Universität Heidelberg
Sommersemester 2022

AG Parallele und Verteilte Systeme (PVS)
Artur Andrzejak

6. Übung zur Vorlesung „Betriebssysteme und Netzwerke“ (IBN)

Abgabedatum: **14.06.2022, 11:00 Uhr**

Hinweise: Das Abgabedatum ist später als üblich. Am 7. Juni wird kein Übungszettel ausgegeben.

Aufgabe 1 (1 Punkt)

Erstellen Sie eine „richtig/falsch” Umfrage mit vier *eigenen* Fragen zu einem der Themen Adressenübersetzung/Paging bzw. Seitentabellen bzw. virtueller Speicher, ähnlich strukturiert wie die Folie „Was ist richtig?” der Vorlesung 10. Beziehen Sie sich dabei auf den Stoff auf den Vorlesungsfolien. Geben Sie zu jeder der Aussagen eine Begründung an, warum diese stimmt bzw. warum nicht. Bitte nur ernste Fragen!

Aufgabe 2 (1 Punkt)

Es wurde beobachtet, dass sich die Zeit zwischen den Seitenfehlern eines Prozesses verdoppelt, wenn der dem Prozess zur Verfügung stehende physische Speicher verdoppelt wird. Angenommen, ein normaler Befehl dauert eine Mikrosekunde, aber wenn der Seitenfehler auftritt, dauert er 2001 Mikrosekunden (d.h. 2 Millisekunden für den Seitenfehler). Wie lange würde ein Programm laufen, das nach 60 Sekunden beendet ist und während dieser Zeit 15000 Seitenfehler erzeugt hat, wenn es doppelt so viel Speicher zur Verfügung hätte?

Aufgabe 3 (1 Punkt)

Betrachten Sie eine Liste von Seitennummern wie ganz unten auf Folie „Second-Chance-Algorithmus” der Vorlesung 12. Angenommen, die R-Bits für die Seiten B bis A sind 11011111. Welche Seiten würde der Second Chance Algorithmus als die nächsten zwei Opfer auswählen? Nehmen Sie an, dass diese Seiten bis zur zweiten Ausführung des Algorithmus nicht verwendet werden (d.h. die R-Bits werden nicht durch die Hardware verändert).

Aufgabe 4 (2 Punkte)

Wie viele Seitenfehler erzeugt der FIFO-Algorithmus mit acht Seiten und vier Seitenrahmen für die folgende Sequenz von Seitenzugriffen: 0, 1, 7, 2, 3, 2, 7, 1, 0, 3, wenn die Seitenrahmen zu Beginn leer sind? Lösen Sie diese Aufgabe auch für den LRU-Algorithmus. Begründen Sie Ihre Ergebnisse.

Aufgabe 5 (1 Punkt)

Wie hängt die Rate der Seitenfehler des optimalen Algorithmus (OPT) mit der Seitenfehlerrate des Algorithmus Least-Recently-Used (LRU) zusammen? Nehmen Sie dabei an, dass die betrachtete Implementation von LRU die exakten Zeitpunkte der Seitenzugriffe verwendet. *Hinweis*: Es gibt einen engen Zusammenhang beider Algorithmen „mathematischer Art", welcher jedoch nicht praktisch verwendbar ist.

Aufgabe 6 (6 Punkte)

Implementieren Sie ein Programm, welches die Seitenersetzungsalgorithmen simulieren kann. Benutzen Sie dabei eine der gängigen Programmiersprachen wie Python, C, C++, Java, C#, ... (ggf. sprechen Sie sich mit Ihrer Tutorin/Ihrem Tutor ab, welche weitere Sprachen akzeptiert werden). Ihr Programm erwartet als Eingabe (Übergabe via Kommandozeilenargumente) (i) die Anzahl an Seitenrahmen und (ii) eine Referenzfolge von Seitenzugriffen. Es soll dann nach und nach eine Repräsentation des jeweiligen Status der Seitenrahmen auf die Standardausgabe ausgeben. Implementieren Sie entweder den Clock-Algorithmus oder den Algorithmus Least-Recently-Used (LRU). Führen Sie das Programm jeweils mit den Referenzfolgen A und B aus der Vorlesung 12 für drei Seitenrahmen aus. Reichen Sie den Quelltext als auch ein Log der Programmausgaben (für mindestens diese zwei Referenzfolgen) als Ihre Lösung ein.

Aufgabe 7 (3+4 Punkte)

Geben Sie *Pseudocode* für ein Programm an, das ein größtes Working Set (WS) in einer Referenzfolge von Seitenzugriffen berechnet, wobei die Definition eines WS aus der Folie *Working Set Definieren und Messen* der Vorlesung 12 verwendet werden soll. Im Detail: das Programm bekommt als Eingabe die Anzahl Δ der letzten zu berücksichtigenden Speicherzugriffe und den Dateinamen einer Datei mit einer Referenzfolge (z.B. pro Zeile: ein Index der zugegriffenen Seite). Es liest die Datei ein, verarbeitet diese, und gibt ein größtes WS (für Δ) aus. Geben Sie die relevanten Datenstrukturen und Variablen an. Bei den I/O-Operationen können Sie vereinfachen, z.B. reicht die Beschreibung „lese Inhalt der Datei mit Dateinamen *name* in ein Array *pageAccesses* ein".

(**Bonusaufgabe, 4 Punkte**): Implementieren Sie ihren Algorithmus in einer gängigen Programmiersprache Ihrer Wahl (wie üblich, sprechen Sie sich mit Ihrer Tutorin/Ihrem Tutor ab), und testen Sie die Implementierung an den Referenzfolgen A und B aus der Vorlesung 12 aus (mit $\Delta = 5$).

Aufgabe 8 (1 Punkt)

Meistens ist die Struktur des Speicherabbilds (sog. image) eines Prozesses so wie in der Vorlesung 3 dargestellt, d.h. Code, Daten und Halde starten auf dem anderen Ende des logischen Adressraumes als der Stack. Vergleichen Sie die folgenden drei Schemen der Speicherverwaltung: (i) reiner linearer

Adressraum (ohne Paging, ohne Segmentierung); (ii) reine Segmentierung, wobei das Betriebssystem die Segmente veraltet; (iii) reines Paging; in Hinblick auf die *Effizienz der Speichernutzung* eines so strukturierten Prozesses. Insbesondere ist damit gefragt, ob ein Prozess mehr RAM belegt, als es wirklich braucht. Erläutern Sie weiterhin für die letzten zwei Schema, was passiert, wenn Halde oder Stack weiter wachsen.