ALP4 SoSe 2013, Di. 16-18

Lösung Übungsblatt 6

Christoph van Heteren-Frese (Matr.-Nr.: 4465677)
Sven Wildermann (Matr.-Nr.: 4567553)
Tutor: Alexander Steen, eingereicht am 30. Mai 2013

Aufgabe 1

TODO CHRIS

Aufgabe 2

TODO CHRIS

Aufgabe 3

Der Effekt des asynchronen Nachrichtenaustauchs lässt sich mit synchronem Austausch mit Hilfe eines Puffers erzeugen. So kann dann mindestens einer der Prozesse eine nicht blockierende Sende-oder Empfangsoperations nutzen. Der Pufferspeicher blockiert nur dann, wenn dieser voll ist. Der Empfänger kann daher jederzeit bereit für eingehende Verbindungen sein, da er diese erst abarbeitet, wenn es für diesen günstig ist. Der Sender hingegen kann damit unabhängig vom Empfänger seine Nachrichten verschicken, da er davon ausgehen kann, dass diese im Puffer des Empfängers landen. Statt der direkten Verbindung: "Sender - Empfänger" gibt es jetzt genau genommen die Verbindung "Sender - Puffer - Empfänger".
Dies funktioniert natürlich auch umgekehrt. Will der Sender zu einem Zeitpunkt mehrere Nachrichten verschicken, kann er diese in seinen eigenen Puffer legen und den Sendevorgang als "abgeschlossen" kennzeichen. Der Puffer versendet dann letztlich (aus technischer Schicht) die Nachrichten.

Aufgabe 4

1. direkte Übersetzung des Codes nach go

```
1  package channel
2
3  import "fmt"
4  import "sync"
5
6  type Kanal struct {
7    botschaft byte
8    s,e,mutex Mutex
9    }
10
```

```
11 func send(x Kanal, c byte){
12   x.mutex.wait()
13   x.botschaft = c
14   x.s.signal()
15   x.e.wait()
16 }
17
18 func init(x Kanal){
19   x.s = 0
20   x.e = 0
21   x.mutex = 1
22 }
23
24 func recv(x Kanal, c byte){
25   x.s.wait()
26   c=x.botschaft
27   x.e.signal()
28   s.mutex.signal()
29 }
```

2. Ablaufreihenfolge angeben, die Verklemmung erzeugt