Fachgebiet Verteilte Systeme
Prof. Dr. Kurt Geihs
FB16 – Elektrotechnik/Informatik
Universität Kassel

VERTEILTE SYSTEME

Betriebssysteme WS 20/21
Übung: Stefan Jakob
jakob@vs.uni-kassel.de
http://www.vs.uni-kassel.de

Übungsblatt 7

Punktzahl gesamt: 50 Punkte

Abgabe der Lösungen bis spätestens 15.01.21, 10:00 Uhr

Aufgabe 1 – Semaphore

16 Punkte

Unten ist die Anordnung von Semaphor-Operationen am Anfang und am Ende von drei Prozessen (A, B, C) dargestellt. Ermitteln Sie für die angegebenen Initialisierungen der Semaphore X, Y, Z, in der Tabelle unten, ob und in welcher/welchen Reihenfolge(n) die Prozesse in einen Deadlock kommen. Betrachten Sie Abläufe bis zu einer Maximaltiefe von 4 (der Deadlock kann im fünften Schritt erfolgen). Wird ein Prozess nur teilweise ausgeführt, schreiben Sie es in geschweiften Klammern (Beispiel in den Übungsfolien). Bitte geben Sie pro Teilaufgabe maximal vier Möglichkeiten an. Falls es zu keinen Deadlocks kommen kann, schreiben Sie bitte in das jeweilige Feld *deadlockfrei* hinein.

Prozess A

```
for(;;) {
  P(Z)
  P(X)
  ...

  V(Y)
}
```

Prozess B

```
for(;;) {
  P(Y)

  ...

  V(X)
  V(X)
}
```

Prozess C

```
for(;;) {
  P(X)
  P(X)
  ...

  V(Z)
}
```

	X	***Y***	***Z***	**Ablauf / Abläufe**
a)	1	0	4	
b)	0	1	0	
c)	3	0	0	
d)	0	1	1	

Aufgabe 2 – Erzeuger-Verbraucher in C++ 34 Punkte

Zwei Konditoren backen mittels der Funktion *backen()* Kuchen. Konditor 1 fertigt alle 2 Sekunden einen Kuchen, Konditor 2 alle 3 Sekunden. Beide Konditoren legen ihre Kuchen in die gleiche Auslage, welche maximal 5 Kuchen umfasst. Ein Verbraucher konsumiert mittels *konsumieren()* die Kuchen sobald sie in der Auslage stehen. Zum Konsumieren benötigt er eine Sekunde Zeit.

a) Benutzen Sie für die erste Variante der Implementierung die Vorlage *ErzeugerVerbraucherMutex*. Initialisieren Sie zunächst beide **Semaphore** mit den entsprechenden Werten. Anschließend sollen Sie die *konsumieren* Methode der Verbraucher Klasse und die *einfuegen* Methode der Konditor Klasse wie oben beschrieben, implementieren. **13 Punkte**

b) Implementieren Sie nun das Szenario mithilfe von std::unique_lock<std::mutex> und std::condition_variable. **13 Punkte**

c) Vergleichen Sie std::mutex und std::unique_lock. **4 Punkte**

d) Vergleichen Sie die Verwendung von std::condition_variable mit dem Konzept von BusyWaiting. **4 Punkte**