Christian-Albrechts-Universität zu Kiel
Institut für Informatik und Praktische Mathematik
Prof. Dr. W.-P. de Roever
Dipl.-Inf. K. Stahl

Nebenläufige Programmierung

Sommersemester 2003

Serie 6

14. Mai 2003

Aufgabe 1 **(4 Punkte)**

Der UNIX Kernel stellt Operationen zur Verfügung, die den folgenden ähnlich sind:

- `sleep()` blockiert den aufrufenden Prozess
- `wakeup()` weckt alle blockierten Prozesse

Jede dieser Operationen ist atomar. Die `wakeup()` Operation weckt alle Prozesse auf, die zum Zeitpunkt wenn `wakeup()` aufgerufen wird blockiert sind.
Entwickel ein `mpd` Programm, dass diese Operationen implementiert. Benutze Semaphoren zur Synchronisation.
Tip: Benutze die Methode "passing the baton".

Aufgabe 2 **(4 Punkte)**

Exercise 4.21 auf Seite 197f.

Aufgabe 3 **(3 Punkte)**

Diese Aufgabe dient zur Einführung der `pthreads` library. Es ist vermutlich nützlich, die folgenden manpages zu lesen: `pthreads`, `mutex`, `semaphore`.

(a) Schreibe ein einfaches `pthreads` Programm mit globalen Variablen `x` und `y`, die initial mit 0 belegt sind. Erzeuge vier Threads welche jeweils `x` inkrementieren und danach `y` 10000 mal inkrementieren. Gebe zum Schluss die Finalwerte der Variablen aus.

(b) Modifiziere das Programm, so dass die Inkrementierungen jeweils atomar ausgeführt werden.

(c) Modifiziere das Programm, so dass gegenseitiger Ausschluss für beide Inkrement-Operationen zusammen gilt, d.h. das Erhöhen von `y` soll im gegenseitigen Ausschluss mit dem Erhöhen von `x` stattfinden.

Hinweis: unter Solaris muss mit `gcc -lpthread -lposix4 file.c` compiliert werden.

Ausgabe: Mittwoch,14. Mai 2003
Abgabe: Mittwoch, 21. Mai 2003