1 Intro: Variablen, einfache Datenypen, Arithmetik

Aufgabe 1.1. (Dokumentation – leicht) Das wichtigste Werkzeug eines jeden angehenden Programmierers ist die Systemdokumentation. Das System vor dem Sie sitzen ist mit einem umfassenden Online-Handbuch ausgestattet, dass sämtliche Funktionen und Programme dokumentiert. Sie können das Handbuch mit dem Programm **man**(1) aufrufen. Tippen Sie

`man` *`Abschnitt Seite`*

um das Handbuch zu *Seite* in *Abschnitt* aufzurufen. Gibt es in allen Abschnitten des Handbuches nur eine Seite zu diesem Begriff, so können Sie *Abschnitt* weglassen. Wir schreiben im Weiteren **printf**(3) für den Begriff **printf** im Abschnitt 3.

Ziel dieser Aufgabe ist es, dass Sie sich mit dem Handbuch vertraut machen. Im weiteren Verlauf des Kurses werden wir annehmen, dass sie in der Lage sind, Einzelheiten zu Bibliotheksfunktionen selber nachzuschlagen. Folgen Sie dazu folgenden Leseaufträgen:

- Überfliegen Sie die Handbuchseite zu **man**(1) und merken Sie sich, was für Inhalte in welchen Abschnitten stehen. In welchen Abschnitten sind Befehle dokumentiert? In welchen Abschnitten Funktionen?
- Zu jedem Abschnitt hat das Handbuch eine Einführungsseite. Lesen Sie zumindest **intro**(3), die Einführung der C-Standardbibliothek.
- Die Handbuchseite **gcc**(1) dokumentiert den C-Kompiler `gcc`. Finden Sie heraus, was die Optionen `-E` und `-S` tun. Damit Sie nicht die gesamte Seite lesen müssen ist es hier sinnvoll, die Suchfunktion des Handbuchs zu verwenden. Tippen Sie `h`, um die Hilfe aufzurufen, hier wird Ihnen erklärt, wie man in Handbuchseiten sucht.
- Sollte Ihnen der Name eines Programmes oder einer Funktion entfallen sein, so können Sie mit **apropos**(1) nach Seiten mit passender Beschreibung suchen. Finden Sie so eine Funktion, die ein Zeichen in einen Großbuchstaben konvertiert.

Aufgabe 1.2. (Schaltjahrberechnung – leicht) Ein Schaltjahr ist ein Jahr, welches durch 400 teilbar ist, oder zwar durch 4 aber nicht durch 100 teilbar ist. Dies entspricht der Formel

$$L(Y) = 4 \mid Y \wedge 100 \nmid Y \vee 400 \mid Y.$$

Schreiben Sie ein Programm, dass sein erste Argument ausliest und berechnet ob es sich um ein Schaltjahr handelt. Kommt Ihr Programm mit der falschen Anzahl an Argumenten zurecht? Wie ist es mit Eingaben, die keine Zahlen sind?

Codebeispiel 1: Beispielausgabe

```
$ ./leap 2008
The year 2008 is a leap year.
$ ./leap 2100
The year 2100 is not a leap year.
$ ./leap 2000
The year 2000 is a leap year.
```

Aufgabe 1.3. (Ostertagberechnung – leicht) Implementieren Sie den Algorithmus von Gauß zur Bestimmung des Ostertages [1] in einem gegebenen Jahr. Nehmen Sie die Jahreszahl als Argument entgegen.

Codebeispiel 2: Beispielausgabe

```
$ ./easter 2024
The easter date of the year 2024 is the 31. of March
$ ./easter 2018
The easter date of the year 2018 is the 1. of April
```

Aufgabe 1.4. (Funktionstafel – leicht) Schreiben Sie ein Programm, dass für die Zahlen $x = 1, 2, \ldots, 100$ die Werte der Ausdrücke x^2, 2^x, und $x!$ bestimmt und tabellarisch ausgibt. Versuchen Sie, die Ausgabe ansprechend zu gestalten, in dem Sie die Zahlen rechts ausgerichtet übereinander schreiben. Lesen Sie ggf. **printf**(3) um zu verstehen, wie man solche Ausrichtung durchführen kann

Vergleichen Sie die Ausgabe Ihres Programmes mit den erwarteten Werten. Sind sie identisch? Falls nicht, warum?

Aufgabe 1.5. (wc(1) – mittel) Schreiben Sie einen einfachen Klon des klassischen UNIX-Programms **wc**(1). Das Programm **wc**(1) ließt die Eingabe und zählt, wie viele Zeichen, Wörter, und Zeilen in ihr vorkommen. Denken Sie darüber nach, wie Sie Wortgrenzen und Zeilenenden erkennen können. Geben Sie am Ende die Anzahlen mit **printf**(3) aus.

Rufen Sie **getchar**(3) in einer Schleife wie folgt auf, um die Eingabe zu lesen. Sie können Ihr Programm dann folgendermaßen aufrufen:

```
./mein_wc <datei.txt
```

```c
    int c;

    /* lese ein Zeichen in c, und solange c nicht EOF
     * (End Of File, Ende der Eingabe) ist, ... */
    while (c = getchar(), c != EOF) {
        /* tue irgendwas mit c */
    }

    /* ist die Schleife beendet, wurde das Eingabeende erreicht */
```

Aufgabe 1.6. (Code verstehen – mittel) Gegeben sei folgender C-Code:

Codebeispiel 3: Mysteriöses Programm

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

int main(int argc, char *argv[])
{
    int a = 0;
    for (int i = 1; i < argc; i++) {
        int b = atoi(argv[i]);
        a += b;
        printf("%d: %d (%d)\n", i, b, a);
    }
    printf("%d\n", a);

    int b = 0;
    if (argc > 1) { b = a/(argc-1); }
    printf("%d\n", b);
}
```

Sind die Aufrufe „legal“? Was wird ausgegeben bei folgenden Eingaben (nicht ausprobieren) und was steht in den Variablen a, b und c zu welchem Zeitpunkt?

1. `$ ./prog abc` (hierzu s. [2, S. 307])
2. `$ ./prog`
3. `$ ./prog 0`
4. `$ ./prog 1 2 3`
5. `$ ./prog 1 2 4`

Aufgabe 1.7. (C-Code in Assembler auf Papier kompilieren – schwer) Gegeben sei folgender C-Code:

Codebeispiel 4: Ausgabe von Konsolenargument

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
    for (int i = 0; i < argc; i++) {
        printf("%d: %s\n", i+1, argv[i]);
    }
}
```

Übersetzen Sie den Code manuell in (in etwa) äquivalenten Assembler-Code (NASM).

Literatur

[1] Wikipedia contributors. *Gaußsche Osterformel: Eine ergänzte Osterformel*. 2018. URL: `https://de.wikipedia.org/wiki/Gau%C3%9Fsche_Osterformel#Eine_erg%C3%A4nzte_Osterformel`.

[2] Open-Std WG14. *C99 Standard*. C99 + TC1-3. N1256. International Standard, Committe Draft. Version C99. ISO/IEC. 7. Sep. 2007. URL: `http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1256.pdf`.