Grundlagen von Betriebssystemen

Speichervirtualisierung mit gekachelter Speicherverwaltung

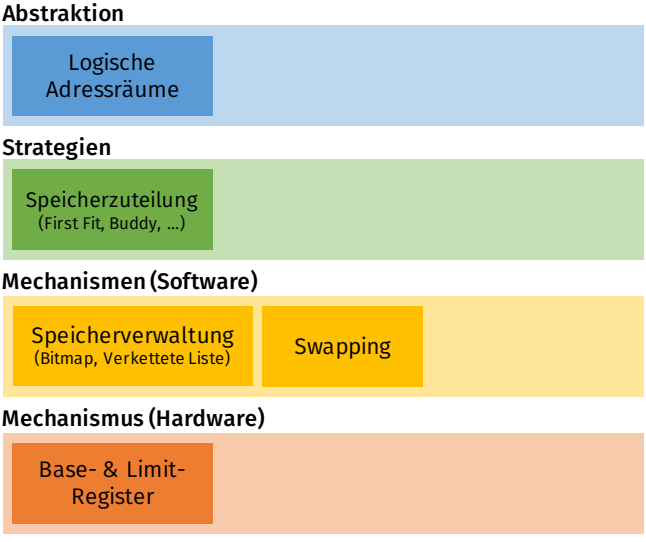
05. November 2020

Peter Ulbrich

Proseminar: Konzepte von Betriebssystemkomponenten

technische universität
dortmund

Rekapitulation



Logische Adressräume

- *Abstrakte Speicheradressen*
- Adressraumsegmentierung
- Dynamische Relokation

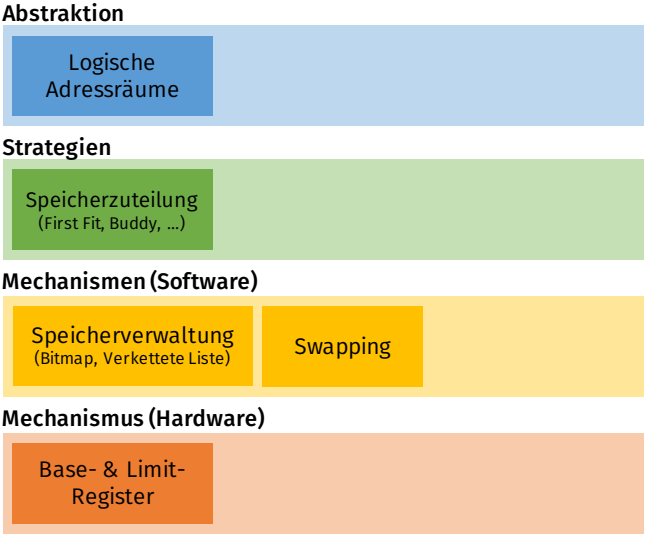
→ Mehrprogrammbetrieb

Speicherverwaltung

- Datenstrukturen
- Zuteilungsalgorithmen

→ **Wo** ist noch Platz?

Rekapitulation



Logische Adressräume

- *Abstrakte Speicheradressen*
- Adressraumsegmentierung
- Dynamische Relokation

→ Mehrprogrammbetrieb

Speicherverwaltung

- Datenstrukturen
- Zuteilungsalgorithmen

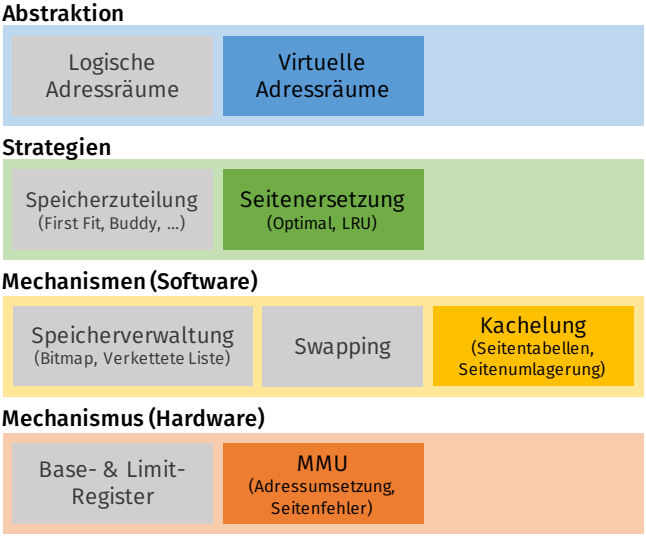
→ **Wo** ist noch Platz?

Grenzen

→ Adressraumüberlauf?

→ Simultaner Mehrprogramm-betrieb?

Lernziele dieser Vorlesung



Virtuelle Adressräume

- *Abstrakter Speicher*

→ Entkopplung Speicherort

Gekachelte Speicherverwaltung

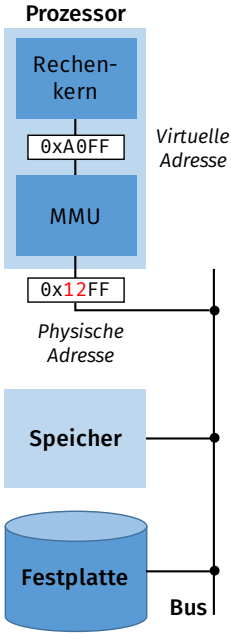
- Seitentabellen
- Adressumsetzung
- Seitenersetzung

→ **Welches** Datum ist **wann** im Speicher?

Gliederung

Gliederung

Grundlagen virtueller Adressräume



Simultanes ausführen von Programmen die potentiell größer sind als der verfügbare physische Speicher

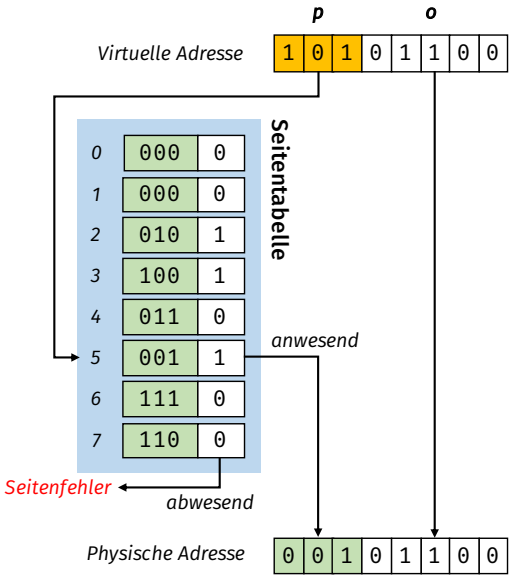
Virtueller Speicher $\rightsquigarrow$ abstrakter Speicher

- Virtueller Adressraum $(A_v) \mapsto$ Partielle Abbildung $f : A_v \mapsto A_p$
- Eine virtuelle Adresse erscheint echt, bildet jedoch **entweder auf den Haupt- oder Ablagespeicher** ab

$\rightarrow$ Mehr Arbeits- als Hauptspeicher

Seitenbasierte (gekachelte) Speicherverwaltung

- Einteilung in (virtuelle) Seiten und (physischen) Seitenrahmen fester Größe
- Adressumsetzung mittels spezieller Hardwareeinheit (MMU)
- Betriebssystem übernimmt die Ein- bzw. Auslagerung



Memory Management Unit

- Verarbeitet virtuelles Adressetupel (p, o):
  - **Seitennummer p** im (linearen) Adressraum des Prozesses
  - **Offset o** innerhalb der Seite
- Bezogen auf die passende Seitentabelle

→ Erzeugt eine physische Adresse, welche auf den Speicherbus gelegt wird

Seitenfehler

- Erfolgt bei nicht abgebildeter Seite (Präsenz-Bit = 0)
- Synchrone Programmunterbrechung (trap)

→ Bedingt Einlagerung der adressierten Seite durch das Betriebssystem

Seitentabellen (1)

Seitentabelle

- Beschreibt den einem Prozess zugewiesenen Adressraum (ggf. mehrere davon)
- Dient der Abbildung von virtuellen Seiten auf Seitenrahmen (physischen Speicher)
- Menge von Seitendeskriptoren mit der virtuellen Seitenummer als Index

Seitentabellen (1)

Seitentabelle

- Beschreibt den einem Prozess zugewiesenen Adressraum (ggf. mehrere davon)
- Dient der Abbildung von virtuellen Seiten auf Seitenrahmen (physischen Speicher)
- Menge von Seitendeskriptoren mit der virtuellen Seitenummer als Index

Seitendeskriptor

Seitendeskriptor

- Beschreibt die statischen/dynamischen Eigenschaften einer Seite:

Seitenrahmennummer:
- Gemäß der Seitengröße ausgerichtete physische Adresse des Seitenrahmens

Attribute:
- Präsenz-Bit (Seite im Hauptspeicher präsent)
- Schutz-Bits (Lese-/Schreib-/Ausführungsschutz)
- Modifikations-Bit (Änderungsindikator, *dirty bit*)
- Referenz-Bit (Zugriffsindikator lesen/schreibend)

Seitentabellen (1)

Seitentabelle

- Beschreibt den einem Prozess zugewiesenen Adressraum (ggf. mehrere davon)
- Dient der Abbildung von virtuellen Seiten auf Seitenrahmen (physischen Speicher)
- Menge von Seitendeskriptoren mit der virtuellen Seitenummer als Index

Seitendeskriptor

Seitendeskriptor

- Beschreibt die statischen/dynamischen Eigenschaften einer Seite:

Seitenrahmennummer:
- Gemäß der Seitengröße ausgerichtete physische Adresse des Seitenrahmens

Attribute:
- Präsenz-Bit (Seite im Hauptspeicher präsent)
- Schutz-Bits (Lese-/Schreib-/Ausführungsschutz)
- Modifikations-Bit (Änderungsindikator, *dirty bit*)
- Referenz-Bit (Zugriffsindikator lesen/schreibend)

⚠ Aufbau ist grundsätzlich stark plattformabhängig, vorgegeben durch die MMU

Umgang mit großen bzw. dünn besetzten Adressräumen

Mehrstufige Seitentabellen

- Nicht benötigte Seitentabellen auslagern
- Adressraumsegmentierung (Text, Daten, Stapel) fördert Löcher im Adressraum
- Aufteilung in zweistufige Seitentabellen
  - 2x 10-Bit-Felder p_1 und p_2 für Seitennummern + 12-Bit-Offset o

  $\rightarrow$ Seitentabellen mit je nur 1024 Einträgen

? Umgang mit großen bzw. dünn besetzten Adressräumen

Mehrstufige Seitentabellen

- Nicht benötigte Seitentabellen auslagern
- Adressraumsegmentierung (Text, Daten, Stapel) fördert Löcher im Adressraum
- Aufteilung in zweistufige Seitentabellen
  - 2x 10-Bit-Felder p_1 und p_2 für Seitenummern + 12-Bit-Offset o

  → Seitentabellen mit je nur 1024 Einträgen

Invertierte Seitentabellen [3]

- Mehrstufigkeit (> 3) wird ineffizient und stößt bei 64-Bit-Rechensystemen schnell an Grenzen ($\approx 2^{52}$ Einträge)
- Seitendeskriptoren lediglich für Seitenrahmen speichern
  - 32 GiB physischer Speicher erfordern lediglich ≈8,4 Mio. Einträge (= 64 MiB)
  - Zuordnung Prozess zu Seite wird schwieriger (Suche statt Index)

  → Hash-Tabellen lösen das Problem

Gliederung

Grundlagen der Seitenersetzung

Zeitmultiplexen von Seitenrahmen durch Umlagerung

Vorraussetzungen die zu erbringen sind:

- Geeignete **Mechanismen** zur Seitenumlagerung (vgl. Folie 4 ff.)
- Ein Seitenrahmen muss zur Ausnahme der angeforderten Seite verfügbar sein
  - Freie Seitenrahmen ermöglichen direkte Speicherzuteilung (vgl. Kapitel 4)
  - Andernfalls muss ein belegter Seitenrahmen freigegeben werden
  - ⚠ Freigeben bedingt ggf. die vorherige Auslagerung
- **Strategie** für die Suche nach einem geeigneten Kandidaten

  → *Welche* Seite im Hauptspeicher ist ersetzbar?

Grundlagen der Seitenersetzung

Zeitmultiplexen von Seitenrahmen durch Umlagerung

Vorraussetzungen die zu erbringen sind:

- Geeignete **Mechanismen** zur Seitenumlagerung (vgl. Folie 4 ff.)
- Ein Seitenrahmen muss zur Ausnahme der angeforderten Seite verfügbar sein
  - Freie Seitenrahmen ermöglichen direkte Speicherzuteilung (vgl. Kapitel 4)
  - Andernfalls muss ein belegter Seitenrahmen freigegeben werden
  - Freigeben bedingt ggf. die vorherige Auslagerung
- **Strategie** für die Suche nach einem geeigneten Kandidaten

  → *Welche* Seite im Hauptspeicher ist ersetzbar?

Die ideale Seitenersetzungsstrategie (Optimaler Algorithmus) [1]

- Ersetze die Seite die *am längsten nicht referenziert* ***wird***
  - Höchste Anzahl an Instruktionen *bis* zum erneuten Zugriff
  - Diese Information ist i.A. nicht Verfügbar ⇝ Vorhersagbarkeit
- Nicht realisierbar, jedoch als Vergleichskandidat geeignet

Der Least-Recently-Used-Algorithmus (LRU)

(Gute) Annäherung an den optimalen Algorithmus

Der Least-Recently-Used-Algorithmus (LRU)

- Ersetze die Seite die *am längsten nicht mehr referenziert* ***wurde***
  - Der Beobachtung folgend, dass Zugriffe meist lokal gehäuft sind
  - Eine alte Referenz deutet darauf hin, dass die Seite nicht mehr gebraucht wird
- LRU ist realisierbar, jedoch **nicht billig**
  - Verkettete Liste aller Seiten, sortiert nach Alter $\rightsquigarrow$ Suche und Manipulation aufwendig

Der Least-Recently-Used-Algorithmus (LRU)

(Gute) Annäherung an den optimalen Algorithmus

Der Least-Recently-Used-Algorithmus (LRU)

- Ersetze die Seite die *am längsten nicht mehr referenziert* ***wurde***
  - Der Beobachtung folgend, dass Zugriffe meist lokal gehäuft sind
  - Eine alte Referenz deutet darauf hin, dass die Seite nicht mehr gebraucht wird
- LRU ist realisierbar, jedoch **nicht billig**
  - Verkettete Liste aller Seiten, sortiert nach Alter $\rightsquigarrow$ Suche und Manipulation aufwendig

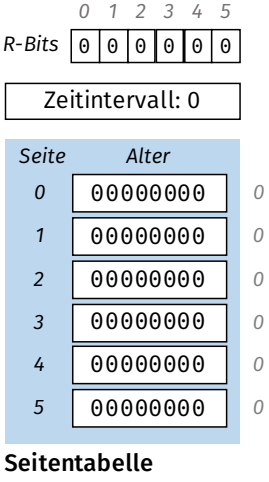
Umsetzung durch Hardwareunterstützung

- Erfordert einen Zähler C, welcher nach jedem Maschinenbefehl inkrementiert wird
- Jeder Eintrag der Seitentabelle wird um ein Feld für C ergänzt, welches bei jedem **Speicherzugriff** aktualisiert
  - Vergrößert den notwendigen Speicherbedarf enorm (C typischerweise 64 Bit)
  - Alternative Umsetzung durch Bitmatrizen ($n \times n$, für n Seitenrahmen)

$\rightarrow$ Betriebssystem sucht den Eintrag mit dem niedrigsten Zählerstand/Binärwert

Entsprechende Hardwareunterstützung fehlt meist (teuer)

Der Aging-Algorithmus



Seitentabelle

Simulation von LRU in Software [4, S.261 ff.]

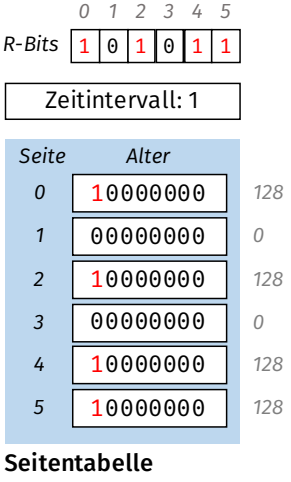
- Reduktion der zeitlichen Auflösung, anstelle von C werden Zeitintervalle gezählt $\rightsquigarrow$ Zeitgeberunterbrechung

$\rightarrow$ Wesentlich geringerer Speicherbedarf (typ. 8 Bit)

Ablauf in jedem Zeitintervall

- Die Zähler für das Alter werden nach rechts geschoben (*vergessen*)
- Das höchstwertige Bit des Zählers wird auf das Referenz-Bit gesetzt (*merken*)

Der Aging-Algorithmus



Seitentabelle

Simulation von LRU in Software [4, S.261 ff.]

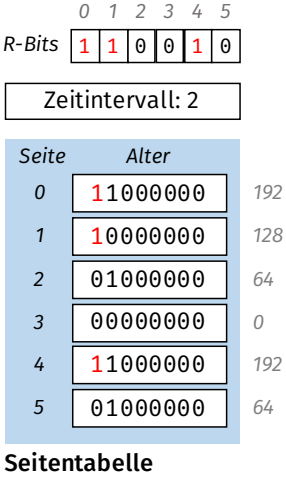
- Reduktion der zeitlichen Auflösung, anstelle von C werden Zeitintervalle gezählt $\rightsquigarrow$ Zeitgeberunterbrechung

$\rightarrow$ Wesentlich geringerer Speicherbedarf (typ. 8 Bit)

Ablauf in jedem Zeitintervall

- Die Zähler für das Alter werden nach rechts geschoben (*vergessen*)
- Das höchstwertige Bit des Zählers wird auf das Referenz-Bit gesetzt (*merken*)

Der Aging-Algorithmus



Seitentabelle

Simulation von LRU in Software [4, S.261 ff.]

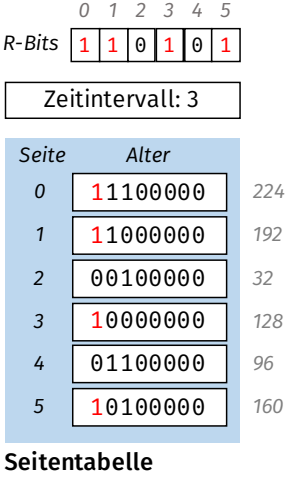
- Reduktion der zeitlichen Auflösung, anstelle von C werden Zeitintervalle gezählt $\rightsquigarrow$ Zeitgeberunterbrechung

$\rightarrow$ Wesentlich geringerer Speicherbedarf (typ. 8 Bit)

Ablauf in jedem Zeitintervall

- Die Zähler für das Alter werden nach rechts geschoben (*vergessen*)
- Das höchstwertige Bit des Zählers wird auf das Referenz-Bit gesetzt (*merken*)

Der Aging-Algorithmus



Seitentabelle

Simulation von LRU in Software [4, S.261 ff.]

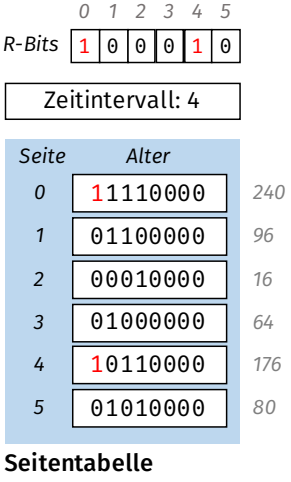
- Reduktion der zeitlichen Auflösung, anstelle von C werden Zeitintervalle gezählt $\rightsquigarrow$ Zeitgeberunterbrechung

$\rightarrow$ Wesentlich geringerer Speicherbedarf (typ. 8 Bit)

Ablauf in jedem Zeitintervall

- Die Zähler für das Alter werden nach rechts geschoben (*vergessen*)
- Das höchstwertige Bit des Zählers wird auf das Referenz-Bit gesetzt (*merken*)

Der Aging-Algorithmus



Seitentabelle

Simulation von LRU in Software [4, S.261 ff.]

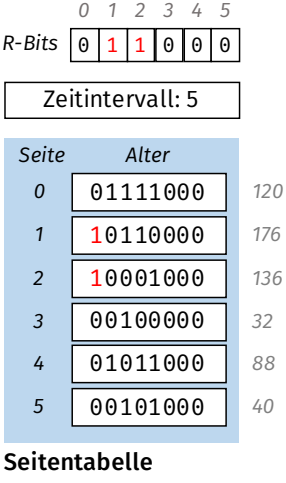
- Reduktion der zeitlichen Auflösung, anstelle von C werden Zeitintervalle gezählt $\rightsquigarrow$ Zeitgeberunterbrechung

$\rightarrow$ Wesentlich geringerer Speicherbedarf (typ. 8 Bit)

Ablauf in jedem Zeitintervall

- Die Zähler für das Alter werden nach rechts geschoben (*vergessen*)
- Das höchstwertige Bit des Zählers wird auf das Referenz-Bit gesetzt (*merken*)

Der Aging-Algorithmus



Seitentabelle

Simulation von LRU in Software [4, S.261 ff.]

- Reduktion der zeitlichen Auflösung, anstelle von C werden Zeitintervalle gezählt $\rightsquigarrow$ Zeitgeberunterbrechung

$\rightarrow$ Wesentlich geringerer Speicherbedarf (typ. 8 Bit)

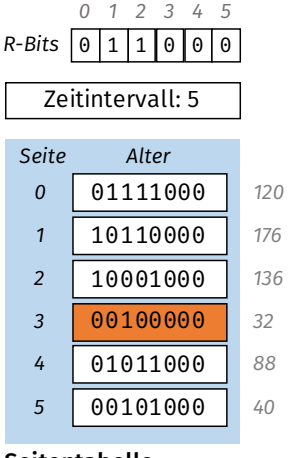
Ablauf in jedem Zeitintervall

- Die Zähler für das Alter werden nach rechts geschoben (*vergessen*)
- Das höchstwertige Bit des Zählers wird auf das Referenz-Bit gesetzt (*merken*)

Bei einem Seitenfehler

- Wird die Seite mit dem niedrigsten Zählerstand entfernt

Der Aging-Algorithmus



Seitentabelle

Simulation von LRU in Software [4, S.261 ff.]

- Reduktion der zeitlichen Auflösung, anstelle von C werden Zeitintervalle gezählt ⇝ Zeitgeberunterbrechung

→ Wesentlich geringerer Speicherbedarf (typ. 8 Bit)

Ablauf in jedem Zeitintervall

- Die Zähler für das Alter werden nach rechts geschoben (*vergessen*)
- Das höchstwertige Bit des Zählers wird auf das Referenz-Bit gesetzt (*merken*)

Bei einem Seitenfehler

- Wird die Seite mit dem niedrigsten Zählerstand entfernt

Gliederung

Resümee und Ausblick

Grundlagen virtuellen Speichers

- Arbeitsspeicher als **Abstraktion des physischen Hauptspeichers**
- **Simultaner Mehrprogrammbetrieb** von Prozessen die potentiell (zu) hohen Speicheranforderungen
- Realisierung in Form einer **gekachelten Speicherverwaltung**
  - Erfordert Unterstützung durch die Hardware: **Memory Management Unit**
  - Mechanismen des Betriebssystems zur **Seitenverwaltung** und **Seitenumlagerung**

Resümee und Ausblick

Grundlagen virtuellen Speichers

- Arbeitsspeicher als **Abstraktion des physischen Hauptspeichers**
- **Simultaner Mehrprogrammbetrieb** von Prozessen die potentiell (zu) hohen Speicheranforderungen
- Realisierung in Form einer **gekachelten Speicherverwaltung**
  - Erfordert Unterstützung durch die Hardware: **Memory Management Unit**
  - Mechanismen des Betriebssystems zur **Seitenverwaltung** und **Seitenumlagerung**

Strategien für die Seitenverwaltung

- Seitenersetzungsalgorithmen
  - **Optimaler Algorithmus** ist nicht realisierbar
  - **Least-Recently-Used-Algorithmus** (teuer) und dessen Annäherung durch **Aging**

Resümee und Ausblick

Grundlagen virtuellen Speichers

- Arbeitsspeicher als **Abstraktion des physischen Hauptspeichers**
- **Simultaner Mehrprogrammbetrieb** von Prozessen die potentiell (zu) hohen Speicheranforderungen
- Realisierung in Form einer **gekachelten Speicherverwaltung**
  - Erfordert Unterstützung durch die Hardware: **Memory Management Unit**
  - Mechanismen des Betriebssystems zur **Seitenverwaltung** und **Seitenumlagerung**

Strategien für die Seitenverwaltung

- Seitenersetzungsalgorithmen
  - **Optimaler Algorithmus** ist nicht realisierbar
  - **Least-Recently-Used-Algorithmus** (teuer) und dessen Annäherung durch **Aging**

→ **Weitere Strategien (z.B. auf Basis der Arbeitsmenge) und Umsetzungsaspekte (z.B. Seitenflattern, Segmentierung) im nächsten Kapitel**

Literaturverzeichnis (1)

[1] Belady, L. A.:
A Study of Replacement Algorithms for a Virtual-storage Computer.
In: *IBM Systems journal* 5 (1966), Nr. 2, S. 78–101

[2] Stallings, W. :
Operating Systems: Internals and Design Principles.
Upper Saddle River, NJ: Pearson/Prentice Hall„ 2009

[3] Talluri, M. ; Hill, M. D. ; Khalidi, Y. A.:
A New Page Table for 64-bit Address Spaces.
In: *Proceedings of the 15th ACM symposium on Operating systems principles*, 1995, S. 184–200

[4] Tanenbaum, A. S.:
Moderne Betriebssysteme.
Pearson Deutschland GmbH, 2009