Crashkurs

Compilieren auf der Console

Zu erst öffnet man das home-Verzeichnis (Orte => Persönlicher Ordner) und legt ein neues Verzeichnis an (Datei => Ordner anlegen). Unter Linux muss man auf die Groß- und Kleinschreibung achten. Leerzeichen in Namen machen auf der Console Probleme. Es ist daher sinnvoll Datei- und Verzeichnisnamen immer klein zu schreiben und ohne Leerzeichen.

Anschließend öffnet man einen Texteditor (Anwendungen => Zubehör => Texteditor) und speichert die Datei im angelegten Verzeichnis ab (Datei => Speichern unter). Für C Dateien hängt man `.c` am Ende des Namens als Dateierweiterung an, für C++ `.cpp` und für Java `.java`. Ohne das weiß der Editor nicht welche Sprache er highlighten soll. Bei Java sollte man darauf achten, dass der Name der Hauptklasse gleichzeitig der Name der Datei ist. Wenn die Hauptklasse (also die, die die Main Methode enthält) zum Beispiel `Foo` heißt, so muss die Datei `Foo.java` heißen. Die Groß- und Kleinschreibung muss man beachten. Leerzeichen sind verboten.

Zum Compilieren speichert man zuerst seinen Quellcode ab und öffnet dann eine Console (Anwendungen => Systemwerkzeuge => Terminal). In der Console sollte etwas von der Art

```
name@rechner(~)$
```

zu sehen sein. Vor dem `$` Zeichen sieht man den aktuellen Pfad in runden Klammern. `~` bezeichnet das Home-Verzeichnis. Mit

```
cd foo
```

wechselt man in das Unterverzeichnis foo des aktuellen Pfads. (Man kann auch einen absoluten Pfad angeben, hier sollte man aber beachten dass / als Trennzeichen benutzt wird.) Zum Compilieren wechselt man in das Verzeichnis wo die Quelldateien sind.

Um C Programme zu übersetzen gibt man

```
gcc -Wall -O2 foo.c
```

ein. C++ Programme werden sehr ähnlich übersetzt.

```
g++ -Wall -O2 foo.cpp
```

In beiden Fällen wird eine ausführbare Datei namens a.out erzeugt. -Wall sorgt dafür, dass Warnungen ausgegeben werden und -O2 dafür, dass der Compiler den Code optimiert. (Achtung: Das O ist ein großes O wie in Otto.) Zum Ausführen gibt man

```
./a.out
```

ein. Man kann die Ein- oder Ausgabe auch umlenken.

```
./a.out < in.txt > out.txt
```

Hier ließt das Programm nun die Eingabe aus der Datei in.txt und gibt diese in die Datei out.txt aus. Man muss nicht immer Ein- und Ausgabe umlenken. Alles was nicht umgelenkt ist, ist an die Console gebunden.

```
./a.out < in.txt
./a.out > out.txt
```

Durch benutzen der Pfeiltasten kann man Kommandos wiederholen.
Java Programme werden wie folgt compiliert.

```
javac Foo.java
```

Zum Ausführen gibt man

```
java Foo
```

ein. Man beachte, dass man hier den Namen der Hauptklasse angibt und nicht wie beim Compilieren den Namen der Quelldatei. Auch bei Java kann man die Ein- und Ausgabe umlenken.

```
java Foo < in.txt
java Foo > out.txt
java Foo < in.txt > out.txt
```

Mit && kann man aus 2 Befehlen einen machen. Dies kann nützlich sein, um sicher zu stellen, dass die ausgeführte Datei immer der aktuellsten Version des Sourcecodes entspricht. Zum Beispiel:

```
g++ -Wall -O2 foo.cpp && ./a.out < in.txt
```

C IO

Die interessanten Funktionen sind printf und scanf aus dem Header <stdio.h>.

```
#include <stdio.h>
int main(){
        int foo;
        if(scanf("%d", &foo) != EOF)
                printf("Ausgabe: %d\n", foo);
        else
                printf("Keine Zahl in der Eingabe\n");
}
```

C++ IO

In C++ kann man die C Funktionen verwenden oder man benutzt cout und cin aus <iostream>.

```
#include <iostream>
using namespace std;

int main(){
        int foo;
        if(cin >> foo)
                cout << "Ausgabe: " << foo << endl;
        else
                cout << "Keine Zahl in der Eingabe" << endl;
}
```

Java IO

Bei Java kann man die Scanner-Klasse verwenden.

```
import java.util.Scanner;

class Main {
        public static void main(String[]args){
                Scanner s = new Scanner(System.in);
                int foo;
                if(s.hasNextInt()){
                        foo = s.nextInt();
                        System.out.print("Ausgabe: ");
                        System.out.println(foo);
                }else
                        System.out.println("Keine Zahl in der Eingabe");
        }
}
```