Technische Universität München
Fakultät für Informatik
Prof. Dr. M. Broy

WS 2002/2003
Übungsblatt 8
13. Dezember 2002

Übungen zur Vorlesung Einführung in die Informatik III

Aufgabe 29 Dining Philosophers

5 Philosophen sitzen um einen runden Tisch herum, zwischen je zwei Philosophen liegt eine Gabel auf dem Tisch. Philosophen beschäftigen sich abwechselnd mit Denken und Essen. Um essen zu können muss ein Philosoph zwei Gabeln aufnehmen, dann isst er eine Weile, dann legt er die Gabeln wieder ab und setzt das Denken fort.

(a) Geben Sie eine Lösung an, die Verklemmung vermeidet, unter Verwendung von Monitoren.

(b) Geben Sie eine Lösung an die zusätzlich noch das Verhungern von Philosophen vermeidet.

Aufgabe 30 Scheduling im Multiplexbetrieb

Im Begleitbuch zur Vorlesung werden zwei einfache Betriebssysteme für Multiplexbetrieb angegeben. Für beide Varianten soll die Abarbeitung von folgendem Beispiel untersucht werden:

- Im Externspeicher liegen drei Programmstapel P_1, P_2 und P_3;
- das Einlesen eines Programmstapels dauert jeweils 7 Zeiteinheiten;
- das Rechnen der Programme P_1, P_2, P_3 dauert 17, bzw. 1, bzw. 4 Zeiteinheiten;
- der Ausdruck eines Programmergebnisses dauert jeweils 5 Zeiteinheiten;
- bei der Programmbearbeitung der zweiten Variante wird ein Programm bis zu 3 Zeiteinheiten gerechnet, bevor es je nach Status wieder in eine Warteschlange eingereiht wird.

a) Geben Sie für jeden der drei Prozesse „reader“, „printer“ und „processor“ der ersten Betriebssystemvariante in Form eines Zeitstrahles an, wann welcher Prozess mit welchem Programm aktiv bzw. idle ist. Die Ausführungszeit der Betriebssystemprozeduren sei dabei gegenüber den zugrundegelegten Zeiteinheiten vernachlässigbar klein.

b) Geben Sie die Zeitstrahlen gemäß Aufgabe a) auch für die zweite Betriebssystemvariante an und ergänzen Sie diese durch die jeweiligen Werte der Warteschlangen „sq“, „pq“ und „cq“.

Im Anhang zum Angabenblatt die beiden Betriebssysteme angeben!

Aufgabe 31 *n* Leser-*m* Schreiber in Java

Implementieren Sie in folgenden Schritten zwei Lösungsvarianten des n Leser und m Schreiber Problems aus Aufgabe 27 in Java.

Realisieren Sie exemplarisch 10 Leser und 1 Schreiber als nebenläufige Threads. Der Schreiber soll in einer Endlosschleife die gemeinsame Programmvariable von 0–100 hochzählen. Alle Leser sollen nebenläufig ihre Id und den Wert der Programmvariablen immer dann auf dem Bildschirm ausgeben, wenn er sich aus ihrer Sicht geändert hat.

a) Implementieren Sie ein Programm `main`, in dem Sie die Leser- und Schreiberthreads erzeugen und den Ablauf Ihrer Lösung beobachten können.

b) Lösung mittels **await**

- Mit `synchronize, wait, notify` und `notifyAll` stehen in Java die Operationen eines Monitor-Konzeptes zur Verfügung. Eine **await** Anweisung existiert dahingegen nicht. Implementieren Sie eine Klasse `LSAwait`, in der Sie die kritischen Bereiche des Leser-Schreiber-Problems in einem Java-Monitor zusammenfassen.

— P —

- Lösen Sie das *n* Leser und *m* Schreiber-Problem mit Hilfe der Klasse `LSAwait`

c) Lösung mittels Semaphore

- Implementieren Sie die Klasse `SemaNat` mit den Operationen `P` und `V` zur Repräsentation natürlichzahliger Semaphore.
- Setzen Sie `SemaNat` zur Lösung des *n* Leser und *m* Schreiber-Problems ein.

<u>Anlage zu Aufgabe 30:</u>

```
proc processor =:
⌈   var nat i := 0;
    await ¬cr_active then
        cr_active, ar, er := true, AR(0) ER(0)
    endwait;
    while p_active do
        i:= (i+1) mod 3;
        await ¬cr_active
            then cr_active, ar, er := true, AR(i), ER(i)
        endwait;
        i:= (i-1) mod 3;
        "Rechne Programmabschnitt i"
        await ¬cp_active
            then cp_active, ap, ep := true, AR(i), ER(i)
        endwait;
        i:=(i+1) mod 3;
    od;                                                   ⌋
```

```
proc processor =:
⌈   var queue pair sq, cq pq := store, emptyqueue, emptyqueue;
    var bool h;
    var bool pjob, rjob := false, false;
    var string status;
    while p_active
```

```
do await true then h:= cr_active endwait;

Leseauftragendbehandlung:
        if ¬h ∧ rjob
        then sq, cq, rjob := rest(sq), stock(cq, first(sq)), false
        fi;

Leseauftragerteilung:
        if ¬h ∧ ¬isemptyqueue(sq)
        then await true then
                cr_active, ar, er := true, s1(first(sq)), s2(first(sq)) endwait;
                rjob= true
        fi;

Druckauftragendbehandlung:
        await true then h := cp_active endwait;
        if ¬h ∧ pjob
        then sq, pq, pjob := stock(sq, first(pq)), rest(pq), false
        fi;

Druckauftragerteilung:
        if ¬h ∧ ¬isemptyqueue(pq)
        then await true then
                cp_active, ap, ep := true, s1(first(pq)), s2(first(pq)) endwait;
                pjob := true
        fi;

Programmbearbeitung:
        if ¬isemptyqueue(cq)
        then    "Rechne bis zu i Zeiteinheiten das Programm in first(cq),
                setze status aus print oder compute"

Einordnen_des_Programms_in_Warteschlange:
        if status = print then cq, pq := rest(cq), stock(pq, first(cq))
                elif status = compute then cq := stock(rest(cq), first(cq))
                else FEHLER
                fi;
        fi;
od;
```