Übung 04

Betriebs- und Kommunikationssysteme
Sommersemester 2021
Prof. Dr.-Ing. Jochen Schiller

Abgabe 21.05.2021, 10:00

Aufgabe 1: Scheduling

Erläutern Sie folgende Begriffe aus den Vorlesungsfolien:

1. Beschreiben Sie die folgenden Scheduling-Strategien in maximal zwei Sätzen. Gehen Sie darauf ein, ob die Strategie präemptiv ist und ob / warum Prozesse verhungern können.
   a) RR – Round Robin
   b) HRRN – Highest Response Ration Next
   c) SPN – Shortest Process Next
   d) SRT – Shortest Remaining Time
   e) FCFCS – First Come First Served

Scheduling per Hand Sie haben in der Vorlesung nicht-präemptive Scheduling-Strategien kennengelernt. In der Vorlesung wurden die Strategien jeweils für nur eine CPU vorgestellt. Sie sind aber leicht so erweiterbar, dass sie ankommende Prozesse auf mehreren CPUs verteilen können.

Betrachten wir als Beispiel die Kassen eines Supermarktes. Jede Kasse entspricht einer CPU und jeder Kunde, der an die Kasse kommt, stellt einen Prozess dar. Die Zahl der Artikel, die ein Kunde eingekauft hat, entspricht der Zeit, die ein Prozess zur Ausführung benötigt. Wir nehmen dabei an, dass die Zeit zum Registrieren eines Artikels an der Kasse konstant ist und setzen diese mit einer Zeiteinheit an, den eigentlichen Bezahlvorgang vernachlässigen wir.

Da es Kunden im Supermarkt nicht zugemutet werden kann, andere Kunden vorzulassen, kommt üblicherweise die FCFS-Strategie zum Einsatz, d. h. wer zuerst die Kasse erreicht, wird zuerst abkassiert. Da die Kunden eines Supermarktes auf ihren Vorteil bedacht sind, stellen sie sich immer an der Kasse an, an der die Wartezeit am geringsten ist, d. h. an der vor ihnen die wenigsten Artikel registriert werden müssen.

Ein Supermarkt habe drei Kassen (K1, K2 und K3). An keiner der Kassen wartet ein Kunde. Nun treffen (fast gleichzeitig) Kunden mit folgenden Artikelanzahlen (in dieser Reihenfolge) ein:

$$6, 15, 23, 10, 3, 13, 15, 7, 20, 40, 19, 4, 6, 21 \qquad \text{(14 Kunden)}$$

O. B. d. A. stellt sich der erste Kunde an K1, der zweite an K2, der dritte an K3 an.

- Dokumentieren Sie die sich ergebenden Warteschlangen an den Kassen K1, K2 und K3.
- Wie viele Zeiteinheiten muss ein Kunde im Durchschnitt warten, bis er an der Kasse ankommt? Nach wie vielen Zeiteinheiten ist er im Durchschnitt fertig?

Aufgabe 2: Scheduling-Algorithmen

In dieser Aufgabe werden Sie verschiedene Scheduling-Algorithmen implementieren. Es werden Ihnen 2 Dateien zur Verfügung gestellt:

- `scheduler.h`: In dieser Datei wurde bereits eine Struktur für die Prozessverwaltung definiert.
  - Die Struktur `process` enthält dabei jeweils einen Zeiger sowohl auf den vorigen (`prev`) als auch auf den nächsten Prozess (`next`).
  - `state` speichert, ob ein Prozess aktuell läuft (`PS_RUNNING`), pausiert (`PS_READY`) oder fertig ist (`PS_DEAD`).
  - Aus `cycles_todo`, `cycles_done` und `cycles_waited` können Sie auslesen, wie viele Prozessorzyklen jeder Prozess noch laufen muss, schon gelaufen ist und gewartet hat.
- `scheduler_wrapper.c`: Mit Hilfe dieser Datei können Sie ihre Implementierung testen, sie enthält einen Satz Prozesse, für welche alle Scheduling-Algorithmen ausgeführt werden. Kompilieren und Linken Sie dafür beide C-Dateien zusammen in ein lauffähiges Programm:

```
$ c99 <OPTIONS> -o programm datei1.c datei2.c
```

Die fünf Funktionen, die Sie in einer `scheduler.c` Datei implementieren sollen haben alle die gleiche Signatur (wie in `scheduler.h` deklariert):

1. `void rr(struct process *head)` – Round Robin
2. `void fcfs(struct process *head)` – First Come First Served
3. `void spn(struct process *head)` – Shortes Process Next
4. `void srt(struct process *head)` – Shortest Remaining Time
5. `void hrrn(struct process *head)` – Highest Response Ratio Next

Die Scheduling-Funktionen bekommen die aktuelle Prozessliste übergeben. Das erste Element der Liste (`head`) enthält keine sinnvollen Daten und ist zu ignorieren. Das letzte Element der Liste hat als Nachfolger (`next`) wieder `head`. Dadurch können Sie herausfinden, dass Sie am Ende der Liste angelangt sind.

Sie sollen in jeder Funktion entsprechend der jeweiligen Scheduling-Algorithmen entscheiden, welcher Prozess als nächstes laufen darf. Dafür soll der Zustand dieses Prozesses auf `PS_RUNNING` geändert (oder belassen) werden. Die Zustände der anderen Prozesse dürfen entsprechend *nicht* `PS_RUNNING` sein.

Darf der Prozess, der zuletzt gelaufen ist, nicht mehr laufen, muss sein Zustand entweder auf `PS_READY` (falls `cycles_todo > 0`) oder `PS_DEAD` (falls `cycles_todo == 0`) geändert werden.

Ändern Sie keine anderen Einträge in der Struktur `Process`.