

## Betriebssysteme im Wintersemester 2008/2009

### Übungsblatt 13

Abgabetermin: 21.01.2009, 13:30 Uhr

#### Aufgabe 51: (H) Seitenersetzungs-Strategien (2+2+2+2 Pkt.)

- a. Die Menge der Seiten sei gegeben durch  $N = \{0, 1, 2, 3, 4\}$  und die Menge der Seitenrahmen, die für die Speicherung der Seiten im Arbeitsspeicher zur Verfügung steht, sei gegeben durch  $\text{Frame}_3 = \{f_0, f_1, f_2\}$ . Auf die fünf Seiten der Menge  $N$  werde in folgender Reihenfolge zugegriffen:

$w = 4 \ 2 \ 0 \ 3 \ 4 \ 2 \ 1 \ 4 \ 2 \ 0 \ 3 \ 1$

Ein Seitenfehler liegt immer dann vor, wenn sich eine referenzierte Seite nicht im Arbeitsspeicher befindet. Dieser ist zu Beginn leer. Ermitteln Sie die Anzahl der Seitenfehler für die folgenden Paging-Strategien, indem Sie alle Veränderungen im Speicher tabellarisch dokumentieren.

- (i) FIFO (First In, First Out)
  - (ii) LIFO (Last In, First Out)
  - (iii) LFU (Least Frequently Used)
- b. Die Menge der Seitenrahmen werde vergrößert zu  $\text{Frame}_4 = \{f_0, f_1, f_2, f_3\}$ . Arbeiten Sie analog Aufgabenteil a) und vergleichen Sie die Seitenfehlerzahlen.

#### Aufgabe 52: (H) Seitenersetzung: Stack-Algorithmen (8 Pkt.)

- a. Was versteht man unter einer Distanzkette (Distance String)? Welche Informationen enthält sie? Warum kann es vorteilhafter sein, die Distanzkette statt der Referenzkette (Reference String) zu betrachten, um ein Paging-System zu bewerten?
- b. Beweisen Sie, dass der FIFO-Algorithmus zur Seitenersetzung kein Stack-Algorithmus ist.

### Aufgabe 53: (P) Buddy-Systeme

(11+5 Pkt.)

Die von Donald E. Knuth entwickelten Buddy-Systeme zur Verwaltung von Speicher nutzen die Binäradressierung aus, um effizient benachbarte Freibereiche zu einem grösseren Freibereich zusammenzufassen. Zur Erinnerung sei das Verfahren im folgenden noch einmal kurz beschrieben:

Die mit dem Buddy-Verfahren verwalteten, zusammenhängenden Speicherbereiche können nur die Länge  $2^k$ , mit  $k = 0, 1, 2, \dots$ , haben. Für einen Speicherbereich der Länge  $length$  wird ein Speicherbereich der Größe  $2^r$ , mit  $r = \lceil \lg length \rceil$ , belegt. Der Gesamtspeicher habe eine Größe von  $2^M$ . Zur Verwaltung der Freibereiche existiert für jede mögliche Länge  $2^k$ , mit  $k = 0, 1, \dots, M$  eine Liste. Für eine Belegung eines Bereichs der Länge  $length$  wird in der entsprechenden Liste nach einem freien Bereich gesucht. Ist bei der Belegung kein Bereich passender Länge frei, so wird ein größerer Bereich genommen und dieser wird halbiert, wobei eine Hälfte in die zugehörige Freiliste aufgenommen wird, während der andere Teil entweder weiter halbiert oder entsprechend der gestellten Anforderung reserviert wird. Bei der Halbierung entstehen zwei **Buddies**. Diese und nur diese können zu einem größeren Freibereich wieder zusammengefaßt werden, falls sie beide wieder frei sind. Zu Beginn existiert nur ein Freibereich der Länge  $2^M$ .

Geben Sie in programmiersprachlicher Notation die *Belege* und *Freigabe* Operationen an, die einen 1Mb großen Speicher gemäß dem oben beschriebenen Buddy-Verfahren verwalten!

### Aufgabe 54: (T) Abstrakter Interpreter

(24 Pkt.)

Wir betrachten ein Paging-System: Jeder Prozess erzeugt eine Folge von Speicherzugriffen. Da jeder Speicherzugriff einer bestimmten virtuellen Seite entspricht, kann man sich den Speicherzugriff eines Prozesses als eine Liste von Seitennummern vorstellen, die Referenzkette (Reference String). Ein Paging-System wird durch die Referenzkette des in Ausführung befindlichen Prozesses, den Seitenersetzungsalgorithmus und die Anzahl  $m$  der physischen Seitenrahmen charakterisiert.

Ein abstrakter Interpreter könnte wie folgt funktionieren: Er enthält eine interne Tabelle  $M$ , die den Zustand des Speichers repräsentiert.  $M$  hat

- $n$  Zeilen, wobei  $n$  die Anzahl unterschiedlicher virtueller Seiten ist,
- so viele Spalten, wie es Speicherzugriffe gibt (also Anzahl der Elemente in der Referenzkette) und
- zwei Teile, nämlich einen oberen Teil, bestehend aus den ersten  $m$  Zeilen der Tabelle, sowie einen unteren Teil, bestehend aus den restlichen  $n - m$  Zeilen.

Wird der Prozess gestartet, so beginnt die Abarbeitung der Referenzkette. Für jede Seite wird geprüft, ob sie im Speicher (d.h. im oberen Teil von  $M$ ) liegt. Falls nicht, ist ein Seitenfehler aufgetreten. Es wird wie folgt vorgegangen:

- Die angeforderte Seite wird in die erste Zeile der jeweiligen Spalte geschrieben.
- Befand sich an dieser Stelle bereits eine Seite, so wandert diese (und auch alle folgenden Seiten) um eine Position nach unten.

Die folgende Referenzkette sei gegeben:

$w = 0 \ 2 \ 1 \ 3 \ 5 \ 4 \ 6 \ 3 \ 7 \ 4 \ 7 \ 3 \ 3 \ 5 \ 5 \ 3 \ 1 \ 1 \ 1 \ 7 \ 1 \ 3 \ 4 \ 1$

- a. Stellen Sie die Tabelle  $M$  auf, und ermitteln Sie dadurch die Anzahl der Seitenfehler, unter der Annahme, dass  $m = 4$  physische Seitenrahmen zur Verfügung stehen.
- b. Welche Ihnen bekannte Seitenersetzungs-Strategie wird durch diesen abstrakten Interpreter realisiert?

- c. Spezifizieren Sie zunächst informell, dann auch formell die Klasse der Keller(Stack)-Algorithmen. Liegt hier ein Keller-Algorithmus vor? Welchen Vorteil haben Keller-Algorithmen gegenüber Algorithmen ohne diese Eigenschaft?
- d. Die sogenannte Distanzkette ist eine abstrakte Repräsentation der Referenzkette. In ihr wird jede Seite durch ihren Abstand vom oberen Ende des Kellers bei einem Zugriff repräsentiert. Der Abstand einer Seite, die nicht im Speicher liegt, sei unendlich. Geben Sie für obiges Beispiel die Distanzkette an.
- e. Entwickeln Sie einen Algorithmus, der es auf der Grundlage einer Distanzkette ermöglicht, die Anzahl der Seitenfehler für verschiedene Speichergrößen vorherzusagen.