Christian-Albrechts-Universität zu Kiel
Institut für Informatik und Praktische Mathematik
Prof. Dr. W.-P. de Roever
Harald Fecher

Sys. Inf. III (Betriebssysteme)

Wintersemester 2004/05

Serie 4

8. November 2004

Thema: Gegenseitiger Ausschluss

Ausgabetermin: 8. November 2004

Abgabe: 15. November 2004 (12:00)

Aufgabe 1 (Produzenten/Konsumenten-Problem mit binären Semaphoren (5 Punkte))
Das Problem der Produzenten-Konsumenten-Koordination über einen beschränkten Puffer soll ein weiteres Mal angegangen werden. Diesmal soll nur eine eingeschränkte Form von Semaphoren zur Verfügung stehen, nämlich sogenannte *binäre* Semaphore. Diese sind gegenüber der allgemeinen Form eingeschränkt, indem sie nur *zwei Werte* annehmen können: 0 und 1.[1] Geben Sie eine Lösung für das Problem mit beschränktem Puffer an, indem Sie folgendes Codefragment vervollständigen! Erläutern Sie die Korrekheit Ihrer Lösung!

```
1  /* producer/consumer Pseudocode-Fragment                          */
2
3  int Buf[N];
4
5  producer ()
6  {
7    int datum, in;
8
9    while (TRUE) {
10
11     datum    = produce();          /* Produzent tut seine Aufgabe      */
12     Buf[in] = datum;                    /* Einfuegen                        */
13     in       = (in + 1) mod N;          /* und modulo weiterz"ahlen         */
14
15   }
16 }
17
18 consumer ()
19 {
```

[1] Läßt man negative Werte zu, z.B., wenn man, der Konvention in Xinu folgend, (mehrere) an der Semaphore wartende Prozesse mit einem negativen Zählerstand vermerkt, müßte man präziser sagen, binäre Semaphore lassen zwei *nicht-negative* Werte zu. Binäre Semaphore können als Spezialfall der "zählenden" Semaphore, wie sie unter anderem in Xinu realisiert sind, angesehen werden. Der Spezialfall besteht darin, daß die Semaphore nur angeben kann, ob eine Resource vorhanden ist (Wert 1) oder nicht (Wert 0), und nicht die Anzahl der freien Resourcen zählen kann.

```
20    int datum, out;
21
22    while (TRUE) {
23
24      datum =  Buf[out];                /* Datum auslesen                */
25      out   =  (out + 1) mod N;         /* weiterz"ahlen                 */
26      consume(datum);
27
28    }
29  }
```

Aufgabe 2 (Verification von Gegenseitigem Ausschluss (5 Punkte)) Beweisen sie für den folgenden Algorithmus (präsentiert in der Vorlesung) gegenseitigen Ausschluss mittels dem invarianten Ansatz.

```
1   C1, C2: Integer range 0..1:=1;
2
3   task body P1 is
4   begin
5     loop
6       Non_Critical_Section_1;
7       C1:=0;
8       loop
9         exit when C2=1;
10        C1:=1;
11        C1:=0;
12      end loop;
13      Critical_Section_1;
14      C1:=1;
15    end loop;
16  end P1;
17
18  task body P2 is
19  begin
20    loop
21      Non_Critical_Section_2;
22      C2:=0;
23      loop
24        exit when C1=1;
25        C2:=1;
26        C2:=0;
27      end loop;
28      Critical_Section_2;
29      C2:=1;
30    end loop;
31  end P1;
```
