Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Prof. Dr. W.-P. de Roever
Jens Schönborn, Christian Motika und Tim Fenten

Nebenläufiges Programmieren

Sommersemester 2008 **Serie 6** 14.5 2008

Ausgabetermin: 14.5 2008

Abgabe: 23.5 2008 (11:00)

Aufgabe 1 (5 Punkte) *Mutual exclusion.* Angenommen ein Computer hat einen *atomaren swap* Befehl. Dieser sei wie folgt definiert:
swap(var1, var2):<tmp = var1; var1 = var2; var2 = tmp; >
tmp ist ein internes Register. *swap* darf zur Lösung nicht geändert und muss *sinnvoll* benutzt werden.

1. Benutze swap um eine Lösung für das critical section Problem für n Prozesse zu entwickeln. Die eventual entry Eigenschaft braucht nicht berücksichtigt werden. Beschreibe gründlich wie deine Lösung funktioniert und warum sie korrekt ist.

2. Gebe unter Verwendung des *swap* Befehls eine Lösung für das critical section Problem an. Sie soll eventual entry Eigenschaft bei weakly fair scheduling gewährleisten. Beachte, daß du nicht deine Lösung aus 1 verwenden kannst um die atomare Aktion im Ticket Algorithmus zu implementieren, da man, bezogen auf das scheduling, unter Verwendung von unfairen Komponenten keine faire Lösung erstellen kann. Du darfst annehmen, daß jeder Prozess eine eindeutige Identität, zum Beispiel Integers von 1 bis n hat. Erkläre deine Lösung, und argumentiere überzeugend warum sie korrekt ist und die eventual entry Eigenschaft bei weakly fair scheduling gewährleistet.

Aufgabe 2 (6 Punkte) *Atomic Broadcast.* Gegeben sei ein Producer und n Consumer die auf einen Buffer zugreifen. Der Producer schreibt Nachrichten in den Buffer, die Consumer lesen sie. Jede Nachricht soll von allen n Consumer Prozessen gelesen werden, bevor der Producer die Nachricht in dem Buffer überschreiben darf.

1. Schreibe ein MPD Programm für dieses Problem mit einelementigem Buffer und mit Semaphoren zur Synchronisation.

2. Nimm an dass der Buffer b Plätze hat. Der Producer kann nur in freie Plätze schreiben, und jede Nachricht muss von allen n Consumer Prozessen gelesen sein, bevor der Platz wiederverwendet werden darf. Verschiedene Consumer dürfen aber unterschiedlich viele Nachrichten gelesen haben. Schreibe ein MPD Programm für dieses generellere Problem mit Semaphoren zur Synchronisation.