Fakultät für Informatik und Mathematik
HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

Betriebssysteme, SoSe 2020

Praktische Übung 5: Semaphore und Mutual Exclusion

Ziele des Labortermins

- POSIX Semaphore und Mutexes
- Synchronisation bei Parallelen Programmen

Aufgabe 5.1: Verständnisfragen

a) Welchen fundamentalen Unterschied gibt es bei Threads gegenüber Prozessen in Hinblick auf die Kommunikation zwischen Ihnen?

b) Was unterscheidet Semaphore von Locks?

c) Beschreiben Sie das Grundprinzip von Petersons Lock.

Für diese und die folgenden Übungen, erhöhen Sie die Anzahl der virtuellen Prozessorkerne für die Virtuelle Maschine. Konfigurieren Sie die Virtuelle Maschine dafür im ausgeschalteten Zustand. Nachdem Sie die Virtuelle Maschine dann gestartet haben, führen Sie `cat /proc/cpuinfo` aus um die Einstellung zu verifizieren.

Aufgabe 5.2: Locking mit Semaphoren und Mutex (praktisch)

Es soll ein Counter implementiert werden, der von mehreren Threads verwendet wird.

a) Erstellen Sie ein Programm mit 16 Threads. Erstellen Sie eine globale Variable `counter`, die Sie mit 0 initialisieren.

   Jeder Thread soll folgende Schleife durchführen:

```
for (int i = 0; i < 10000; i++) {
  counter++;
}
```

Warten Sie bis alle Threads beendet sind und geben dann den Wert von `counter` aus. Führen Sie das Programm ohne Synchronisierung aus.

b) Parallelisieren Sie Ihr Programm mit Semaphoren.

c) Parallelisieren Sie Ihr Programm mit POSIX Mutex.

Aufgabe 5.3: Array-Operation parallelisieren (praktisch)

Ähnlich zur vorhergehenden Übung, soll folgende Schleife parallelisiert werden:

```
extern int A[];
int x = 0;
for (int i = 0; i < 10000; i++) {
  x += A[i];
}
```

Sie finden im Ordner `array` die Datei `array.c`, die die Daten enthält und dazu gelinkt werden muss.

a) Führen Sie Ihr Programm ohne Thread-Parallelisierung aus.

b) Überlegen Sie sich eine Strategie mit der Sie die Schleife effizient auf 8 Threads aufteilen können.

c) Verteilen Sie das Programm auf 8 Threads indem Sie den Zugriff auf `x` durch ein Mutex-Lock schützen.

d) Überlegen Sie sich eine bessere Alternative. *Tipp: Das Programm kann ganz ohne Synchronisierung implementiert werden. Beachten Sie dabei die Berechnung und Zusammenführung von Teilergebnissen.*