Vorlesung Systemnahe Programmierung – WS 2014/15

Prof. Dr. Matthew Smith, Dr. Matthias Frank, Sergej Dechand

3. Übungszettel

Ausgabe: Mi 22.10.2014; Abgabe bis zum (Freitag) 31.10.2014, 23:59 Uhr
Die Vorführung erfolgt in der Woche vom 03.11.2014 bis zum 07.11.2014 in Ihrer Übungsgruppe.

Alle Programme müssen im Poolraum unter Linux kompilierbar bzw. lauffähig und ausreichend kommentiert und mit einem Makefile versehen sein. Die Lösungen sind bei Ihrem Tutor während Ihrer Übungsgruppen vorzuführen. Die Abgabe erfolgt mittels Ihres SVN-Repositories jeweils in dem Verzeichnis /tag/AufgabeXY.

Aufgabe 7: Fehlersuche in Programmen (2 Punkte)
In das folgende Programm haben sich einige syntaktische und semantische Fehler eingeschlichen, die Sie erkennen und korrigieren sollen.
Tipp: Für die Fehlersuche in komplexeren Programmen ist es sinnvoll, statt der Einführung von Hilfsausgaben einen Debugger zu verwenden. Dieser ermöglicht eine Schrittweise Programmausführung, bei der in jedem Schritt eine Beobachtung aller Variablen im Sichtbarkeitsbereich durchgeführt werden kann. Ein prominenter Debugger für C-Programme ist dabei der GNU Debugger (gdb). Im Internet finden sich zahlreiche Einführungen in die Bedienung von gdb.
Finden und korrigieren Sie die im folgenden Programm vorhandenen Fehler. Protokollieren Sie die Fehler in einer „README“-Datei. Das Programm können Sie auch von unserer Homepage herunterladen:

```c
#include <stdio.h>

int main(void) {
  int number1 /* first summand */
  int number2 /* second summand */

  printf("Bitte erste ganze Zahl eingeben: ");
  scanf("%i", number1);

  printf("Bitte zweite ganze Zahl eingeben: ");
  scanf("%i", number2);

  if (number1 = number2) {
       printf("Die eingegebenen Zahlen sind identisch.\n");
  }

  printf("Die Summe von %i und %i ist %i.\n",
         number1, number2, number1 + number2);

  return 0;
```

Aufgabe 8: Matrix-Operationen (4 Punkte)

Implementieren Sie ein Programm für die Generierung von Matrizen und ein Programm für die Multiplikation und Addition von Matrizen. Das Programm soll die beiden Eingabematrizen aus je einer Datei einlesen, die Matrix-Multiplikation bzw. Addition durchführen und die Ergebnis-Matrix wieder in eine Datei zurückschreiben.

Hinweise zum Matrix-Textdateiformat:

- Jede Zeile der Matrix ist in einer separaten Zeile der Textdatei gespeichert, wobei die x-te Zeile der Matrix in der x-ten Zeile der Textdatei zu finden ist.
- Die einzelnen Zahlenwerte einer Matrixzeile werden jeweils durch ein einzelnes Leerzeichen getrennt in die entsprechende Zeile der Datei geschrieben.
- Die Berechnungen sollen positive und negative Ganzzahlen unterstützen. Um auch große Zahlenwerte zu unterstützen, soll intern mit dem Datentyp *long* gearbeitet werden.

Das **Matrixgenerierungsprogramm** erhält die Anzahl der Zeilen und Spalten als Parameter. Sie können die Matrix entweder direkt in eine Datei schreiben lassen oder die Matrix auf der Standardausgabe ausgeben lassen, um sie anschließend in eine Datei umzuleiten.
Beispiel: `matrixgen 1000 1000 > mymatrix`
Es sollen sowohl positive als auch negative Zahlen generiert werden.

Das **Matrixmultiplikations- bzw. Matrixadditionsprogramm** soll folgende Anforderungen erfüllen:

- Die Operation (Multiplikation, Addition), die Dateinamen der Eingabedateien und der Dateiname der Ausgabedatei sind als Kommandozeilenparameter zu übergeben.
- Die Ein- und Ausgabedateien sind Textdateien im oben beschriebenen Format.
- Es sollen keine festen Grenzen für die Größe der Matrizen vorgegeben werden. Dementsprechend muss der Speicher für die Matrizen dynamisch allokiert werden.
- Während des Einlesens der Eingabedateien soll das Programm die Eingabedaten und die Kompatibilität der zu multiplizierenden Matrizen überprüfen und die Bearbeitung gegebenenfalls mit einer Fehlermeldung abbrechen.

Aufgabe 9 – Experimente mit GNU gprof (2 Punkte)
Mit Hilfe eines Profilers lässt sich das Laufzeitverhalten eines Programms analysieren. So ist es insbesondere möglich festzustellen, wie viel Zeit die Programmausführung in den einzelnen Programmteilen verbringt oder wie häufig einzelne Funktionen im Rahmen einer Programmausführung aufgerufen werden. Das Programm ***GNU gprof*** *ist ein solcher Profiler. Das Ziel dieser Aufgabe besteht darin, Sie ein wenig mit der Funktionsweise von gprof vertraut zu machen und die Funktionsweise anhand des Programms zur Matrixmultiplikation und -addition auszuprobieren.*

a) Informieren Sie sich über die grundsätzliche Funktionsweise und Verwendung von GNU gprof anhand der offiziellen Dokumentation:

   - http://sourceware.org/binutils/docs/gprof/

b) Analysieren Sie das Laufzeitverhalten Ihres Programms zur Addition und Multiplikation von Matrizen aus Aufgabe 7.

   - Welche Einzelschritte sind für diese Analyse erforderlich?
   - Wie teilt sich die Zeit, die für die Ausführung des Programms benötigt wird, auf die einzelnen Teile Ihrer Implementierung auf?
   - Vergleichen Sie die Ergebnisse für die Addition und Multiplikation. Welche Unterschiede fallen auf?

Dokumentieren Sie Ihr Vorgehen und Ihre Ergebnisse (inklusive der relevanten Teile der Ausgabe von gprof) in einem Versuchsprotokoll.

Aufgabe 10 – Netzwerkprogrammierung Quickies (2 Punkte)

a. Was ist ein Client(-Programm), was ist ein Server(-Programm)? Fordert ein Server üblicherweise Dienste von einem Client an?

b. Welche Informationen werden von einem Prozess verwendet, um einen Prozess auf einem anderen Host zu identifizieren?

c. Erklären Sie, aus welchen Gründen sich ein Anwendungsentwickler dafür entscheiden könnte, eine Anwendung über UDP statt über TCP auszuführen.