Universität Hamburg
Fachbereich Informatik
Wissenschaftliches Rechnen
Prof. T. Ludwig, H. Lenhart, U. Körner, N. Hübbe

Übungsblatt 3 zum Praktikum
Paralleles Programmieren für
Geowissenschaftler im SS 2012
Abgabe: 9.05.2012

Aufgabe 3: FORTRAN Debugging Übungsaufgaben

Dieses Übungsblatt umfasst im Wesentlichen Aufgaben zum Debugging von FORTRAN Programmen.

Dazu werden auf der Webseite Programme bereitgestellt, die mit **gdb** oder **Valgrind** auf Fehler untersucht werden sollen. Die Korrektur soll so erfolgen, dass fehlerhafte Zeilen/Blöcke auskommentiert werden und die korrigierten Zeilen/Blöcke in unmittelbarer Nähe stehen. Zusätzliche Kommentarzeilen sollten die Korrektur leicht nachvollziehbar machen. (Ausnahme: Game of Life)

Wie gehabt, sollten Probleme auftauchen, wenden Sie sich bitte an die Mailingliste:

`PPG-12@wr.informatik.uni-hamburg.de`

Aufgabe 3A: Game of Life - Vollständig (90 Punkte)

In der neuen Aufgabe soll das Konzept von Conway's "Game of Life" in einem Programm **vollständig** umgesetzt werden. Als Basis dient Euer Programm, das Ihr für das Übungsblatt 2 erstellt habt. Nähere Informationen sind auf dem letzten Übungsblatt zu finden.

Diesmal sollen die Ecken der Matrix jeweils mit einem "Beacon" Muster ausgefüllt werden. Die Abfragen an den Rändern müssen entsprechend angepasst bzw. ergänzt werden.

Es sollen die "Lebenszyklen" der verschiedene Muster wieder mit 10 Iterationen durchlaufen und die Muster ensprechend dargestellt werden.

Aufgabe 3B: Abzählfehler (60 Punkte)

Als Basis dient das Programm count3_err.f90, welches auf Fehler untersucht werden soll.(Methode beliebig)

Diese Variante des Abzählprogramms liefert falsche Werte. Korrigiert es, so dass es die korrekte Reihenfolge liefert. Beschreibt mit wenigen Sätzen Eure Vorgehensweise (Debug-Methode, etc.).

Aufgabe 3C : Sortieralgorithmen (120 Punkte + 60 Bonuspunkte)

Das Testprogramm sorter.f95 (runterzuladen auf der Website) implementiert zwei unterschiedliche Sortieralgorithmen, enthält aber drei unterschiedliche Bugs:

Einfach: Sorgt dafür, dass **Valgrind** keine verlorenen Blöcke mehr auflistet. Die beiden Fehlermeldungen "Invalid read of size 4" und "Process terminating with default action of signal 11 (SIGSEGV)", die Valgrind ausgibt, gehören zum nächsten Fehler und sind für diesen Aufgabenteil unerheblich. Entscheident ist die Ausgabe unter "LEAK SUMMARY".

Mittelschwer: **Bucket sort** ist ein Sortieralgorithmus, bei dem die Werte in einem Durchgang in eine Anzahl von Eimern mit unterschiedlichen Wertebereichen sortiert werden. Am Ende eines Sortierschrittes können sich also beliebig viele verschiedene Werte in den verschiedenen Eimern befinden. Wenn also alle Werte aus Eimer 0 vor den Werten aus Eimer 1 stehen, ist bereits eine grobe Sortierung erreicht. Diese Implementation sortiert die Daten vollständig, indem sie alle Eimer, in denen etwas drin ist, anschließend mit dem selben Verfahren durchsortiert. Leider ist sie nicht ganz korrekt, was zum Absturz des Programmes führt. Benutzt **gdb**, um herauszufinden, warum es zu dem fehlerhaften Speicherzugriff kommt.
Tips: Wie wird der Index berechnet, und welche Werte haben die Variablen, die an seiner Berechnung beteiligt sind? Und wie kommt es zu den vielen ähnlichen Zeilen in der Ausgabe von gdb's `bt` Befehl?

Schwer: **Insertion sort** sortiert die Zahlen, wie ein Kartenspieler seine Karten sortiert. Er fängt mit einer Karte in der Hand an, und fügt jeweils die nächste Karte an der richtigen Stelle ein, bis er alle Karten auf der Hand hat. Insertion sort geht ähnlich vor: Es geht mit einem Zähler i von links nach rechts durch das zu sortierende Array durch. Bevor i hochgezählt wird, ist das Subarray bis zur Stelle i sortiert. Nach dem Hochzählen ist der Wert an der Stelle i im Normalfall nicht an der richtigen Stelle, er wird aus dem Array herausgenommen. Anschließend werden alle Werte links von i, die größer als der einzusortierende Wert sind, um eine Position nach rechts verschoben. Am Ende wird der einzusortierende Wert an die freigewordene Stelle geschrieben. Leider produziert die Implementation in sorter.f95 am Anfang des Arrays eine mehr oder weniger lange Sequenz von gleichen Werten, die so in den Eingabedaten nicht vorhanden war. Sorgt dafür, dass das nicht passiert. **Diesen Bug zu finden ist optional!**

Abgabe

Die auf dem Cluster lauffähigen FORTRAN Programme sollen als Quellcode mit der Angabe der Gruppe (Personen in der Gruppe) bis zum Mittwoch den 9.5.2012 geschickt werden an:

hermann.lenhart@informatik.uni-hamburg.de

D.h. **nicht an die Mailingliste** schicken und **bitte NUR den Quellcode** schicken!

Als Subject im Kopf der Mail bitte die Angabe: PPG-12 Blatt3 und die Liste der Familiennamen der Personen in der Übungsgruppe. Die **fehlerfreien FORTRAN Programme** sollen **mit Kommentaren zur Fehlerbehebung** versehen als Attachment anhängt werden.