Universität Konstanz
Informatik & Informationswissenschaft
Prof. Dr. Marcel Waldvogel / Klaus Herberth

Programmierkurs 3
WS 2016/2017

6. Übungsblatt

(Ausgabe: 1. Dezember 2016 — Abgabe bis: 8. Dezember 2016, 4:00)

Aufgabe 1: Namen (0 Punkte)

Schreiben Sie ihren Namen auf alle Abgaben und benennen Sie die Dateien wie angegeben.

Aufgabe 2: Scope (10 Punkte)

Schauen Sie sich die Datei `scope.c` an und beantworten Sie innerhalb des Dokuments folgende Fragen:

1. Welche Bezeichner sind deklariert und wie ist ihr scope?
2. Was sind die Vor- und Nachteile, Variablen so spät wie möglich zu definieren?
3. Was ist an diesem Beispiel falsch?

Aufgabe 3: Alles steht Kopf (20 Punkte)

Schreiben Sie ein Programm (`kopf.c`) das seine gesamte Eingabe char-weise in umgekehrter Reihenfolge ausgibt (dies zerstört unter anderem Unicode-Text — macht nichts). Steht am Ende der Eingabe ein Zeilenumbruch, so soll dieses eine Zeichen nicht an den Anfang der Ausgabe verschoben werden, sondern am Ende stehen bleiben.

```
$ echo Hallo | ./a.out
ollaH
```

Allozieren Sie zunächst nur wenig Speicher (z.B. 1 KB) und verdoppeln Sie ihn wenn er gefüllt ist. Das Programm sollte mit "beliebig" großen Dateien (d.h., begrenzt nur durch den verfügbaren Hauptspeicher) funktionieren, die beliebige Daten (auch Binärdaten) enthalten können. Falls nicht genügend Speicher alloziert werden kann, muss das Programm mit einer Felermeldung und Rückgabewert 1 (z.B. `return 1` in `main`, oder `err (3)`) abbrechen.

Aufgabe 4: Slab Allocator (30 Punkte)

Ihrem Arbeitgeber war langweilig und er hat versucht, einen eigenen Slab Allocator zu schreiben. Leider ist er nicht sehr weit gekommen und nun liegt es an ihnen, die fehlenden Teile zu implementieren. In `slab.h` hat er schon alle benötigten Funktionen und Strukturen deklariert die Sie verwenden sollen, auch steht ihnen schon ein entsprechendes Testprogramm (`main.c`) und einige Funktionsrümpfe (`slab.c`) zur Verfügung. Da ihr Arbeitgeber mehrere Programmieren angestellt hat und Ihnen auch ihre verdiente Entlohnung zukommen lassen möchte, schreiben Sie bitte Ihren Namen an den Anfang des Quellcodes. Verändern Sie unter keinen Umständen das Testprogramm, da ihre Arbeit hiermit kontrolliert werden soll.

```
$ pkgcc main.c slab.c -o slab
$ ./slab lin < main.c
enqueue(#include "slab.h" )
enqueue(#include <stdio.h> )
enqueue(#include <string.h>)
dequeue(#include <string.h>)
dequeue(#include <stdio.h> )
...
Completed!


Dump follows
- maxElems=80 usedElems=64 elemSize=32
Blocks:
- Block with 10 elements
- Block with 10 elements
- Block with 10 elements
- Block with 10 elements
- Block with 10 elements
- Block with 10 elements
- Block with 10 elements
- Block with 10 elements
16 free members found

64 texts found:
NULL static size_t expon
---- entialGrowth(slabHe
---- ader *sh)
---- exponentially by %z
...
```

Oben sehen Sie die entsprechenden Anweisungen um das Programm zu kompilieren und auch welche Ausgabe Sie ungefähr zu erwarten haben. Bitte beachten Sie, dass `...` keine wirkliche Ausgabe darstellt, sondern der Abkürzung dient. `pkgcc` steht in diesem Beispiel für gcc mit all seinen in der Vorlesung gezeigten Flags.

Bonus (3 Punkte): Nutzen Sie die Funktionen `lock` und `unlock` um Ihren Code thread safe zu machen.