Ludwig-Maximilians-Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Datenbanksysteme

University of Munich
Institute for Computer Science
Database and Information Systems

Vorlesung Einführung in die Informatik: Systeme und Anwendungen im SS 2001

Erinnerung:

Am Freitag, den 8. Juni 2001, findet die Übung am CIP-Pool in der Oettingenstraße statt. (Gruppeneinteilung auf Übungsblatt 4)

Übungsblatt 3

Organisatorisches:

Abgabetermin: Freitag 8. Juni 2001, 13:00 (strikt)

per Email an die Korrektoren gemäß dem Anfangsbuchstaben desjenigen, der die Email absendet.

A – D	funkc@dbs.informatik.uni-muenchen.de	Caroline Funk
E – J	martinj@dbs.informatik.uni-muenchen.de	Johanna Martin
K – R	reisslh@dbs.informatik.uni-muenchen.de	Harald Reissl
S – Z	wiedeman@dbs.informatik.uni-muenchen.de	Florian Wiedemann

Wichtig! Die Email muß des Weiteren folgende Informationen enthalten:
Name, Vorname und Matrikelnummer aller Mitglieder des Teams, das die Aufgabe gemeinsam bearbeitet hat.

Aufgabe 4 (insgesamt 25 Punkte)

Die speisenden chinesischen Philosophen

Zu den klassischen Synchronisationsaufgaben zählt das Problem der speisenden Philosophen:

In einem Elfenbeinturm leben fünf Philosophen. Der Tageszyklus eines jeden Philosophen besteht abwechselnd aus Essen und Denken. Die fünf Philosophen essen an einem runden Tisch, auf dem in der Mitte eine Schüssel mit Nudeln steht. Jeder Philosoph hat seinen festen Platz am Tisch, und zwischen zwei Plätzen liegt jeweils genau ein Stäbchen.

Das Problem der Philosophen besteht nun darin, daß die Nudeln nur mit zwei Stäbchen zu essen sind. Darüber hinaus darf jeder Philosoph nur das direkt links und das direkt rechts neben ihm liegende Stäbchen zum Essen benutzen. Das bedeutet, daß zwei benachbarte Philosophen nicht gleichzeitig essen können.

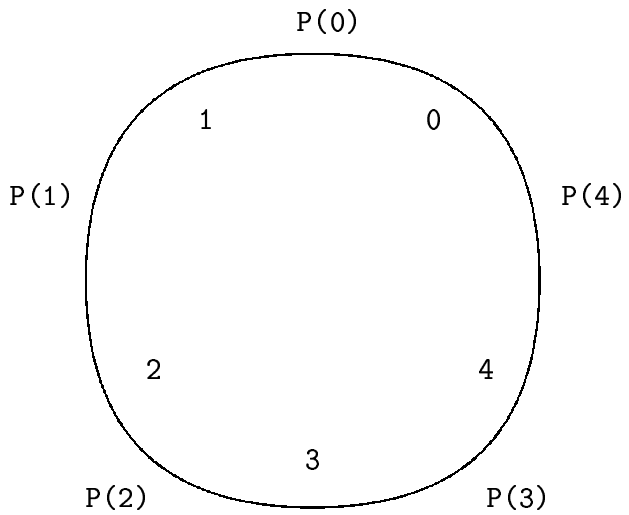

Abbildung 1: Speisende Philosophen

Vorschlag 1 (12 Punkte)

Für eine Lösung des Philosophenproblems seien die folgenden 5 binären Semaphore definiert:

```
VAR S(0), S(1), S(2), S(3), S(4):  binarysemaphore ;
```

wobei jeder der 5 Semaphore mit 1 initialisiert ist:

```
S(0).value := 1 ; S(0).queue := [] ;
S(1).value := 1 ; S(1).queue := [] ;
S(2).value := 1 ; S(2).queue := [] ;
S(3).value := 1 ; S(3).queue := [] ;
S(4).value := 1 ; S(4).queue := [] ;
```

Jeder Philosoph i, mit $i \in \{0, ..., 4\}$, führe den folgenden Algorithmus aus.
Er versucht zuerst das Stäbchen links von ihm zu bekommen und danach das Stäbchen rechts von ihm. Hat er beide Stäbchen, so ißt er. Danach legt er zuerst das linke und danach das rechte Stäbchen zurück.
`denken` und `essen` sind andere Programme, deren Ausführung je nach Philosoph und nach Aufrufzeitpunkt unterschiedlich lang dauern kann.

```
P(i):
LOOP
  denken;
  wait(S(i));
  wait(S((i+1) mod 5));
  essen;
  signal(S(i));
  signal(S((i+1) mod 5));
END LOOP;
```

Ein Ablaufprotokoll würde wie folgt initialisiert sein, wobei einer der Philosophen mit dem Denken anfängt. (Da wir nur einen Prozessor voraussetzen, denken die Philosophen also

quasiparallel. Das Synchronisationsproblem ändert sich dadurch aber nicht wesentlich.)
Für die Semaphore sind die `(value,queue)`- Paare angegeben.

P(0)	P(1)	P(2)	P(3)	P(4)	S(0)	S(1)	S(2)	S(3)	S(4)
			denken		(1,[])	(1,[])	(1,[])	(1,[])	(1,[])
				denken	(1,[])	(1,[])	(1,[])	(1,[])	(1,[])
			⋮		⋮	⋮	⋮	⋮	⋮
			denken		(1,[])	(1,[])	(1,[])	(1,[])	(1,[])

Dann hört einer der Philosophen — z.B. `P(0)` — zum Denken auf und will essen.

⇒ Zeigen Sie an Hand eines Ablaufprotokolls, daß es möglich ist, daß sich die 5 Philosophen verklemmen.

Bedenken Sie, daß sich die Philosophen genau dann verklemmt haben, wenn jeder Prozeß `P(i)` in der Warteschlange (`queue`) eines Semaphors 'hängt'. Das obige Ablaufprotokoll ist also so weiterzuschreiben, daß sich ein derartiger Zustand einstellt. (*Achten Sie auf die korrekte Veränderung der Semaphore bei jedem Programmschritt im Ablaufprotokoll. Die angegebenen Werte der Semaphore sind die Werte nach Ausführung der jeweils im linken Teil des Ablaufprotokolls angegebenen Programmanweisungen.*)

Um die Größe des Ablaufprotokolls nicht ausufern zu lassen, verwenden Sie bitte folgende Abkürzungen:

```
w(i)    statt  wait(S(i));
w(i+)   statt  wait(S((i+1) mod 5));
s(i)    statt  signal(S(i));
s(i+)   statt  signal(S((i+1) mod 5));
```

Vorschlag 2 (4 Punkte)

Nur vier Philosophen dürfen sich gleichzeitig an den Tisch setzen. Dies kann durch Einführung eines zusätzlichen Zählsemaphors `tisch` erreicht werden, dessen Zählvariable mit 4 initialisiert.

```
VAR tisch:  semaphor;
tisch.count := 4 ; tisch.queue := [] ;
```

⇒ Ändern Sie unter Verwendung des Zählsemaphors `tisch` den "Anweisungsteil" — zw. `LOOP` und `END LOOP;` — des Philosophen-Programms `P(i)` so ab, daß gewährleistet ist, daß höchstens vier Philosophen gleichzeitig ihr linkes Stäbchen nehmen können und somit immer mindestens ein Philosoph auch sein zweites Stäbchen nehmen und damit essen kann.

Vorschlag 3 (4+5 Punkte)

Der Philosoph 4 glaubt schlauer zu sein als die anderen. Er greift zuerst nach seinem rechten Stäbchen.

Dies bedeutet, jeder Philosoph i, mit $i \in 0, ..., 3$ führt den folgenden (alten) Algorithmus aus:

```
P(i):

LOOP
  denken;
  wait(S(i));
  wait(S((i+1) mod 5));
  essen;
  signal(S(i));
  signal(S((i+1) mod 5));
END LOOP;
```

und der Philosoph 4 führt folgenden Algorithmus aus:

```
P(4):

LOOP
  denken;
  wait(S(0));
  wait(S(4));
  essen;
  signal(S(0));
  signal(S(4));
END LOOP;
```

⇒ Zeigen Sie an Hand eines Ablaufprotokolls, daß Philosoph 4 trotz seiner Schlauheit nicht notwendigerweise als erster zu 2 Stäbchen kommt und als erster zu essen anfangen kann.

⇒ Als wievielter — 2., 3. ... — kommt Philosoph 4 im für ihn ungünstigsten Fall zum Essen?

Diese Frage soll unter den folgenden Annahmen beantwortet werden.

1. Die Prozessorvergabe ist fair, d.h. alle Prozesse im Zustand `ready` befinden sich in einer Warteschlange und bei Prozessorwechsel kommt der schon am längsten wartende Prozeß den Prozessor. Wechselt ein Prozeß den Zustand von `blocked` zu `ready`, so tritt er zu diesem Zeitpunkt in die Warteschlange der `ready`-Prozesse ein.
2. Die Warteschlagen (`queue`) der Semaphore sind ebenfalls fair, d.h. der schon am längsten wartende Prozeß bekommt die kritische Ressource.
3. Sie können annehmen, daß nach der Erzeugung der 5 Philosophenprozesse alle etwa zur gleichen Zeit das erste Mal hungrig werden ("Nach dem morgendlichen Aufstehen und vor dem Frühstück wird wenig gedacht.")
4. Sie können annehmen, daß kein Philosoph ewig denkt oder ewig ißt.
5. Machen Sie jedoch **keine** Annahmen über die relative Dauer der Abarbeitung der Programme `denken` und `essen`. Dies kann bei den verschieden Philosophen zu unterschiedlichen Zeitpunkten ganz verschieden sein.

Machen Sie sich klar, daß Philosoph 4 mit seiner Schlauheit zwar sich selbst nicht direkt geholfen hat, jedoch zumindest die Verklemmungsgefahr beseitigt hat.