Grundlagen der Systemsoftware

SoSe 2015 · Übungsblatt 6 (VSIS) · Abgabe: Do. 02.07.2015 (bis 10:00 Uhr)

Aufgabe 1: Speicherverwaltung (14 Punkte)

Es wird das Innenleben einer *Memory Management Unit* (MMU) mit Seitenadressierung betrachtet, wobei eine Seitentabelle mit 16 Seiten verwendet wird, virtuelle Adressen sind 16 Bit, die physikalischen Adressen sind 15 Bit lang und die Wortlänge sei 1 Byte[1].

a) Wie groß ist eine Seite?

b) (optional) Woraus besteht ein typischer Seitentabelleneintrag und welche Bedeutung hat insbesondere das *Present/Absent*-Bit? Bei höchstens wievielen Einträgen der Seitentabelle kann im gegebenen Fall das *Present/Absent*-Bit auf 1 („Present“) gesetzt sein?

c) Gegeben sei ein Ausschnitt der Seitentabelle zu einem Zeitpunkt (vgl. Abbildung rechts). Welche physikalische Adresse ergibt sich für folgende virtuelle Adressen?

   i) 0x5fe8

   ii) 0xfeee

   iii) 0xa470

   iv) 0x0101

	Seiten-rahmen-nr.	Present/ Absent-Bit
15	000	0
14	000	0
13	...	...
12	...	...
11	...	...
10	000	1
9	000	0
8	110	1
7	...	...
6	...	...
5	001	1
4	...	...
3	...	...
2	111	1
1	010	1
0	101	1

d) Um eine möglichst optimale Seitengröße zu wählen, muss man zwischen mehreren zum Teil gegenläufigen Faktoren abwägen. Bitte nennen Sie mindestens 2 Faktoren, die für eher kleine Seiten sprechen, und mindestens einen Grund, welcher für eher große Seiten spricht.

e) Sei $p = 4\,\text{MiB}$ die durchschnittliche Prozessgröße im Speicher und $L = 8\,\text{B}$ die Länge eines Seitentabelleneintrags. Erläutern Sie kurz den Zusammenhang zwischen der Seitentabellengröße im Speicher und der internen Fragmentierung bei verschiedenen Seitengrößen und ermitteln Sie die optimale Seitengröße s in KiB unter der Annahme, dass im Durchschnitt die Hälfte der letzten Seite eines Prozesses leer bleibt. Geben Sie dabei den vollständigen Rechenweg an und erklären Sie Ihre Annahmen.

Aufgabe 2: Seitenersetzungsalgorithmen (14 Punkte)

a) Gegeben sei ein Seitenspeicher der Größe 3. Zum Zeitpunkt $t = 0$ sei der Speicher leer. Illustrieren Sie (z. B. anhand von Tabelle 1) den Zustand des Seitenspeichers bei Verwendung des (a) *optimalen Seitenersetzungsalgorithmus* und (b) LRU-Algorithmus (*Least Recently Used*) zu den Zeitpunkten 1 bis 15 für die Referenzkette 1, 2, 3, 4, 5, 6, 1, 3, 1, 6, 3, 5, 4, 2, 1. Markieren Sie die Spalten, in denen ein Seitenalarm (engl. „page fault“) auftritt.

[1] Hilfe zur Beantwortung dieser Fragen finden Sie in weiterführender Literatur, bspw. in Andrew S. Tanenbaum, *Moderne Betriebssysteme*, 3. Auflage, Abschnitte 3.3.1ff (*Paging*) und 3.5.3ff (*Seitengröße*)

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Seite	1	2	3	4	5	6	1	3	1	6	3	5	4	2	1
Fault	×	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Seiten im Speicher	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.
		.	.	.	.	.	.	.	.	.	.	.	.	.	.
			.	.	.	.	.	.	.	.	.	.	.	.	.

Tabelle 1: Zustand des Seitenspeichers für $t = 1$ bis $t = 15$.

b) (optional) Der optimale Seitenersetzungsalgorithmus wird gerne für theoretische Betrachtungen herangezogen. Warum kommt er nicht in realen Betriebssystemen zum Einsatz (max. 3 Sätze)?

c) (optional) Ergibt „*LRU*“ als Seitenersetzungstrategie Sinn, sofern das „L“ in „LRU“ für „Last“ und nicht, wie korrekterweise, für „Least“ stehen würde ?

Aufgabe 3: Synchronisation (12 Punkte)

Der Gebrauch von Semaphoren kann anhand der Lösung des klassischen Reader-Writer-Problems anschaulich demonstriert werden. Bei dem Reader-Writer-Problem gibt es zwei verschiedene Klassen, die beide auf einen gemeinsamen Datenbereich zugreifen. Instanzen der Klasse `Writer` schreiben in ihrer Hauptmethode (`processWriter()`) in den gemeinsamen Datenbereich und benötigen daher den exklusiven Zugriff. Instanzen der Klasse `Reader` hingegen greifen in ihrer Hauptmethode (`processReader()`) nur lesend auf den Datenbereich zu und können daher auch parallel auf dem gemeinsamen Datenbereich agieren.

a) Entwerfen Sie eine Lösung für das Reader-Writer-Problem, indem Sie die Hauptmethoden (`processWriter()` und `processReader()`) der beiden Klassen in Pseudocode angeben. Der Zugriff auf den gemeinsamen Datenbereich erfolgt durch die Methoden `readData()` (im Falle eines Readers) und durch die Methode `writeData()` (im Falle eines Writers). Verwenden Sie in Ihrer Lösung folgende Hilfsvariablen und geben Sie deren Initialisierungswerte mit an:

   - ein binäres Semaphor `W`, um den kritischen Anschnitt zu schützen, in dem die eigentlichen Datenzugriffe (durch `writeData()` bzw. `readData()`) erfolgen
   - eine Zähler-Variable `NumberOfActiveReaders`, die die Anzahl der aktiven Reader angibt, die gerade auf den gemeinsamen Datenbereich zugreifen
   - ein binäres Semaphor `Mutex`, welches dafür sorgt, dass die Zähler-Variable `NumberOfActiveReaders` nur durch einen Reader zur Zeit modifiziert werden kann

b) Werden bei Ihrer Lösung bestimmte Prozesse benachteiligt? Wenn ja, deuten Sie bitte kurz (max. 5 Sätze) einen möglichen Lösungsweg an.