Freie Universität Berlin

Übung 5

Betriebs- und Kommunikationssysteme
Barry Linnert, Sommersemester 2019

Ausgabe 17.05.2019
Abgabe 31.05.2019, 10:15

Bitte beachten Sie die allgemeinen Hinweise auf Übungszettel 1

Aufgabe 1: FIFO-Scheduler

Gegeben sei folgende Scheduling-Strategie mit vier Prioritätsklassen. Jeder Klasse ist eine eigene FIFO-Warteschlange und ein Quantum zugeordnet:

Klasse	Quantum	Priorität
0	1	höchste
1	4	
2	16	
3	∞	niedrigste

Es wird *Process Aging* verwendet, d.h. wenn das Zeitquantum eines Prozesses abgelaufen ist, wird er an das Ende der Warteschlange mit nächstniedriger Priorität gestellt. Neuen Prozessen ist eine bestimmte Priorität zugeordnet. Sie werden an das Ende der Warteschlange ihrer Prioritätsklasse angehängt. Es wird am Ende jeder Zeiteinheit überprüft welcher Prozess am Anfang der nicht leeren Warteschlange mit der höchsten Priorität steht und dieser dann im nächsten Schritt bearbeitet.

Folgende Prozesse sollen betrachtet werden:

Prozess	Ankunftszeit	Priorität	Laufzeit
A	0	0	4
B	0	1	7
C	1	0	1
D	2	1	4
E	5	3	20
F	10	1	3
G	13	0	2
H	15	1	3
I	16	1	3

Ein Prozess, der zum Zeitpunkt t ankommt, wird erst ab dem $(t + 1)$-ten Zeitpunkt berücksichtigt, d.h. er kann frühestens eine Zeiteinheit nach seiner Ankunft die CPU verwenden.

a) Geben Sie für die ersten 20 Zeiteinheiten an, welchem Prozess Rechenzeit zugeteilt wird.

b) Geben Sie für alle Prioritätsklassen an, welche Prozesse sich nach Ablauf der ersten 20 Zeiteinheiten noch in der Warteschlange befinden (in der korrekten Reihenfolge), wie viele Zeiteinheiten jeder einzelne Prozess noch laufen muss und wie viele Zeiteinheiten von seinem Quantum noch übrig sind.

Aufgabe 2: Implementierung eines Schedulers in C

In dieser Aufgabe soll eine Prozessverwaltung mit Hilfe einer doppelt verketteten Liste simuliert werden. Führen Sie Fehlerbehandlung durch und erläutern Sie im Quellcode (als Kommentar) warum Sie sich für genau diese Fehlerbehandlung entschieden haben. Die Ausgaben sollen auf `stdout` ausgegeben werden, die Fehler auf `stderr`. Ihr Programm muss mit folgenden Compilerflags ohne Warnungen und / oder Fehler compilieren:

```
\$ gcc -std=c11 -o <program.out> -Wall -Wextra -pedantic <program.c>
```

Wählen Sie nach Belieben **drei** der folgenden sechs Scheduling Algorithmen aus und implementieren Sie diese:

- `struct Process* rr(struct Process* head, struct Process* current);`
- `struct Process* fcfs(struct Process* head, struct Process* current);`
- `struct Process* spn(struct Process* head, struct Process* current);`
- `struct Process* srt(struct Process* head, struct Process* current);`
- `struct Process* hrrnPreemptive(struct Process* head, struct Process* current);`
- `struct Process* hrrnNonPreemptive(struct Process* head, struct Process* current);`

Die einzelnen Elemente der Prozessliste sind wie folgt definiert:

```
struct Process {        // Elemente der verketteten Liste
    uint64_t cycles_done;   // Anzahl bereits durchlaufener Zyklen
    uint64_t cycles_waited; // Anzahl bereits gewarteter Zyklen
    uint64_t cycles_todo;   // Anzahl der zu verarbeitenden Zyklen
    uint64_t pId;           // Prozess ID

    struct Process* next;   // Doppelt verkettete Liste
    struct Process* prev;
};
```

Die Liste, aus diesen Elementen erstellt hat folgende Eigenschaften:

- doppelte Verkettetung, d.h. jedes Element kennt sowohl seinen Vorgänger als auch seinen Nachfolger
- das erste Element ist ein Dummy, d.h. es trägt keine Informationen
- Kreisverkettung, d.h das erste Element (Dummy) zeigt auf das nächste und das letzte Element zeigt wieder auf das Erste

In der Datei `scheduler.c` sind die Funktionskörper der Scheduling-Funktionen gegeben, die sie implementieren sollen. Die Funktionen sollen nichts weiter tun, als aus der Prozessliste den Prozess auszuwählen, der als nächstes laufen soll. Sie müssen die Statusinformationen der Prozesse **NICHT** aktualisieren!

Testen

Zum testen wird ihnen die Datei `sched_test.c` zur Verfügung gestellt. **AN DIESER DATEI MÜSSEN SIE NICHTS VERÄNDERN!** Die Datei enthält einen Beispiel Testfall. Gehen sie aber davon aus, dass ihr Tutor ihre Abgabe noch mit weiteren Testfällen überprüft!