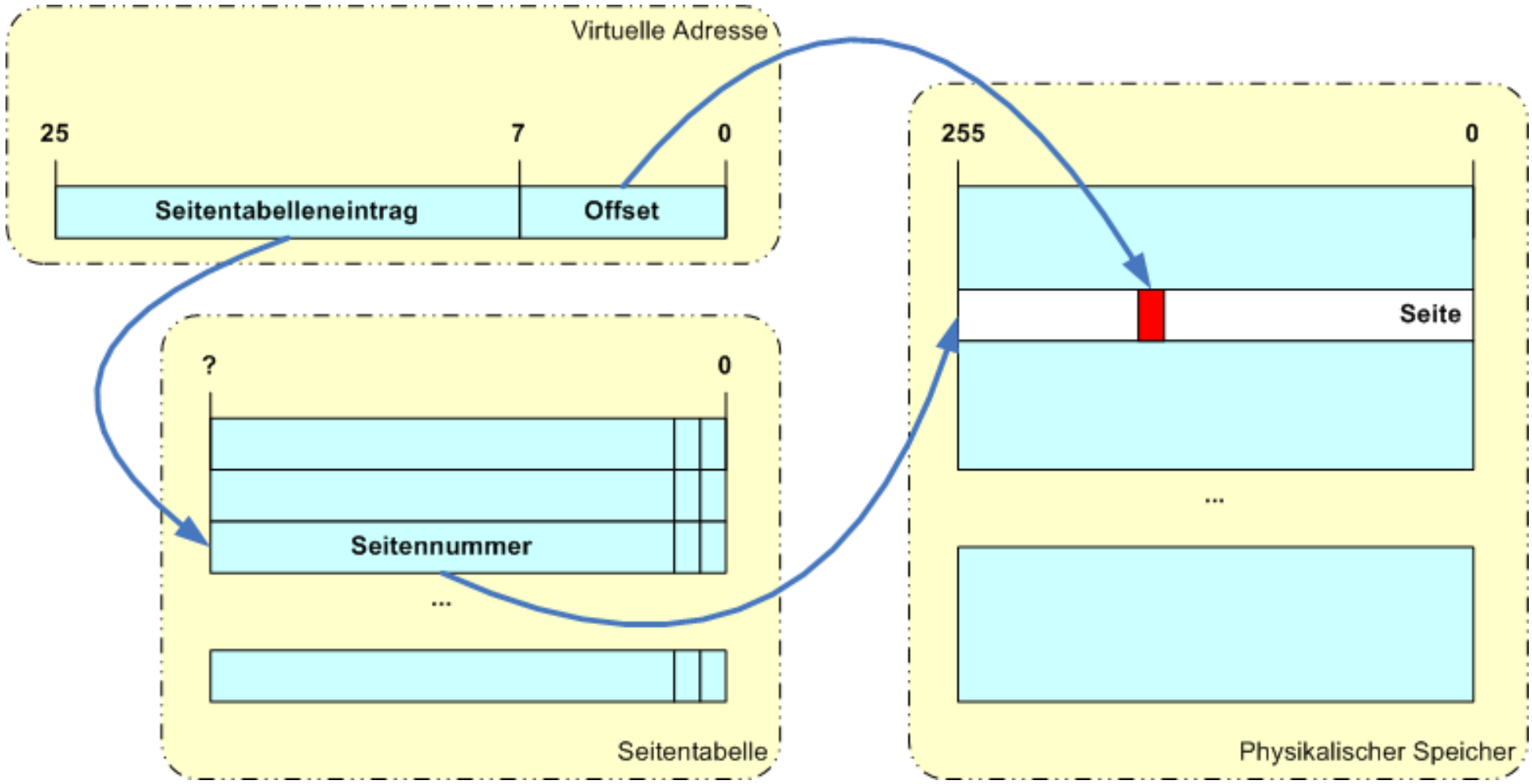
Agenda

- **Adressumsetzung**
- **Seitenersetzungsalgorithmen**

Adressumsetzung (I)

- **Beispielsystem**
  - 26 Bit Computer
  - 256 Byte Seitengröße
  - 2 Bit Status pro Seiteneintrag
  - einstufige Umsetzung mit Seitentabellen
  - 24 MByte RAM
  - 5 MByte belegt durch Betriebssystem
  - 8 Prozesse

Adressumsetzung (II)



Adressumsetzung (III)

- **Größe des virtuellen Adressraums**
  - 2^26 Bit = 67108864 Byte = 64 MByte
- **Anzahl der Seitentabelleneinträge**
  - 2^26 Bit / 256 Byte = 262144 Einträge
- **Größe eines Seitentabelleneintrages**
  - 26 Bit – 8 Bit Offset + 2 Bit Status = 20 Bit
- **Größe der Seitentabelle**
  - 20 Bit * 262144 Einträge = 5242880 Bit = 0,625 MByte

Adressumsetzung (IV)

- **Maximales Workingset eines Prozesses**
  - Gesamt RAM: 24 MByte
  - - Seitentabellen für 8 Prozesse (8*0,625 MByte = 5 MByte)
  - - Speicherbelegung durch Betriebssystem (5 MByte)
  - = 24 MByte – 10 MByte = 14 MByte maximales Workingset

Seitenersetzungsalgorithmen Überblick (I)

- **Begrenzte Workingset-Größe**
  - Zugriff auf nicht im physikalischen Speicher vorhandene Seite
  - Einlagerung der benötigten Seite
  - Ersetzung einer anderen Seite
- **Ziel: möglichst wenige Page-Faults**
- **Verschiedene Algorithmen / Heuristiken**
  - Workingset-Verwaltung und Seitenersetzung
  - Vergleich über Referenzzugriffsliste

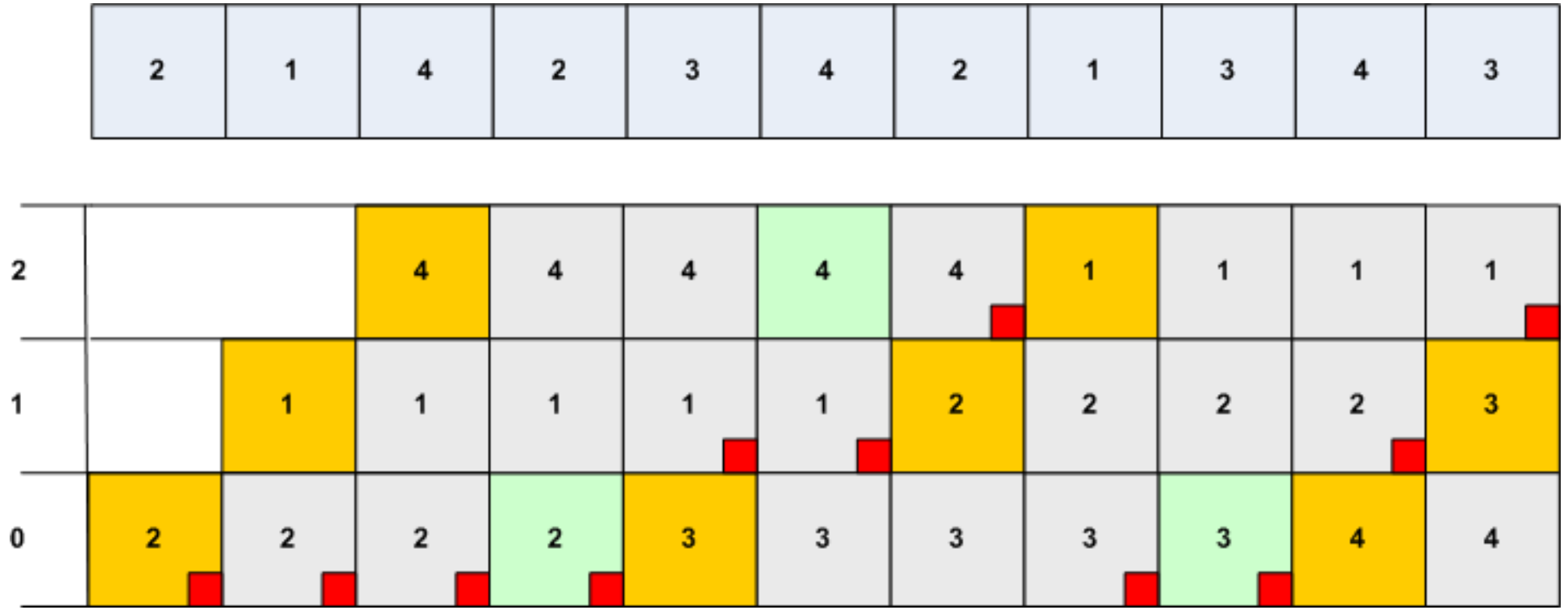
Seitenersetzungsalgorithmen Überblick (II)

- **Strategien zur Seitenersetzung**
  - First-In-First-Out (FIFO)
  - Second-Chance (SC)
  - Least-Recently-Used (LRU)
  - Optimaler Algorithmus

- **Unterschiede**
  - Page-Fault-Rate
  - Implementierbarkeit

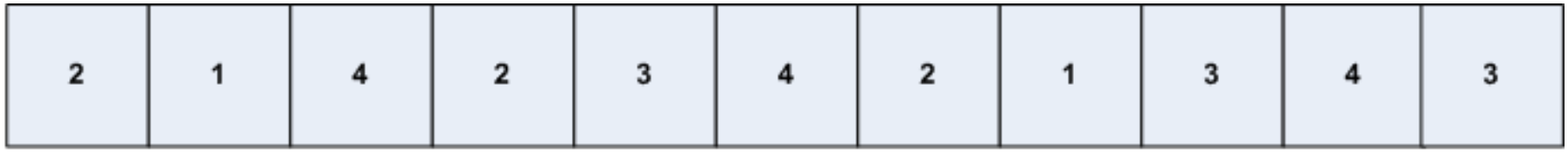
Seitenersetzungsalgorithmen
First-In-First-Out (FIFO)

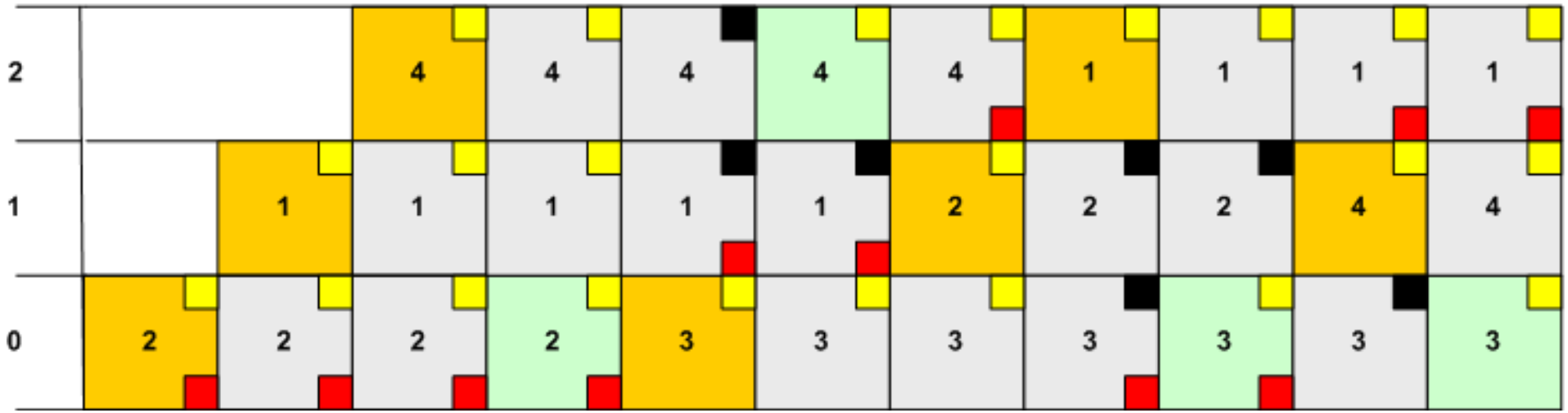
- Ersetzung der „ältesten“ Speicherseite
  - Queue der Speicherseiten, einfache Implementierung



Seitenersetzungsalgorithmen
Second-Chance (SC)

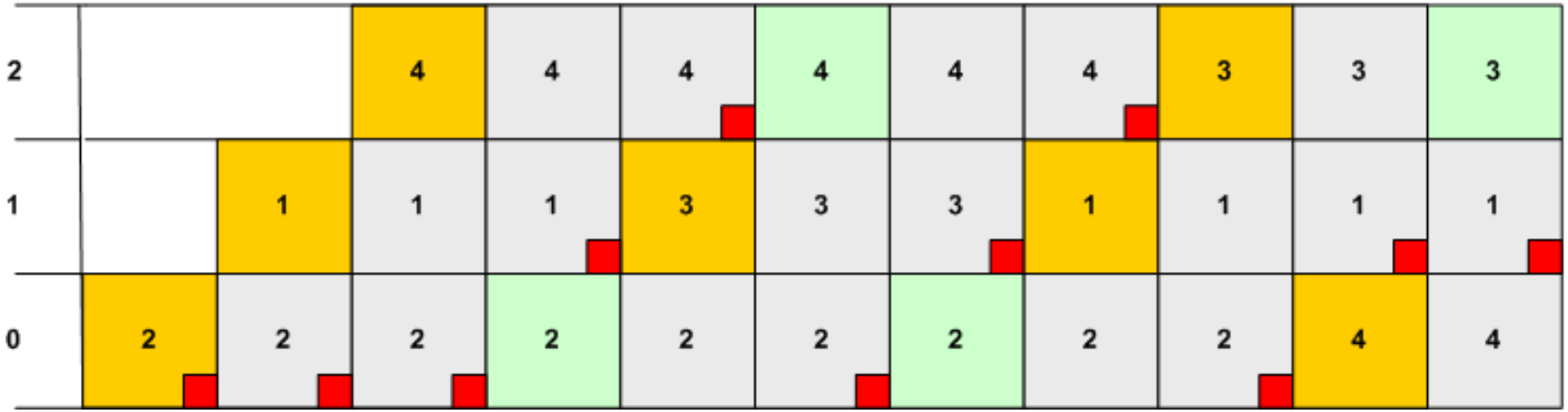
- Ersetzung der „ältesten“ Seite,
  auf die nicht „kürzlich“ zugegriffen wurde
  - einfache Implementierung möglich über Statusbits der Seiten





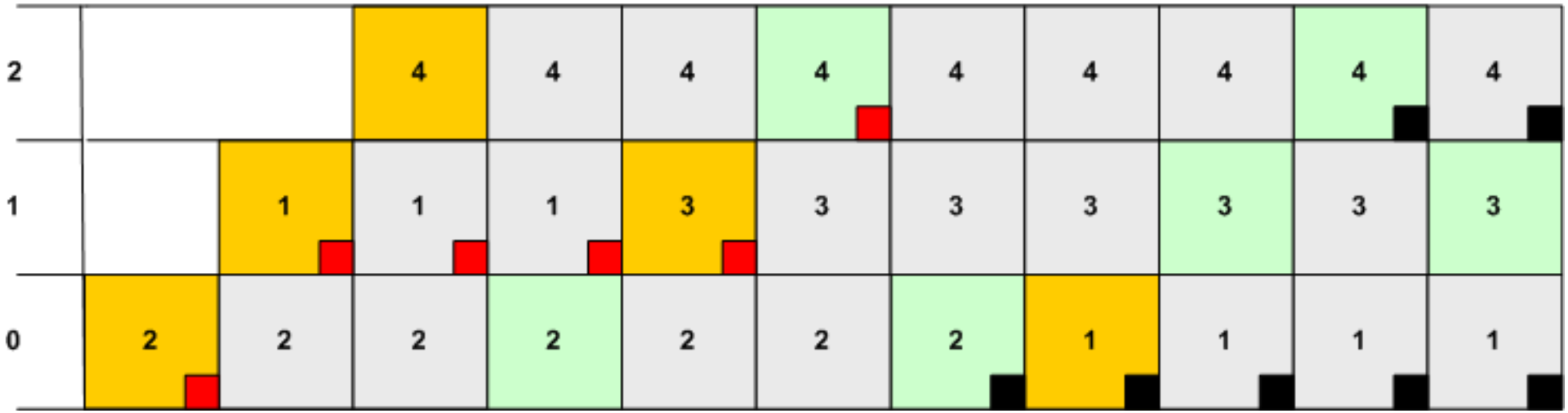
Seitenersetzungsalgorithmen
Least-Recently-Used (LRU)

- Ersetzen der Seite, die am längsten nicht mehr verwendet wurde
  - Nur aufwändig implementierbar; nach Zugriff alle Seiten prüfen



Seitenersetzungsalgorithmen
Optimaler Algorithmus

- Ersetzen der Seite, die am längsten nicht mehr benötigt wird
  - Nicht implementierbar; abhängig vom Programmablauf



Belady‘s Anomaly

- **Untersuchung: Einfluss der Anzahl der Seitenrahmen**
  - Intuition: mehr verfügbare Seiten → weniger Page-Faults, (bei gleicher Zugriffssequenz)
  - Übungsblatt: Gegenbeispiel für FIFO-Algorithmus

Seitenersetzungsalgorithmen „Reale“ Betriebssysteme (I)

- **Weitere Aspekte der Speicherseitenersetzung**

  - *Demand-Paging*: Einlagern der fehlenden Seite + einige Seiten, die (logisch) davor und/oder danach liegen
  - *Prefetcher*: Überwachen der benötigten Seiten beim Booten oder Starten einer Anwendung → beim nächsten Start berücksichtigen

  - Lokale Seitenersetzungsalgorithmen:
    - Nur auf Workingset eines Prozesses bezogen
  - Globale Seitenersetzungsalgorithmen:
    - Auf alle vorhandenen Speicherseiten bezogen

Seitenersetzungsalgorithmen
„Reale“ Betriebssysteme (II)

- **Windows**
  - Kombination von lokalen und globalen Konzepten
    - Global: Workingset-Trimming
    - Lokal: Aging-Algorithmus → LRU Nachbildung
- **Linux (2.6.11 Kernel)**
  - Keine lokalen Seitenersetzungsalgorithmen
    - Ziel: Optimale Ausnutzung des Arbeitsspeichers
  - Page Frame Reclaiming Algorithm
    - Globaler Ansatz → freie Seiten generieren
    - Kombination verschiedenster Heuristiken
      - Art der Seitenverwendung (Caches)
      - Einfaches LRU (*in-use* / *unused*)
      - Seitenstatus (*un-modified* / *modified*)