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 (Gruppen 1, 3, 5, ...) oder Dienstag nachdem das nächste Blatt erschienen ist (Gruppen 2, 4, 6, ...) jeweils bis 12 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 Aufgaben sind über AsSESS (`http://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 von 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)

Lernziel dieser Aufgabe ist der Umgang mit der UNIX-Systemumgebung und dem C-Compiler. Darüber hinaus sollen mit dem Schreiben eines einfachen C-Programms erste Schritte in dieser Programmiersprache getan werden.

Theoriefragen: Systemumgebung (2 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, experimentiert mit den in der Tafelübung vorgestellten UNIX-Kommandos.

1. Wozu dienen die UNIX-Kommandos *chown*, *chgrp* und *chmod*? Mit welchem Kommando kann man die Datei *beispiel.txt* der Gruppe *studis* zuordnen?
2. Mit welchem Kommando kann ein Ordner */home/studi/beispiel1/* mitsamt seiner Inhalte in den Ordner */home/studi/beispiel2/unterordner/* kopiert werden?

(Fortsetzung der Theoriefragen nach der Programmieraufgabe)

[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

Programmierung in C (4 Punkte)

3.
- Erstellt ein Programm, das die Anzahl der Schaltjahre nach dem gregorianischen Kalender in einem gegebenen Zeitraum bestimmen soll. Das Programm soll mit einem Aufruf von startjahr=1853, endjahr=2013 folgendes ausgeben:

  "Im Zeitraum von 1853 bis 2013 gab es 39 Schaltjahre"

- Implementiert dazu folgende Funktion:

```
void schaltjahre(int startjahr, int endjahr)
```

- Die Implementierung soll in der Datei `schaltjahre.c` abgegeben werden.
- Bei fehlerhaften Eingaben soll eine Fehlermeldung ausgegeben werden.
- Die Bedingungen für die Bestimmung von Schaltjahren nach dem gregorianischen Kalender sind in der folgenden Abbildung dargestellt:

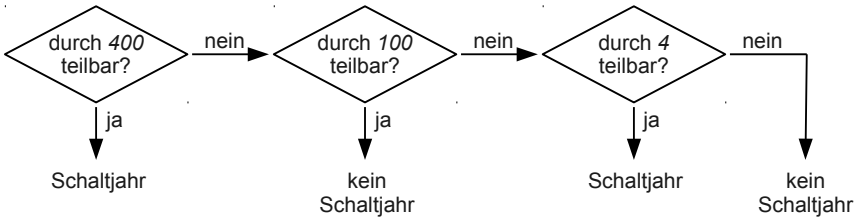

Abbildung 1: Berechnung der Schaltjahre

- Beachtet ebenfalls, dass die heutige Schalttagsregelung und damit der gregorianische Kalender erst seit 1582 existieren. Vor der Einführung des gregorianischen Kalenders vorhandene Schaltjahre sollt ihr nicht berücksichtigen.

- Legt darüber hinaus eine Reihe von unterschiedlichen Variablen (verschiedene Typen; global, lokal; initialisiert, uninitialisiert, statisch) an und lasst deren Adresse im Hauptspeicher ausgeben.

Theoriefragen (Fortsetzung): Variablen in C (4 Punkte)

4. Welchen Abstand (in Bytes) haben die Adressen zweier nacheinander in `main()` angelegter lokaler Variablen vom Typ char? Welche der beiden Adressen ist größer? Begründet Eure Antwort!
5. Warum ist eine globale Integer-Variable an einer völlig anderen Adresse?

6. Betrachtet das folgende C-Programm:

```c
#include <stdio.h>

const double PI = 3.14159;
int a;

double umfang(unsigned int radius) {
        return 2 * PI * radius;
}

int main(void) {
        double u = umfang(3);
        return 0;
}
```

   - In welchen Bereichen des Speicherlayouts (Segmente) befinden sich die Funktion *umfang*(), die Variablen *PI*, *a* und *u* sowie der Parameter *radius*?

Tipps zu den Programmieraufgaben:

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

Abgabe: bis spätestens Donnerstag, 02. Mai 12:00 (Übungsgruppen 1./3./5./...) bzw. Dienstag, 07. Mai 12:00 (Übungsgruppen 2./4./6./...).