

4 Second Chance (5)

- Second chance zeigt FIFO Anomalie
 - ◆ Wenn alle Referenzbits gleich 1, wird nach FIFO entschieden
- Erweiterung
 - ◆ Modifikationsbit kann zusätzlich berücksichtigt werden (*Dirty bit*)
 - ◆ vier Klassen: (0,0), (0,1), (1,0) und (1,1) mit (Referenzbit, Modifikationsbit)
 - ◆ Suche nach der niedrigsten Klasse (Einsatz im MacOS)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1997

E-Memory.doc: 1997-12-17 12:11

E.69

Reproduzieren oder andere Verwendung dieser Vorlesung oder in Lehrwerken an der Universität Erlangen-Nürnberg ist strafbar. Beleid der Zustimmung des Autors.

5 Freiseitenpuffer

- Statt eine Seite zu ersetzen wird permanent eine Menge freier Seiten gehalten
 - ◆ Auslagerung geschieht im „voraus“
 - ◆ Effizienter: Ersetzungszeit besteht im Wesentlichen nur aus Einlagerungszeit
- Behalten der Seitenzuordnung auch nach der Auslagerung
 - ◆ Wird die Seite doch noch benutzt bevor sie durch eine andere ersetzt wird, kann sie mit hoher Effizienz wiederverwendet werden.
 - ◆ Seite wird aus Freiseitenpuffer ausgetragen und wieder dem entsprechenden Prozeß zugeordnet.

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1997

E-Memory.doc: 1997-12-17 12:11

E.70

Reproduzieren oder andere Verwendung dieser Vorlesung oder in Lehrwerken an der Universität Erlangen-Nürnberg ist strafbar. Beleid der Zustimmung des Autors.

6 Seitenanforderung

- ▲ Problem: Zuordnung der Kacheln zu mehreren Prozessen
- Begrenzungen
 - ◆ Maximale Seitenmenge: begrenzt durch Anzahl der Kacheln
 - ◆ Minimale Seitenmenge: abhängig von der Prozessorarchitektur
 - Mindestens die Anzahl von Seiten nötig, die theoretisch bei einem Maschinenbefehl benötigt werden (z.B. zwei Seiten für den Befehl, vier Seiten für die adressierten Daten)
- Gleiche Zuordnung
 - ◆ Anzahl der Prozesse bestimmt die Kachelmenge, die ein Prozeß bekommt
- Größenabhängige Zuordnung
 - ◆ Größe des Programms fließt in die zugeteilte Kachelmenge ein

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1997

E-Memory.doc: 1997-12-17 12:11

E.71

Reproduzieren oder andere Verwendung dieser Vorlesung oder in Lehrwerken an der Universität Erlangen-Nürnberg ist strafbar. Beleid der Zustimmung des Autors.

6 Seitenanforderung

- Globale und lokale Anforderung von Seiten
 - ◆ lokal: Prozeß ersetzt nur immer seine eigenen Seiten
 - Seitenfehler-Verhalten liegt nur in der Verantwortung des Prozesses
 - ◆ global: Prozeß ersetzt auch Seiten anderer Prozesse
 - bessere Effizienz, da ungenutzte Seiten von anderen Prozessen verwendet werden können

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1997

E-Memory.doc: 1997-12-17 12:11

E.72

Reproduzieren oder andere Verwendung dieser Vorlesung oder in Lehrwerken an der Universität Erlangen-Nürnberg ist strafbar. Beleid der Zustimmung des Autors.

E.8 Seitenflattern (*Thrashing*)

- Ausgelagerte Seite wird gleich wieder angesprochen
- ◆ Prozeß verbringt mehr Zeit mit dem Warten auf das Beheben von Seitenfehler als mit der eigentlichen Ausführung
- Ursachen
 - ◆ Prozeß ist nahe am Seitenminimum
 - ◆ zu viele Prozesse gleichzeitig im System
 - ◆ schlechte Ersetzungsstrategie
- ★ Lokale Seitenanforderung behebt Thrashing zwischen Prozessen
- ★ Zuteilung einer genügend großen Zahl von Kacheln behebt Thrashing innerhalb der Prozeßseiten
- ◆ Begrenzung der Prozeßanzahl

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1997

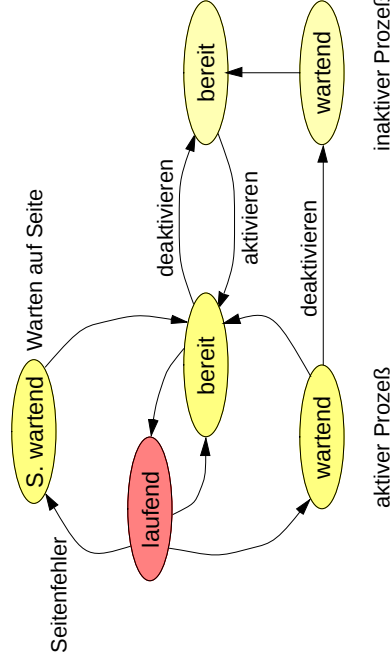
E-Memory.doc: 1997-12-17 12:11

E.73

Reproduzieren oder die Weiterverbreitung dieser Vorlesung ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg. Jegliche andere Weiterverbreitung ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg.

1 Deaktivieren von Prozessen

- Einführung von „Superzuständen“



- ◆ inaktiver Prozeß benötigt keine Kacheln; Prozeß ist vollständig ausgelagert (swapped out)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1997

E-Memory.doc: 1997-12-17 12:11

E.74

Reproduzieren oder die Weiterverbreitung dieser Vorlesung ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg. Jegliche andere Weiterverbreitung ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg.

1 Deaktivieren von Prozessen (2)

- Sind zu viele Prozesse aktiv, werden welche deaktiviert
- ◆ Kacheln teilen sich auf weniger Prozesse auf
- ◆ Verbindung mit dem Scheduling nötig
 - Verhindern von Aushungern
 - Erzielen kurzer Reaktionszeiten
- ◆ guter Kandidat: Prozeß mit wenigen Seiten im Hauptspeicher
 - geringe Latenz bei Wiedereinlagerung bzw. wenige Seitenfehler bei Aktivierung und Demand paging

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1997

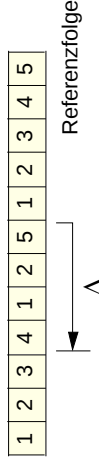
E-Memory.doc: 1997-12-17 12:11

E.75

Reproduzieren oder die Weiterverbreitung dieser Vorlesung ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg. Jegliche andere Weiterverbreitung ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg.

2 Arbeitsmengenmodell

- Menge der Seiten, die ein Prozeß wirklich braucht (*Working set*)
- ◆ kann nur angenähert werden, da üblicherweise nicht vorhersehbar
- Annäherung durch Betrachten der letzten Δ Seiten, die angesprochen wurden
- ◆ geeignete Wahl von Δ
 - zu groß: Überlappung von lokalen Zugriffsmustern
 - zu klein: Arbeitsmenge enthält nicht alle nötigen Seiten



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1997

E-Memory.doc: 1997-12-17 12:11

E.76

Reproduzieren oder die Weiterverbreitung dieser Vorlesung ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg. Jegliche andere Weiterverbreitung ist ohne schriftliche Genehmigung der Universität Erlangen-Nürnberg.

2 Arbeitsmengenmodell (2)

- Beispiel: Arbeitsmengen bei verschiedenen Δ

Referenzfolge	1	2	3	4	1	2	5	1	2	3	4	5
$\Delta = 3$	Seite 1	x	x	x		x	x	x	x	x		
	Seite 2		x	x	x		x	x	x	x	x	
	Seite 3			x	x	x					x	x
	Seite 4											x
	Seite 5				x	x					x	x
$\Delta = 4$	Seite 1	x	x	x	x	x	x	x	x	x	x	
	Seite 2		x	x	x	x	x	x	x	x	x	x
	Seite 3			x	x	x	x				x	x
	Seite 4					x	x	x	x			x
	Seite 5									x	x	x

3 Bestimmung der Arbeitsmenge

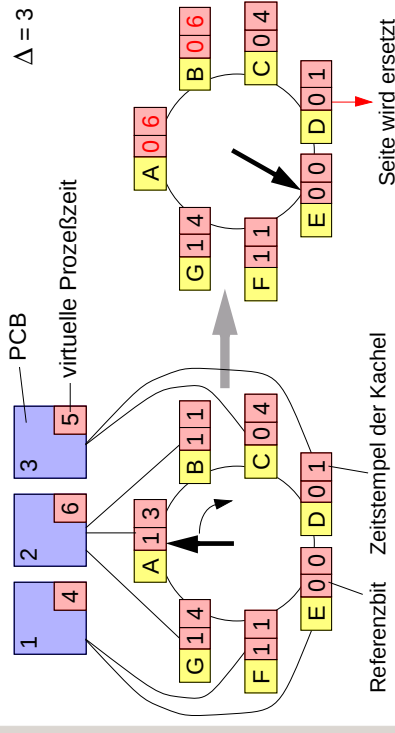
- Annäherung der Zugriffe durch die Zeit
 - bestimmtes Zeitintervall ist ungefähr proportional zu Anzahl von Speicherzugriffen
- Virtuelle Zeit des Prozesses muß gemessen werden
 - nur die Zeit relevant, in der der Prozeß im Zustand laufend ist
 - Verwalten virtueller Uhren pro Prozeß
- Referenzbit, Altersangabe und Timer-Interrupt
 - jede Seite erhält eine Altersangabe (Zeitintervall ohne Benutzung)
 - durch regelmäßigen Interrupt wird mittels Referenzbit die Altersangabe fortgeschrieben (bei Benutzung auf Null gesetzt, ansonsten erhöht); dabei werden nur die Seiten des gerade laufenden Prozesses „gealtert“
 - Seiten mit Alter $> \Delta$ sind nicht mehr in der Arbeitsmenge des jeweiligen Prozesses

3 Bestimmung der Arbeitsmenge (2)

- Ungenau: System ist aber nicht empfindlich auf diese Ungenauigkeit
- Verringerung der Zeitintervalle: höherer Aufwand, genauere Messung
- Algorithmus WSClock (Working set clock)
 - arbeitet wie Clock
 - Seite wird nur dann ersetzt, wenn sie nicht zur Arbeitsmenge ihres Prozesses gehört oder der Prozeß deaktiviert ist
 - Bei Zurücksetzen des Referenzbits wird die virtuelle Zeit des jeweiligen Prozesses eingetragen, die z.B. im PCB gehalten und fortgeschrieben wird
 - Bestimmung der Arbeitsmenge erfolgt durch Differenzbildung von virtueller Zeit des Prozesses und Zeitstempel in der Kachel

3 Bestimmung der Arbeitsmenge (3)

- WSClock Algorithmus

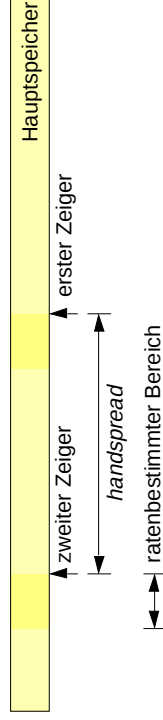


3 Bestimmung der Arbeitsmenge (4)

- ▲ Zuordnung zu einem Prozeß nicht immer möglich
- ◆ gemeinsam genutzte Seiten in modernen Betriebssystemen eher die Regel als die Ausnahme
 - Seiten des Codesegments
 - Shared libraries
 - Gemeinsame Seiten im Datensegment (*Shared memory*)
- ◆ moderne System bestimmen meist eine globale Arbeitsmenge von Seiten

4 Ersetzungsstrategie bei Solaris

- Prozeß *pageout* arbeitet Clock-Strategie ab
- ◆ Prozeß läuft mehrmals die Sekunde (4x)
- ◆ adaptierbare Rate: untersuchte Seiten pro Sekunde
- ◆ statt ein Zeiger: zwei Zeiger
 - am ersten Zeiger werden Referenzbits zurückgesetzt
 - am zweiten Zeiger werden Seiten mit gelöschtem Referenzbit ausgewählt
 - nötig, weil sonst Zeitspanne zwischen Löschen und Auswählen zu lang wird (großer Hauptspeicher; 64 MByte entsprechen 8.192 Seiten)
 - Zeigerabstand einstellbar (*handspread*)

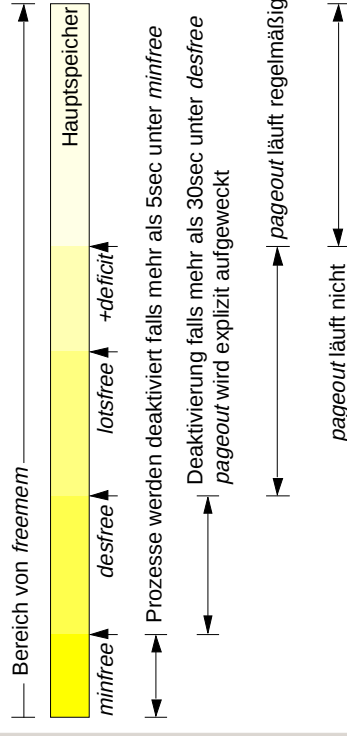


4 Ersetzungsstrategie bei Solaris (2)

- ◆ ermittelte Seiten werden ausgelagert (falls nötig) und
- ◆ in eine Freiliste eingehängt
- ◆ aus der Freiliste werden Kacheln für Einlagerungen angefordert
- ◆ Seitenfehler können unbenutzte Seiten aus der Freiliste wieder zurückfordern (*Minor page faults*)

4 Ersetzungsstrategie bei Solaris (3)

- Verhalten von *pageout* orientiert sich an Größe der Freiliste (Menge des freien Speichers)

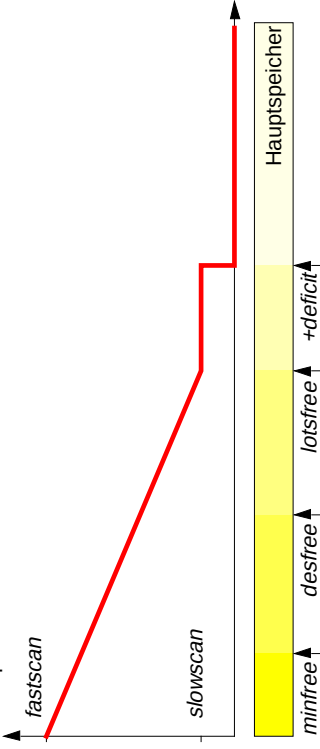


- ◆ *deficit* wird dynamisch ermittelt (0 bis *lotsfree*) und auf *lotsfree* addiert
- entspricht Vorschau auf künftige große Speicheranforderungen

4 Ersetzungsstrategie bei Solaris (4)

- Seitenuntersuchungsrate des *pageout* Prozesses

Seiten pro Sekunde



- ◆ je weniger freier Speicher verfügbar ist, desto höher wird die Untersuchungsrate
- ◆ *slowscan* und *fastscan* sind einstellbar

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1997

E-Memory.doc: 1997-12-17 12.11

E.85

Reproduktion oder die nicht-Vermittlung dieser Vorlesung ist ein Verstoß gegen die Urheberrechte der Universität Erlangen-Nürnberg. Jegliche Art der Reproduktion ist untersagt.

4 Ersetzungsstrategie bei Solaris (5)

- Weitere Parameter
 - ◆ *maxpgio*: maximale Transferrate bei Auslagerungen (vermeidet Plattensaturierung)
 - ◆ *autoup*: Zeitdauer des regelmäßigen Auslagerns alter Seiten durch den Prozeß *flushd* (Default: alle 30 sec)
- Aktivieren und Deaktivieren (*Swap in*, *Swap out*)
 - ◆ Auswahl wird dem Scheduler überlassen
 - ◆ Deaktivierung wird lediglich von Speicherverwaltung angestoßen

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1997

E-Memory.doc: 1997-12-17 12.11

E.86

Reproduktion oder die nicht-Vermittlung dieser Vorlesung ist ein Verstoß gegen die Urheberrechte der Universität Erlangen-Nürnberg. Jegliche Art der Reproduktion ist untersagt.

4 Ersetzungsstrategie bei Solaris (6)

- Typische Werte

- ◆ *minfree*: 1/64 des Hauptspeichers (Solaris 2.2), 25 Seiten (Solaris 2.4)
- ◆ *desfree*: 1/32 des Hauptspeichers (Solaris 2.2), 50 Seiten (Solaris 2.4)
- ◆ *lotsfree*: 1/16 des Hauptspeichers (Solaris 2.2), 128 Seiten (Solaris 2.4)
- ◆ *deficit*: 0 bis *lotsfree*
- ◆ *fastscan*: min(1/4 Hauptspeicher, 64 MByte) pro Sekunde (Solaris 2.4)
- ◆ *slowscan*: 800 kBytes pro Sekunde (Solaris 2.4)
- ◆ *handspread*: wie *fastscan* (Solaris 2.4)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1997

E-Memory.doc: 1997-12-17 12.11

E.87

Reproduktion oder die nicht-Vermittlung dieser Vorlesung ist ein Verstoß gegen die Urheberrechte der Universität Erlangen-Nürnberg. Jegliche Art der Reproduktion ist untersagt.

E.9 Zusammenfassung

- Freispeicherverwaltung
 - ◆ Speicherrepräsentation, Zuteilungsverfahren
- Mehrprogrammbetrieb
 - ◆ Relokation, Ein- und Auslagerung
 - ◆ Segmentierung
 - ◆ Seitenadressierung, Seitenadressierung und Segmentierung, TLB
 - ◆ gemeinsamer Speicher
- Virtueller Speicher
 - ◆ Demand paging
 - ◆ Seitenersetzungsstrategien: FIFO, B₀, LRU, 2nd chance (Clock)
- Seitenflattern
 - ◆ Super-Zustände, Arbeitsmengenmodell

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1997

E-Memory.doc: 1997-12-17 12.11

E.88

Reproduktion oder die nicht-Vermittlung dieser Vorlesung ist ein Verstoß gegen die Urheberrechte der Universität Erlangen-Nürnberg. Jegliche Art der Reproduktion ist untersagt.