PHILIPPS-UNIVERSITÄT MARBURG D-35032 Marburg
Fachbereich Mathematik und Informatik Hans Meerwein Straße
Prof. Dr. R. Loogen
Dipl.Inf. J.Berthold

21.April 2005

Übungen zu „Parallele Programmierung“, SS 2005

Nr. 2, Abgabe der Aufgaben: 28.April 2004 vor der Vorlesung

Hinweise:
Am Mittwoch, dem 27.5.2005 findet wegen des *Tags der Informatik* keine Vorlesung statt.

Übungsaufgaben können in Gruppen von max. zwei Personen abgegeben werden.
Geben Sie Programme bitte ausgedruckt und zusätzlich per Mail ab.

Prozesse in MPD

In den Beispielen und der ersten Übung kamen bereits Prozeduren und (neben dem `co`-Konstrukt) die Anweisung `fork` vor. MPD stellt ein weiteres Konstrukt zur Verfügung: mit `process` können Prozesse definiert und gleich gestartet werden, ohne explizit eine Prozedur dafür zu definieren.

```
resource meinProgramm()
  int i,j

  # starte 25 konkurrierende Prozesse
  process meinProzess[i = 1 to 5,j = 8 downto 0 by -2] {
    # Anweisungen können i und j als Parameter benutzen
    write("Prozess (",i,",",j,") ...")
  }

  # Rest konkurrierend zu erzeugten Prozessen ausgeführt
  write("Hauptprozess ...")

  final { # nach Ende aller gestarteten Prozesse der resource:
      write("Programm beendet.")
  }
end meinProgramm
```

Der Rumpf des Prozesses wird als konkurrierender Thread abgespalten, der Hauptprozess läuft weiter (nicht synchronisiert). Den gleichen Effekt kann man mit `fork` und einer passenden Prozedur erzielen.
Durch die Quantifizierung im Beispiel wird gleich eine ganze Serie von Prozessen gestartet. Dabei sind `i` und `j` die Aktualparameter der implizit definierten Prozedur.

Bitte wenden!

Aufgaben

2.1 **Präfixsummen** Gegeben sei eine Liste von ganzen Zahlen $x = [x_1, \ldots, x_n]$ $(n \geq 1)$. 4 Punkte

Entwerfen Sie einen parallelen Algorithmus zur Bestimmung aller Präfixsummen

$$S_i := \sum_{j=1}^{i} x_j \quad (1 \leq i \leq n)$$

dieser Liste.

Analysieren Sie Ihr Verfahren hinsichtlich der Anzahl der nötigen Arbeitsschritte und der insgesamt durchgeführten Operationen und bestimmen Sie, welcher Speed-up mit Ihrem Verfahren erzielt werden kann.

2.2 **PCAM - Entwurf** 4 Punkte

Für ein Wort w über dem Alphabet $\Sigma = \{a, \ldots z\}$ bezeichne $\overline{w}$ die Spiegelung von w, d.h. $\overline{\varepsilon} = \varepsilon$, $\overline{aw} = \overline{w}a$ $\forall a \in \Sigma, w \in \Sigma^*$

Eine Sprache $W \subset \Sigma^*$ heiße *palindromisch*, falls $\forall w \in W : \overline{w} \in W$

Entwerfen und diskutieren Sie *nach der PCAM-Methode* einen parallelen Algorithmus, der aus einer als Datei vorliegenden Wortliste die maximale palindromische Teilmenge bestimmt.

2.3 **Semaphoren zur Zuteilung beschränkter Ressourcen** 4 Punkte

Ein Flugplatz verfüge über s Startbahnen und einen Tower zur Überwachung des Flugverkehrs. Dieser Flugplatz wird von $f > s$ Flugzeugen angeflogen, die dort wiederholt landen und nach einer Weile wieder starten.

In der Zeit zwischen Landen und Starten blockiert ein Flugzeug eine der Startbahnen. Während eines Lande- oder Startvorgangs benötigt ein Flugzeug ununterbrochenen Kontakt mit dem Tower (und selbstverständlich eine Startbahn).

Simulieren Sie diese Vorgänge über ein MPD -Programm mit Semaphoren. Verwenden Sie den Wartebefehl `nap` und `random`, um zufällige Wartezeiten zu erzeugen.