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	{C} {A} DeadLock A D C {A} DeadLock {B} {C} {A} DeadLock {A} {C} DeadLock
b)	0	1	0	D C {A} DeadLock
c)	3	0	0	C A B C {A} DeadLock C {A} {C} DeadLock C {C} {A} DeadLock
d)	0	1	1	D A {C} DeadLock {A} D C DeadLock B C {A} DeadLock {A} B {A} C DeadLock

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**

c) std::mutex ist eine Datenstruktur, mit der sichergestellt werden kann, dass parallele Threads nicht gleichzeitig auf die gleichen Daten zugreifen. mutex unterstützt kein recursive locking.

std::unique_lock ist ein Wrapper für std::mutex. Das heißt es verwaltet einen mutex, der über den Konstruktor übergeben wird. Außerdem werden die Funktionen von mutex erweitert, bspw. um das oben genannte recursive locking. Ein weiterer Vorteil von unique_lock ist, dass der mutex beim Aufruf des Konstruktors unlocked wird, also bspw. wenn der Scope von unique_lock verlassen wird oder ein Fehler im Programm bzw. einem Thread auftritt. Somit lassen sich Deadlocks durch ausfallende Threads verhindern.

d) siehe .txt.