

CHRISTIAN-ALBRECHTS-UNIVERSITÄT ZU KIEL

Institut für Informatik

Prof. Dr. W.-P. de Roever

Jens Schönborn und Jan Waller



Nebenläufige und verteilte Programmierung

Wintersemester 2006/07

Serie 1

18. Oktober 2006

Thema: Gegenseitiger Ausschluß

Ausgabetermin: 18. Oktober 2006

Abgabe: 23. Oktober 2006 11:00 im Schrein

Organisatorisches Wir bitten euch die Übungen (außer Mitt- und Endsemestertest) in Zweiergruppen ab zu geben. Die Übungen werden Montags ausgeteilt und sollen am Montag der folgenden Woche bis 11:00 im Schrein abgegeben werden. In der Woche der Abgabe werden sie dann in den Übungen besprochen und zurückgegeben.

Diese Serie dient dazu sich mit der Programmiersprache MPD vertraut zu machen. Quellen des MPD-Compilers und der zugehörigen Werkzeuge, Informationen zur Benutzung und Beispielprogramme können im Netz gefunden werden unter:

<http://www.cs.arizona.edu/mpd>

<http://www.cs.arizona.edu/mpd/programs/tutorial.html>

Die Programmierwerkzeuge sind frei verfügbar und können auch zu Hause installiert werden. Sie sind ebenfalls im Uni-Netz installiert, und zwar im Verzeichnis

`/home/provers/bin/mpd`

Sollten Probleme mit den installierten Tools auftreten, bitte Mail an Jens:

`jes@informatik.uni-kiel.de`

Um diese Programme nutzen zu können braucht man lediglich seinen Pfad um dieses Verzeichnis zu erweitern, bei Benutzung der Shell bash muss dazu in der Datei `~/.bashrc` die folgende Zeile angehängt werden:

```
export PATH="/home/provers/bin/mpd:$PATH"
```

Unter csh/tcsh muss entsprechend `~/.csh` bzw. `~/.tcsh` erweitert werden um:

```
setenv PATH "/home/provers/bin/mpd:$PATH"
```

Die folgenden Werkzeuge stehen danach zur Verfügung:

mpd Compiler

mpdl Linker

mpdm Makefile maker (für komplexere Programme)

mpdprof mpd profile, liest trace files von Programmabläufen und liefert eine lesbare Darstellung

Programmierlösungen sollten kommentiert sein, grundlegende Ideen und Vorgehensweisen sollten erläutert werden. Die Abgabe der Programme erfolgt zunächst als Ausdruck, zusammen mit zwei Probeläufen. Stichprobenartig wird es eine *Korrektur in Anwesenheit* geben, bei der die Lösungen erläutert werden und vorgeführt werden sollen.

Aufgabe 1 (4 Punkte) [Exercise 1.7] Programmieren vier Programme zur Lösung des Quadraturproblems, gegeben in Section 1.5:

1. ein sequentielles Programm das eine feste Anzahl von Intervallen benutzt,
2. ein sequentiell rekursives Programm das adaptive Quadratur benutzt,
3. ein parallel iteratives Programm welches eine feste Anzahl von Intervallen nutzt und
4. ein rekursives paralleles Programm mit adaptiver Quadratur.

Die Programme sollen als Kommandozeilenargumente die linken und rechten Grenzen für die Quadratur bekommen, sowie die Anzahl der Intervalle (bei (a) und (c)) bzw. den Wert von EPSILON (bei (b) und (d)).

Verwende die Funktion `sin(x) * exp(x)`.

Aufgabe 2 (4 Punkte) Schreibe ein Programm `differ`, welches eine vereinfachte Version des Unix `diff` ist. Als Kommandozeilenargumente sollen die Namen zweier Files angegeben werden. Das Programm soll alle Zeilen ausgeben, in denen sich die Zeilen unterscheiden, und zwar in der folgenden Form:

```
filename1 lineNumber: line from file 1
filename2 lineNumber: line from file 2
```

Ist eine Datei länger als die andere, so soll auch je eine Ausgabezeile für jede dieser Extrazeilen ausgegeben werden.

Schreibe eine zweite parallele Version dieses Programms, benutze dazu drei Prozesse: zwei die die Eingabedateien lesen und einen Prozess der die Zeilen vergleicht und die Ausgaben macht.

Die lesenden Prozesse sollten *double buffering* benutzen: lese in einen zweiten Eingabepuffer während der erste untersucht wird. Die Prozesse müssen synchronisieren um garantieren zu können, dass Puffer nicht überschrieben werden bevor sie fertig untersucht worden sind. Dazu können "flag" Variablen und *busy waiting* benutzt werden.