Übungen zu Computergrundlagen WS 2016/2017

Übungsblatt 13: Programmieren in C II

3. Februar 2017

Allgemeine Hinweise

- Abgabetermin für die Lösungen ist **Freitag, 10.02.2017, 11:00 Uhr**.
- Schickt die Lösungen bitte per Email an Euren Tutor:
  - Montag 11:30 – 13:00 Uhr: Julian Michalowsky (`jmichalowsky@icp.uni-stuttgart.de`)
  - Montag 14:00 – 15:30 Uhr: Frank Uhlig (`fuhlig@icp.uni-stuttgart.de`)
  - Dienstag 14:00 – 15:30 Uhr: Patrick Kreissl (`pkreissl@icp.uni-stuttgart.de`)
  - Dienstag 15:45 – 17:15 Uhr: Kai Szuttor (`kai@icp.uni-stuttgart.de`)
  - Donnerstag 09:45 – 11:15 Uhr: Frank Maier (`fmaier@icp.uni-stuttgart.de`)
  - Donnerstag 15:45 – 17:15 Uhr: Evangelos Tzaras (`etzaras@icp.uni-stuttgart.de`)

Aufgabe 13.1: Wörter zählen in C (10 Punkte)

Ziel dieses Übungsblattes ist es, das Pythonskript `/group/cgl/2016/13/occurrence.py` in C zu implementieren. Das Pythonskript zählt die Häufigkeit der verschiedenen Wörter in einem Text und gibt diese aus. Schaut Euch das Skript zunächst an und versucht zu verstehen, wie es funktioniert. Überlegt Euch vorallem, welche Features von Python benutzt werden, die es in C so nicht gibt und die daher von Euch nachgebaut werden müssen!

Hinweise:

- Zum Testen des Programmes könnt Ihr die Dateien `gpl-3.0.txt` (General Public License) und `mobydick.txt` (Moby Dick) im Verzeichnis `/group/cgl/2016/13` verwenden.
- Dokumentation der meisten C-Bibliotheksfunktionen bekommt man über den `man`-Befehl. So zeigt der Befehl `man getline` die Hilfe für die C-Funktion `getline` an.
- Zum Kompilieren Eures C-Programms verwendet wieder am besten den Befehl

```
gcc -std=gnu99 -O3 -o occurrence occurrence.c
```

  da die `getline`-Funktion eine POSIX-Erweiterung ist.

Teilaufgaben

- **13.1.1** Schreibt ein C-Programm, das eine Datei zeilenweise einliest und den Inhalt auf dem Bildschirm ausgibt. Der Dateiname soll auf der Kommandozeile übergeben werden können. (2 Punkte)

  Hinweise:

  - Verwendet den Befehl `getline`.
  - Lest den Abschnitt "Example" in der *Linux*-Manpage von `getline`. Er zeigt ein Beispielprogramm. Unter Mac OS X ist die Manpage knapper und das Beispiel etwas komplizierter.

- **13.1.2** Erweitert das Programm aus der vorigen Aufgabe so, dass es die einzelnen Zeilen mit Hilfe der Funktion `strtok` in einzelne Wörter zerlegt. Verwendet als *delim* dafür die Zeichenkette `delimiters` aus dem Pythonskript. (1 Punkt)
- **13.1.3** Erweitert das Programm aus der vorigen Aufgabe so, dass es die einzelnen Wörter mit Hilfe der Funktion `tolower` vollständig in Kleinbuchstaben umwandelt. (2 Punkte)
- **13.1.4** Erweitert das Programm aus der vorigen Aufgabe so, dass es die verschiedenen Wörter zählt und das Ergebnis ausgibt. (4 Punkte)

  Hinweise:

  - Erzeugt jeweils für die Wörter und deren Anzahl Arrays, die entsprechend das Wort und die dazugehörige Anzahl speichern. Das n-te Wort gehört dann zum n-ten Eintrag in der Liste mit den Häufigkeiten.
  - Speichert die Wörter als Array von Strings, also Zeigern auf Zeichen.
  - Wenn ein Wort im Text gefunden wird, dann muss das Programm zunächst schauen, ob das Wort bereits in der Liste ist, und wenn ja, an welcher Position. Wenn das der Fall ist, muss der entsprechende Zähler erhöht werden.
  - Wenn das Wort nicht existiert, müssen die beiden Arrays erweitert werden. Der neue String wird an die Liste der Wörter angehängt und der neue Zähler mit 1 initialisiert, da wir ja bereits ein Aufkommen dieses Worts beobachtet haben.
  - Caveat Emptor! Wenn das Wort an die Liste der Wörter angehängt wird, muss die Funktion `strdup` verwendet werden, sonst sind die Ergebnisse vermutlich etwas seltsam. Überlegt Euch, warum das so ist, und besprecht Eure Überlegungen mit Eurem Tutor. Wann kann die gedankenlose Verwendung von `strdup` zum Problem werden? (*Keine* Aufgabe zum Abgeben!)
- **13.1.5** Messt die Laufzeit des Pythonskripts und des C-Programmes anhand von Moby Dick. Welches davon ist schneller? Warum? (1 Punkt)