Institut für Informatik
Prof. Dr. Stefan Conrad

Sommersemester 2021
Übung 2

Hardwarenahe Programmierung
Gruppe 17 (Florian)

In dieser Übung liegt der Fokus auf dem Umgang mit Strings in C-Programmen.

Wichtig:

- *Denken Sie daran, Strings zu terminieren und Variablen zu initialisieren!*
- *Denken Sie daran, ihre Abgabe hochzuladen und die Tests im ILIAS zu absolvieren!*
- *Achten Sie darauf, dass Sie Ihre Lösung im ILIAS auch bei der richtigen Übung abgeben, also zum Beispiel nicht die Lösungen dieses Übungsblattes bei Übung 1 hochladen.*
- *Achten Sie darauf, dass Sie nicht die Abgabe für ein vorheriges Übungsblatt hochladen!*
- *Laden Sie ihre Abgabe aus dem ILIAS runter, um sich zu vergewissern, dass Sie beim zippen keine Dateien vergessen haben.*

Aufgabe 1 *Strings verbinden (**Pflichtaufgabe**)*

(a) Schreiben Sie eine Funktion `strings_verbinden`, welche drei Strings entgegen nimmt und diese miteinander verkettet in einem Ausgabestring speichert. Zum Beispiel würden die Strings „E“, „n“ und „te“ zu „Ente“ verbunden werden.

Implementieren Sie die Funktion in einer Datei mit Namen `strings_verbinden.c`. Wählen Sie eine der beiden folgenden Funktionssignaturen:

```
void strings_verbinden(char zusammen[], char str1[], char str2[], char str3[])
void strings_verbinden(char *zusammen, char *str1, char *str2, char *str3)
```

Sie dürfen hierbei annehmen, dass das `char`-Array mit Namen `zusammen`, in dem Sie den String speichern, genau groß genug ist.

WICHTIG: Bei dieser Aufgabe darf neben der Funktion `strlen` keine andere Bibliotheksfunktion verwendet werden. Bei anderen Aufgaben sind dann wieder alle Funktionen aus dem C99-Standard erlaubt, solange nichts anderes angegeben wurde.

(b) Verwenden Sie den Kommandozeilenbefehl `make run`, der die Tests baut, kompiliert und ausführt. Wenn die Tests durchlaufen heißt dies allerdings noch nicht, dass Ihr Code auch fehlerfrei ist. Achten Sie deshalb wie immer darauf, dass Sie nicht aus uninitialisierten Speicherbereichen lesen und auch sonst keine Speicherzugriffsfehler begehen.

Aufgabe 2 *Groß- und Kleinbuchstaben (**Pflichtaufgabe**)*

Schreiben Sie ein Programm, dass einen String zu `snake_case` übersetzt. Genauer sollen Großbuchstaben durch einen Unterstrich gefolgt vom entsprechenden Kleinbuchstaben ersetzt werden. Zeichen außer A–Z sollen nicht modifiziert werden.

(a) Implementieren Sie eine Funktion `void zu_snake_case(char *string)`, die einen String als Parameter übergeben kriegt und diesen entsprechend modifiziert. Zum Beispiel würde der String „`eineVariable`“ zu „`eine_variable`“ übersetzt werden.

Tipp: Da der String hierbei länger wird bietet es sich an, rückwärts zu arbeiten, um den Eingabestring nicht zu früh zu überschreiben.

Nehmen Sie an, dass exakt ausreichend Speicher für den Ausgabestring vorhanden ist.

WICHTIG: Bei dieser Aufgabe darf die Funktion `tolower` nicht verwendet werden. Sehen Sie sich eine ASCII-Tabelle an, um zu verstehen, wie man diese Funktionalität selber implementieren kann. Bei anderen Aufgaben sind dann wieder alle Funktionen aus dem C99-Standard erlaubt, solange nichts anderes angegeben wurde.

(b) Wir haben Ihnen Unit-Tests und eine Makefile-Datei zur Verfügung gestellt. Sie können die Tests mit dem Befehl `make run` bauen und ausführen. Testen Sie Ihre Funktion mit den bereitgestellten Unit-Tests und stellen Sie sicher, dass alle erfüllt sind. Schauen Sie sich außerdem den Aufbau des Makefiles an, damit Sie in Zukunft selbst Makefiles erstellen können, um Ihren Kompilier- und Testprozess zu vereinfachen.

Mit `make snake_case` wird das Programm `snake_case` erzeugt, womit Sie ihre Funktion auf der Kommandozeile ausprobieren können, wenn Sie möchten.

Aufgabe 3 *URLs finden (**Pflichtaufgabe**)*

Gegeben ist ein Text, in dem bis zu 10 URLs enthalten sind. In dieser Aufgabe sollen diese URLs gefunden und in ein Array geschrieben werden. Der Einfachheit halber wird angenommen, dass URLs entweder mit `https://`, `http://` oder `www.` beginnen und aus den Zeichen a – z, A – Z, 0 – 9 und `!#$%&'()*+,-./:;=?@[]~_` bestehen. Weiterhin passen alle URLs in das dafür vorgesehene Array.

Zum Beispiel enthält folgender Text die drei URLs *https://www.example.com*, *http://example.com* und *www.example.com*:

```
Eine URL: https://www.example.com
Noch eine URL:http://example.com
Dritte URL: {www.example.com}
```

- Implementieren Sie die Funktion `urls_finden`, die die Strings aus `text` in das Array `urls` schreibt und die Anzahl der gefundenen URLs zurück gibt.

  ```
  int urls_finden(char urls[10][150], char *text);
  ```

- Das Array `urls` ist nicht initialisiert; es kann also irgendwelche Werte beinhalten.
- Führen Sie die Unit-Tests mit dem Befehl `make run` aus.