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

mobile and distributed systems group

Tutoriumsblatt 3
Betriebssysteme im WiSe 2020/2021

Zum Modul D

Tutorium: Die Aufgaben werden in einem Tutorien-Video vorgestellt, das am 18. November 2020 (17 Uhr) veröffentlicht wird.

Aufgabe T6: User-Level-, Kernel-Level-Threads und Prozesse

(– Pkt.)

Betrachten Sie nun folgendes Fallbeispiel: Gegeben ist eine Anwendung, die bisher single-threaded abläuft, also nur aus einem einzigen Thread besteht. Diese Anwendung bietet eine Eingabe-Schnittstelle, über die der Benutzer mathematische Ausdrücke auswerten lassen kann. Die Anwendung soll nun so modifiziert werden, dass der Benutzer während einer laufenden Auswertung eines Ausdrucks weitere Berechnungen starten oder auch Zwischenergebnisse einer der aktiven Auswertung abfragen kann.

Im Folgenden wird angenommen, dass ein Benutzer die Berechnung der ersten hunderttausend Primzahlen anfordert.

a. Warum muss man bei diesem Szenario unbedingt einen neuen Thread oder Prozess für die Berechnung starten?

b. Es werden folgende Alternativen erwogen:

   - Berechnung der Primzahlen in einem neuen User-Level-Thread starten.
   - Berechnung der Primzahlen in einem neuen Kernel-Level-Thread starten.
   - Berechnung der Primzahlen in einem neuen Prozess starten.

   Bewerten Sie jede der drei Möglichkeiten hinsichtlich der folgenden Gesichtspunkte. Verwenden Sie dazu eine geeignete tabellarische Darstellung.

   (i) Aufwand für die Generierung des neuen Threads/Prozesses

   (ii) Kommunikation/Datenaustausch zwischen der bisherigen Anwendung und dem neuen Thread/Prozess

   (iii) Abwicklung des Schedulings (Wer ist für das Scheduling des neuen Threads/Prozesses verantwortlich?)

   (iv) Ausführung auf einer Multiprozessorumgebung

c. Basierend auf Ihrer Bewertung in der Teilaufgabe b), welche Möglichkeit würden Sie hier wählen? Begründen Sie Ihre Entscheidung.

Aufgabe T7: Grundlagen von Threads

(– Pkt.)

a. Nennen Sie zwei Gründe, warum es nicht sinnvoll ist, zu viele Threads zu verwenden.

b. Nennen Sie zwei Gründe, warum Threads sinnvoll/wichtig sind.

Aufgabe T8: Java: Koordination von Threads

(– Pkt.)

In dieser Aufgabe sollen Sie eine Lösung implementieren, die es ermöglicht, Züge koordiniert über einen **eingleisigen** Streckenabschnitt (AB) fahren zu lassen. Der Streckenabschnitt AB unterliegt folgenden Einschränkungen:

– Der Streckenabschnitt AB verfügt über genau ein Gleis, d.h. es kann gleichzeitig nur in genau eine Richtung gefahren werden (entweder West oder Ost).

– Es können sich maximal drei Züge gleichzeitig auf dem Streckenabschnitt befinden.

– Jeder Zug verlässt den Streckenabschnitt nach endlicher Zeit.

Die Klassen `TrainNet` und `Train` sind bereits gegeben. Sie können sich den Quelltext von der Website zur Vorlesung herunterladen.

Die Klasse `TrainNet` erzeugt einen Streckenabschnitt AB (Instanz der Klasse `RailAB` und startet die Züge (Instanzen der Klasse `Train`).

Die Klasse `Train` repräsentiert Züge und ist als Thread implementiert. Innerhalb der `run()`-Methode werden auf die Instanz der Klasse `RailAB` die Methoden `goEast()` und `goWest()` aufgerufen. Diese dienen dazu, einen Zug auf den Streckenabschnitt von West nach Ost bzw. von Ost nach West zu schicken. Zudem wird die Methode `leaveAB()` aufgerufen, durch deren Aufruf ein Zug den Streckenabschnitt AB wieder verlässt.

Implementieren Sie nun die Klasse `RailAB` unter Berücksichtigung der oben genannten Einschränkungen. Die Lösung muss frei von Deadlocks sein und darf Züge nicht unnötig blockieren. Implementieren Sie

a. einen passenden Konstruktor (siehe Klasse `TrainNet`),

b. die Methode `goWest()`, welche die Züge, die nach Westen fahren, koordiniert,

c. die Methode `goEast()`, welche die Züge, die nach Osten fahren, koordiniert, und

d. die Methode `leaveAB()`, welche von den Zügen aufgerufen wird, die den Streckenabschnitt wieder verlassen.