Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Prof. Dr. W.-P. de Roever
Jens Schönborn, Christian Motika und Tim Fenten

Nebenläufiges Programmieren

Sommersemester 2008 16.06.2008

Serie 11
Endsemestertest 1.Teil

Ausgabetermin: 16.06.2008

Abgabe: 27.06.2008 (11:00)

Diese Serie soll in Einzelarbeit gelöst werden!

Aufgabe 1 (8 Punkte) [*One-Lane Bridge / Lindaunis*] Es soll die einspurige Lindaunis-Brücke modelliert werden. Siehe auch *http://de.wikipedia.org/wiki/Lindaunisbrücke*. Autos überqueren die Brücke von Norden nach Süden sowie in umgekehrter Richtung. Autos, die in der gleichen Richtung unterwegs sind, dürfen gemeinsam auf die Brücke fahren. Autos in die andere Richtung dürfen dann nicht auf die Brücke.

Zusätzlich muss gelegentlich ein Zug über die Brücke, in diesem Fall darf also kein anderes Fahrzeug auf die Brücke. Es passt immer nur ein Zug zur Zeit auf die Brücke.

Von Zeit zu Zeit wird die Brücke hochgeklappt, um wartende Segelboote passieren zu lassen. Ist die Brücke hochgeklappt, so dürfen natürlich keine Züge und Autos auf die Brücke. Es ist aber möglich, dass mehrere Boote gleichzeitig die Brücke passieren. Unterschiedliche Fahrtrichtungen brauchen hierbei nicht modelliert zu werden.

Entwickle genau einen Server-Prozess, der den Verkehr über die Brücke unter Berücksichtigung der einzelnen Bedingungen des *Critical Section Problems* (mindestens Mutual Exclusion, No Unnecessary Delay, Eventual Entry) regelt. Nimm an, dass die Autos, Züge und Boote Client Prozesse sind. Benutze asynchrones Message passing für die Prozessinteraktion. Schreibe eine Simulation deiner Lösung in MPD. Die Verkehrsteilnehmer sollen immer wieder versuchen die Brücke zu überqueren. Jeder Teilnehmer soll eine zufällige Zeit lang auf der Brücke verweilen, und nach dem Verlassen eine zufällige Zeit lang warten, bis er erneut versucht, die Brücke zu passieren. Teilnehmer, die gleichzeitig auf der Brücke dürfen sich gegenseitig überholen.

Erkläre und kommentiere deinen Code gründlich auch in Bezug auf die einzelnen Bedingungen des *Critical Section Problems* (mindestens Mutual Exclusion, No Unnecessary Delay, Eventual Entry).

Tips:

1. Channels werden in MPD anders als in dem Buch mit dem Keyword op deklariert. Schaut euch bei Problemen auch das Tutorial an.

2. Falls euch die zündende Idee fehlt, schaut euch Readers/Writers (und den Barbershop) nochmal an.

3. Queues dürfen wie folgt mit Channels implementiert werden.

Listing 1: FIFO queue mittels channels implementiert

```
resource fifoqueue()
  op fifo (int id); # MPD Deklaration eines Channels
  # die im Buch angegebene Operation empty(ch) auf Channels gab's nicht
  # und da receive blocking ist wird
  # die Anzahl der in der Queue enthaltenen Elemente gespeichert
  int nfifo = 0;

  procedure dequeue() returns int id{
    if (nfifo > 0) {
      nfifo = nfifo - 1;
      receive fifo(id);
    } else {
      id=-1;
    }
  }

  procedure enqueue(int id){
    nfifo = nfifo + 1;
    send fifo(id);
  }

  process enqueuer{
    while(true){
      nap(int (random(100,500)));
      int element = int (random(0,100));
      printf("Element %i enqueued\n",element)
      enqueue(element);
    }
  }

  process dequeuer{
    while(true){
      nap(int (random(100,500)));
      int element = dequeue();
      printf("                          Element %i dequeued\n",element)
    }
  }
end
```