Christian-Albrechts-Universität zu Kiel
Institut für Informatik und Praktische Mathematik
Prof. Dr. W.-P. de Roever
Dipl. Inf. K. Baukus · Dr. M. Steffen

Informatik III

Wintersemester 2000/01

Handout 1

6. November 2000

Handout 1: Gegenseitiger Ausschluß: Dekker

Ausgabetermin: 6. November 2000

Übungsettel 1 bestand aus einer Reihe von Versuchen, gegenseitigen Ausschluß mittels *Software* zu garantieren.[1] Die angestrebten Lösungen waren alle in der einen oder anderen Form unbefriedigend (zu starr und unflexibel, keine Garantie des gegenseitigen Ausschlusses, Verklemmungsgefahr). Eine naheliegende Frage also ist, ob und wie die Beispiele erweitert werden können, sodaß man von einer zufriedenstellenden Lösung des Problems sprechen kann.

Eine Lösung des Problems (für zwei Prozesse) stammt von *Dekker*. Die Aufgaben des ersten Zettel können als Vorstufen zu Dekkers Algorithmus gesehen werden. Die Variable `turn` der ersten Aufgabe, die zu einem starren Wechsel der beiden Prozesse führte, wird (unter dem Namen `favored` in dem unteren Programm) dazu verwendet, nicht denjenigen Prozeß zu bestimmen, der als Einziger als nächstes eintreten darf *und muß,* sondern denjenigen, der *Vortritt* hat, falls beide Partner Zutritt zum kritischen Bereich wollen. Wie in Aufgabe 3 des ersten Zettel, verwendet man zwei Variable (`P0_wants_to_enter` und `P1_wants_to_enter`) um den Betretenswunsch pro Prozeß anzuzeigen. Die Gefahr aus Aufgabe 3, daß beide Prozesse sich *verklemmen*, indem sie "gleichzeitig" den Betretenswunsch äußern, wird entschärft, indem die Variable `favored` in diesem Fall die Pattsituation bricht. Der Code für Prozeß `P0` ist auf der folgenden Seite angegeben.

[1] Wie man es ansonsten bewerkstelligt, insbesondere, welche hardwaregestützten Möglichkeiten verwendet werden oder wie Semaphore helfen, wird im weiteren Verlauf der Vorlesung verdeutlicht.

```
//  Dekkers Algorithmus f"ur Gegenseitigen Ausschlu"s
//  boolean  P0_wants_to_enter, P1_wants_to_enter = false;

  process P0 =
  begin

    while true                              /* Endlosschleife      */
    do
      some_actions;                         /* unkritischer Code   */

      P0_wants_to_enter := true;
      while  P1_wants_to_enter
      do
        if    favored = 1
        then  P0_wants_to_enter := false;
              while favored = 1 do skip od;
              P0_wants_to_enter := true
        fi
      od;
      critical_section_0;
      favored := 1;
      P0_wants_to_enter := false;
    od;
  end;
```