Vorlesung Betriebssysteme WS 2018/2019

Aufgabenblatt 5 vom 24. Januar 2019

(Vorstellung der Lösungen bei den Tutoren bis zum 08.02.2019)

1. Beantworten Sie die folgenden Fragen und erläutern Sie die Antworten Ihrem Tutor.

   (a) Erklären Sie die Begriffe *Physischer Speicher*, *Virtueller Speicher*, *Paging*, *Page-Fault*, *Page-Replacement-Algorithmus*, *Swapping* und *Interne/Externe Fragmentierung*.

   (b) Erläutern Sie anhand eines Schaubilds wie Seiteneinträge auf x86 aufgebaut sind. Wofür stehen die Statusbits?

   (c) Erläutern Sie die Rollen der *Standby-*, *Modified-*, *Free-* und *Zero page-*Listen in der Windows Speicherverwaltung. Welche Eigenschaften haben Speicherseiten, die in diesen Listen enthalten sind?

2. Ein Prozess benutzt *Speicherseiten*, die beim Zugriff in verfügbare Seitenrahmen im Hauptspeicher geladen werden. Alle Seitenrahmen sind zu Beginn leer.

   Simulieren Sie das Verhalten der Seitenersetzungsstrategien *FIFO* und *LRU* in der Zugriffssequenz

   3 0 2 4 3 0 5 3 0 2 4 5

   unter der Annahme, dass

   - a) 3 Seitenrahmen
   - b) 4 Seitenrahmen

   zur Verfügung stehen.

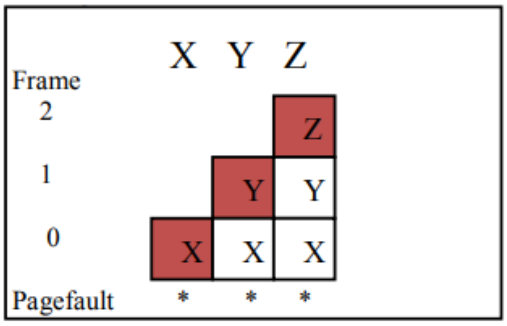

Abbildung 1: Beispieldiagramm

Zeichnen Sie dazu Diagramme, welche die Belegung der Seitenrahmen zu jedem Zeitpunkt zeigen, wie in Abbildung 1 dargestellt und erläutern Sie diese Diagramme Ihrem Tutor.

Markieren Sie in Ihren Diagrammen Seitenzugriffsfehler und berechnen Sie das Trefferverhältnis. Welches Verhalten lässt sich beim Vergleich der Seitenersetzungsstrategien beobachten?

3. Gegeben sei ein Rechnersystem mit 36-Bit Adresszeigern und folgenden Eigenschaften:

   - Die Seitengröße (*Page-Frame Size*) beträgt 16 KiB
   - Die virtuelle Speicherverwaltung unterstützt lediglich eine Ebene von Seitentabellen, d.h. zu jedem Prozess gehört genau eine Tabelle, die den gesamten virtuellen Speicher auf den physikalischen abbildet. Diese Tabellen können nicht ausgelagert werden.
   - Der Rechner hat insgesamt 1 GiB physikalischen Speicher. Das Betriebssystem benötigt mindestens 128 MiB Speicher, der nicht ausgelagert werden kann. In diesem Speicherbereich sind evtl. Seitentabellen noch nicht eingerechnet.

   Beantworten Sie zu diesem Rechnersystem folgende Fragen (der Rechenweg muss ersichtlich sein):

   (a) Welche maximale Speichergröße unterstützt dieses System?

   (b) Wie viele Einträge enthält eine Seitentabelle?

   (c) Wie viel physikalischer Speicher wird von einer Seitentabelle belegt? Jeder Seitentabelleneintrag soll die Seitennummer und 10 Steuerungs-Bits enthalten.

   (d) Wenn drei Prozesse in diesem System laufen, wie groß kann das Working-Set eines Prozesses maximal sein?

4. Ein *Multilevel Queue Scheduler* verwaltet pro Prioritätsstufe eine Liste von lauffähigen Threads. Bei der Auswahl des als nächtes laufenden Threads werden die Listen entsprechend ihrer Priorität inspiziert, d.h. die Liste mit einer höheren Priorität wird vor eine Liste mit niedrigerer Priorität durchsucht. Innerhalb einer Liste wird Round Robin zur Auswahl eines Threads verwendet. Bei preemptivem Scheduling wird der ausgewählte Thread durch den Dispatcher für die Länge eines Quantums zur Ausführung gebracht. Nach Ablauf des Quantums wird der Thread unterbrochen, und der Warteschlange zurückgeführt.

   Das vorgestellte Verfahren birgt das Risiko, dass Threads mit niedriger Priorität verhungern können (*starvation*). Implementieren Sie im gegebenen Programmrahmen (`osmthread.zip`) ein Ihnen bekanntes Verfahren, um dieses Problem zu lösen. Modifizieren Sie dazu die Funktionen `rr_enqueue` und/oder `rr_dequeue`, sowie die Struktur `tblock_t` innerhalb der Datei `schedule.c`.

   Wir empfehlen Ihnen die Aufgabe auf einem aktuellen GNU/Linux zu lösen.

   Vergleichen Sie das Verhalten Ihrer Implementierung mit der Theorie (zum Beispiel mit Hilfe eines Gantt-Diagramms) und bewerten Sie die Laufzeitkomplexität Ihres Verfahrens.

   Verpacken Sie die Source-Code Dateien und ein Makefile in ein ZIP-Archiv und laden Sie dieses Archiv im Abgabesystem hoch.