

Betriebssysteme im Wintersemester 2008/2009

Übungsblatt 8

Abgabetermin: 03.12.2008, 13:30 Uhr

Aufgabe 32: (H) Apfelplantage

Das Erzeuger/Verbraucher-Problem soll anhand einer 'virtuellen Apfelplantage' geübt werden. Auf einer Apfelplantage arbeiten 2 Feldarbeiter und ein Koch. Feldarbeiter müssen essen, um Äpfel pflücken zu können, mit einer Portion Apfelmus hat sich ein Feldarbeiter genug gestärkt um 2 Äpfel zu pflücken (außer Apfelmus gibt es nichts zu essen). Danach ist er wieder hungrig. Der Koch kann aus 12 Äpfeln 8 Portionen Apfelmus kochen, jedoch muss er sich für die schwere Arbeit auch mit einer Portion Apfelmus stärken, vor dem Kochen hat er satt zu sein, danach ist er wieder hungrig. Zu Beginn seien 3 Portionen Apfelmus und 6 Äpfel in der Vorratskammer der Apfelplantage vorhanden.

- a. Der genannte Ablauf führt sicher zu einem Deadlock. Begeben Sie sich mithilfe eines Petri-Netzes auf die Fehlersuche. Zeichnen Sie
 - das Petrinetz
 - einen repräsentativen Ausschnitt des Erreichbarkeitsgraphen dieses Petri-Netzes (Der Ausschnitt des Graphen sollte mindestens 2 Äste, die zu einem Deadlock führen, zeigen.). Geben Sie auch eine Abschätzung über die Anzahl der Äste des Erreichbarkeitsgraphen an. Versuchen Sie im Erreichbarkeitsgraphen (wenn anwendbar) Markierungsmengen wieder zu verwerten, d.h. duplizieren Sie diese Menge nicht, wenn dieselbe Menge auf 2 Pfaden erreichbar ist.
- b. Weshalb bleibt das Apfelplantagen-Beispiel sicher in einem Deadlock stecken?
- c. Woran erkennt man (im allgemeinen) an einem Erreichbarkeitsgraphen, dass ein Ablauf sicher einen Deadlock erzeugt?
- d. Geben Sie 2 mögliche Änderungen des Apfelplantagen-Beispiels an, so dass der Ablauf möglicherweise Deadlockfrei läuft.

Aufgabe 33: (T) Deadlocks: Modellierung

(6 Pkt.)

In dieser Aufgabe werden zwei Methoden zur Modellierung von Deadlocks in Computersystemen betrachtet: der *Resource-Allocation-Graph* und das *Prozessfortschrittsdiagramm*.

- Erläutern Sie beide Verfahren jeweils anhand eines konkreten Beispiels.
- Vergleichen Sie die beiden Modellierungsvarianten anhand sinnvoller Kriterien.

Aufgabe 34: (K) Deadlocks: Wiederholung

(7 Pkt.)

Gegeben sei ein Computersystem mit vier relevanten Betriebsmitteln/Ressourcen, die zum betrachteten Zeitpunkt t_0 ungebunden sind. Alle Betriebsmittel sind exklusiv. Drei Prozesse konkurrieren um die Betriebsmittel und tätigen die in der folgenden Tabelle skizzierten Anforderungen. Ein Tabelleneintrag $A(R_n)$ in der Zeile P_i und der Spalte t bedeutet, dass der Prozess P_i zum Zeitpunkt $t_0 + t$ auf das Betriebsmittel R_n zugreifen möchte. Eine Anforderung wird sofort in eine feste Ressourcenbindung umgesetzt, wenn die angeforderte Ressource zum jeweiligen Zeitpunkt frei ist. Andernfalls muss der Prozess so lange warten, bis das Betriebsmittel wieder freigegeben wurde.

	1	2	3	4	5	6	7
P_1	$A(R_1)$				$A(R_4)$		
P_2		$A(R_2)$				$A(R_1)$	
P_3			$A(R_4)$	$A(R_3)$	$A(R_2)$		

- Skizzieren Sie alle Ressourcenbindungen und -anforderungen zum Zeitpunkt $t_0 + 7$ in einem Resource-Allocation-Graph.
- Entscheiden Sie anhand des Graphen, ob sich das System zu diesem Zeitpunkt in einem Deadlock befindet. Begründen Sie Ihre Antwort.
- Deadlocks lassen sich durch eine lineare Ordnung F auf den Ressourcen vermeiden, d.h. auf Betriebsmittel dürfen nur in einer vorher bestimmten Reihenfolge zugegriffen werden. Es sei die Ordnung $F(R_n) = n$ gegeben, also Ressourcenanforderungen können nur in aufsteigender Reihenfolge erfüllt werden. Welches Problem entsteht hierbei für das obige Beispiel, wenn Sie annehmen, dass die Reihenfolge der Ressourcenanforderungen für einen Prozess nicht verändert werden darf? Welche Schlussfolgerung lässt dies über das Verfahren der linearen Ordnung auf Ressourcen zur Deadlock-Vermeidung zu?
- Gibt es überhaupt eine Ordnung $F(R_n)$, sodass dieses Problem speziell für das gegebene Beispiel nicht auftritt? Geben Sie F an oder beweisen Sie, dass ein solches F nicht existiert.

Aufgabe 35: (P) Scheduler-Simulation: Implementierung (8 Pkt.)

Implementieren Sie einen einfachen Priority-Scheduler in Java. Die Java-Klassen `Simulation`, `Scheduler`, `PriorityProcess` und `P` sind bereits gegeben. Aus Platzgründen wurden die Klassen nicht auf dieses Übungsblatt gedruckt. Bitte laden Sie sich die Dateien von der Info3-Homepage herunter. Verwenden Sie außerdem die Klasse `Process`, die Sie in Aufgabe 30 programmiert haben.

- a. Auf der Info3-Homepage finden Sie die Klasse `PriorityScheduler`. Ergänzen Sie den fehlenden Code an der dafür markierten Stelle. Verwenden Sie die Methoden `Thread.suspend()` und `Thread.resume()`, falls erforderlich. Achten Sie darauf, alle relevanten Aktionen durch Verwendung von `PriorityProcess.printInfo()` zu dokumentieren. Kommentieren Sie Ihre Lösung sinnvoll.
- b. Beschreiben Sie kurz Sinn und Ablauf der `main`-Methode der Klasse `Simulation`.
- c. Starten Sie die Simulation. In der Programmausgabe werden auch die Ausführungszeiten der Prozesse angegeben, wenn diese fertig gerechnet haben. Führen Sie die Simulation drei mal durch und dokumentieren Sie die Zeiten in einer Tabelle. Wie erklären Sie sich eventuelle Differenzen?