1 Allgemeine Hinweise zu den BS-Übungen

- Die Aufgaben sind *in Dreiergruppen* zu bearbeiten (Aufgabe 0 in Ausnahmefällen noch allein oder zu zweit). Der Lösungsweg und die Programmierung sind gemeinsam zu erarbeiten.
- Die Gruppenmitglieder sollten gemeinsam an der gleichen Tafelübung teilnehmen. Die Lösung wird jeweils komplett bewertet und den Gruppenmitgliedern gleichermaßen angerechnet.
- Die Übungsaufgaben müssen abhängig von der Tafelübung entweder bis zum Donnerstag bevor das nächste Blatt erscheint (Übungsgruppen in ungeraden Kalenderwochen) oder Dienstag nachdem das nächste Blatt erschienen ist (Übungsgruppen in geraden Kalenderwochen) jeweils bis 10 Uhr abgegeben werden. In darauffolgenden Tafelübungen werden teilweise einzelne abgegebene Lösungen besprochen, teilweise auch eine Musterlösung.
- Die abgegebenen Antworten/Programme werden automatisch auf Ähnlichkeit mit anderen Abgaben überprüft. Wer beim Abschreiben[1] erwischt wird, verliert ohne weitere Vorwarnung die Möglichkeit zum Erwerb der Studienleistung in diesem Semester!
- Die optionalen Aufgaben (davon wird es jeweils eine auf den Aufgabenblättern 0–3 geben) sind ein Stück schwerer als die „normalen“ und geben *keine* zusätzlichen Punkte für das jeweilige Aufgabenblatt – aber jeweils ein „Bonus-Sternchen“ ★. Wenn ihr drei Sternchen sammelt, müsst ihr das letzte Aufgabenblatt (A4) nicht bearbeiten!
- Die Aufgaben sind über AsSESS (`https://ess.cs.tu-dortmund.de/ASSESS/`) abzugeben. Dort gibt *ein* Gruppenmitglied die erforderlichen Dateien ab und nennt dabei die anderen beteiligten Gruppenmitglieder (Matrikelnummer, Vor- und Nachname erforderlich!). Namen und Anzahl der abzugebenden C-Quellcodedateien variieren und stehen in der jeweiligen Aufgabenstellung; Theoriefragen sind grundsätzlich in der Datei `antworten.txt`[2] zu beantworten. Bis zum Abgabetermin kann eine Aufgabe beliebig oft abgegeben werden – es gilt die letzte, vor dem Abgabetermin vorgenommene Abgabe.
- Sobald eine Abgabe von den Betreuern korrigiert wurde, kann die korrigierte Lösung ebenfalls im AsSESS eingesehen werden.

Aufgabe 0: Erste Schritte in C (10 Punkte)

Das Lernziel dieser Aufgabe ist der Umgang mit der UNIX-Systemumgebung und dem C-Compiler. Darüber hinaus sollt ihr euch durch das Schreiben eines einfachen C-Programms mit dieser Programmiersprache vertraut machen.

Theoriefragen: Systemumgebung (5 Punkte)

Macht euch zunächst mit der Systemumgebung – im IRB-Pool (am besten in der Rechnerübung!), in der auf der Veranstaltungswebseite zur Verfügung gestellten virtuellen Maschine oder in einer eigenen Linux-Installation zu Hause – vertraut. Öffnet ein Terminal-Fenster und experimentiert mit den in der Tafelübung vorgestellten UNIX-Kommandos.

1. Erklärt in eigenen Worten, welche Rolle die Umgebungsvariable `PATH` spielt.
2. Erklärt den Unterschied zwischen den Befehlen `rm(1)` und `rmdir(1)`.
3. Inwieweit unterscheiden sich die Signale `SIGKILL` und `SIGTERM`?

[1] Da wir im Regelfall nicht unterscheiden können, wer von wem abgeschrieben hat, gilt das für Original **und** Plagiat.
[2] reine Textdatei, codiert in ISO-8859-15 oder UTF-8

4. Macht euch mit dem UNIX-Kommando **man(1)** vertraut. Wie kann man eine Handbuchseite als HTML-Dokument anzeigen?
5. Gebt eine Möglichkeit an, um auf der UNIX-Konsole die aktuelle Wochennummer anzuzeigen.

Theoriefragen (Fortsetzung): Variablen in C (2 Punkte)

6. Betrachtet das folgende C-Programm:

```c
#include <stdio.h>
#include <math.h>

const double luftreibung = 0.47; /* Koeffizient einer Kugel */
const double g = 9.81;
double luftdichte;

double endgeschwindigkeit(double masse, double flaeche) {
        /* Berechne die maximale Fallgeschwindigkeit eines Objektes */
        return sqrt((2 * masse * g) / (luftdichte * flaeche * luftreibung));
}

int main(void) {
        double v; /* Endgeschwindigkeit */
        luftdichte = 1.225; /* Luft bei 15 Grad */
        v = endgeschwindigkeit(5.0, 0.5);
        printf("%lf\n", v); /* Ausgabe */
        return 0;
}
```

   - In welchen Bereichen des Speicherlayouts (Segmente) befinden sich die Funktion `endgeschwindigkeit()`, die Variablen `luftreibung`, `luftdichte`, `g` und `v`, sowie die Parameter der Funktion `endgeschwindigkeit()`? Warum befinden sich die Variablen `luftreibung` und `luftdichte` in unterschiedlichen Segmenten?
   - **Tipp:** Wollt ihr obigen Code compilieren, müsst ihr noch das Argument „`-lm`“ (der erste Buchstabe ist ein kleines L, kein großes i) anfügen, damit gegen die math-Bibliothek gelinkt wird. Das Compilieren ist jedoch für diese Aufgabe nicht notwendig.

Programmierung in C (3 Punkte)

7. Wir wollen an dieser Stelle eine reduziertere Version des UNIX-Standardprogramms **id** implementieren.

   Anstelle einer vollständigen Ausgabe soll es genügen, dass die Benutzer- und Gruppen-ID ausgegeben wird. Ein Beispielaufruf sähe folgendermaßen aus:

```
titan@Unidesk:~$ ./id
user: 1000
group: 100
```

   Schaut euch dafür die Funktionen **getuid(2)** und **getgid(2)** an.

   Die Implementierung soll in der Datei `id.c` abgegeben werden.

★ Programmieren in C - Extended Edition!

Erweitert eure **id**-Implementierung so weit, dass es möglich ist, einen Nutzernamen als Argument zu übergeben. Falls ein Nutzername übergeben wird, soll nicht die Benutzer- und Gruppen-ID des

aufrufenden Benutzers ausgegeben werden, sondern die des übergebenen Nutzers. An dieser Stelle müsst ihr keine Fehlerbehandlung implementieren, falls der Benutzer nicht existiert.

Schaut euch für die Bearbeitung dieser Aufgabe **`getpwnam(3)`** an.

Die veränderte Version soll als `id_extended.c` abgegeben werden.

Beispiel für Programmaufrufe:

```
titan@Unidesk:~$ ./id
user: 1000
group: 100
titan@Unidesk:~$ ./id root
user: 0
group: 0
```

Tipps zu den Programmieraufgaben:

- Kommentiert euren Quellcode ausführlich, so dass wir auch bei Programmierfehlern im Zweifelsfall noch Punkte vergeben können!
- Die Programme sollen sich mit dem gcc auf den Linux-Rechnern im IRB-Pool übersetzen lassen. Der Compiler ist dazu mit folgenden Parametern aufzurufen:
  `gcc -std=c11 -Wall -o id id.c`
  Alternativ könnt ihr die Programme auch in C++ schreiben, der Compiler ist dazu mit folgenden Parametern aufzurufen:
  `g++ -Wall -o id id.c`
  Weitere (nicht zwingend zu verwendende) Compilerflags, die dafür sorgen, dass man sich näher an die Standards hält, sind: `-Wpedantic -Werror`

Abgabe: bis spätestens Donnerstag den 04. Mai 10:00 (Übung in ungerader Kalenderwoche), bzw. Dienstag den 09. Mai 10:00 (Übung in gerader Kalenderwoche)