Fachbereich Mathematik
Prof. Dr. Martin
Dipl.-Math. Andrea Peter
Dipl.-Math. Christine Schönberger

TECHNISCHE UNIVERSITÄT DARMSTADT

SS 2008
16./17.04.08

1. Tutoriumsblatt zur „Algorithmischen Diskreten Mathematik“

Übung am Computer

In dieser Praxisübung am Computer wollen wir Ihnen einen Einstieg in die Programmiersprache `C` geben. Wir gehen hier einen gewissen Spagat ein, da die Vorkenntnisse sehr unterschiedlich sind. Daher umfasst dieses Blatt sowohl einige Basics für Einsteiger in Linux, als auch erste Programmieraufgaben in `C`. Wenn Sie sich schon etwas mit Linux auskennen, können Sie die ersten Aufgaben schnell überfliegen.

Wir werden hier nur einige Basics zu Linux erwähnen, die Sie benötigen, um die übrigen Aufgaben sinnvoll bearbeiten zu können. Weitere Informationen, Befehle, Hinweise finden Sie z.B. im ersten Kapitel des Skriptes zur Computerorientierten Mathematik, dass Sie unter “Material” von unserer Webseite herunterladen können.

Aufgabe C1 (Einloggen)
Loggen Sie sich am Rechner ein. Im unteren Bereich des Bildschirm sollten Sie nun eine Menüleiste sehen, und oben links eine Liste virtueller Bildschirme. Außerdem sollte sich ein Terminalfenster geöffnet haben. Falls sich kein Fenster geöffnet hat, öffnen Sie nun eins, indem sie auf den entsprechenden Button in der Menüleiste klicken.

Einige Basics

1. Verzeichnisse erstellen und löschen
   - Mit `mkdir Verzeichnisname` können Sie ein leeres Verzeichnis erstellen (“make directory”).
   - Mit `rmdir Verzeichnisname` können Sie ein leeres Verzeichnis löschen (“remove directory”).
2. Arbeitsverzeichnisse wechseln
   - Mit `cd Verzeichnisname` wechseln Sie in das nächsttiefere Verzeichnis `Verzeichnisname` (“change directory”).
   - Mit `cd` gelangen Sie ins home-Verzeichnis des Benutzers.
   - Mit `cd ..` wechseln Sie ins nächsthöhere Verzeichnis.
3. Inhalte von Verzeichnissen anzeigen
   - `ls` (“list”) wird der Inhalt des Verzeichnisses aufgelistet.
4. Dateien
   - `touch Dateiname` erzeugt eine leere Datei namens `Dateiname`.
   - Mit `rm Dateiname` wird die Datei namens `Dateiname` gelöscht.
   - Mit `cp Dateiname1 Dateiname2` wird die erste Datei auf die zweite kopiert. Dabei können die Dateinamen auch von der Form `Verzeichnisname/Dateiname` sein.

   VORSICHT: Mit `rm` wird die Datei unwiderruflich gelöscht und kann nicht wie vielleicht von Windows gewöhnt einfach wieder aus einem Papierkorb gefischt werden!
5. Hilfe

   Um herauszufinden was Programme eigentlich tun, kann man unter Linux die sogenannten “Manual-” und “Info-Seiten” nutzen.
   - Durch Eingabe von `man Programmname` werden die Manualseiten zum Programm `Programmname` im Terminalfenster geöffnet.
   - Durch Eingabe von `info Programmname` entsprechend die Infoseiten.

   Durch diese kann man dann mit Hilfe der ↑, ↓ Tasten und der Bild ↑ und Bild ↓ Tasten navigieren, beenden kann man sie mit `q`.
   - Weiß man noch gar nicht, welches Programm man benötigt, kann man `apropos` benutzen. Genaueres z.B. unter `man apropos` ;-)

Aufgabe C2 (Verzeichnisse)
In dieser Aufgabe soll ein Verzeichnisbaum angelegt werden. Sollten Sie sich Ihren Praktikumsaccount mit anderen Studenten teilen, dann sollte zunächst jeder dieser Studenten im Heimatverzeichnis des Praktikumsaccounts ein Verzeichnis mit seinem Namen anlegen und in dieses wechseln.

(a) Erstellen Sie die Verzeichnisse `Tut01` und `Data`.

(b) Erstellen Sie unter dem neuen Verzeichnis `Tut01` ein weiteres Verzeichnis namens `Heute` und wechseln Sie in dieses.

(c) Erstellen Sie weitere Verzeichnisse mit Unterverzeichnissen. Welche dieser neu erstellten Verzeichnisse können Sie mit Hilfe von `rmdir` löschen? Welche nicht und warum nicht?

Aufgabe C3 (Dateien)

(a) Wechseln Sie wieder in das Verzeichnis `Tut01`.

(b) Erzeugen Sie Dateien `hello.c` und `daten.dat`.

(c) Kopieren Sie die Datei `daten.dat` in den Ordner `Data`.

(d) Lassen Sie sich den Inhalt des Verzeichnisses `Tut01` anzeigen.

(e) Finden Sie heraus, wie man sich mit `ls` z.B. auch den Autor der Datei anzeigen lassen kann.

Ran ans Programmieren!

In diesem zweiten Teil starten wir nun unsere ersten C-Versuche. Dabei wollen wir wie folgt vorgehen: Wir lesen abwechselnd Teile des zweiten Kapitel des Skripts zur Computer-orientierten Mathematik und machen Übungen dazu. Bitte beachten Sie, dass wir in dieser Veranstaltung nur ein kleine Grundlage zum Programmieren geben können, Programmieren aber lernt man wie den Marathonlauf nur durch andauerendes Trainieren. Bitte verstehen Sie diese Übung als Hilfe zur Selbsthilfe (Trainingsplan). Der Marathontrainer steht hinter Ihnen.

Aufgabe C4 (Hello World!)

(a) Öffnen Sie die Datei `hello.c` in einem Editor Ihrer Wahl. (Sie können dazu z.B. `gedit hello.c &` eingeben.)

(b) Implementieren Sie das „Hello World!"-Programm aus dem Skript, übersetzen/kompilieren Sie es und lassen Sie es laufen.

(c) Ändern Sie das Programm so ab, dass es jetzt

```
Hello!
This is my first program in C.
```

ausgibt.

Aufgabe C5 (Fibonacci)

(a) Implementieren, übersetzen und testen Sie das Fibonacci-Programm aus dem Skript. Haben Sie eine (intuitive) Ahnung, was die einzelnen Zeilen des Codes bedeuten?

(b) Ändern Sie das Programm so ab, dass alle Fibonacci-Zahlen bis 100 ausgegeben werden und außerdem am Ende die Anzahl der ausgegebenen Fibonacci-Zahlen auf dem Bildschirm erscheint.

(c) Ändern Sie das Programm so ab, dass in der `while`-Schleife nicht nur `sum` ausgegeben wird, sondern stattdessen ein Text der Bauart:

```
low = 0, high = 1, sum = 1.
```

Rufen Sie zu diesem Zweck die Funktion `printf()` nach Möglichkeit nur einmal für jede ausgegebene Zeile auf. Hilfe finden Sie mit `man 3 printf` (die „3" ist wichtig!).

Lesen Sie im Skript den Unterabschnitt zu Variablen und Variablentypen. Der Rechner arbeitet im Grunde nur mit Nullen und Einsen. Daher wollen wir uns die Binärdarstellung von ganzen Zahlen ein bisschen genauer anschauen.

Aufgabe C6 (Binäre Zahlen)

Wir betrachten positive ganze Zahlen im Dualsystem (d.h. binäre Zahlen), die genau acht Stellen haben (möglicherweise mit führenden Nullen). Dabei bezeichnen wir die einzelnen Stellen als Bits und numerieren diese von rechts nach links durch:

Auf diesen Zahlen sei die normale (binäre) Addition mit der folgenden Ausnahme definiert:

$$11111111 + 00000001 \quad = \quad 00000000 \tag{1}$$

(Die Operation in (1) sorgt dafür, dass niemals Zahlen mit mehr als acht binären Stellen auftreten und ist genau so auch in Computern implementiert.) Subtraktion von solchen Zahlen betrachten wir zunächst nicht.

(a) Überlegen Sie sich ein Verfahren, wie man eine ganze Zahl zwischen 0 und 255 (inklusive) in eine Binärzahl umwandeln kann.

(b) Wie lautet die binäre Darstellung von 16, 32, 127 und 213?

Aufgabe C7 (Überlauf)

Wieso führt das folgende Code-Fragment zu einer Endlosschleife?

```
short i;
unsigned short j = 40000;

for (i = 0; i < j; i++) {
  printf("%d\n", i);
}
```

Aufgabe C8 (Division)

Was gibt das folgende Programm aus? Hätten Sie diese Antwort erwartet? Wie kann man dies ändern?

```
#include <stdio.h>

float answer;

int main (void)
{
    answer = 1/3;
    printf("The answer is %f.\n", answer);

    return 0;
}
```

Aufgabe C9 (Fehlersuche)

Finden Sie alle Fehler im folgenden Programm.

```
#include <stdio.h>

/* In diesem Programm stecken einige Fehler */
/* Finden Sie diese

int summe;
int zahl_1;
int 2.zahl;

/* Nun die Hauptfunktion */
int main (void)
{
    summe = zahl_1 + 2.zahl;
    printf("Die Summe von %d und %d ist %f.\n", zahl_1, 2.zahl, summe);

    return 0;
}
```

Suchen Sie diese erst auf dem Papier. Versuchen Sie anschließend das Programm zu kompilieren und es anhand der Fehlermeldungen zu debuggen.