Prof. Dr. Franz Schweiggert — Institut für Angewandte Informationsverarbeitung — 6. Mai 2009
Michael Mattes — Blatt 3

Systemnahe Software (II) im Sommersemester 2009

Abgabetermin: 13. Mai 2009

8 Vektormultiplikation (6 Punkte)

Als Vorbereitung zur nächsten Aufgabe sollen Sie ein Programm `vmult.c` schreiben, welches zwei ganzzahlige Vektoren miteinander multipliziert und den resultierenden Wert als Exitstatus zurückliefert. Das Endergebnis soll sich im Bereich zwischen 0 und 254 befinden, allerdings können die Zwischenergebnisse diesen Bereich verlassen – beachten Sie dies. Falls der Bereich verlassen wurde, soll der Exitstatus 255 sein.

Die Daten erhält `vmult.c` über die Kommandozeile (d.h. argc und argv), die Dimension der Vektoren soll aus der Anzahl der Argumente berechnet werden. Falls die Anzahl der erhaltenen Werte ungerade ist, liegt ein Fehler vor und das Programm soll auch hier den Exitstatus 255 liefern.

Hier ein Beispiel, bei dem die Vektoren (1,1,1,4) und (-1, -2, -3, 5) miteinander multipliziert werden:

```
$ ./vmult 1 1 1 4 -1 -2 -3 5
$ echo $?
14
$
```

9 Parallele Matrixmultiplikation (24 Punkte)

Achtung Forklawine: Programmieren Sie diese Aufgabe nur zuhause oder auf einem Arbeitsplatzrechner!

In fast allen Anwendungen von Mathematik in der Technik wird an irgendeiner Stelle die Multiplikation zweier quadratischer Matrizen verwendet. Da die meisten Rechner heutzutage mehr als einen Prozessorkern besitzen, wäre der klassische Ansatz der seriellen Berechnung aller Matrix-Einträge Zeitverschwendung.

Ihre Aufgabe ist es daher, eine parallele Matrixmultiplikation `mmult.c` zu implementieren. Dabei berechnet jeder Kindprozess genau einen Eintrag der Ergebnismatrix, indem er `vmult.c` (über `exec()`) ausführt. *Der Hauptprozess soll keinerlei Berechnungen durchführen, sondern nur die Kindprozesse koordinieren.*

Die Werte der Matrizen soll `mmult.c` über die Standardeingabe lesen, und zwar diesmal als ASCII-Klartext. Zunächst wird hier die Dimension n der Matrizen angegeben (n <= 50), danach folgen in der gleichen Zeile (von links nach rechts, dann von oben nach unten) die n*n Einträge der ersten Matrix und anschließend die n*n Einträge der zweiten Matrix. Für das Einlesen können Sie `scanf()` verwenden.

Man bekommt keinen Geschwindigkeitszuwachs, wenn man mehr parallel laufende Prozesse hat, als das System Prozessorkerne besitzt. Deswegen soll es möglich sein, die Anzahl der parallel laufenden Kindprozesse zu begrenzen. Die Begrenzung wird über die Kommandozeile festgelegt. Wird dort nichts angegeben, sollen maximal 5 Kindprozesse zur gleichen Zeit laufen.

Sollte eine Multiplikation den erlaubten Wertebereich verlassen, soll das Programm eine entsprechende Fehlermeldung ausgeben. Im Erfolgsfall ist die Matrix in einer lesbaren Form auszugeben.

In folgendem Beispiel werden zwei Matrizen mit maximal 2 Kindprozessen multipliziert:

```
$ cat input
3 1 0 1 0 2 0 0 0 -3 4 5 6 7 8 9 -3 -4 -5
$ ./mmult 2 <input
  1   1   1
 14  16  18
  9  12  15
$
```

Viel Erfolg!