Lehrstuhl für Connected Mobility
TUM School of Computation, Information and Technology
Technische Universität München

Grundlagen - Betriebssysteme und Systemsoftware

IN0009, WiSe 2023/24

Übungsblatt 5

20. November 2023 – 24. November 2023

Hinweis: Mit * gekennzeichnete Teilaufgaben sind ohne Lösung vorhergehender Teilaufgaben lösbar.

Aufgabe 1 Vorbereitung

Vor der Übung sollten Sie...

- die Funktionsweise von Semaphoren und Mutex verstanden haben.
- die Problematik des Producer-Consumer-Problems verinnerlicht haben.

Aufgabe 2 Netzwerkkarte

Wir betrachten das folgende, stark vereinfachte Beispiel für ein Erzeuger-Verbraucher-Problem: Eine Datei soll über ein Netzwerk auf einen Computer transferiert werden. Die Netzwerkkarte `N` des Computers empfängt blockweise Datenpakete und legt diese im Buffer `B` (Kapazität: `n`) ab, von wo aus sie nach und nach entnommen und auf die Festplatte `F` gespeichert werden. Um wechselseitigen Ausschluss zu erreichen, sei folgender Lösungsversuch mit dem Mutex `wa` als Pseudocode gegeben:

```
1   Deklaration:
2   wa(1);
3
4   Netzwerkkarte N:
5   while(true) {
6     <empfange Datenblock>;
7     down(wa);
8     <schreibe Datenblock in B, falls Platz frei, sonst warte>;
9     up(wa);
10  }
11
12  Festplatte F:
13  while(true) {
14    down(wa);
15    <entnimm Datenblock aus B, falls vorhanden, sonst warte>;
16    up(wa);
17    <schreibe Datenblock auf Festplatte>;
18  }
```

Hinweis: Die Pseudocode-Operation zum Deklarieren eines Semaphors mit Namen `name` mit Startwert `n` lautet `name(n);` Das Semaphor unterstützt die Operationen `up(name)` und `down(name)`.

a) Laufen beide Prozesse verklemmungsfrei? Welche Situationen führen zu Verklemmungen?

b) Geben Sie eine verbesserte Version an, in der keine Probleme mehr auftreten, indem Sie zwei Semaphore geeignet deklarieren und geeignete Aufrufe von `down` und `up` einfügen.

c) Welche Probleme treten auf, wenn Sie in Ihrer verbesserten Lösung die Reihenfolge der `down`-Operationen für `wa` und Ihrer beiden zusätzlichen Semaphore vertauschen?

Noch Fragen? Dann schaut doch mal im Zulip-Stream vorbei:
`https://zulip.in.tum.de`

Aufgabe 3 U-Bahn Chaos

Wir betrachten die Strecke der U6 zwischen Garching-Forschungszentrum (**GF**) und Fröttmaning (**F**). Da zur Zeit gebaut wird, herrscht zwischen Garching-Hochbrück (**GH**) und F eingleisiger Betrieb. Im Folgenden modellieren wir die Synchronisation des Streckenabschnitts GF $\Longleftrightarrow$ F. Gegeben ist: im Bahnhof GF haben nur zwei Züge Platz, die Kapazität des Bahnhofs F ist unbegrenzt. Ergänzen Sie im Folgenden den unten stehenden Code.

a)* Fügen Sie einen Mutex hinzu, sodass es auf dem eingleisigen Abschnitt zu **keiner Kollision** kommen kann. Ist aktuell ein Zug im eingleisigen Abschnitt, so muss der nächste im letzten Bahnhof vor der Baustelle warten.

b)* Führen Sie mittels Semaphoren Zähler ein, die dafür sorgen, dass in den Bahnhöfen GF und F **jeweils niemals weniger als null** Züge sind. Sorgen Sie dafür, dass in GF **niemals mehr als zwei** Züge sind. Sind in GF bereits zwei Züge, so darf in F kein weiterer Richtung GF ausfahren. **Am Anfang seien in GF ein Zug, in F drei**.

c)* Verhindern Sie, dass auf dem Streckenabschnitt GF $\Longleftrightarrow$ GH in beiden Richtungen zusammen mehr als **zwei** Züge unterwegs sind.

```
// Deklarationen




// ---
```

```
// Thread
Fahre_in_richtung_F
{


  <Fahre aus GF aus>


  <Fahre in GH ein>


  <Fahre aus GH aus>


  <Fahre durch eingleisigen Abschnitt>


  <Fahre in F ein>


}
```

```
// Thread
Fahre_in_richtung_GF
{


  <Fahre aus F aus>


  <Fahre durch eingleisigen Abschnitt>


  <Fahre in GH ein>


  <Fahre aus GH aus>


  <Fahre in GF ein>


}
```

Noch Fragen? Dann schaut doch mal im Zulip-Stream vorbei:
https://zulip.in.tum.de