TECHNISCHE UNIVERSITÄT
CHEMNITZ

Professur
Betriebssysteme

6. Übungsblatt

Grundlagen der Anwendungsunterstützung

Christine Jakobs und Laura Morgenstern

Aufgabe 1

Jede der unten dargestellten Funktionen wird durch einen Thread abgearbeitet. Nehmen Sie an den markierten Stellen entsprechende Ergänzungen in Form von Signalen vor, die bewirken, dass die Zahlen von 1 bis 100 in jedem Fall in aufsteigender Ordnung ausgegeben werden. Welche globalen Variablen müssen eingeführt werden? Welche Initialisierungen sind in der `main()`-Funktion vorzunehmen?

```
void thread_ungerade()
{
  unsigned int i;

  for(i = 1; i <= 100; i += 2)
  {
    // ???
    printf("%d\n", i);
    // ???
  }
}
```

```
void thread_gerade()
{
  unsigned int i;

  for(i = 2; i <= 100; i += 2)
  {
    // ???
    printf("%d\n", i);
    // ???
  }
}
```

Aufgabe 2

Eine typische Art der Prozesssynchronisation wird durch das Erzeuger-Verbraucher-Problem beschrieben. Folgende Eigenschaften sind für das EVP kennzeichnend:

- Genau 2 Prozesse greifen auf einen gemeinsamen Speicher zu.
- In dem Speicher können n Datensätze abgelegt werden.
- Einer dieser Prozesse legt die Daten im Speicher ab (Erzeuger).
- Der andere Prozess ruft sie ab (Verbraucher).
- Ein Über- und Unterlauf des Speichers soll verhindert werden.

a) Implementieren Sie das EVP mittels Semaphoren.

```
void erzeuger()
{
  unsigned int i = 0;

  while(1)
  {
    /* ??? */

    buffer[i] = /* ... */;
    i = (i + 1) % N;

    /* ??? */
  }
}
```

```
void verbraucher()
{
  unsigned int i = 0;

  while(1)
  {
    /* ??? */

    /* ... */ = buffer[i];
    i = (i + 1) % N;

    /* ??? */
  }
}
```

b) Welche Vereinfachung ergibt sich, wenn unbegrenzter Speicherplatz zur Verfügung steht?

Aufgabe 3

Gegeben seien folgende zwei Prozesse, die mit einer Ausnahme vom Betriebssystem beliebig umgeschaltet werden können:

> Prozess B soll `empfangen_?(`K_1`, y)` **vor** dem Sendevorgang in Prozess A ausführen.

Beide Prozesse kommunizieren über die Kanäle K_1 und K_2. Die Kanäle besitzen eigenen Speicher, d.h. sie arbeiten wie Behälter.

A

```
⋮
x := 15;
senden_a(K1, x);
z := 4;
empfangen_s(K2, z);
print(z);
⋮
```

B

```
⋮
y := 12;
empfangen_?(K1, y);
y := y + 4;
senden_a(K2, y);
⋮
```

Bestimmen Sie, wie sich die Empfangsart in Prozess B auf die Ausgabe von Prozess A auswirkt.