Institut für Informatik

Arbeitsgruppe Autonome Intelligente Systeme

Freiburg, 15. Dezember 2015

Systeme I: Betriebssysteme
Übungsblatt 8

Aufgabe 1 (1+2 Punkte)

Semaphore werden verwendet, um den zeitlichen Ablauf von Prozessen zu synchronisieren. Fügen Sie in den Code der folgenden Teilaufgaben Semaphore und ihre Methodenaufrufe `up()` und `down()` ein, um den korrekten zeitlichen Ablauf zu garantieren. Sie dürfen keine anderen Konstrukte (Zähler usw.) verwenden. Beschreiben Sie außerdem kurz in eigenen Worten wie Ihre Lösung funktioniert. Erläutern Sie bei der Initialisierung jedes Semaphors kurz dessen Aufgabe.

a) Zwei Prozesse sollen parallel jeweils ein Teilergebnis berechnen. Der erste Prozess soll danach beide Teilergebnisse addieren.

Stellen Sie mithilfe von Semaphoren sicher, dass der Prozess A erst dann die Summe bildet, wenn der Prozess B sein Teilergebnis berechnet hat.

Gemeinsame Initialisierung

```
1 a := 0
2 b := 0
3 summe := 0
4 // definieren und initialisieren Sie hier Ihre Semaphore
```

Prozess A

```
5 a := berechne_Teilergebnis_a()
6 summe := a + b
```

Prozess B

```
b := berechne_Teilergebnis_b()
```

b) Drei Arbeiter sollen möglichst schnell eine Maschine warten. Die Arbeiter sollen parallel ihre Aufgaben durchführen, allerdings natürlich nur, wenn die Maschine ausgeschaltet ist.

Stellen Sie mithilfe von Semaphoren sicher, dass die Arbeiter nur dann arbeiten, wenn die Maschine ausgeschaltet ist.

Gemeinsame Initialisierung

```
1 // definieren und initialisieren Sie hier Ihre Semaphore
```

Arbeiter A

```
2
3 Zahnrad_wechseln()
4
```

Arbeiter B

```
Schrauben_anziehen()
```

Arbeiter C

```
Maschine_ausschalten()
Material_nachfuellen()
Maschine_einschalten()
```

.

Aufgabe 2 (1+1+1+1+1 Punkte)

Drei Prozesse (p_1, p_2, p_3) eines Systems müssen nach folgendem Schema auf drei Ressourcen (A, B, C) zugreifen:

Prozess 1

```
1: Anforderung B
2: Anforderung A
3: Freigabe B
4: Anforderung C
5: Freigabe A
6: Freigabe C
```

Prozess 2

```
1: Anforderung A
2: Anforderung C
3: Freigabe A
4: Anforderung B
5: Freigabe C
6: Freigabe B
```

Prozess 3

```
1: Anforderung B
2: Anforderung A
3: Anforderung C
4: Freigabe B
5: Freigabe A
6: Freigabe C
```

Die Prozesse werden pseudoparallel abgearbeitet und es kann jederzeit ein Prozesswechsel stattfinden. Die einzelnen Operationen jeder Zeile sind atomar.

a) Zeichnen Sie ein Ressourcendiagramm (siehe Vorlesung Kap. 6: Deadlocks, Folie „Ressourcendiagramm: Deadlock“) mit Prozess 1 auf der horizontalen und Prozess 2 auf der vertikalen Achse. Die Markierungen auf den Achsen entsprechen den Zeitpunkten, zu denen die zugehörige Code-Zeile ausgeführt wird.

b) Woran erkennen Sie an dem Diagramm aus a), dass ein Deadlock bei jeder beliebigen Ausführungsreihenfolge garantiert ausgeschlossen ist?

c) Zeichnen Sie ein weiteres Ressourcendiagramm mit Prozess 2 auf der horizontalen und Prozess 3 auf der vertikalen Achse.

d) Zeichen Sie in dem Diagramm aus c) eine Ressourcenspur (auf der Vorlesungsfolie schwarze gestrichelte Linie) ein, die zu einem Deadlock zwischen diesen beiden Prozessen führt, und geben Sie die dazugehörige Ausführungsreihenfolge an.

e) Erweitern Sie die Ausführungsreihenfolge aus d) um die Ausführung von Prozess 1, sodass alle drei Prozesse in einem Deadlock enden. Zeichen Sie für diese Ausführungsreihenfolge den Belegungs-Anforderungs-Graphen, wie er in der Vorlesung vorgestellt wurde; geben Sie dabei die einzelnen Zwischenschritte des Graphen an.

Aufgabe 3 (2+2 Punkte)

Vier Prozesse (p_1, p_2, p_3, p_4) greifen auf fünf Ressourcenklassen $(K_1, K_2, K_3, K_4, K_5)$ zu. Der Vektor insgesamt verfügbarer Ressourcen ist $V = (5, 7, 7, 6, 9)$, die Maximalanforderungsmatrix ist

$$M = \begin{pmatrix} 3 & 6 & 5 & 2 & 4 \\ 2 & 1 & 7 & 4 & 5 \\ 4 & 2 & 4 & 1 & 2 \\ 4 & 5 & 2 & 4 & 4 \end{pmatrix}$$

und die Belegungsmatrix zum Zustand Z_1 ist

$$E_{Z_1} = \begin{pmatrix} 0 & 2 & 1 & 2 & 1 \\ 1 & 0 & 1 & 0 & 4 \\ 2 & 2 & 0 & 0 & 2 \\ 0 & 2 & 1 & 3 & 2 \end{pmatrix}.$$

Dieser Zustand Z_1 ist laut Bankieralgorithmus „sicher“ (dies brauchen Sie hier nicht zu zeigen).

Die Prozesse p_1 und p_4 sind rechenbereit und fordern in zwei separaten Schritten eine Ressource aus der Klasse K_2 an. Das Betriebssystem muss nun mithilfe des Bankier-Algorithmus entscheiden, welchen der beiden Prozesse es als nächstes ausführen kann ohne einen Deadlock zu riskieren.

Geben Sie für die folgenden Zustände mittels des Bankieralgorithmus an, ob sie „sicher“ oder „unsicher“ sind. Begründen Sie Ihre Aussage und geben Sie für jeden Zwischenschritt des Algorithmus den Ressourcenrestvektor F an:

a) Der Zustand Z_2, der sich aus dem ursprünglichen Zustand Z_1 ergibt, wenn das Betriebssystem dem Prozess p_1 eine weitere Ressource aus der Klasse K_2 zuteilt.

b) Der Zustand Z_3, der sich aus dem ursprünglichen Zustand Z_1 ergibt, wenn das Betriebssystem stattdessen dem Prozess p_4 eine weitere Ressource aus der Klasse K_2 zuteilt.

Abgabe: Als PDF-Datei im Ilias bis 21. Dezember 2015, 23:59 Uhr