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 EST

2. Juli 2003

Bearbeitung dieser Serie in Einzelarbeit!

Alle Lösungsideen sollten zumindest kurz erläutert werden.

Aufgabe 1 (4 Punkte)

[*Bakery algorithm*]

(a) Modifiziere den Bakery Algorithmus (coarse grained, Fig. 3.10 auf S. 112) derart, dass die `turn[i]` Werte begrenzt werden.
Hinweis: Die atomare Anweisung welche `turn[i]` setzt darf nicht geändert werden, das await-Statement wohl. Neue Variablen, z.B. ein `next`, dürfen eingeführt werden.

(b) Modifiziere ebenfalls die fine-grained Variante von Fig. 3.11 auf S. 114.
Hinweis: Es darf hier kein Fetch-and-Add verwendet werden. Man kann beispielsweise versuchen, die turn-Werte von einem Prozess neu anordnen zu lassen. Pseudo-Code reicht aus um dieses zu beschreiben.

Aufgabe 2 (4 Punkte)

[*Message passing using semaphores*]
Sei `synch_send` ein Kommando für synchrones Senden und `receive` das übliche (synchrone) Empfangen. Beide Kommandos sind also blockierend, d.h. jedes der Kommandos erst terminieren nachdem das Gegenstück aufgerufen und die Nachricht ausgetauscht wurde.

Benutze Semaphoren (für wechselseitigen Ausschluss und zum Signalisieren) und entwickle eine Implementierung (Pseudocode) von `synch_send(msg)` und `receive(msg)`. Nimm an, dass es einen Pufferplatz der Größe 1 gibt, um die Nachricht vom Sender zum Empfänger zu kopieren.

Aufgabe 3 (4 Punkte)

[*Minimum of a set*]
Exercise 8.8 auf S. 417 (kein lauffähiges Programm nötig).

Aufgabe 4 (4 Punkte)

[*Stable marriage*]
Exercise 8.18 auf S. 420, *nur Teil (a)* (kein lauffähiges Programm nötig). Erläutere insbesondere die Lösungsstrategie.

Ausgabe: Mittwoch, 2. Juli 2003
Abgabe: Mittwoch, 9. Juli 2003