UNIVERSITÄT OSNABRÜCK

ESS

marcel.koeppen [☺] uni-osnabrueck.de
mluetkedreim [☺] uni-osnabrueck.de

Übung zur Vorlesung
Betriebssysteme
Wintersemester 2021/2022

Übungsblatt 1: Erste Schritte in C

➪ **Abgabe der Lösungen bis Montag, 25. Oktober 14:00 im AsSESS**

Organisatorisches

- Die Aufgaben sind *in Dreiergruppen* zu bearbeiten. Der Lösungsweg und die Programmierung sind gemeinsam zu erarbeiten.
- Es wird ein wöchentliches Übungsblatt (Bearbeitungszeit: eine Woche) und parallel dazu etwa zweiwöchentlich ein Programmierblatt (Bearbeitungszeit: zwei Wochen) geben.
- Die Übungsaufgaben müssen jeweils montags bis 14 Uhr abgegeben werden. In den darauffolgenden Tafelübungen werden die Lösungen der Aufgaben besprochen.
- 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 Prüfungszulassung in diesem Semester!
- Die Aufgaben sind über AsSESS (`https://ess.cs.uos.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[2] variieren und stehen in der jeweiligen Aufgabenstellung; Antworten auf Theorie- bzw. Textfragen sind grundsätzlich gesammelt in der Datei `antworten.txt`[3] zu beantworten. Bis zum Abgabetermin kann eine Aufgabe beliebig oft abgegeben werden – es gilt die letzte vor der Abgabefrist vorgenommene Abgabe. Dabei müssen immer *alle* Dateien hochgeladen werden.
- Sobald eine Abgabe von den Betreuern korrigiert wurde, kann die korrigierte Lösung ebenfalls im AsSESS eingesehen werden.

Aufgabe 1: Systemumgebung (1+1+1+2 = 5 Punkte)

1. Erklären Sie kurz, wozu das Programm **man** gut ist. Was ist der Unterschied zwischen den Aufrufen `man printf` und `man 3 printf`?
2. Mit welchem Aufruf des Systemprogramms **chmod (1)** lassen sich für eine Datei geheim.txt im lokalen Verzeichnis die Zugriffsrechte so einstellen, dass der Besitzer die Datei nur lesen und schreiben, die Gruppe die Datei nur lesen und andere Benutzer die Datei weder lesen, schreiben, noch ausführen dürfen?
3. Geben Sie einen Aufruf an, mit dem die C-Quellcodedatei `programm.c` mit dem GNU C-Compiler in die ausführbare Datei `programm` übersetzt werden kann.
4. Ausgehend vom Home-Verzeichnis sollen rekursiv alle C-Quellcode-Dateien – also Dateien mit den Endungen „.c“ und „.C“ – gesucht werden, die innerhalb der letzten 10 Minuten geändert wurden. Geben Sie einen Befehl an, mit dem die entsprechenden Dateien aufgelistet werden.

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

Aufgabe 2: Variablen in C (1+2 = 3 Punkte)

1. Gegeben sei folgender Programmcode:

```c
#include <stdio.h>

struct dice {
    int num_faces;
};

//Transformiert den Würfel in einen d20
void d20(struct dice d) {
    d.num_faces = 20;
}

int main(void) {
    struct dice d;
    d.num_faces = 6;
    printf("Hallo vom %d-seitigen Würfel!\n", d.num_faces);
    d20(d);
    printf("Hallo vom %d-seitigen Würfel!\n", d.num_faces);
}
```

   Welche Ausgabe hat das Programm, wenn es ausgeführt wird?
   Erfüllt die Funktion `d20(...)` ihren Zweck? Falls nicht, erklären Sie den Fehler.

2. Gegeben sei folgender Programmcode:

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

bool flag = true;
int ergebnis;

int main(void) {
    int i;
    int j = 100;

    for (i = 0; i < j; i++) {
        if (flag) {
            j = j - 2;
        } else {
            j++;
        }

        flag = !flag;
    }
    ergebnis = j;
    printf("Das Ergebnis ist: %d\n", ergebnis);
}
```

   Geben Sie an, in welchen Speichersegmenten die Variablen **`flag`**, **`ergebnis`**, **`i`** und **`j`** während der Ausführung abgelegt sind.
   Wieso sind die Variablen **`flag`** und **`ergebnis`** nicht im selben Segment?

Aufgabe 3: Programmieren in C (5 Punkte)

Schreiben Sie eine Funktion mit der Signatur `int euclidean_gcd(int a, int b)`, welche mithilfe des euklidischen Algorithmus den größten gemeinsamen Teiler der Zahlen `a` und `b` berechnet und

diesen zurückgibt. Beachten Sie dabei, dass die Parameter auch 0 sein können.

Zur Erinnerung: Der euklidische Algorithmus führt in jedem Schritt eine Division mit Rest aus, beginnend mit den Zahlen `a` und `b`. In jedem weiteren Schritt wird die Division mit dem Divisor und dem Rest des vorherigen Schritts ausgeführt. Wenn bei einer Division ein Rest von 0 entsteht, ist der entsprechende Divisor der größte gemeinsame Teiler.

Geben Sie Ihren Quellcode der Funktion `euclidean_gcd` (bitte ohne main-Funktion) in der Datei **euclid.c** ab.

Aufgabe 4: Ein kleines Systemprogramm (3 Punkte)

Mit dem Systemprogramm **id (1)** kann man sich in der Shell ausgeben lassen, welche Benutzer- und Gruppen-IDs ein bestimmter Benutzer hat. In dieser Aufgabe soll eine vereinfachte Version dieses Programms erstellt werden, das die Benutzer-ID und primäre Gruppen–ID des aufrufenden Benutzers ausgibt.

Für einen Benutzer mit der Benutzer-ID 1234 und Gruppen-ID 4711 soll die Ausgabe folgendermaßen aussehen:

```
Benutzer-ID: 1234
Gruppen-ID: 4711
```

Zum Ermitteln der Benutzer- und Gruppen-ID können Sie die Funktionen **getuid (2)** und **getgid (2)** verwenden. Denken Sie daran, die für die benutzten System- und Bibliotheksfunktionen benötigten Include-Dateien einzubinden.

Geben Sie Ihre Lösung in der Datei **miniid.c** ab.