1 Intro

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. (echo(1) – leicht) Implementieren Sie das klassische Unix-Programm **echo**(1). Das Programm **echo**(1) gibt seine Kommandozeilenargumente mit Leerzeichen dazwischen aus. Tippen Sie auf der Konsole `man 1 echo`, um die Dokumentation des Programms zu lesen.

Aufgabe 1.3. (Fakultät – leicht) Schreiben Sie ein Programm, dass zu einer auf der Kommandozeile übergebenen Zahl die Fakultät berechnet. Erinnern Sie sich, dass

$$n! = 1 \cdot 2 \cdot \ldots \cdot (n-1) \cdot n$$

und versuchen Sie $n!$ mit Hilfe einer Schleife zu berechnen. Verwenden Sie folgenden Programmrahmen.

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[])
4 {
5     int n, n_fakultaet;
6
```

```
7     /* haben wir genug Argumente? */
8     if (argc != 2) {
9         printf("Aufruf: %s n\n", argv[0]);
10        /* bei main: 0 ist Erfolg, alles andere Fehlschlag */
11        return (1);
12    }
13
14    /* lese n aus Argumenten */
15    n = atoi(argv[1]);
16
17    /* hier Code zur Berechnung von n_fakultaet einfuegen */
18
19    printf("n! = %d\n", n_fakultaet);
20
21    return (0);
22 }
```

Aufgabe 1.4. (Funktionstafel – mittel) 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
```

```
1     int c;
2
3     /* lese ein Zeichen in c, und solange c nicht EOF
4      * (End Of File, Ende der Eingabe) ist, ... */
5     while (c = getchar(), c != EOF) {
6         /* tue irgendwas mit c */
7     }
8
9     /* ist die Schleife beendet, wurde das Eingabeende erreicht */
```