INF FAKULTÄT FÜR INFORMATIK

Institut für Intelligente Kooperierende Systeme (IKS)
Prof. Dr. David Hausheer

Technische Informatik 2 - SS 2018 - Theoretische Übung 3

Abgabe: 20.05.2018

Technische Informatik II

Übung 3: Speicherverwaltung

Aufgabenstellung

Aufgabe 1: Konzepte

a) Bilden Sie durch ein Sequenz-Diagramm die verschiedenen Stufen der Bearbeitung eines Programms – vom Quellenprogramm (Quellcode) bis es sich als ein binäres ausführbares Image im Speicher befindet – ab. Identifizieren Sie in diesem Diagram diejenigen Stufen, wo eine Adressbindung möglich ist und assoziieren Sie jede Bindungsmöglichkeit mit der Relokationsflexibilität, die jede Bindungsentscheidung ermöglicht.

b) Bei welchen Arten von Adressbindung ist die Unterscheidung zwischen virtuellem und physikalischem Adressraum notwendig?

c) Gegeben sei ein 128MiB grosser Speicher und zwei Prozesse A und B, die jeweils 60MiB resp. 70MiB Speicher benötigen. Nennen Sie zwei Techniken, die für die gleichzeitige Ausführung von Prozessen A und B eingesetzt werden können.

d) Gegeben sei ein Betriebssystem, das über *First-fit* Speicherzuteilung verfügt. Warum kann bei diesem System das Phänomen der externen Fragementierung auftreten? Nennen Sie jeweils ein Verfahren, das die Auswirkungen der externen Fragmentierung i) mildern oder ii) komplett eliminieren kann.

e) Ein System verwende zur Verwaltung des Speichers *Segmentierung*. Beschreiben Sie kurz, wieso zwei Prozesse, die ein gemeinsames Segment zum Austausch von dynamisch generierten Programm Codes besitzen (Shared Memory), eine identische Segment ID für dieses haben sollten.

f) Gegeben sei ein Betriebsystem, das über Demand Paging mit *lokalem Seitenersatz* (*local page replacement*) verfügt. Ein Prozess, der auf diesem System gestartet wird, tätigt insgesamt *p* Speicherzugriffe. Für die total vom Prozess benötigten Seiten *n*, werden beim Starten *m* leere Frames alloziert. Diese Allokation ändert sich nicht während des Verlaufs des Prozesses. Unabhängig vom eingesetzten Seitenersatzalgorithmus (*Page Replacement Algorithm*), bestimmen Sie:

1. **Die untere Schranke für die Anzahl der Seitenfehler** (Page Faults), die der obige Prozess verursacht.
2. **Die obere Schranke für die Anzahl der Seitenfehler** (Page Faults), die der obige Prozess verursacht.

Aufgabe 2: Virtueller Speicher – Demand Paging

Gegeben sei ein Speicherverwaltungssystem mit virtuellem Speicher und folgender Konfiguration:

- **Virtuelle Adressen**: 44-Bit
- **Seitengrösse**: 4KiB
- **Seitentabelle**: Im Hauptspeicher, einstufig (*Single-level*), ohne Flag-Bits

a) Beschreiben Sie wie in diesem System eine virtuelle Adresse auf eine physikalische Adresse abgebildet wird (Skizze!).

b) Wie gross ist der minimale Anteil des Hauptspeichers, der durch eine derartige Seitentabelle belegt wird?

c) Verbessern Sie das ursprüngliche System indem Sie den Speicherverbrauch der Seitentabelle optimieren. Verwenden Sie dafür eine zweistufige Seitentabelle (*hierarchisches Paging*). Zusätzlich achten Sie darauf, dass eine Seitentabelle der zweiten Ebene genau in einer Seite gespeichert werden kann. Skizzieren Sie das verbesserte System und beschreiben Sie die Abbildung der virtuellen auf die physikalischen Adressen. Wie gross ist jetzt der minimale Anteil vom Hauptspeicher der von einer derartigen Seitentabelle belegt wird?

d) Gegeben sei folgender Code:

```
int array2D [1024][4];
int i, j = 0;

for (i=0; i < 1024; ++i){
     for (j=0; j < 4; ++j){
          array2D[i][j] = i+j;
     }
}
```

Der obige Code wird in Form eines Prozesses von einem Betriebssystem mit Demand Paging ausgeführt. Das Paging System wird von der Aufgabe a) übernommen. Die Seitenersatzstrategie sei LRU.
Dem Prozess werden für Code und Daten (Seitentabellen ausgenommen) 3 Frames statisch zugewiesen. Es wird angenommen, dass beim Starten des Prozesses die ganze Seitentabelle in den Hauptspeicher geladen wird und über den ganzen Verlauf des Prozesses dort verbleibt (*memory resident*). Vergleichen Sie unter zwei verschiedenen Annahmen die Anzahl Page Faults, welche die Ausführung dieses Prozesses verursacht:

i) Das zwei-dimensionale Array `array2D` wird in row-major Reihenfolge gespeichert (`array2D[0][0], array2D[0][1], array2D[0][2], array2D[0][3], array2D[1][0], ...`).

ii) Das zwei-dimensionale Array wird in column-major Reihenfolge gespeichert (`array2D[0][0], array2D[1][0], ..., array2D[1023][0], array2D[0][1], ...`).

Aufgabe 3: Seitenersatz-Algorithmen

Geben Sie in den untenstehenden Tabellen nach jedem Seitenzugriff die geladenen Seiten an. Ordnen Sie die Seiten in der Reihenfolge in der sie ersetzt werden; platzieren Sie die Seite, welche als erste ersetzt wird, zuoberst.

a) Gegeben ist folgende Zugriffssequenz auf fünf diskrete Seiten:

`3,2,1,0,3,2,4,3,2,1,0,4,2,3,2,1,0,4`

Der Algorithmus zum Ersetzen der Seiten ist FIFO.
Nehmen Sie zunächst an, es stehen 3 Page Frames zur Verfügung:

	3	2	1	0	3	2	4	3	2	1	0	4	2	3	2	1	0	4
1.																		
2.																		
3.																		

Nehmen Sie dann an, es stehen 4 Page Frames zu Verfügung:

	3	2	1	0	3	2	4	3	2	1	0	4	2	3	2	1	0	4
1.																		
2.																		
3.																		
4.																		

Vergleichen Sie die totale Anzahl Page Faults mit den zwei unterschiedlichen Speichergrössen.
Was beobachten Sie? Wie heisst dieses Phänomen?

b) Lösen Sie Aufgabe a) noch einmal, aber mit Least Recently Used (LRU) Page-Replacement.

	3	2	1	0	3	2	4	3	2	1	0	4	2	3	2	1	0	4
1.																		
2.																		
3.																		

	3	2	1	0	3	2	4	3	2	1	0	4	2	3	2	1	0	4
1.																		
2.																		
3.																		
4.																		

Vergleichen Sie wieder die Anzahl Page Faults für die zwei Speichergrössen. Wie unterscheidet sich dieser Fall vom FIFO-Seitenersatz?

c) Nun wollen wir das oben beobachtete Verhalten des LRU Algorithmus bezüglich Speichergrösse verallgemeinern. Konkret wollen wir folgende Behauptung beweisen: Unter LRU wird die Anzahl Page Faults nicht grösser wenn wir den physikalischen Speicher um ein Frame vergrössern.

Zu diesem Zweck definieren wir für eine gegebene Speichergrösse von i Page Frames die Menge $LRU_i(n)$ als die Menge der Seiten, die der LRU-Algorithmus nach n Seitenzugriffen im Hauptspeicher hat.
Zeigen Sie nun, dass Folgendes immer gilt:

$$LRU_i(n) \subseteq LRU_{i+1}(n).$$

Dies impliziert dann direkt die Behauptung.
Tipp: Benutzen Sie Induktion als Beweismethode und verwenden Sie Seiten-Zugriffe als einzelne Schritte der Induktion.

Aufgabe 4: Speicher-Fragmentierung

a) Was sind die notwendigen Bedingungen damit fragmentierter Speicher verdichtet werden kann?

i. Virtuelle Adressen werden zur Ladezeit auf physikalische abgebildet.
ii. Virtuelle Adressen werden zur Laufzeit auf physikalische abgebildet.
iii. Es können nur Speicherblöcke fixer Grösse alloziert werden

b) Gegeben seien 5 freie Speicherblöcke (A–E) mit folgenden Grössen:

A: 100KiB
B: 500KiB
C: 200KiB
D: 300KiB
E: 600KiB

Nun kommen 4 neue Prozesse (1–4) mit folgendem Speicherbedarf in der gegebenen Reihenfolge in eine FIFO Queue:

1: 212KiB
2: 417KiB
3: 112KiB
4: 426KiB

Wie werden die Prozesse (1–4) jeweils den Speicherlücken (A–E) zugeordnet, wenn der *First-fit, Best-fit,* oder *Worst-fit* Algorithmus angewendet wird? Füllen Sie die untenstehende Tabelle gemäss dem Beispiel in der ersten Zeile aus.

Algorithmus	A	B	C	D	E
Beispiel	4	3	2	1	–
First-fit					
Best-fit					
Worst-fit					

c) Wieviel interne Fragmentierung verursachen die drei Algorithmen im obigen Beispiel (in KiB)?

d) Welcher der drei Algorithmen (First-fit, Best-fit, Worst-fit) benötigt am wenigsten CPU-Operationen um einen Prozess einer Speicherlücke zuzuordnen? Begründen Sie Ihre Wahl.

e) Wir betrachten folgendes System: Das System ist Byte-adressiert und verwendet Paging. Die Pagegrösse beträgt X Bytes, $X > 1$. Jeder Prozess hat eine konstante Grösse von L Bytes. Ferner gilt $L \gg X$, L ist unabhängig von X. Jeder einzelne Prozess belegt im virtuellen Speicher einen linearen zusammenhängenden Bereich. Es befinden sich N Prozesse im Hauptspeicher

(i) Überlegen Sie sich zuerst, was in punkto Fragmentierung der *schlechtestmögliche* Fall ist, also unter welchen Umständen der durch Fragmentierung verlorene Speicher *maximal* ist. Geben Sie dann eine Formel an für die maximale interne Fragmentierung.

(ii) Was ist wohl die *durchschnittliche* Fragmentierung? Begründen Sie Ihre Definition und geben Sie eine Formel für die durchschnittliche Fragmentierung an.

(iii) Ist es möglich, dass gar keine Fragmentierung auftritt? Begründen Sie Ihre Antwort und geben Sie eine Formel für die *minimale* Fragmentierung an.