Prof. Dr. Th. Letschert

THM TECHNISCHE HOCHSCHULE MITTELHESSEN
MNI Mathematik, Naturwissenschaften und Informatik

NVP – Nebenläufige und Verteilte Programme

Aufgabenblatt 13

Aufgabe 1

Ein bekannter Algorithmus zur Berechnung der Primzahlen ist das *Sieb des Eratosthenes*. Der Algorithmus ist wahrscheinlich schon seit tausenden von Jahren bekannt und wird an vielen Stellen beschrieben.[1] Es handelt sich um ein Verfahren zur Berechnung der Primzahlen zwischen 2 und einer oberen Grenze n.

Man erstellt dazu eine Tabelle der Zahlen von 2 bis n und von links nach rechts gehend streicht man alle Vielfachen der jeweils nächsten ungestrichenen Zahl. Eine sequentielle Definition des Algorithmus' ist:

```
object Sieve_Main extends App {

  def sieve(n: Int): List[Int] = {
    val num = Array.tabulate(n+1)(i => if(i>=2) i else -1)
    var p = 2
    while (p*p <= n) { // num(i) = -1: i ist gestrichen
      for (i <- 2*p to n by p) { // streiche Vielfache
        num(i) = -1
      }
      do { // gehe zu nächsten ungestrichenen Zahl
        p = p+1
      } while (num(p) < 0)
    }
    num.toList.filter(_ > 0)
  }

  println(sieve(100))

}
```

Das Sieb kann mit einem System von kommunizierenden Prozessen realisiert werden.

Diese Variante des Siebens von Primzahlen ist ein Allzeit-Klassiker der nebenläufigen Programmierung, lange bekannt, nicht seit Jahrtausenden, aber seit vielen Jahrzehnten, weit verbreitet wohl durch den epochalen Artikel von C.A.R. Hoare über kommunizierende Prozesse[2] und seitdem ein "virales" Beispiel, das jeder "Nebenläufige" einmal selbst implementiert haben sollte.

Die Prozesse bilden eine Pipeline von Filtern. Jeder Filter empfängt einen Strom an Zahlen und sendet einen Strom an Zahlen an seinen Nachfolger. Die erste Zahl, die ihn erreicht, ist eine Primzahl p, deren Vielfache er im Folgenden aussortieren wird. Jeder Filter bekommt und bestimmt damit eine Primzahl p und filtert aus dem Strom, der durch ihn fließt, alle Vielfache von p aus. Ein Filter entspricht damit der äußeren Schleife des sequentiellen Algorithmus' oben.

Eine dynamisch erzeugte Pipeline könnte so gestaltet werden, dass ein Filter, den eine Primzahl erreicht und der keinen Nachfolger hat, seinen Nachfolger erzeugt. Auf diese Weise wird die Pipeline im Fluge erzeugt.

Schreiben Sie eine Anwendung, die ein dynamisches, sich selbst entfaltendes Netz aus Filter-Threads aufspannt und mit

[1] Siehe beispielsweise Wikipedia: https://de.wikipedia.org/wiki/Sieb_des_Eratosthenes.

[2] Hoare, C. A. R. *Communicating Sequential Processes.* Commun. ACM 21.8 (1978): 666-677.
online, z.B.: http://cs2.ist.unomaha.edu/ stanw/papers/p666-hoare.pdf

diesem die Primzahlen in einem bestimmten Bereich berechnet.

Aufgabe 2

Schreiben Sie eine Funktion die die Primzahlen in einem bestimmten Bereich mit einem statischen, vor Beginn der Berechnung aufgespannten Pipeline-Netz berechnet.

Eine Pipeline muss ausreichend viele Filter enthalten, für jede Primzahl einen. Leider gibt es keinen Algorithmus zur Bestimmung der Zahl der Primzahlen in einem bestimmten Bereich – ausser dem, sie zu bestimmen. Nutzen Sie den sequentiellen Algorithmus um die notwendige Zahl der Filter im Voraus zu bestimmen.

Aufgabe 3

Lassen Sie die berühmten Philosophen wieder speisen. Jeder Philosoph wird durch einen Prozess (Thread) modelliert. Die Interaktion der Philosophen ist jetzt jedoch auf den Austausch von Nachrichten beschränkt. Gemeinsame Ressourcen sind verboten: Keine Speicherplätze, keine Mutexe, Semaphore, Monitore etc. auf die mehr als ein Philosoph zugreifen kann. Alles nicht erlaubt.

Lösen Sie das Synchronisationsproblem der Philosophen mit aktiven Monitoren.