Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Christian Böhm
Sebastian Goebl

München, 02.05.2016

Einführung in die Informatik: Systeme und Anwendungen

SS 2016

Übungsblatt 4: Semaphore

Abgabe bis 09.05.2016, 14:00 Uhr

Besprechung am 09./10.05.2016

Aufgabe 4-1 *Semaphore* (8 Punkte)

Hausaufgabe

Das Problem des *schlafenden Barbiers* ist ein bekanntes Beispiel für die Anwendung von Semaphoren zur Synchronisation von Prozessen. Der Barbier arbeitet in einem Laden mit einem Barbierstuhl und einem Warteraum, in dem Platz für vier Kunden vorhanden ist. Es gelten folgende Verhaltensregeln:

- Der Barbier verbringt seine Zeit mit Haare schneiden, Kassieren und Schlafen (bis ihn ein Kunde weckt).
- Ein Kunde darf den Laden nicht betreten, wenn der Warteraum voll ist.
- Hat ein Kunde den Laden betreten, nimmt er im Warteraum Platz.
- Wenn der Barbier frei ist, bedient er einen der wartenden Kunden.
- Wenn ein Kunde bedient wurde, muss dieser den Barbier bezahlen.

Geben Sie Prozessbeschreibungen für den Barbier und die Kunden an. Die Synchronisation soll dabei über Zählsemaphore erfolgen. Folgende Semaphorvariablen stehen zur Verfügung:

```
VARIABLES warteraumFrei, barbierBereit, kundeBereit,
rasurFertig, zahlung: SEMAPHORE
```

Aufgabe 4-2 *Semaphore* (2+6 Punkte)
Hausaufgabe

Betrachten Sie die folgende Variante des Erzeuger-Verbraucher-Problems. Gegeben seien ein Erzeuger E, der Produkte erzeugt und in eine Zwischenablage legt, sowie ein Verbraucher V, der Produkte aus der Zwischenablage nimmt und dann verbraucht. Die Kapazität der Zwischenablage ist auf ein Produkt beschränkt.

Der Erzeuger E führt über die Anzahl der erzeugten Produkte und die Anzahl der in der Zwischenablage abgelegten Produkte Buch. Beim Erzeugen wird die Integer-Variable *erzeugt* um 1 erhöht. Beim Ablegen wird die Integer-Variable *abgelegt* um 1 erhöht.

Der Verbraucher V führt über die Anzahl der aus der Zwischenablage genommenen Produkte und die Anzahl der verbrauchten Produkte Buch. Beim Nehmen wird die Integer-Variable *genommen* um 1 erhöht. Beim Verbrauchen wird die Integer-Variable *verbraucht* um 1 erhöht.

Erzeuger und Verbraucher seien durch folgende Prozessbeschreibungen gegeben. Die einzelnen Aktionen sind dabei zusätzlich mit einer Nummer versehen.

```
VARIABLES nichtVoll, nichtLeer: SEMAPHORE
init(nichtLeer, 0);
init(nichtVoll, 1);

VARIABLES erzeugt, abgelegt, genommen, verbraucht: INT
erzeugt = 0;
abgelegt = 0;
genommen = 0;
verbraucht = 0;
```

```
   PROZESS E
     BEGIN
       WHILE true DO {
(1)      erzeugt = erzeugt + 1;
(2)      wait(nichtVoll);
(3)      abgelegt = abgelegt + 1;
(4)      signal(nichtLeer);
       }
     END
```

```
PROZESS V
  BEGIN
    WHILE true DO {
(1)   wait(nichtLeer);
(2)   genommen = genommen + 1;
(3)   signal(nichtVoll);
(4)   verbraucht = verbraucht + 1;
    }
  END
```

(a) In dieser Variante des Erzeuger-Verbraucher-Problems ist der Platz in der Zwischenablage auf ein Produkt beschränkt. Begründen Sie kurz, warum in diesem Fall allein durch die Semaphoren *nichtVoll* und *nichtLeer* der wechselseitige Ausschluss der Zugriffe auf die Zwischenablage gewährleistet ist.

(b) Vervollständigen Sie die die Prozessablauftabelle auf dem Arbeitsblatt, so dass genau der angegebene Endzustand erreicht wird. Insbesondere sollen am Ende des Ablaufs 4 Produkte erzeugt, 3 Produkte abgelegt, 2 Produkte genommen und 1 Produkt verbraucht worden sein.

Hinweise: Sie benötigen nicht alle zur Verfügung stehenden Zeilen für eine richtige Lösung. In die Spalten nichtVoll und nichtLeer sind sowohl die Werte als auch die Warteschlangen der Semaphoren einzutragen. Benutzen Sie eine neue Zeile für jede einzelne Aktion der Prozesse. Die Zustandsänderungen der Variablen sollen eindeutig nachvollziehbar sein. Die Menge *ready* enthält stets alle laufenden Prozesse, die nicht blockiert sind.