UNIVERSITÄT SIEGEN

Fakultät IV
Betriebssysteme und verteilte Systeme
Prof. Dr. rer. nat. Roland Wismüller

Aufgabenblatt 6

(Besprechung am Mo., 26.05.25 und Fr., 30.05.25)

Vorlesung Betriebssysteme und nebenläufige Programmierung
Sommersemester 2025

Aufgabe 1: Streng abwechselnde Synchronisation

Gegeben sind die beiden folgenden Threads:

Thread 0:	**Thread 1:**

while (true)
{
  print(”1”);
}
	while (true)
	{
	  print(”2”);
	}

Diese beiden Threads sollen **mit Hilfe von Semaphoren** so synchronisiert werden, dass ihre Ausgaben immer streng abwechselnd erfolgen. Die Ausgabe sollte also **`1 2 1 2 1 2`**... sein.

a) Geben Sie eine Lösung für diese Synchronisation an!

b) Das Java-Programm, das Sie auf der Vorlesungs-Webseite [1] finden, erlaubt Ihnen unter anderem, eine falsche Lösung für das Problem zu simulieren. Starten Sie das Programm dazu mit

```
java -cp Demo.jar Demo Abwechselnd wrong=1
```

Erzeugen Sie zunächst einen Ablauf, der die richtige Ausgabe produziert. Versuchen Sie dann, einen Ablauf zu erzeugen, bei dem eine unzulässige Ausgabe entsteht. Welche möglichen Ausgaben erzeugt dieses Programm?

Aufgabe 2: Implementierung eines Semaphors (Pflichtaufgabe für die Studienleistung! Abgabe bis So., 01.06., 23:59 über moodle)

Implementieren Sie eine Java Klasse **`Semaphore`**, die ein (zählendes) Semaphor realisiert. Verwenden Sie dazu **ausschließlich** die Synchronisationskonstrukte, die die Sprache Java zur Verfügung stellt (`synchronized`, `wait()`, `notify()`, `notifyAll()`); benutzen Sie also **keine** Klassen aus den Paketen `java.util.concurrent` bzw. `BSSync`! Ihre Klasse soll folgenden Aufbau haben:

```
public class Semaphore
{
  // Erzeugt neues Semaphor mit initialem Wert
  public Semaphore(int val) { ... }

  // P–Operation
  public void P() { ... }

  // V–Operation
  public void V() { ... }
}
```

[1] `https://www.bs.informatik.uni-siegen.de/lehre/material/bs1/demos/Demo.jar`

Erstellen Sie zusätzlich ein Hauptprogramm, um Ihre Lösung zu testen. Erzeugen Sie dazu zwei Threads, die jeweils eine Meldung ausgeben. Synchronisieren Sie die beiden Threads dann mit Hilfe von zwei Semaphoren so, daß ihre Ausgaben streng abwechselnd erfolgen (siehe vorherige Aufgabe).

Aufgabe 3: Erzeuger/Verbraucher Synchronisation

Betrachten Sie die Lösung des Erzeuger/Verbraucher-Problems aus der Vorlesung:

```
Semaphore mutex = 1;
Semaphore full = 0;
Semaphore empty = N;
```

Erzeuger:	**Verbraucher:**

while (true)
{
  item = Produce();
  P(empty);
  P(mutex);
  insert_item(item);
  V(mutex);
  V(full);
}
	while (true)
	{
	  P(full);
	  P(mutex);
	  item = remove_item();;
	  V(mutex);
	  V(empty);
	  Consume(item);
	}

a) Ist die Reihenfolge der P-Operationen wichtig? Das heißt: was würde passieren, wenn man im Erzeuger erst **`P(mutex)`** und dann **`P(empty)`** schreibt? Oder analog im Verbraucher erst **`P(mutex)`** und dann **`P(full)`**?

b) Ist die Reihenfolge der V-Operationen wichtig? Das heißt: was würde passieren, wenn man im Erzeuger erst **`V(full)`** und dann **`V(mutex)`** schreibt? Oder analog im Verbraucher erst **`V(empty)`** und dann **`V(mutex)`**?

Aufgabe 4: Synchronisation mit Semaphoren

Betrachten Sie das folgende Threadsystem:

```
// Semaphore und globale Variable
Semaphore S = 2;
Semaphore T = 0;
```

// Thread A	**// Thread B**	**// Thread C**

while (true)
{
  P(S);
  print("A");
  V(T);
}
	while (true)
	{
	  P(S);
	  print("B");
	  V(T);
	}
		while (true)
		{
		  P(T);
		  print("C");
		  V(S);
		}

Welche der folgenden Aussagen sind richtig?

a) Alle drei Threads können gleichzeitig die Prozedur print() ausführen.
b) Es wird immer zuerst ein A ausgegeben.
c) Es wird nie zuerst ein C ausgegeben.
d) Die Ausgabe kann mit ABA ... beginnen.
e) Die Ausgabe kann mit ABCC ... beginnen.
f) Die Ausgabe kann die Folge ... CCCCC ... enthalten.
g) Das System kann sich verklemmen.

Anmerkung: dies ist eine Aufgabe im Klausurstil.

Aufgabe 5: Leser-Schreiber Synchronisation

Geben Sie eine Lösung in einer Pseudo-Programmiersprache für ein modifiziertes Leser/Schreiber (*Reader/Writer*) Problem (siehe Vorlesung Kapitel 3.8) unter Verwendung von **Semaphoren** an. Die Modifikation besteht darin, dass von allen Lesern nur maximal 3 gleichzeitig auf die Daten zugreifen dürfen.

Aufgabe 6: Steuerung eines Warenautomaten

Ein Warenautomat enthält zu Beginn 200 Chips-Tüten (er ist damit vollständig gefüllt) und führt eine Endlosschleife aus, deren Rumpf die folgenden Schritte enthält:

- Warten, wenn der Automat leer ist,
- Warten, bis ein Kunde 5 Euro eingeworfen hat,
- Erniedrigen eines Zählers für die Anzahl der Chips-Tüten und Chips-Tüte ausgeben.

Es gibt mehrere Kunden, die jeweils folgende Schritte ausführen:

- Warten, solange ein anderer Kunde den Automaten benutzt,
- den Betrag in fünf einzelnen Eurostücken einwerfen,
- Warten, bis die Ware kommt (das kann durchaus länger dauern, da erst der Lieferant kommen muß, wenn der Automat leer ist),
- den Automaten wieder freigeben.

Zudem gibt es einen Lieferanten, der alle zwei Tage kommt und den Automaten wieder vollständig (mit maximal 200 Chips-Tüten) nachfüllt.

Geben Sie eine Lösung in einer Pseudo-Programmiersprache für die drei Prozesse (Automat, Kunde und Lieferant) an. Identifizieren Sie alle Synchronisationsbedingungen zwischen den Prozessen. Benutzen Sie selbst definierte Semaphore, um eine korrekte Synchronisation zu implementieren.