Institut für Informatik

Arbeitsgruppe Autonome Intelligente Systeme

Freiburg, 7. Februar 2017

Systeme I
Übungsblatt 14 - Wiederholung

Aufgabe 1 (2+1 Punkte)

Wechselseitiger Ausschluss - Petersons Algorithmus

In der Vorlesung wurden mehrere Lösungsversuche vorgestellt, mit denen eine Softwarelösung für den wechselseitigen Ausschluss gefunden werden sollte. Hier geht es um den *Peterson-Algorithmus*:

Gemeinsame Initialisierung

```
1  flag[0] := false;
2  flag[1] := false;
3  turn := 0;
```

Prozess 0

```
4  wiederhole
5  {
6    flag[0] := true;
7    turn := 1;
8    solange (flag[1] = true und turn = 1)
9      tue nichts;
10
11   Anweisung 1      } kritische Region
12   Anweisung 2      }
13   ...              }
14
15   flag[0] := false;
16
17   Anweisung 3      } nichtkritische Region
18   Anweisung 4      }
19   ...              }
20 }
```

Prozess 1

```
wiederhole
{
  flag[1] := true;
  turn := 0;
  solange (flag[0] = true und turn = 0)
    tue nichts;

  Anweisung 5      } kritische Region
  Anweisung 6      }
  ...              }

  flag[1] := false;

  Anweisung 7      } nichtkritische Region
  Anweisung 8      }
  ...              }
}
```

Die Anweisungen in den kritischen und nichtkritischen Regionen ändern nichts an den Variablen `flag[0]`, `flag[1]` und `turn`.

Im Folgenden soll per Widerspruchsbeweis gezeigt werden, dass diese Lösung den wechselseitigen Ausschluss auf die kritische Region garantiert, d.h., dass die beiden Prozesse niemals gleichzeitig in der kritischen Region sein können.

Nehmen Sie dazu an, dass die Prozesse 0 und 1 zu einem Zeitpunkt t beide in der kritischen Region seien. Die Zeitpunkte, zu denen die Prozesse 0 und 1 die `solange`-Schleife zuletzt verlassen haben, seien t_0 und t_1. Ohne Beschränkung der Allgemeinheit sei $t_0 > t_1$.

Der Beweis beruht auf einer Fallunterscheidung darüber, aus welchem Grund Prozess 0 die `solange`-Schleife zum Zeitpunkt t_0 verlassen konnte. Den Fall `turn` $\neq$ `1` brauchen Sie an dieser Stelle nicht zu betrachten.

a) Betrachten Sie den Fall, dass `flag[1] = false` zum Zeitpunkt t_0. Zeigen Sie, dass diese Annahme zum Widerspruch führt.

b) Welchen Nachteil hat Petersons Lösungsversuch? Wie kann man diese Art von Nachteil umgehen (allgemein, nicht auf Petersons Algorithmus bezogen)?

Aufgabe 2 (3 Punkte)

Produzenten/Konsumenten-Problem

In der Vorlesung haben Sie das *Produzenten/Konsumenten-Problem* kennengelernt. Sie sehen hier eine Variante der Lösung aus der Vorlesung. Es wurden lediglich bei der Prozedur `producer` die Reihenfolge der Befehle `down(empty);` und `down(mutex);` vertauscht:

```
Semaphore mutex; count_mutex := 1;
Semaphore empty; count_empty := MAX_BUFFER;
Semaphore full;  count_full  := 0;
```

```
1  Prozedur producer
2  {
3    wiederhole
4    {
5      item := produce_item();
6
7      down(mutex);     /* Reihenfolge */
8      down(empty);     /* vertauscht  */
9
10     insert_item(item);
11
12     up(mutex);
13     up(full);
14   }
15 }
16
17 Prozedur consumer
18 {
19   wiederhole
20   {
21     down(full);
```

```
22      down(mutex);
23
24      item := remove_item();
25
26      up(mutex);
27      up(empty);
28
29      consume_item(item);
30    }
31  }
```

Funktioniert diese Variante der ursprünglichen korrekten Lösung fehlerfrei? Beweisen Sie entweder, dass Deadlocks bei diesem Algorithmus garantiert ausgeschlossen sind, oder geben Sie eine Ausführungsreihenfolge an, die zu einem Deadlock führt.

Aufgabe 3 (2+2+2 Punkte)

Deadlocks

Zwei Prozesse wollen auf vier Ressourcen `A`, `B`, `C` und `D` zugreifen. Folgende Tabelle zeigt, in welcher Reihenfolge die Prozesse Ressourcen anfragen und freigeben. Wir vernachlässigen hier Befehle, die zwischen den Anforderungen und Freigaben stehen.

Prozess 1	Prozess 2
1: Anforderung B	1: Anforderung A
2: Anforderung A	2: Anforderung D
3: Anforderung C	3: Freigabe A
4: Freigabe B	4: Anforderung B
5: Anforderung D	5: Anforderung C
6: Freigabe C	6: Freigabe B
7: Freigabe D	7: Freigabe C
8: Freigabe A	8: Freigabe D

a) Zeichnen Sie in das Diagramm in Abbildung 1 auf Seite 4 die Bereiche ein, in der beide Prozesse auf eine Ressource zugreifen würden. Die horizontale Achse repräsentiert den Programmfortschritt von Prozess 1 und die vertikale Achse repräsentiert den Programmfortschritt von Prozess 2. Die mit einer Nummer versehenen horizontalen und vertikalen Linien sind die Zeitpunkte in der Programmausführung, bei deren Überschreitung die entsprechend nummerierte Zeile des Prozesses ausgeführt wird.

b) In diesem Szenario kann es zu einem Deadlock kommen. Zeichnen Sie den Bereich ein, in dem ein Deadlock unvermeidlich ist und markieren Sie die Stelle, an dem der Deadlock eintritt. Begründen Sie kurz, wieso an dieser Stelle der Deadlock eintritt.

c) Geben Sie schriftlich in Form von Stichpunkten eine Ausführungsreihenfolge an, die deadlockfrei ist, und eine, die zu einem Deadlock führt.

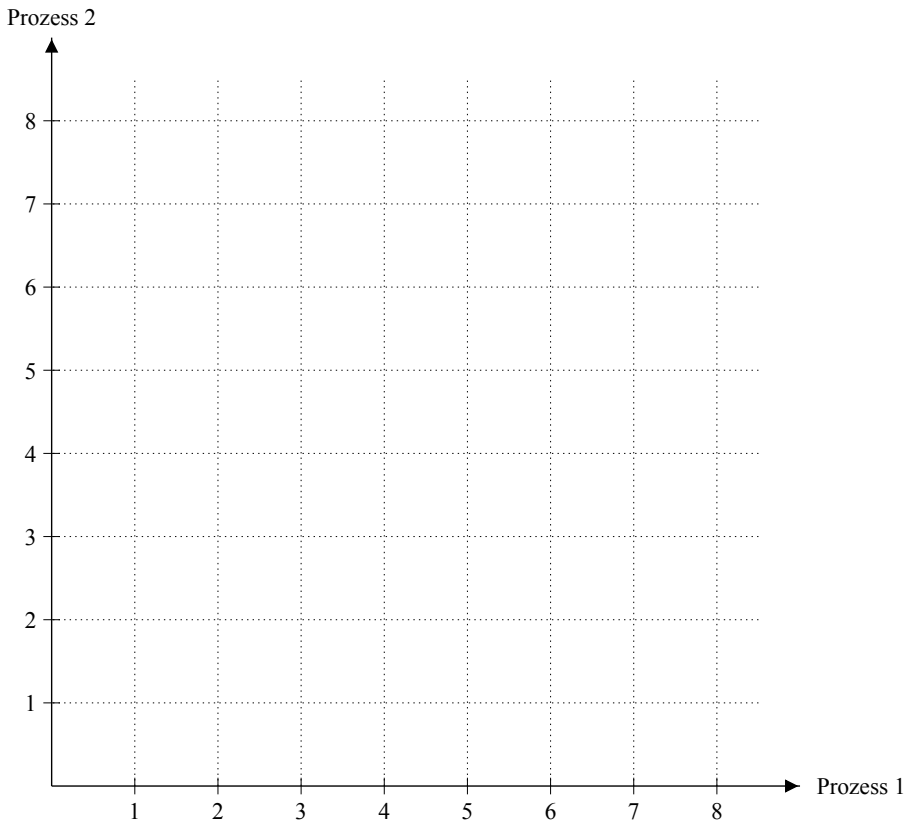

Abbildung 1: Zeichnen Sie hier die Bereiche ein.

Aufgabe 4 (4+1+1+1 Punkte)

Deadlocks und Bankieralgorithmus

a) Nennen Sie die vier Voraussetzungen, die erfüllt sein müssen, damit ein Ressourcen-Deadlock auftreten kann und erklären Sie jeden Punkt kurz.

b) Beim Bankieralgorithmus wurde der Begriff des „sicheren Zustands“ verwendet. Wie lautet die Definition eines sicheren Zustandes?

c) Führt jeder unsichere Zustand unweigerlich in einen Deadlock? Begründen Sie Ihre Antwort.

d) Drei Prozesse p_1, p_2 und p_3 greifen auf Ressourcen einer einzigen Ressourcenklasse zu. Insgesamt stehen $V = 7$ Ressourcen zur Verfügung. Für die maximale Anzahl M_i von Ressourcen, auf die die Prozesse zugreifen werden, und für die Anzahl von Ressourcen E_i, die die Prozesse schon erhalten haben, gilt:

	M_i	E_i
p_1	6	2
p_2	3	2
p_3	6	2

Ist dieser Zustand ein „sicherer Zustand“? Begründen Sie Ihre Antwort.

Aufgabe 5 (3+2+3 Punkte)

Sicherheit

a) Entscheiden Sie, ob die folgenden Aussagen richtig oder falsch sind.

	Behauptung	richtig	falsch
1.	Bei der Caesar-Chiffre lässt sich der Schlüssel bereits aus einem Klartext-Chiffrat-Paar berechnen.	☐	☐
2.	AES gilt für Schlüssellängen ab 192bit als empirisch sichere Stromchiffre.	☐	☐
3.	Eine SSH-Verbindung ist stets gegen Man-in-the-middle-Attacken sicher.	☐	☐
4.	Blockchiffren werden wegen ihrer beweisbaren Sicherheit verwendet.	☐	☐
5.	Pseudozufallszahlengeneratoren erzeugen deterministische Zahlenfolgen basierend auf einer zufälligen Initialisierung.	☐	☐
6.	Ein mit Hilfe des Diffie-Hellman-Verfahrens berechneter Schlüssel wird häufig im Nachgang für symmetrische Verschlüsselung verwendet.	☐	☐

b) Alice möchte mit dem ElGamal-Verfahren den Klartext $P = 2$ an Bob schicken. Bobs öffentlicher Schlüssel lautet $K_{\text{public, Bob}} = (p, g, h) = (17, 3, 12)$.

   1) Nehmen Sie an, dass Alice die Zufallszahl $y = 2$ wählt. Geben Sie den Inhalt der Nachricht von Alice an Bob an.

   2) Angenommen, Alice verwendet stets das gleiche y. Welches Problem ergibt sich?

c) Wir betrachten eine Blockchiffre, für die als Betriebsmodus ein modifiziertes Cipher-Block-Chaining (CBC)-Verfahren verwendet wird, in dem der Initialisierungsvektor nicht zufällig ist, sondern aus dem kryptographischen Hash des Klartexts besteht.

   Angenommen, es existieren nur zwei verschiedene Klartexte (z.B. „ja“ oder „nein“). Die beiden möglichen Klartexte seien dem Angreifer bekannt. Der Angreifer besitzt außerdem Zugriff auf ein Klartext-Chiffrat-Paar für einen der möglichen Klartexte aus früherer Kommunikation. Ist die Kommunikation noch vertraulich? Begründen Sie oder geben Sie einen Angriff an.

Abgabe: Als PDF-Datei im Ilias bis zum 13. Februar 2017, 23:59 Uhr; zu diesem Übungsblatt findet kein Tutorium statt