Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Claudia Linnhoff-Popien

mobile and distributed systems group

Betriebssysteme im Wintersemester 2017/2018
Übungsblatt 12

Abgabetermin: 29.01.2018, 18:00 Uhr

Besprechung: Besprechung der T-Aufgaben in den Tutorien vom 22. – 26. Januar 2018
Besprechung der H-Aufgaben in den Tutorien vom 29. Januar – 02. Februar 2018

Ankündigungen: Die **Klausur** findet am **8. Februar 2018 von 18.30 - 20.30 Uhr** statt. Bitte melden Sie sich **bis spätestens 5. Februar 2018, 12:00 Uhr** über Uniworx zur Klausur **an** bzw. **ab**.

Aufgabe 51: (T) Semaphore und Prozessabläufe

(– Pkt.)

Wir betrachten im Folgenden die drei Prozesse P_1, P_2 und P_3, die ihren Zugriff auf gemeinsam genutze Ressourcen über die drei Zählsemaphore a, b und c synchronisieren. Für jeden der drei Prozesse zeigen die folgenden Programmsequenzen den Zugriff der Prozesse auf die entsprechenden Semaphore.

Prozess P_1

```
1  p1() {
2      ...
3      wait(a);
4      wait(a);
5      wait(a);
6      ...
7      signal(b);
8      signal(b);
9      ...
10 }
```

Prozess P_2

```
1  p2() {
2      ...
3      wait(c);
4      wait(c);
5      wait(c);
6      ...
7      signal(b);
8      ...
9  }
```

Prozess P_3

```
1  p3() {
2      ...
3      wait(b);
4      ...
5      signal(a);
6      signal(c);
7      ...
8  }
```

Ermitteln Sie für die vier folgenden Initialisierungen der Semaphore a, b und c, in welcher Reihenfolge die Prozesse P_1, P_2 und P_3 geschedult werden müssen, um möglichst viele dieser Prozesse vollständig abzuarbeiten und geben Sie die Bezeichner der vollständig abgearbeiteten Prozesse an. Gehen Sie dabei von einem preemptiven Scheduling Algorithmus aus. Verwenden Sie folgende Darstellung zur Veranschaulichung der Abläufe, die in einem Deadlock bzw. einem blockierten Zustand enden oder zu einem erfolgreichen Durchlauf führen. Markieren Sie mögliche Deadlocks bzw. Blockierungen:

Prozess	Ausgeführte Codezeile
...	...

Hinweis: Eine Codezeile soll erst dann als ausgeführt angesehen werden, wenn der entsprechende Funktionsaufruf zurückgekehrt ist.

Initialisierung	i)	ii)	iii)	iv)
Semaphor a	2	2	1	3
Semaphor b	0	0	1	0
Semaphor c	2	3	2	0

Aufgabe 52: (T) Nachbildung eines Zählsemaphor in Java

(– Pkt.)

Schreiben Sie eine Java Klasse `SimpleCountingSemaphore`, die es ermöglicht, Instanzen zu erzeugen, welche in Analogie zu der von Dijkstra vorgeschlagenen Datenstruktur eines Zählsemaphor verwendet werden können. Verwenden Sie dazu den Java `synchronized`-Mechanismus. Verwenden Sie außerdem eine *minimale* Anzahl an `wait()`- und `notify()`-Aufrufen!

Hinweis: Für die Lösung dieser Aufgabe reicht es aus, wenn Sie das auftreten einer etwaigen `InterruptedException` ohne weitere Fehlerbehandlung abfangen.

Des Weiteren sollen Sie sich in dieser Aufgabe die Unterschiede und Gemeinsamkeiten zwischen Java-Synchronisation und Monitoren klar machen.

a. Beschreiben Sie das grundlegende Konzept der Synchronisation in Java. Gehen Sie dabei auf die Verwendung von Objekt-Locks, Threads und Warteschlangen ein.

b. Wie werden `wait()` und `signal()` in Java umgesetzt?

c. Erläutern Sie den grundlegenden Unterschied des Java Synchronisationsmechnismus im Gegensatz zu „echten“ Monitoren (ohne Beachtung der Signalisierungsmechanismen).

d. Beschreiben Sie die Einschränkungen bei der Verwendung von `wait()` und `notify()` gegenüber „echten“ Monitoren.

   Hinweis: Überlegen Sie sich dazu, wie im Falle echter Monitore bzw. im Falle des Java Synchronisierungsmechanismus der nächste aktive Thread selektiert wird.

e. Überlegen Sie sich, wie diese Einschränkungen umgangen werden können.

f. Es gibt zwei Modelle, wie die Ausführung in einem Monitor nach dem Aufruf von `signal()` fortfährt (A: signalisierender Prozess, B: aufgeweckter Prozeß):

   - Signal-and-wait: A muß nach seiner Signalisierung den Monitor sofort freigeben und warten, bis B den Monitor verlassen hat oder auf eine andere Bedingung wartet.
   - Signal-and-continue: B muß warten, bis A den Monitor verlassen hat oder auf eine andere Bedingung wartet.

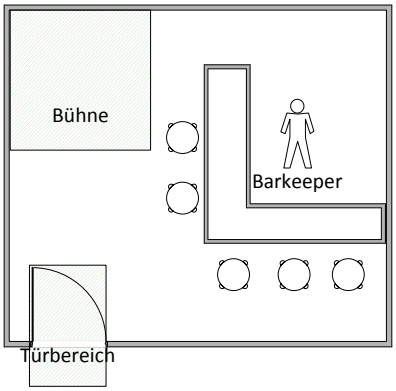
   Welchem Modell folgt Java?

Aufgabe 53: (H) Semaphore

(16 Pkt.)

In dieser Aufgabe sollen Sie das Konzept der Semaphore am Beispiel einer kleinen Studentenbar umsetzen. Dazu soll das Betreten und Verlassen der Bar simuliert werden, welche `5` Barhocker bereitstellt, die von den Gästen genutzt werden können. Die Gäste (welche hier als Prozesse angesehen werden können) betreten bzw. verlassen die Bar über eine Tür. Damit es nicht zu Auseinandersetzungen kommt, darf sich zu jedem Zeitpunkt nur eine Person im Türbereich aufhalten. Auch dieser Umstand soll von Ihnen modelliert werden. Es dürfen sich stets nur maximal so viele Gäste in der Bar befinden (inklusive einer Person im Türbereich), wie Barhocker vorhanden sind.
Nachdem ein Gast die Bar betreten und auf einen Hocker Platz genommen hat, versucht er drei Getränke zu sich zu nehmen. Dazu signalisiert er dem Barkeeper, der sich bereits hinter dem Tresen befindet, für jedes Getränk einzeln einen Bestellwunsch. Der Gast wartet, bis sein Getränk zubereitet wurde, was ihm vom Barkeeper signalisiert wird. Solange keine Bestellung aufgegeben wurde, ist der Barkeeper untätig. Er kann zu jedem Zeitpunkt nur eine Bestellung konzentriert abarbeiten. Deshalb muss sichergestellt werden, dass zu jeder Zeit nur ein Gast mit dem Barkeeper interagiert. Nach der Bearbeitung einer Bestellung soll der Barkeeper wieder allen Gästen zur Verfügung stehen. Nachdem ein Gast drei Getränke konsumiert hat, verlässt er die Bar durch den Türbereich. Da

alle Gäste den Barkeeper gut kennen und implizit anschreiben, kann der Bezahlvorgang außer Acht gelassen werden.
Die nachfolgende Abbildung zeigt den schematischen Aufbau.



Beantworten Sie nun auf Grundlage dieses Szenarios folgende Aufgaben:

a. Was sind die kritischen Bereiche bei diesem Problem?

b. Tragen Sie in eine Tabelle der folgenden Form die aus Ihrer Sicht benötigten Semaphoren ein, um den beschrieben Sachverhalt zu synchronisieren. Geben Sie zu jedem von Ihnen angedachten Semaphor einen Bezeichner, den Typ und eine kurze Beschreibung, wofür er verwendet werden soll, an. Die beiden Semaphoren `bestellwunsch` und `getränk_fertig` sind bereits gegeben.

Bezeichner	Typ	Zweck
`bestellwunsch`	Zählsemaphor	Signalisiert dem Barkeeper einen Bestellwunsch
`getränk_fertig`	Zählsemaphor	Signalisiert einem Gast die Fertigstellung seines Getränks
⋮		

c. Geben Sie in Pseudocode an, wie die benötigten Semaphore initialisiert werden müssen. Verwenden Sie dabei die Notation, die bei den gegebenen Semaphoren verwendet wurde:

Pseudocode	Bedeutung
`init(bestellwunsch, 0);`	Initialisiert den Semaphor `bestellwunsch` mit dem Wert 0.
`init(getränk_fertig, 0);`	Initialisiert den Semaphor `getränk_fertig` mit dem Wert 0.
⋮	

d. Vervollständigen Sie nun den Pseudocode für einen Gast (den Code-Rahmen dafür können Sie von der Website zur Vorlesung herunterladen), so dass auch mehrere Gäste stets synchronisiert werden. Dabei soll das Betreten bzw. Verlassen des Lokals durch die Gäste sowie deren Getränkebestellung simuliert werden. Beachten Sie, dass der Barkeeper immer nur einen Gast bedienen kann. Der Pseudocode des Barkeepers ist bereits gegeben (ebenfalls auf der Webseite zur Vorlesung).

Verwenden Sie für den Zugriff auf Ihre Semaphore folgende Notation:

Pseudocode	Beispiel	Bedeutung
`wait(<semaphor>);`	`wait(mutex);`	Erniedrigt den Wert des Semaphor `mutex` um eins
`signal(<semaphor>);`	`signal(mutex);`	Erhöhe den Wert des Semaphor `mutex` um eins
`<Aktion>`	`<Getränk entgegennehmen>`	Führe die gelistete Aktion aus
`//Kommentar`	`//Zubereitung abwarten`	Kommentarzeile

Aufgabe 54: (H) Einfachauswahlaufgabe: Prozesskoordination

(5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“). Nennen Sie dazu in Ihrer Abgabe explizit die jeweils ausgewählte Antwortnummer ((i), (ii), (iii) oder (iv)). Eine korrekte Antwort ergibt jeweils einen Punkt. Mehrfache Antworten oder eine falsche Antwort werden mit 0 Punkten bewertet.

<table>
<tr><td colspan="4">a) Wie bezeichnet man den Prozessormodus, dem der Prozessor bei der Ausführung von Anwendungsprogrammen zugeordnet ist und damit keine privilegierten Operationen ausgeführt werden können?</td></tr>
<tr><td>(i) Private Mode</td><td>(ii) Public Mode</td><td>(iii) User Mode</td><td>(iv) Kernel Mode</td></tr>
<tr><td colspan="4">b) Was entspricht der Zeitspanne, die ein Prozess im System verbringt, ohne dass er ausgeführt wird (nicht im Zustand „running“)?</td></tr>
<tr><td>(i) Wartezeit</td><td>(ii) Bedienzeit</td><td>(iii) Antwortzeit</td><td>(iv) Verweildauer</td></tr>
<tr><td colspan="4">c) Angenommen das Erzeuger/Verbraucher-Problem wurde mit Hilfe von Semaphoren gelöst. Dabei regelt ein binärer Semaphor <code>s</code> den wechselseitigen Zugriff auf den gemeinsamen Speicher, ein Zählsemaphor <code>b</code> repräsentiert die Anzahl der belegten Plätze und ein weiterer Zählsemaphor <code>p</code> gibt die Anzahl der freien Speicherplätze (wovon <code>MAX > 0</code> zur Verfügung stehen) an. Die Semaphoren werden wie folgt initialisiert: <code>init(s, 1); init(b, 0); init(p, MAX).</code>

Welche Aussage bezüglich des folgenden Quellcodes für den Erzeuger und den Verbraucher, welche parallel ausgeführt werden, ist korrekt?
<pre>(* Erzeuger: *)
REPEAT
  <erzeuge Element>;
  wait(s);
  wait(p);
  <Element in Speicher legen>;
  signal(s);
  signal(b);
UNTIL FALSE;</pre><pre>(* Verbraucher: *)
REPEAT
  wait(b);
  wait(s);
  <El. aus Speicher nehmen>;
  signal(s);
  signal(p);
  <verbrauche Element>;
UNTIL FALSE;</pre></td></tr>
<tr><td>(i) Die Lösung ist korrekt.</td><td>(ii) Der Verbraucher kann niemals ein Element aus dem Speicher nehmen.</td><td>(iii) Der Erzeuger kann niemals ein Element in den Speicher legen.</td><td>(iv) Es kann ein Deadlock entstehen.</td></tr>
<tr><td colspan="4">d) Was ist kein Bestandteil eines Monitors (Softwaremodul)?</td></tr>
<tr><td>(i) eine oder mehrere Prozeduren</td><td>(ii) eine Liste der Prozesse, die den Monitor verlassen haben</td><td>(iii) eine Warteschlange für ankommende Prozesse</td><td>(iv) lokale Daten</td></tr>
<tr><td colspan="4">e) Wie viele Prozesse dürfen sich zu jeder Zeit maximal in einem Monitor befinden?</td></tr>
<tr><td>(i) 3</td><td>(ii) 2</td><td>(iii) 1</td><td>(iv) 0</td></tr>
</table>