Virtuelle Speicherverwaltung

Konzepte von Betriebssystem-Komponenten

Olessia Usik
olessia@freenet.de

20. Juni 2005

Gliederung

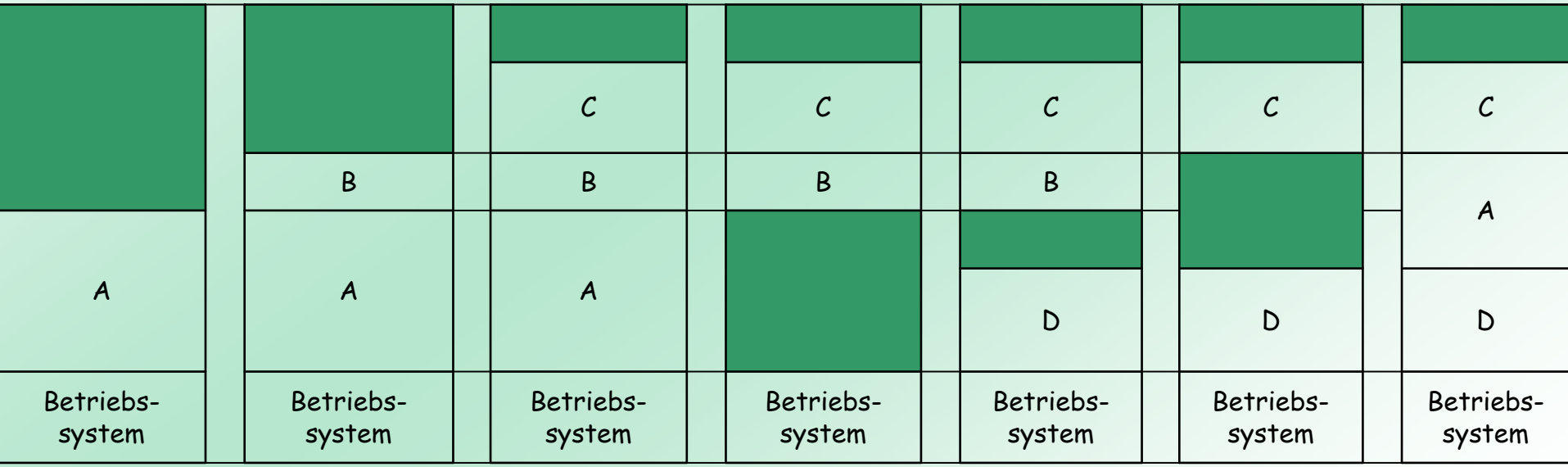
1. Einleitung
2. Swapping
3. Virtuelle Speicherverwaltung
   - 3.1 Segmentorientierter Speicher
   - 3.2 Seitenorientierter Speicher
   - 3.3 Paging vs. Segmentierung
   - 3.4 Pagingstrategien
   - 3.5 Behandlung von Seitenfehlern
4. Virtuelle Speicherverwaltung bei verschiedenen Betriebssystemen
   - 4.1 Unix
   - 4.2 Windows 2000
5. Zusammenfassung

Gliederung

2. Swapping

Swapping: Ein- bzw. Auslagern von nicht aktiven Prozessen

- es wird nur wirklich vorhandener physikalischer Hauptspeicher benutzt



2. Swapping

Für die Verwaltung von Speicher gibt es zwei Möglichkeiten:
- Speicherverwaltung mit **Bitmaps**
- Speicherverwaltung mit **verketteten Listen** (Freibereichslisten)

Positionierungsstrategien:

1. First Fit: Wähle den *ersten freien Bereich*,
der groß genug für das Programm ist.

2. Next Fit: Wähle den *nächsten freien Bereich*,
der groß genug für das Programm ist.
Starte an der Stelle,
an der das letzte Programm eingefügt wurde.

3. Best Fit: Wähle den *kleinstmöglichen freien Bereich*,
der groß genug für das Programm ist.

4. Worst Fit: Auswahl des jeweils *größten Loches*

5. Quick Fit: Getrennte *Listen für Löcher*
in einigen gebräuchlicheren Größen

2. Swapping

Nachteile von Swapping:

- Der Platz für Programm und Daten
  ist durch die **Hauptspeicherkapazität beschränkt**.

- **Problematisch** bei den Prozessen, die **viele E/A-Operationen** ausführen

- Der **Zugriff auf den Speicherbereich** eines fremden Prozesses
  muss durch das Betriebssystem explizit **verhindert** werden

- Die zusammenhängende Belegung von Hauptspeicher für ein ganzes Programm
  verschärft das Fragmentierungsproblem.

- Der **komplette Adressraum** eines Prozesses wird beim Prozesswechsel
  auf den Hauptspeicher **ausgelagert** und ein anderer Adressraum **eingelagert**

- Extrem **aufwendiger Prozesswechsel**

► Diese Nachteile werden durch das Konzept des virtuellen Speicher behoben.

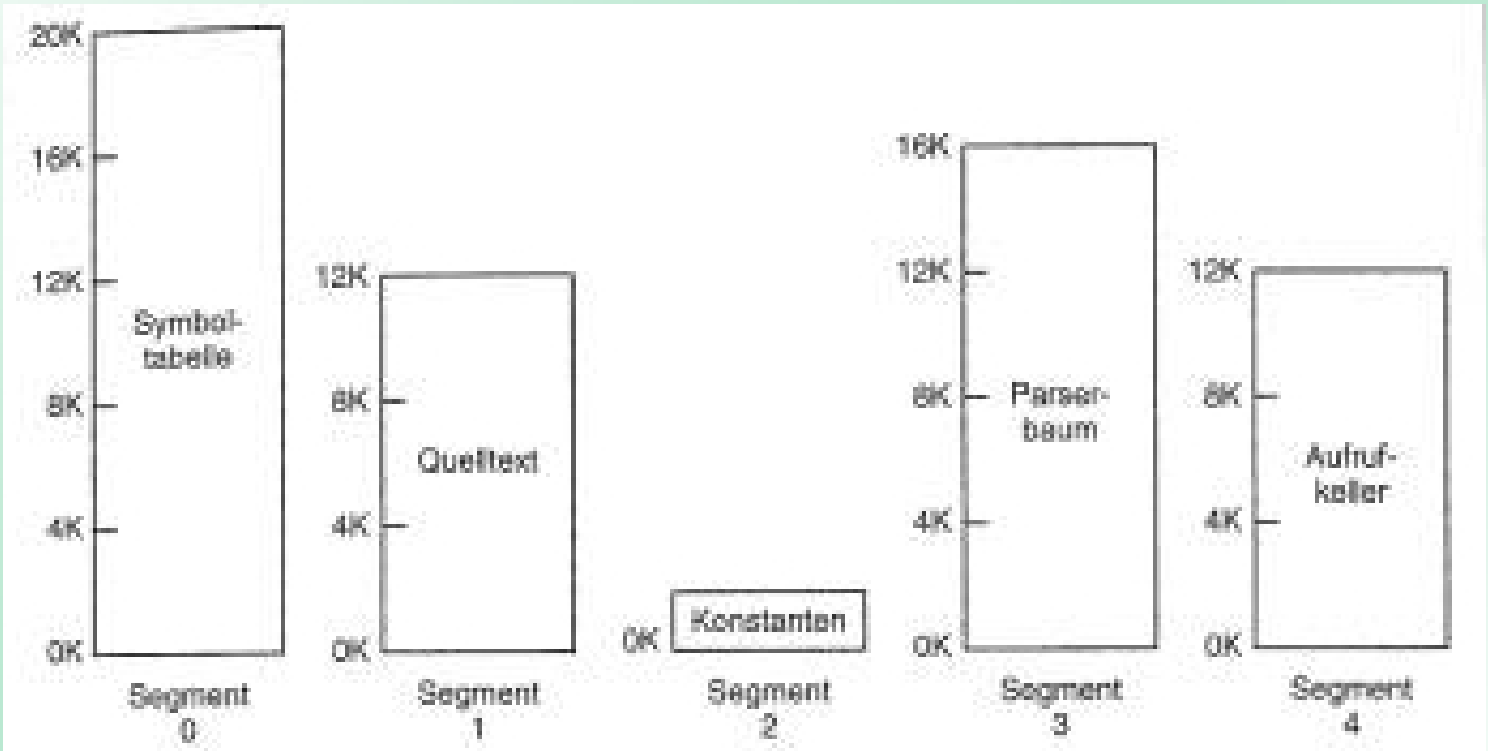
Gliederung

1. Einleitung
2. Swapping
3. Virtuelle Speicherverwaltung
   - 3.1 Segmentorientierter Speicher
   - 3.2 Seitenorientierter Speicher
   - 3.3 Paging vs. Segmentierung
   - 3.4 Pagingstrategien
   - 3.5 Behandlung von Seitenfehlern
4. Virtuelle Speicherverwaltung bei verschiedenen Betriebssystemen
   - 4.1 Unix
   - 4.2 Windows 2000
5. Zusammenfassung

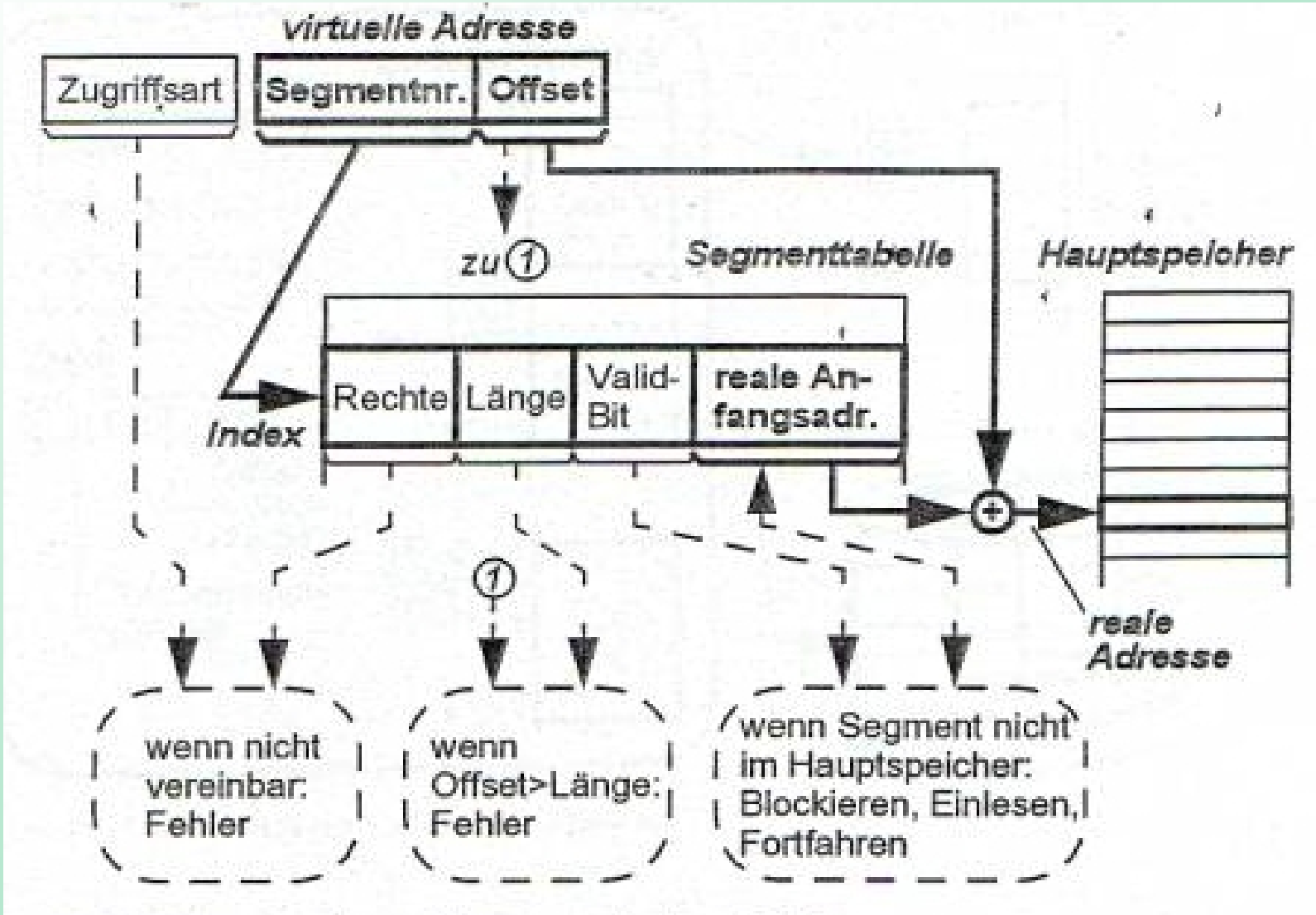
3.1 Segmentorientierter Speicher

Segment:

- unterschiedlich große Einheiten
- von einander unabhängige Adressräume



3.1 Segmentorientierter Speicher



3.1 Segmentorientierter Speicher

Vorteile:

- Die Segmente entsprechen unmittelbar
  den **logischen Speichereinheiten** eines Prozesses

- Die Segmente können **individuell**
  durch verschiedene **Zugriffsrechte** geschützt werden

- Die Segmente können **einzeln**
  auf den Plattenspeicher **verdrängt werden**

- Durch **eigene Segmenttabelle**
  sind die Prozesse wirksam voneinander abgeschottet

- **leichte Zusammenarbeit** von Prozessen möglich

- **einfache Verwaltung** von Datenstrukturen, die ihre Größe ändern

3.1 Segmentorientierter Speicher

Nachteile:

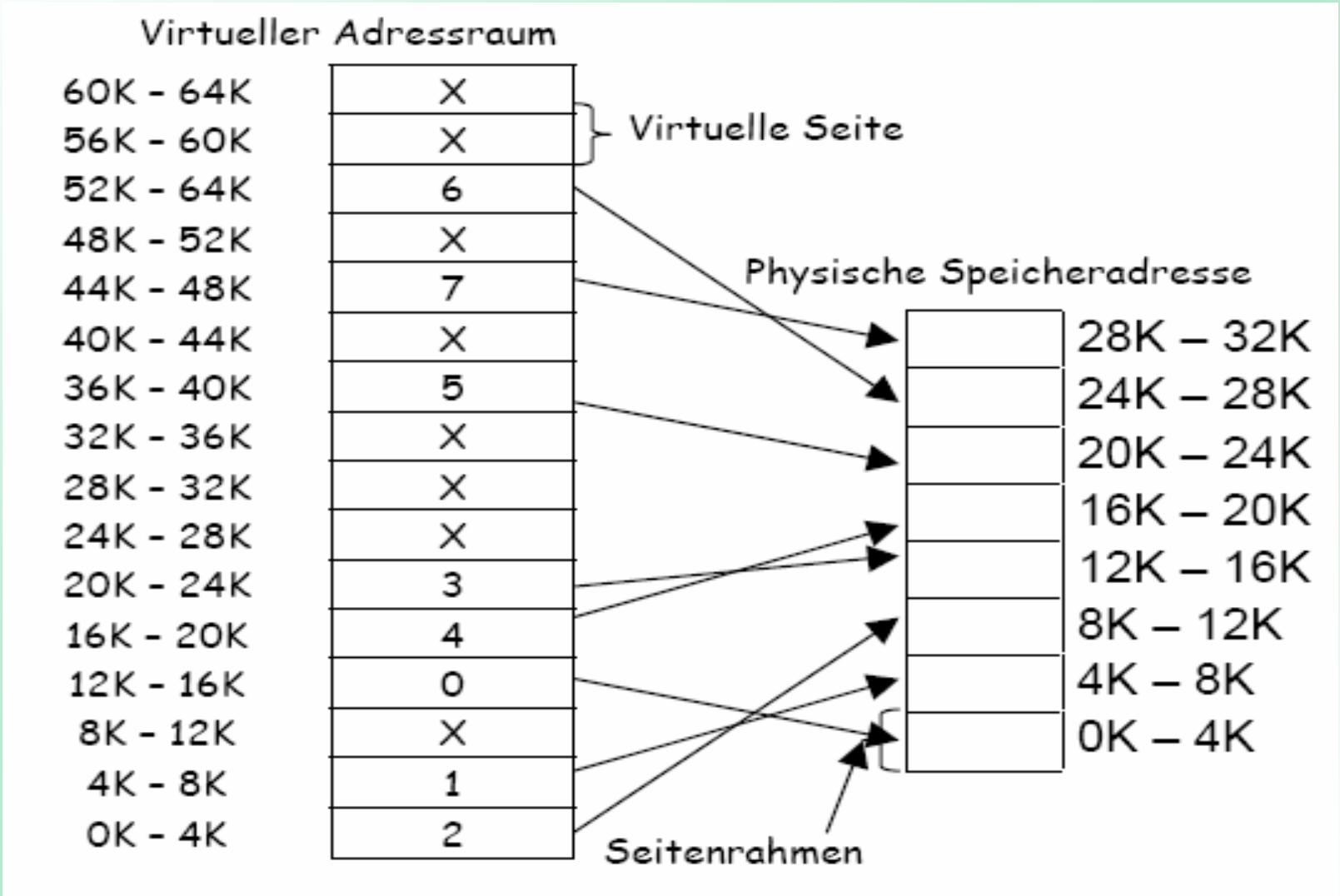
- zusammenhängender Speicherbereich

- Verschnittprobleme, da die Segmente unterschiedlich groß sind

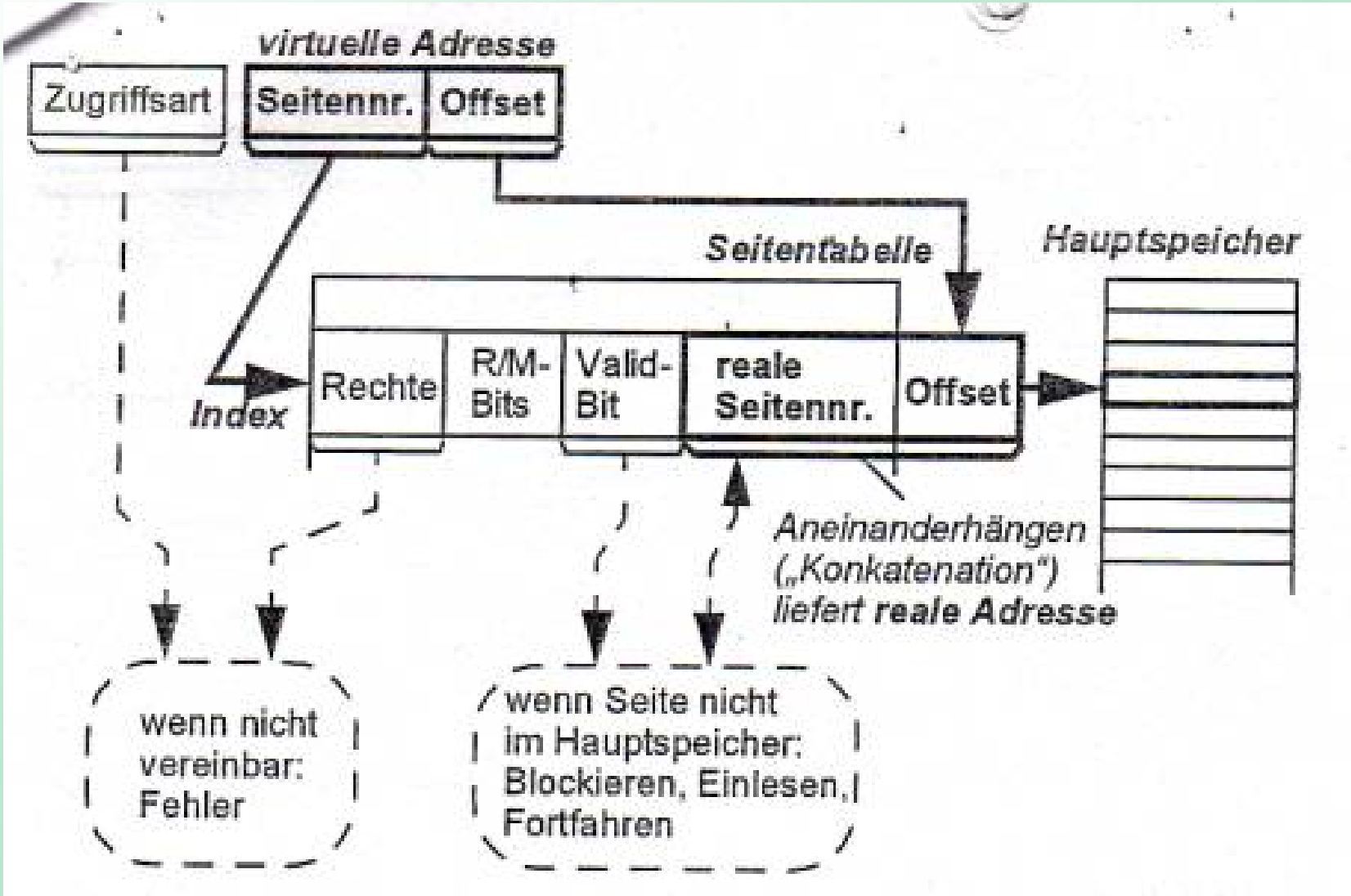
- Vollständiger Segment muss bei einem Zugriff **im Hauptspeicher** stehen

Gliederung

3.2 Seitenorientierter Speicher



3.2 Seitenorientierter Speicher



3.2 Seitenorientierter Speicher

Zur **Speicherung der Seitentabelle** gibt es mehrere Möglichkeiten:

- Einfach vollständig im Hauptspeicher

-Translation Lookaside Buffer (TLAB oder **TLB**)

- **Mehrstufige** Seitentabellen

- **Invertierten** Seitentabelle

3.2 Seitenorientierter Speicher

Vorteile:

- einfache Speicherverwaltung

- **kein Verschnittproblem**

- **leichterer Zugriff** auf den Plattenspeicher

- es werden **nur die benötigten Daten** in den Hauptspeicher geladen

- Prozesse sind von einander abgeschottet (durch **eigene Seitentabellen**)

3.2 Seitenorientierter Speicher

Nachteile:

- virtueller Speicher **spiegelt nicht den logischen Aufbau** der Prozesse wieder

- im Vergleich zu Segmenten etwas schwerere Identifikation, der Schutz und das Sharing individueller logischer Einheiten

- **hoher Speicherbedarf** für Seitentabellen

- **zeitaufwendige Adressberechnung**

Gliederung

3.4 Pagingstrategien

- **Ladestrategie**: Sie legt fest, zu welchem Zeitpunkt welche Seiten in den Hauptspeicher geladen wird

  Demand Paging: lädt eine Seite nur dann, wenn sie **unmittelbar** benötigt wird

  Prepaging: lädt Seiten **vorzeitig**

3.4 Pagingstrategien

- **Speicherzuteilungsstrategie**:

  entscheidet darüber,
  wie viele **Seitenbereiche** des Hauptspeichers
  den einzelnen Prozessen **zugeteilt** werden.

Working-Set-Strategie:

  versucht, für einen Prozess
  genau die **Seiten** im Hauptspeicher zu **halten**,
  auf denen er gerade **arbeitet**.

Page-Fault-Frequency-Strategie:

► **vergrößer**t den Bereich, wenn die **Seitenfehlerrate** des Prozesses einen oberen Schwellenwert **übersteigt**

► **verkleinert** ihn, wenn die **Fehlerrate** einen unteren Schwellwert **unterschreitet**

3.4 Pagingstrategien

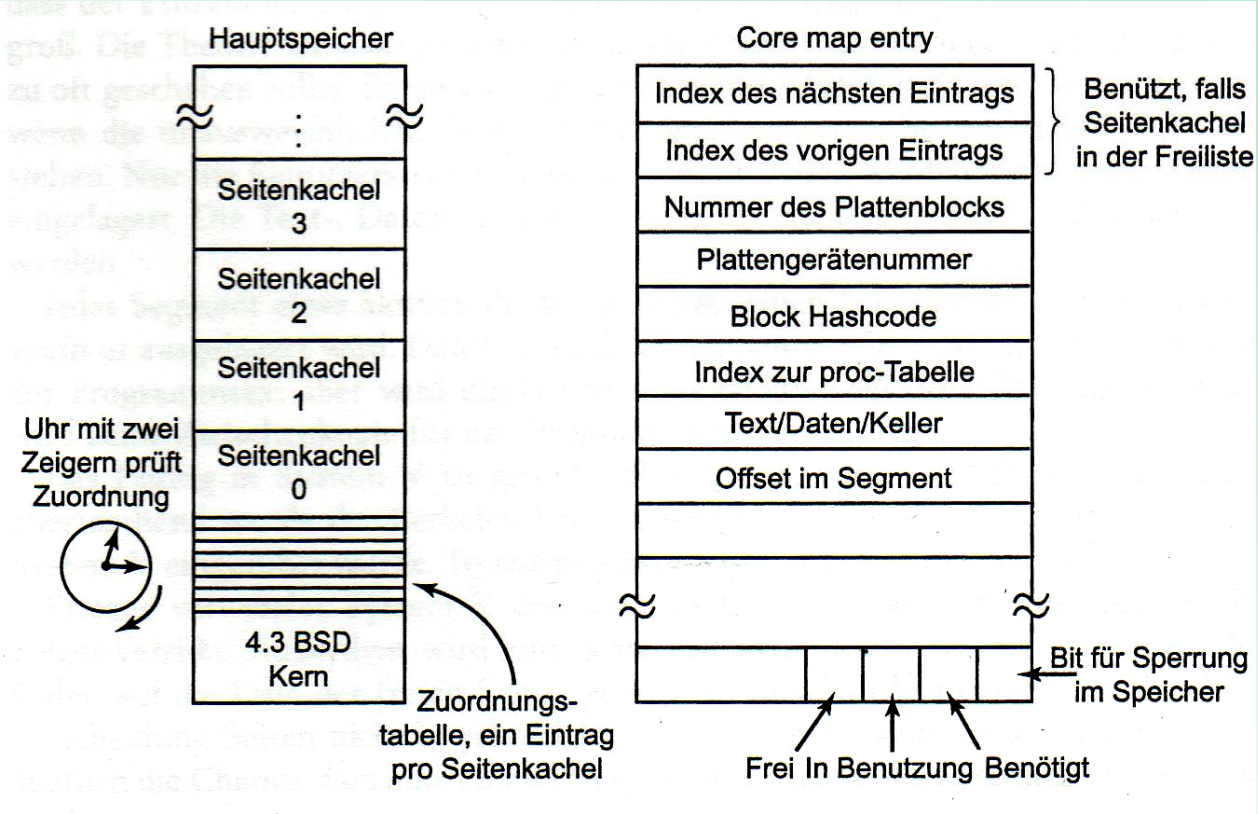
- **lokal:** Prozess ersetzt nur immer seine eigene Seiten
  - **statische Zuteilung** von Seiten pro Prozess
  - Seitenfehler-Verhalten liegt in der Verantwortung des Prozesses

- **global:** Prozess ersetzt auch Seiten anderer Prozesse
  - **dynamische Verhalten** der Prozesse berücksichtbar
  - **bessere Effizienz**

Gliederung

4.1 Unix

- Früher: Swapping -> Heute: **Demand Paging**



4.1 Unix

- Paging wird von **Page Daemon** realisiert

  - wird periodisch gestartet
  - Überprüfung, ob die Anzahl an freien Seiten zu klein geworden ist

► Falls ja

-> Auslagerung von Seiten auf die Festplatte
gemäß der eingesetzten Verdrängungsstrategie

► Falls nein

-> Blockierung des Paging-Daemons bis zum nächsten Sollzeitpunkt

4.1 Unix

- globaler **Zwei-Zeiger-Uhr-Algorithmus** als Seitenersetzungsalgorithmus

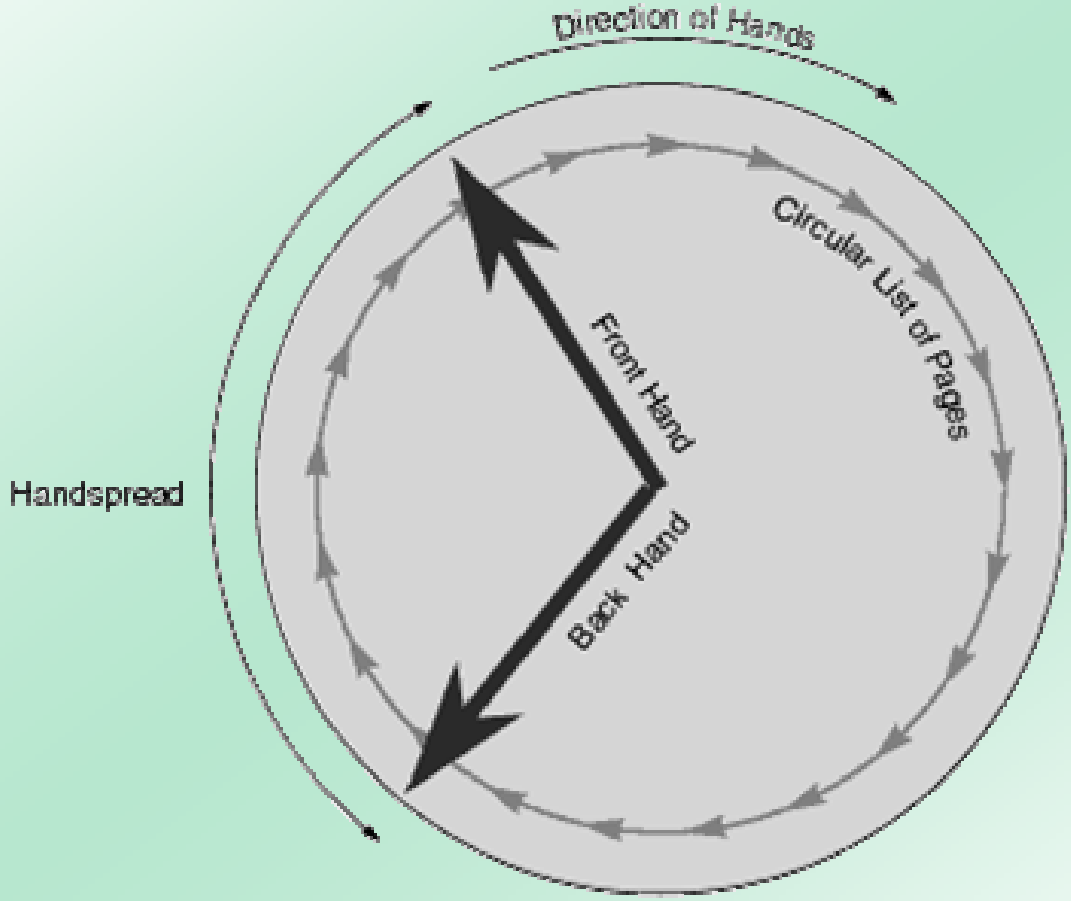


Figure 7.13 Two-handed clock algorithm

4.1 Unix

- Um Thrashing zu vermeiden wird **Swapping** eingesetzt:

- Gibt es ein Prozess, der länger als 20 Sekunden untätig ist?

  ► Falls ja
  -> wird derjenige ausgelagert, der am längsten untätig war

  ► Falls nein
  -> die vier größten Prozesse werden untersucht und
  derjenige ausgelagert, der sich am längsten im Speicher befindet

- Soll ein bereiter Prozess wieder zurück eingebracht werden?
  -> es wird nach Werten entschieden,
  die von verschiedenen Größen abhängen

4.1 Unix

- Besonderheiten bei **System V**

- einfacher Uhralgorithmus

  - statt *losfree*: *min* unb *max*

Gliederung

4.2 Windows 2000

• Ein einziger, linearer 4-GB-Adressraum je Prozess

• Segmentierung wird nicht unterstützt -> **Seitenadressierung**

• Bei Zuordnung eines Bereiches des virtuellen Adressraums wird die Datenstruktur **VAD** (Virtual Address Descriptor) angelegt.

► In VAD wird folgendes festgehalten:

- Der Bereich der abgebildeten Adressen
- Die zuständige Auslagerungsdatei auf dem Hintergrundspeicher
- Der Offset
- Der Schutzcode

► Erste Berührung der Seiten führt zur Erzeugung des Seitentabellenverzeichnis
-> Eintrag des Zeigers in VAD

4.2 Windows 2000

- **Kein Prepaging** vorhanden

► Beim Prozessstart sind keine Seiten im Speicher
► Dynamische Einlagerung bei Seitenfehler

20	3	1	1	1	1	1	1	1	1	1
Seitenkachel	**Nicht benutzt**	**G**	**L**	**D**	**A**	**C**	**Wt**	**U**	**W**	**V**

G: Seite global für alle Prozesse
L: Große (4GB) Seite (large)
D: Seiteninhalt ungültig (dirty)
C: Caching ein-/ausgeschaltet
A: Auf Seite wurde zugegriffen (accessed)
Wt: Write through (kein Caching)
U: Seite sichtbar im Benutzermod.
W: Seite ist beschreibbar (write)
V: Gültiger Eintrag (valid)

4.2 Windows 2000

- Arten von **Seitenfehlern**:

  - Die referenzierte Seite ist nicht belegt
  - Eine Schutzverletzung ist aufgetreten
  - Auf eine geteilt genutzte Seite wurde geschrieben
  - Der Stack muss wachsen
  - Die referenzierte Seite ist belegt, aber nicht eingelagert.

- Bei Seitenfehler werden **benachbarte Seiten** auch eingelagert

4.2 Windows 2000

- Ziel bei Seitenersetzungsalgorithmus:
  Gewisser Prozentsatz aller Seiten soll freigehalten werden

► Keine Auslagerung bei Seitenfehlern erforderlich

4.2 Windows 2000

- **Working-Set Verfahren**

- besteht aus eingelagerten Seiten eines Prozesses

- wird durch zwei Parameter beschrieben:
  -> die minimale und die maximale Größe

- Zu Beginn ist das Minimum des Working-Sets auf einen Wert zwischen 20 und 50 und das Maximum zwischen 45 und 345 gesetzt

- Jeder Prozess startet mit demselben Minimum und Maximum

- Entscheidung bei Seitenfehler
► Größe des WS < MIN -> Seite wird hinzugefügt
► Größe des WS > MAX -> Seite wird aus WS entfernt
(aber nicht aus dem Speicher)

- lokaler Algorithmus

4.2 Windows 2000

- **Balance-Set-Manager**:

  Sind genügend freie Seiten vorhanden?

NEIN: Start des Working-Set-Managers zur Seitenfreigabe

- Prüfreihenfolge anhand von Leerlaufzeiten
  und Größe der Prozess wird festgelegt

- Prozesse mit Working-Set kleiner als Minimum
  oder mit einer großen Anzahl von Seitenfehlern werden ignoriert

Gliederung

5. Zusammenfassung

Swapping:

- Betriebssystem kann **mehr Prozesse** ausführen kann,
  als im Speicher Platz haben

- Die Prozesse, für die kein Platz ist, werden auf die Festplatte ausgelagert.

- Verwaltung von freien Speicher
  mit Hilfe einer **Bitmap** oder einer **Freibereichsliste**

5. Zusammenfassung

Virtueller Speicher:

-> Paging

- einfachste Form: Adressraum wird in **Seiten** zerlegt
- verschiedene **Seitenersetzungsalgorithmen**
    - Aging
    - WSClock.

-> Segmentierung

- Vorteile bei Datenstrukturen, die ihre **Große ändern**
- **Linken** wird vereinfacht
- **gemeinsame** Benutzung von Code und Daten
- Möglichkeit die Segmente durch verschiedene **Zugriffsrechte** zu schützen

-> Hybride Formen

Gliederung

Quellen

[1] Andrew S. Tanenbaum: „Moderne Betriebssysteme“
2. Auflage, Pearson Studium, 2002

[2] Rüdiger Brause: „Betriebssysteme. Grundlagen und Konzepte“
3. Auflage, Springer Verlag, 2004

[3] Carsten Vogt: „Betriebssysteme“
1.Auflage, Spektrum Akademischer Verlag, 2001

[4] www.ibr.cs.tu-bs.de/lehre/ss04/ bsn/BSN-SoSe04-Kap03-Speicherverwaltung-3S.pdf
aufgerufen am 16.04.2005

[5] http://wwwcs.uni-paderborn.de/cs/ag-kao/de/teaching/ss05/bs1/script/BS_SS05_kap3_2seiten.pdf
aufgerufen am 02.06.2005

Fragen?