Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion
CAU Kiel, Informatik und Prakt. Mathematik
Dr. Frank Huch

24118 Kiel
Christan-Albrechts-Platz 4
Tel.: 0431 / 880-7277

13. Übung „Funktionale Programmierung“

Abgabe am 13. Juli 2005 vor der Vorlesung

Aufgabe 1 20 Punkte

a) Erweitern Sie die in der Vorlesung vorgestellten dinierenden Philosophen um eine Nummerierung, so dass eine Ausgabe möglich ist, welcher Philosoph isst bzw. denkt.
Erweitern Sie das Programm außerdem um die Deadlockvermeidung mittels zurücklegen des linken Stabs, wenn der rechte Stab nicht verfügbar ist.
Implementieren Sie außerdem die in der Vorlesung diskutierte Deadlockvermeidung, bei der ein Philosoph die Stäbe in umgekehrter Reihenfolge nimmt.

b) Überarbeiten Sie die in der Vorlesung vorgestellte `Chan`-Implementierung, so dass `emptyChan` auf keinen Fall blockiert. Hierzu sollten Sie den `Chan` um eine Größeninformation erweitern, die die Zahl der Elemente im `Chan` angibt.

c) Implementieren Sie einen abstrakten Datentyp `FinChan`, für Kanäle mit fixer Kapazität ≥ 1. Die Kapazität soll bei der Konstruktion mittels `newFinChan` festgelegt werden. Im Unterschied zum Datentyp `Chan`, sollen Schreiber suspendieren, wenn der `FinChan` (unter Berücksichtigung der Kapazität) voll ist. Ein `FinChan` mit Kapazität 1 soll sich also genau wie eine `MVar` verhalten.

d) Implementieren Sie einen Stack-Server `stack`, welcher als Thread gestartet werden kann. Verwenden Sie folgende Signatur:

```
data StackOp a = Push a | Pop (MVar a)
stack :: MVar (StackOp a) -> IO (MVar a)
```

Die Argument-`MVar` von `stack` dient als Anfrage Kanal. Die `Pop`-Anfrage enthält zusätzlich eine (leere) `MVar`, in die der Stack-Server die Antwort schreibt.

Können Sie den Stack-Server implementieren ohne explizit eine Datenstruktur zur Speicherung der Stackeinträge zu verwenden? Die Idee hierbei ist die Verwendung des Laufzeitkellers.

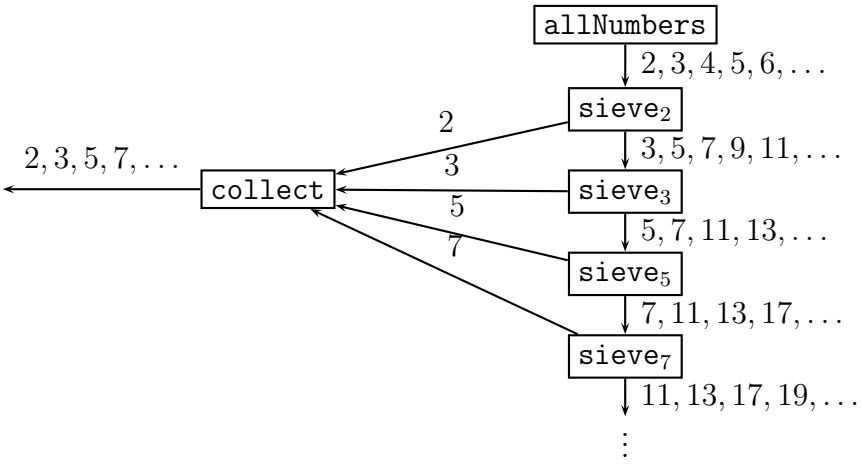
Zusatzaufgabe 2 10 Sonderpunkte

Auf der Web-Seite zur Vorlesung finden Sie ein Erlang-Modul, welches eine kleine Zähler-GUI implementiert. Zum Lösen dieser Aufgabe ist es nicht notwendig die Implementierung der Gui zu verstehen!
Erweitern Sie das Testprogramm `main()/outputMessages()` so, dass die Button der Gui folgende Bedeutung erhalten:

- `Start` der Zähler (und alle seine Clone) beginnt automatisch hochzuzählen (ungefährt im Sekundentakt).
- `Stop` der Zähler (und alle seine Clone) stoppt.
- `Copy` startet eine Kopie des Zählers, die sich zunächst gleich wie der Zähler verhält, aber ein eigenständiger Zähler und kein Clone ist.
- `Clone` der Zähler in dem dieser Button gedruckt wird, wird geklont. Clone und ursprünglicher Zähler verhalten sich identisch.

Zusatzaufgabe 3 10 Sonderpunkte

In dieser Aufgabe sollen Sie mit Hilfe nebenläufiger Erlang Prozesse das Sieb des Eratosthenes implementieren. Definieren Sie ein Modul `primes`, welches einen Prozeß zur Verfügung stellt, der auf Anfrage die Primzahlen der Reihe nach ausgibt.
Definieren Sie zunächst einen Prozeß `allNumbers`, welcher auf Anfrage durch einen anderen Prozeß alle Zahlen (ab zwei) ausgibt. Dann können Sie einen Prozeß `sieve` definieren, welcher von einem anderen Prozeß (initial der Prozeß `allNumbers`) jeweils weitere Zahlen erfragt. Die erste solche Zahl ist immer eine Primzahl. Aus allen weiteren Zahlen soll dieser Prozeß die Vielfachen dieser Primzahl herausfiltern und diese auf Anfrage weiterleiten. Verwenden Sie, zum Prüfen ob eine Zahl Vielfaches einer Primzahl ist, die Erlang-Infix-Operation `rem`, welche den Divisionsrest liefert.
Folgendes Bild soll das Vorgehen verdeutlichen:



Bei der Implementierung müssen Sie darauf achten, dass ein `sieve` Prozeß einen neuen `sieve` Prozeß erst dann erzeugt, wenn er vom `collect` Prozeß nach seiner ersten Primzahl gefragt wurde. Sonst erzeugen Sie direkt unendliche viele `sieve` Prozesse, was natürlich zu einem Speicherüberlauf führt.
Implementieren Sie auch eine Funktion, welche die Primzahlberechnung startet und die Primzahlen der Reihe nach abfragt und ausgibt.
Wieviele Erlang Prozesse verwendet Ihr System, zum Zeitpunkt bei dem die Primzahl 7727 ausgegeben wird? Wie oft wurde die Nachricht 7727 von einem zu einem anderen Prozeß verschickt? Wird sie bei der Ausgabe weiterer Primzahlen noch einmal verschickt?