Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Prof. Dr. W.-P. de Roever
Jens Schönborn, Christian Motika und Tim Fenten

Nebenläufiges Programmieren

Sommersemester 2008 — **Serie 1** — 9. April 2008

Ausgabetermin: 9. April 2008

Abgabe: 18. April 2008 11:00

Organisatorisches: Wir bitten Euch die Übungen (außer Mitt- und Endsemestertest) in Zweiergruppen abzugeben. Die Übungen werden Mittwochs ausgeteilt und sollen am Freitag der darauffolgenden Woche bis 11:00 abgegeben werden. Programmierlösungen sollen kommentiert werden, grundlegende Ideen und Vorgehensweisen sollen dabei erläutert werden. Die Abgabe der Programme erfolgt zunächst als Ausdruck, zusammen mit zwei Probeläufen sowie als Mail an `jes@informatik.uni-kiel.de`. Stichprobenartig kann es eine *Korrektur in Anwesenheit* geben, bei der die Lösungen erläutert und vorgeführt werden sollen.

Aufgabe 0 (StudiDB) Sofern noch nicht geschehen, meldet Euch in der StudiDB für die richtige Übung an.
Beachtet bitte, dass es zwei Übungen gibt: 080123 für Bachelors, die die Vorlesung als Projektvorbereitungsmodul hören und 080111 für allen anderen.
Weitere Informationen findet Ihr auf der Webseite zur Vorlesung:

`http://www.informatik.uni-kiel.de/inf/deRoever/SS08/ConcurrentProgramming`

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

```
http://www.cs.arizona.edu/mpd
http://www.cs.arizona.edu/mpd/programs/tutorial.html
http://www.cs.arizona.edu/mpd/language.html
http://www.cs.arizona.edu/mpd/SRbook.appendixC.html
```

Die Programmierwerkzeuge sind frei verfügbar und können auch unter Linux installiert werden.

MPD im Uni-Netz

MPD ist im Uni-Netz im folgenden Verzeichnis installiert

```
/home/provers/bin/mpd
```

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

Sollten Probleme mit den im Uni-Netz installierten Tools auftreten, bitte eine Mail an Jens: `jes@informatik.uni-kiel.de`

MPD auf Linux

Wir können leider nicht für jede Distribution/Version Unterstützung bei der Installation anbieten, waren aber mit folgenden Informationen erfolgreich:
Hilfreiche Links auf der MPD-Seite:

- `http://www.cs.arizona.edu/mpd/download/`
- `http://www.cs.arizona.edu/mpd/install.html`

1. MPD herunterladen und entpacken
2. sind die Pakete lex und yacc installiert, alternativ gehen auch flex und bison
3. in der Datei "Configuration" den Pfad zum C-Compiler richtig setzen
4. den Installationsanweisungen aus oben genanntem Link folgen
5. der Fehler "incompatible implicit declaration of built-in function" wird durch ein fehlendes include der entsprechenden Library in der jeweiligen Datei verursacht

Benutzer von Debian Linux sollten sich bei Problemen die Seite
`http://www.cs.arizona.edu/mpd/download/` anschauen.
Falls Fragen auftreten könnt Ihr sie gerne per Mail an Tim Fenten stellen
`tif@informatik.uni-kiel.de`. Wir können aber nicht garantieren jede Frage auch beantworten zu können.

Aufgabe 1 (4 Punkte) [Exercise 1.7] Programmiere 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 Prgramm 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.