Verteilte Systeme (I7, Bc)

Blatt 01a: Arbeitsumgebung Linux, Threads

Wintersemester 2024/25

Bearbeitung im Praktikum ab 10.10.2024

Entwicklungsplattform und weitere Randbedingungen

- Die Referenzplattform für das Praktikum ist **Linux Ubuntu 22.04**.
- Prinzipiell können Sie die Übungsaufgaben aller Blätter auf Ihrem eigenen Linux-System (eigenes Notebook oder VM) lösen.
  - Je nach Übungsblatt müssen Sie dann mehr oder weniger Software selbst installieren.
- Prinzipiell können Sie für die Lösung der Übungsaufgaben auch mit anderen Plattformen arbeiten. Es wird aber empfohlen, mit Ubunut 22.04 zu arbeiten, weil dies die Plattform ist, die für die Befragung verwendet wird (eine gemeinsame VM für alle Studierende).
- Für die Übungsaufgaben von Blatt 1 können Sie auch einen Arbeitsplatzrechner im Labor benutzen (falls vorhanden).
  - Starten des Linux Systems: Beim Booten des Arbeitsplatzrechners eine Desktop-Version auswählen.
  - Einloggen mit Standard-Login, d.h. `benutzername@hs-coburg.de`
- Für die Übungsaufgaben des Blatt 1 benötigen Sie als Software i.W. eine Java-Entwicklungsplattform (JDK).
- Nutzung einer Entwicklungsumgebung (IDE) wie z. B. NetBeans oder Eclipse: Sie können im Praktikum eine Entwicklungsumgebung verwenden.
  Aber: Bei der Abgabe der Lösungen ist zu differenzieren zwischen den Aufgabenblättern. Bei manchen können Sie ein NetBeans-Projekt abgeben, bei anderen nur Quellcode und eine Bauanleitung, die ohne IDE funktioniert. Eclipse-Projekte werden generell **nicht** akzeptiert.
- Bei der Befragung:
  - Das Starten von Programmen muss stets ohne IDE erfolgen.
  - Je nach Aufgabenblatt ist Nutzung NetBeans erlaubt, Nutzung von Eclipse ist nicht erlaubt,
- Alternative zur IDE: Pseudoterminal, javac zum Compilieren und zum Editieren 'Vim' bzw. 'gvim' oder 'Geany' oder 'Nano' und ggf. Shellskripte.

Hinweise

- Sie erhalten rechtzeitig eine eigene Labor-VM mit der Plattform, die für die Befragung genutzt wird. Diese Labor-VM wird auf einem Server der Hochschule gehostet. Mit dieser Labor-VM können Sie dann Ihre Lösungen testen oder auch erstellen.
- Wenn Sie in Ihrer eigenen Labor-VM weitere Software installieren wollen, können Sie das tun, da Sie für Ihre eigene Labor-VM Root-Rechte erhalten. Beachten Sie aber: diese Zusatzsoftware wird Ihnen bei der Befragung **nicht** zu Verfügung stehen.
- Falls Ihre Lösung auf Ihrer eigenen Plattform läuft, bedeutet das nicht, dass Ihre Lösung auf der Plattform für die Befragung läuft. Deshalb sollten Sie Ihre Lösung auf der Befragungsplattform testen, d.h auf der Labor-VM, die Sie von uns erhalten.

- Typischer Fehler beim Portieren von Lösungen von einer Linux-Maschnine auf eine andere (z.B. bei Übernahme einer Lösung von zuhause auf die Labor-VM):
  - Bei der Portierung gehen auf der Dateiebene die Ausführungsrechte verloren. Problemlösung: mit `chmod +x dateiname` Ausführungsrechte einrichten.
- Beachten Sie auch, ob es zu den Aufgaben Zulieferungen im Moodle-Kurs gibt.
- Die Aufgabenstellungen haben teilweise Freiheitsgrade. Es ist Ihre Aufgabe, diese Freiheitsgrade sinnvoll auszufüllen.

Vorgaben

- Bei Blatt 01 sind **keine** IDE-Projekte für die Abgabe erlaubt.
- Alle Klassen von Lösungen sollen bei Blatt 01 im Paket '`threadslernen`' liegen.
- Die Lösungsdateien müssen den Hauptaufgaben eindeutig zuordenbar sein. Deshalb ist folgendermaßen vorzugehen:
  - Die Dateien liegen in einem Unterpaket von '`threadslernen`', welches die Hauptaufgabennummer im Namen des Unterpakets entält, z.B. für Aufgabe 2 lautet der Name des Unterpakets '`aufgabe2`'. D.h. eine Klasse mit Namen '`SumComputation`' liegt dann im Paket '`threadslernen.aufgabe2`'. und die Datei zur Klasse heißt '`SumComputation.java`'. Eine Unterscheidung in Unteraufgaben, z. B. Aufgabe 2.1 und 2.2 wird nicht mehr vorgenommen.

Aufgabe 1: Java-Programmierung

1.1 "Hello World"-Programm

Schreiben Sie in der Programmiersprache Java ein "Hello World"-Programm, welches auch alle seine Aufrufparameter ausgibt. Die Startklasse des Programms soll `HelloWorld` heißen.

Beachten Sie: Die Startklasse soll im Paket '`threadslernen`' liegen.

Übersetzen Sie dieses Programm und führen Sie es mit mehreren Aufrufparametern aus.

Benutzen Sie für diese Aufgabe keine IDE!

Lernziele

- Entwicklungsplattform kennen lernen
- Kenntnisse in Java-Programmierung auffrischen

Aufgabe 2: Gesamtsummenbildung

Gegeben sei der folgende Bauplan für ein Java-Programm, welches mit Hilfe von Threads die Elemente zweier Zahlenarrays aufaddiert:

```
package threadslernen;
import  java.util.Date;
public class  SumComputation {
  final static int SIZE = 100000;
  private int sum = 0;
  private int[] sa1 = new int[SIZE];
  private int[] sa2 = new int[SIZE];
  private long starttime;
  private long endtime;
  public static void main(String[] args) {
     SumComputation prog = new SumComputation();
     prog.execute(args);
    }

    // Methode zum Zugriff auf sum
    void addToSum(int i) {
        sum = sum + i;
    }

    private void execute(String[] args) {
     System.out.println("Starte Programm SumComputation");

        // Initialisiere bei sa1 alle Elemente mit 1, bei sa2 mit 2
       ...
      Thread worker1 = ... // Erzeuge Worker-Thread 1
      Thread worker2 = ... // Erzeuge worker-Thread 2
      ...  // Starte Worker-Thread 1
      ...  // Starte Worker-thread 2
      // Warte auf das Ende von Worker-Thread 1
      // Warte auf das Ende von Worker-Thread 2
      ...
      System.out.println("Programm: sum = " + sum);
      System.out.println("Beende Programm SumComputation");
    } // execute


    // Implementiere eine private Klasse Worker für die Worker-Threads
    private class ...
      // Hauptmethode des Worker-Threads implementieren
       System.out.println("Starte worker " + id + " der Klasse " +
         getClass().getName());
```

```
          // Thread soll 1 Sekunde schlafen
          ...
          // Thread addiert die Zahlen seines Arrays einzeln zur Gesamtsumme mit
Hilfe von addToSum()
          ...
        // vor dem Ende des Worker-Threads die Ergebnisausgabe
        System.out.println("Worker " + id + ": sum = " + sum);
        System.out.println("Beende worker "+id+" der Klasse "+
          getClass().getName());


} // class
```

2.1 Implementieren

Ergänzen den Bauplan zu einem korrekten Programm bzw. modifizieren Sie gegebenenfalls dieses Programm, so dass es ohne Warnungen kompilierbar und ohne Absturz ablauffähig ist.

2.2 Korrekte Synchronisation

Diese Teilaufgabe kommt erst auf dem Blatt 01b.

2.3 Zeitmessung

Ergänzen Sie Ihre Lösung um eine Zeitmessung. Messsen Sie die Laufzeit des Programms auf Millisekundengenauigkeit und zwar einmal mit und einmal ohne Synchronisation.

Hinweise

- Threads kann man warten lassen mit der Methode `sleep(millisekunden)`.
- Verwenden Sie zur Zeitmessung die Klasse java.util.Date.
- Es reicht eine Gesamtaufgabenlösung zu dieser Aufgabe abzugeben, aus der die anderen Lösungen der Teilaufgaben durch Hinzufügen oder Wegnehmen von Kommentaren erstellt werden können.
- Bei der Programmausführung kann notwendig werden, dass man die Optimierungen des JIT-Compilers ausschaltet, um Effekte wie Race-Conditions zu sehen. Durch die Option `-Djava.compiler=NONE` schaltet man den JIT-Compiler aus. Konkret also Ihr Programm starten mit `java -Djava.compiler=NONE <Startklasse>`

Lernziele

- Threads erzeugen können
- Zugriff auf gemeinsame Ressource synchronisieren können