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

Übungsblatt 3: Aufgabe 4

LÖSUNGSVORSCHLAG zu Vorschlag 1

Wenn alle fünf Philosophen quasi-gleichzeitig ihre erste wait-Operation (`wait(S(i));`) ausführen, d.h. alle quasi-gleichzeitig ihr linkes Stäbchen nehmen, folgt daraus eine Deadlock-Situation, da kein Philosoph das zweite Stäbchen nehmen kann. Bei Ausführung der zweiten wait-Operation (`wait(S((i+1) mod 5));`) werden alle Philosophen blockiert.

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,[])
w(0)					(0,[])	(1,[])	(1,[])	(1,[])	(1,[])
	w(1)				(0,[])	(0,[])	(1,[])	(1,[])	(1,[])
		w(2)			(0,[])	(0,[])	(0,[])	(1,[])	(1,[])
			w(3)		(0,[])	(0,[])	(0,[])	(0,[])	(1,[])
				w(4)	(0,[])	(0,[])	(0,[])	(0,[])	(0,[])
w(0+)					(0,[])	(0,[P(0)])	(0,[])	(0,[])	(0,[])
	w(1+)				(0,[])	(0,[P(0)])	(0,[P(1)])	(0,[])	(0,[])
		w(2+)			(0,[])	(0,[P(0)])	(0,[P(1)])	(0,[P(2)])	(0,[])
			w(3+)		(0,[])	(0,[P(0)])	(0,[P(1)])	(0,[P(2)])	(0,[P(3)])
				w(4+)	(0,[P(4)])	(0,[P(0)])	(0,[P(1)])	(0,[P(2)])	(0,[P(3)])

Alle Prozesse befinden sich in der Warteschlange eines Semaphors.

LÖSUNGSVORSCHLAG zu Vorschlag 2

Der "Anweisungsteil" (ohne das `denken`) des Philosophen-Programms wird zusätzlich mit `wait(tisch)` und `signal(tisch)` geklammert.
Dadurch ist gewährleistet, daß höchstens vier Philosophen quasi-gleichzeitig ihr linkes Stäbchen nehmen können und somit immer mindestens ein Philosoph auch sein zweites Stäbchen nehmen und damit essen kann.

Es ergibt sich als Lösung, daß jeder Philosoph i, mit $i \in \{0, ..., 4\}$ den folgenden Algorithmus ausführt:

```
P(i):

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

LÖSUNGSVORSCHLAG zu Vorschlag 3

1. Frage

Bei dieser Lösung wird kein zusätzlicher Semaphor benötigt. Vier Philosophen greifen wie bisher zuerst zu ihrem linken Stäbchen; der fünfte Philosoph greift zuerst zu seinem rechten Stäbchen. Dadurch wird ebenfalls erreicht, daß immer mindestens ein Philosoph auch sein zweites Stäbchen nehmen und somit essen kann. Das folgende Ablaufprotokoll ist eine mögliche Antwort auf diese Frage.

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,[])
w(0)					(0,[])	(1,[])	(1,[])	(1,[])	(1,[])
				w(0)	(0,[P(4)])	(1,[])	(1,[])	(1,[])	(1,[])
w(0+)					(0,[P(4)])	(0,[])	(1,[])	(1,[])	(1,[])
essen					(0,[P(4)])	(0,[])	(1,[])	(1,[])	(1,[])

Philosoph 4 hängt nach dem 2. Schritt (nach dem `denken`) in der Warteschleife von Stäbchen 0. Damit scheidet er für's erste aus. Es bleiben also 5 Stäbchen für 4 Philosophen, damit bekommt einer von ihnen 2 Stäbchen und kann essen.

Philosoph 4 ist so lange blockiert bis Philosoph 1 gegessen hat.

2. Frage

Im ungünstigsten Fall kommt Philosoph 4 zum essen, nachdem alle anderen gegessen haben, einige von ihnen u.U. schon mehrmals.

Folgende Tabelle gibt einen ungünstigen Ablauf an.
(Links die Stäbchen, die ein Prozeß hat. Hat ein Prozeß 2 Stäbchen, ißt er.)

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

3 Philosophen haben also schon gegessen. Während `P(0)` als 4. ißt — u.U. auch schon früher — kann folgendes passieren:

`P(2)` und `P(3)` können abwechselnd mit den Stäbchen 2, 3 und 4 essen so oft sie wollen (? in Tabelle unten).
Falls `P(1)` wieder hungrig wird, ändert das nichts, da sein Essensdrang durch den essenden Philosophen `P(0)` blockiert wird, und folglich `P(2)` und `P(3)` nicht blockieren kann.

Nach endlicher Zeit hört `P(0)` zu essen auf.

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

Jetzt will Philosoph `P(4)` Stäbchen 4 nehmen und es kommt auf den Zustand von Semaphor `S(4)` an, was passiert. Er bekommt Stäbchen 4, falls `S(4)=(1,[])` und bekommt es nicht, falls `S(4)=(0,[])` was bedeutet ob `P(3)` gerade ißt oder nicht.