# ÃIJbung 1

Rasmus Diederichsen

October 13, 2014

## 1

### Aufgabe 1.2

```c
#include <stdio.h>
/** Print shit to console
 * @return 0 on success
 */
int main(void)
{
   printf("Hello World!\n");
   return 0;
}
```

### Aufgabe 1.3

```
# Wall: show most warnigns
# Wstrict-prototypes: warn when function prototypes or definitions omit argument
# types
# ansi: compile according to ANSI standard (c89), accept only code compliant
# with that standard. Enable trigraphs! :)
# -pedantic: show all warnings according to strict ISO C/C++
hello: hello.c
        gcc -Wall -Wstrict-prototypes -ansi -pedantic hello.c -o hello
clean:
        rm hello
test:
        ./hello
```

### Aufgabe 1.4

- Standardmäßig liegen Bibliotheken in `/usr[/local]/lib/` während sich die Header in `/usr[/local]/include/` befinden
- Der Präprozessor durchsucht standardmäßig alle Pfade, die im `C_INCLUDE_PATH` stehen, neben den obigen Standardverzeichnissen und einigen anderen.
- Für Bibliotheken ist die Variable `LIBRARY_PATH` nützlich

- Um Umgebungsvariablen persisent zu setzen, bietet sich eine Anweisung der Form `export $Variable = Wert` an, die in eine der Dateien geschrieben wird, die die Shell beim Start einliest (etwa `.profile` oder `.bash_login` oder `.bashrc` für GNU Bash).

- `-I/dir` weist den Präprozessor an, *dir* ebenfalls nach Headern zu durchsuchen, `-L` hat dieselbe Funktion für Libraries. `-lfoo` Weist den Linker an, gegen die Bibliothek `foo` bzw. das Archiv `libfoo.a` zu linken.

- Die Syntax mit spitzen Klammern weist den Präprozessor an, nach der Header-Datei in den oben angegebenen Standardverzeichnissen sowie den durch die `-I`-Option spezifizierten zu suchen. Diese Syntax ist für System-Header gedacht.

  Schreibt man stattdessen Anführungszeichen, weist man auf selbstgeschriebene Header hin und es wird zunächst im Arbeitsverzeichnis, dann in den durch die `-iquote`-Option spezifizierten Verzeichnissen und zuletzt ebenfalls in den Standardverzeichnissen gesucht.

- Beim statischen Linken werden alle Bibliotheksfunktionen direkt in das entstehende Programm integriert, sodass diese nicht zwangsläufig auf dem System vorhanden sein müssen, auf dem es ausgeführt wird. Beim dynamischen Linken wird eine Bibliothek nicht direkt in die ausführbare Datei gesteckt, sondern diese bedient sich zur Laufzeit aus ihrem Code. Dies erfordert, dass die Bibliothek (`.so`, `.dll`, `.dylib`) auf dem System vorhanden ist. Man kann Einfluss auf das Linken nehmen, indem man selbst angibt, ob gegen dynamische Libraries gelinkt werden soll. `ld` akzeptiert hierzu die Flags `-bstatic` und `-bdynamic` bzw. systemabhängige Varianten. Hierbei müssen die benötigten Libraries dann als Archive vorliegen, falls das Kompilieren gelingen soll.

- Mit `ldd` oder `otool` kann man herausfinden, welche dynamischen Bibliotheken von einer Datei (ausführbar, object file, `.a` oder `.so`) benötigt werden.

- `nm` zeigt die Symboltabelle der Bibliothek an.

- Man kann mit `dlopen` zur Laufzeit Bibliotheken laden. Wenn die Bibliothek auf dem eigenen System weder statisch noch dynamisch vorhanden ist, muss man vermutlich diesen Weg wählen.

- Bibliotheken stehen unter bestimmten Lizenzen, die beeinflussen, ob und wie man sie im eigenen Programm benutzen kann, will man dieses verteilen.

- Ein Paketmanager übernimmt die Aufgabe, Programme und Programmteile für ein System zu kompilieren, sodass der Benutzer nicht selbst dafür sorgen muss, dass alle Dependencies vorhanden sind und die Software richtig installiert wird. Man kann prinzipiell die gesamte Arbeit von Hand machen, wenn man eine hohe Schmerztoleranz hat.