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

mobile and distributed systems group

Übungsblatt 10
Betriebssysteme im WiSe 21/22

Zum Modul K

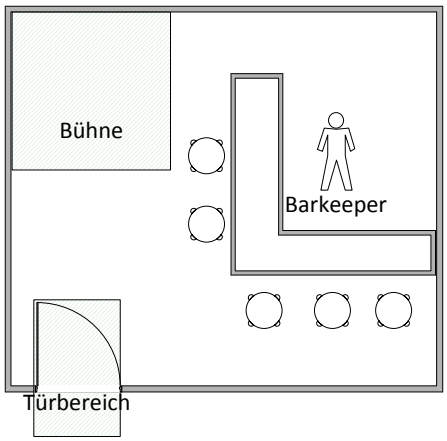
Abgabetermin: am 09.01.2021 bis 17:59 auf Uni2Work

Besprechung: vom 10. – 14. Januar 2022 in den Übungsgruppen

Aufgabe Ü25: Semaphore

(17 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 in Ihrer Modellierung 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.



Bearbeiten Sie nun auf Grundlage dieses Szenarios folgende Aufgaben:

a. Was sind die kritischen Bereiche bei diesem Problem?

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

Bezeichner	Zweck
`bestellwunsch`	Signalisiert dem Barkeeper einen Bestellwunsch
`getraenk_fertig`	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(getraenk_fertig, 0);`	Initialisiert den Semaphor `getraenk_fertig` mit dem Wert 0.

d. Vervollständigen Sie den folgenden Pseudocode für einen Gast, 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, was von Ihnen modelliert werden soll. Der Pseudocode des Barkeepers ist bereits gegeben.

```
1  Barkeeper() {
2      while(true) {
3          //auf Bestellung warten
4          wait(bestellwunsch);
5          //Fertigstellung signalisieren
6          signal(getraenk_fertig);
7          <Getraenk an den Gast geben>;
8      }
9  }
```

```
1   Gast() {
2
3
4
5       <die Bar betreten>;
6
7
8
9       <auf Hocker Platz nehmen>;
10
11
12
13      for(Getraenke = 0; Getraenke < 3; Getraenke++) {
14
15
16
17          //Bestellung aufgeben
18
19
20
21
22          //Zubereitung abwarten
23
24
25
26
27           <Getraenk entgegennehmen>;
28
29
30
31      }
32
33
34
35      <die Bar verlassen>;
36
37
38
39  }
```

Verwenden Sie für den Zugriff auf Ihre Semaphore die folgende Notation:

Pseudocode	Beispiel	Bedeutung
`wait(<semaphor>);`	`wait(mutex);`	Verringert den Wert des Semaphor `mutex` um eins
`signal(<semaphor>);`	`signal(mutex);`	Erhöht den Wert des Semaphor `mutex` um eins
`<Aktion>`	`<die Box verlassen>`	Führe die gelistete Aktion aus
`//Kommentar`	`//Fertigstellung signalisieren`	Kommentarzeile

Aufgabe Ü26: 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.

a) Was ist keine der 3 atomaren Operationen, mit denen ein Semaphor `S` verändert werden kann?

(i) `init(S, Anfangswert)`	(ii) `wait(S)` oder auch `P(S)`	(iii) `block(S)` oder auch `B(S)`	(iv) `signal(S)` oder auch `V(S)`

b) Für eine korrekte Lösung des wechselseitigen Ausschlusses müssen drei Bedingungen erfüllt sein. Was ist keine davon?

(i) Mutual Exclusion	(ii) Correlated Blocking	(iii) Progress	(iv) Bounded Waiting

c) Angenommen das Erzeuger/Verbraucher-Problem wurde mit Hilfe von Semaphoren gelöst. Dabei regelt ein binärer Semaphor `s` den wechselseitigen Zugriff auf den gemeinsamen Speicher, ein Zählsemaphor `b` repräsentiert die Anzahl der belegten Plätze und ein weiterer Zählsemaphor `p` gibt die Anzahl der freien Speicherplätze (wovon `MAX > 0` zur Verfügung stehen) an. Die Semaphoren werden wie folgt initialisiert: `init(s, 1); init(b, 0); init(p, MAX).`

Welche Aussage bezüglich des folgenden Quellcodes für den Erzeuger und den Verbraucher, welche parallel ausgeführt werden, ist korrekt?

```
(* Erzeuger: *)                          (* Verbraucher: *)
REPEAT                                   REPEAT
  <erzeuge Element>;                       wait(b);
  wait(s);                                 wait(s);
  wait(p);                                 <El. aus Speicher nehmen>;
  <Element in Speicher legen>;             signal(s);
  signal(s);                               signal(p);
  signal(b);                               <verbrauche Element>;
UNTIL FALSE;                             UNTIL FALSE;
```

(i) Es kann ein Deadlock entstehen.	(ii) Der Verbraucher kann niemals ein Element aus dem Speicher nehmen.	(iii) Der Erzeuger kann niemals ein Element in den Speicher legen.	(iv) Die Lösung ist korrekt.

d) Was entspricht der Zeitspanne, die ein Prozess im System verbringt, ohne dass er ausgeführt wird (nicht im Zustand „running“)?

(i) Wartezeit	(ii) Bedienzeit	(iii) Antwortzeit	(iv) Verweildauer

e) Welche Aussage bezüglich des klassischen Erzeuger/Verbraucher- Problems ist falsch? Dabei handelt es sich zum Beispiel um zwei Prozesse, die über eine gemeinsam genutzte Datenstruktur (gemeinsam genutzten Speicherbereich) Informationen austauschen (analog zu Aufgabenteil c)).

(i) Die Summe aus der Anzahl der verfügbaren Speicherplätze `p` und der Anzahl der belegten Speicherplätze (Bestand) `b` ergibt bei einer korrekten Implementierung des Erzeugers und Verbrauchers zu jedem Ausführungszeitpunkt den Wert `MAX`; (`p + b = MAX`)

(ii) Der Verbraucher kann nur so lange Objekte aus der Datenstruktur entnehmen, bis diese leer ist.

(iii) Erzeuger und Verbraucher dürfen nicht gleichzeitig auf die gemeinsam genutzt Datenstruktur (den gemeinsam genutzten Speicherbereich) zugreifen.

(iv) Der Erzeuger kann nur so viele Objekte in die Datenstruktur einfügen, wie der Speicherbereich Kapazität bietet.