PHILIPPS-UNIVERSITÄT MARBURG
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen
Dipl.Inf. J.Berthold

D-35032 Marburg
Hans Meerwein Straße

18.April 2005

Übungen zu „Parallele Programmierung“, SS 2005

Nr. 1, Abgabe und Besprechung der Aufgaben: 25.April 2005 in der Übung*

Hinweise:
* Das erste (d.h. *dieses*) Übungsblatt erscheint zur ersten Übung und ist auch dort wieder abzugeben. Die folgenden Übungsblätter werden *in der Vorlesung* aus- und zurückgegeben.

Benutzung von Linux, Netz des Fachbereichs

In den praktischen Übungen zur Vorlesung werden Programmiersprachen und Systeme benutzt, die nur unter Linux verfügbar sind. Auch wenn das HOME-Verzeichnis unter Windows als Laufwerk *U:* sichtbar ist, müssen Sie mit Hilfe einer Konsole (Shell) Programme übersetzen und ausführen. Machen Sie Sich möglichst bald mit der Bedienung einer Linux-Konsole und den gängigsten Kommandos vertraut.

Windows Aufruf des *SSH-Client* (Programme-Network) und Angabe eines Rechnernamens. Dies öffnet eine Konsole auf dem angegebenen Rechner. Sie können mit Windows-Programmen in Ihrem HOME arbeiten und die Konsole zum Übersetzen und Ausführen verwenden.

Linux Sie können einen der Linux-Rechner in den Räumen D4424 oder D5437 benutzen. Dort arbeiten Sie unter einer grafischen Benutzeroberfläche und können eine Konsole öffnen (Systemwerkzeuge-Terminal).

Benutzbare Rechner (in den PC-Arbeitsräumen) sind die folgenden:
`namibe, juba, dodoma, kitwe, bujumbura, laayoune, ibadan, zinder`

Programmiersprache MPD

Im ersten Teil der Vorlesung wird die Modellsprache MPD (Multithreaded, Parallel, and Distributed Programming) benutzt. Die Sprache ist am Fachbereich unter Linux installiert (*/app/lang/parallel/mpd*). Zur Benutzung fügen Sie */app/lang/parallel/mpd/bin* zu Ihrem Suchpfad hinzu, am besten permanent in der Datei *.tcshrc* in Ihrem HOME-Verzeichnis.

```
setenv PATH ${PATH}:/app/lang/parallel/mpd/bin
```

Die Syntax der Sprache MPD ist ähnlich wie die von C (die Vorgängersprache SR war dagegen syntaktisch an Pascal angelehnt).

- Anweisungen und Deklarationen werden durch Semikolon oder Zeilenwechsel getrennt.
- Blöcke von Anweisungen und Deklarationen werden in geschweifte Klammern gesetzt. Parameter für Prozeduren (und andere Konstrukte) stehen in runden Klammern, Indices für Arrayzugriffe und Schleifenvariablen in eckigen.
- Variablen müssen (irgendwo!) vor der ersten Benutzung deklariert werden,

- Einzeilige Kommentare werden mit # eingeleitet, mehrzeilige Kommentare in `/* ...*/` eingeschlossen.
- Die zentralen Konzepte in MPD sind virtuelle Machinen (`vm`), Ressourcen (`ressource`) und globale Objekte (`global`).

  Eine *virtuelle Machine* ist ein Adressbereich (und entspricht i.W. einer "realen" Maschine im Netz). Implizit wird stets eine VM für das Hauptprogramm angelegt.

  Eine *Ressource* ist eine Schablone ähnlich wie eine Klassendefinition: sie definiert eine Einheit von lokalen Variablen, Prozeduren und Prozessen und kann mehrfach instanziert werden. Die im Sourcecode *als letztes* definierte Resource ist das Hauptprogramm und wird beim Programmstart automatisch einmal instanziert.

  Ressourcen können andere Ressourcen importieren und ihre Operationen aufrufen. Ebenso werden auch *globale Objekte* importiert; sie sind aber in einem gesamten Adressbereich (VM) für alle importierenden Ressourcen-Instanzen die gleichen.

Auf der Vorlesungsseite finden Sie ein Dokument mit einer Zusammenfassung der wichtigsten vordefinierten Operatoren und Funktionen von MPD . Das Dokument bezieht sich auf den Vorgänger SR, gilt allerdings bis auf die Zuweisung (`=` statt `:=`) und die Vergleichsoperation (`==` statt `=`) auch für MPD .
Die erste Übung enthält Aufgaben, in denen Sie das Prozedurkonzept von MPD kennen lernen. Eine Prozedur wird mit `op` deklariert und mit `proc` definiert. Alternativ kann man Prozeduren auch mit `procedure` definieren, falls keine externe Deklaration nötig ist. Prozeduren können mit Hilfe von `co ...oc` konkurrierend gestartet werden, wobei dieses Konstrukt auch automatisch für eine Synchronisation am Ende sorgt.

Aufgaben

1.1 Binomialkoeffizienten — 7 Punkte

Bekanntlich lassen sich die Binomialkoeffizienten neben der geschlossenen Formel auch über die Rekursionsformel $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$ bestimmen. Das Berechnungsschema ist bekannt als das *Pascalsche Dreieck*.

(a) Schreiben Sie ein MPD -Programm, das Binomialkoeffizienten mit Hilfe einer rekursiven Prozedur berechnet. / 3

(b) Modifizieren Sie Ihr Programm, so dass es ein *global definiertes*, zweidimensionales Array benutzt. Das Array soll am Ende das Pascalsche Dreieck bis zum gewünschten Wert enthalten und schrittweise berechnet werden. / 3

Wie viele Prozesse benutzt Ihr Programm, wie viele Arbeitsschritte führen die Prozesse aus? Welche Probleme könnten bei einer Parallelisierung auftreten? / 1

1.2 Rekursives Doppeln — 5 Punkte

Das Verfahren des „rekursiven Doppelns“ erlaubt die Anwendung einer assoziativen binären Operation auf n Zahlen mit $\lfloor \frac{n}{2} \rfloor$ Prozessoren in $\lceil \log_2 n \rceil$ Arbeitsschritten.

Zeigen Sie, dass eine entsprechende Berechnung mit $p < \lfloor \frac{n}{2} \rfloor$ Prozessoren mindestens

$$T_{p,\min} = \lceil \log_2 p \rceil + \left\lceil \frac{n - 2^{\lceil \log p \rceil}}{p} \right\rceil$$

Schritte benötigt.