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 9** 04.06.2008

Ausgabetermin: 04.06.2008

Abgabe: 13.06.2008 (11:00)

Aufgabe 1 (6 Punkte) Betrachte das folgende Speicherzuweisungsproblem: Nehme an, dass es die beiden Operationen `request(amount)` und `release(amount)` gibt, wobei `amount` ganzzahlig sei. Sobald ein Prozess `request` aufruft, wird er dadurch verzögert, bis mindestens `amount` freier Speicher zur Verfügung steht, welcher dem Prozess dann auch zugewiesen wird. Durch den Aufruf von `release` kann ein Prozess wieder Speicher freigeben. Desweiteren ist zu beachten, dass ein Prozess unter Umständen nicht den gesamten zugewiesenen Speicher auf einmal freigeben muss! Notwendige Operationen (e.g., `insert()`, `remove()`) auf geordneten Listen dürfen als gegeben angenommen werden. Dabei ist die der Liste zugrundeliegende Ordnung anzugeben.

(a) Entwickle eine Implementierung der Funktionen `request` und `response`, welche die *shortest-job-next allociation policy* (SJN) verwenden. Insbesondere sollen Anforderungen kleinerer Speicherbereiche eine höhere Priorität gegenüber Anforderungen grösserer Bereiche haben. Die Lösung sollte in der Form jener aus [Andrews, Fig. 4.14] entsprechen.

(b) Entwickle eine Implementierung der Funktionen `request` und `response`, welche eine *first-come-first-served allocation policy* (FCFS) verwenden. Insbesondere könnten also anstehende/wartende Speicheranforderungen verzögert werden, obwohl ausreichend freier Speicher zur Verfügung steht.

Aufgabe 2 (4 Punkte) In dieser Aufgabe soll der *sleeping barber*-Monitor aus [Andrews, Fig. 5.10] in Bezug auf die verwendete *signaling discipline* genauer untersucht werden:

(a) Einige `while`-Schleifen können unter Annahme der *Signal and Continue discipline* durch `if`-Abfragen ersetzt werden. Finde heraus *welche* der *while*-Schleifen ersetzt werden können, begründe dies stichhaltig und modifiziere den Monitor entsprechend.

(b) Ist der Monitor, wie gegeben, korrekt unter Annahme der *Signal and Wait discipline*? Falls dies der so ist, begründe dies stichhaltig, andernfalls modifiziere zusätzlich den Monitor, so dass er korrekt arbeitet.

(c) Ist der Monitor, wie gegeben, korrekt unter Annahme der *Signal and Urgent Wait discipline*? Falls dies der so ist, begründe dies stichhaltig, andernfalls modifiziere zusätzlich den Monitor, so dass er korrekt arbeitet.

Aufgabe 3 (3 Punkte) In dieser Aufgabe soll der *separate disk scheduler*-Monitor, gegeben wie in [Andrews, Fig. 5.13], nähere Betrachtung finden:

(a) Ändere den Monitor so ab, dass er den *elevator*-Algorithmus verwendet. Gebe zu Deiner Lösung auch eine Monitor-Invariante DISK_SCAN an.

(b) Ändere den Monitor so ab, dass er den *shortest-seek-time*-Algorithmus verwendet. Gebe zu Deiner Lösung auch eine Monitor-Invariante DISK_SST an.

Erkläre anhand der Invariante und mittels Kommentaren die Funktionsweise. Die Invariante muss nicht formal bewiesen werden, eine kurze Begründung reicht hier aus.