Kapitel 5

Einführung

Die Programmiersprache C ist eine der am häufigsten verwendeten Programmiersprachen in Wissenschaft und Technik. Sie ist sehr viel näher an der Maschine (dem Computer) angesiedelt als zum Beispiel MATLAB.

C ist eine Compile–Sprache. Das Übersetzen der Programme besteht aus zwei Komponenten, dem eigentlichen Compilieren und dem Linken. Der Compiler erzeugt aus dem Quelltext einen für den Rechner lesbaren Objektcode. Der Linker erstellt das ausführbare Programm, indem es in die vom Compiler erzeugte Objektdatei Funktionen (siehe auch Kapitel **??**) aus Bibliotheken (Libraries) einbindet. Der Begriff Compiler wird häufig auch als Synonym für das gesamte Entwicklungssystem (Compiler, Linker, Bibliotheken) verwendet.

Diese Vorgehensweise ist in Gegensatz zu MATLAB, wo die Befehle während der Laufzeit eingelesen und dann abgearbeitet werden. Die Herangehensweise von MATLAB ist deutlich langsamer.

Für die Bearbeitung der Übungsaufgaben werden Editor und C–Compiler bereitgestellt. Es handelt sich dabei um frei erhältliche (kostenlose) Programme (`gcc`), die auf LINUX-Betriebssystemen arbeiten. Es handelt sich bei C um eine weitgehend standardisierte Programmiersprache. Jedoch ist C–Compiler nicht gleich C–Compiler. Die Vergangenheit hat gezeigt, dass nicht alle Programme unter verschiedenen Compilern lauffähig sind. Wer mit einem anderen als mit dem `gcc`–Compiler arbeitet, hat unter Umständen mit Schwierigkeiten bei der Kompatibilität zu rechnen.

Dieser Teil der Vorlesung basiert auf der Vorlesung mit dem gleichen Titel aus dem Wintersemester 2005/2006, gehalten von Erik Wallacher.

5.1 Das erste C–Programm

Traditionell besteht das erste C–Programm darin, dass die Meldung *Hallo Welt* auf dem Bildschirm ausgegeben werden soll.
Quelltext: (HalloWelt.c):

```
/* HalloWelt.c */
#include <stdio.h>

int main()
{
  printf("Hallo␣Welt␣\n"); /* "\n new line */
  return 0;
}
```

Folgende Strukturen finden wir in diesem ersten einfachen Programm vor:

- *Kommentare*
  werden mit /* eingeleitet und mit */ beendet. Sie können sich über mehrere Zeilen erstrecken und werden vom Compiler (genauer vom Präprozessor) entfernt.
- *Präprozessordirektiven*
  werden mit # eingeleitet. Sie werden vom Präprozessor ausgewertet. Die Direktive `#include` bedeutet, dass die nachfolgende Headerdatei einzufügen ist. Headerdateien haben die Dateinamenendung (Suffix) .h. Die hier einzufügende Datei stdio.h enthält die benötigten Informationen zur standardmäßigen Ein- und Ausgabe von Daten (standard input/output).
- Das *Schlüsselwort* main markiert den Beginn des Hauptprogramms, d.h. den Punkt, an dem die Ausführung der Anweisungen beginnt. Auf die Bedeutung des Schlüsselwortes void und der Klammer () wird später detaillierter eingegangen.
- Syntaktisch (und inhaltlich) zusammengehörende Anweisungen werden in Blöcken zusammengefasst. Dies geschieht durch die Einschließung eines Blocks in geschweifte Klammern:

```
{
  erste Anweisung
  ...
  letzte Anweisung
}
```

- Die erste Anweisung, die wir hier kennenlernen, ist *printf()*. Sie ist eine in stdio.h deklarierte Funktion, die Zeichenketten (Strings) auf dem Standardausgabegerät (Bildschirm) ausgibt. Die auszugebende Zeichenkette wird in Anführungsstriche gesetzt.
  Zusätzlich wird eine Escapesequenz angefügt: \n bedeutet, dass nach der Ausgabe des Textes *Hallo Welt* eine neue Zeile begonnen wird. *Anweisungen innerhalb eines Blocks werden mit Semikolon ; abgeschlossen.*

Der übliche Suffix für C-Quelldateien ist .c und wir nehmen an, dass der obige Code in der Datei hallo.c abgespeichert ist.
Die einfachste Form des Übersetzungsvorgangs ist die Verwendung des folgenden Befehls in der Kommandozeile:

```
gcc HalloWelt.c
```

gcc ist der Programmname des GNU-C-Compilers. Der Aufruf des Befehls erzeugt die ausführbare Datei a.out (a.exe unter Windows). Nach Eingabe von

`./a.out` (bzw. ./a.exe)

wird das Programm gestartet. Auf dem Bildschirm erscheint die Ausgabe "Hallo Welt". (Das Voranstellen von ./ kann weggelassen werden, falls sich das Arbeitsverzeichnis ./ im Suchpfad befindet. Durch Eingabe von

```
export PATH=$PATH:.
```

wird das Arbeitsverzeichnis in den Suchpfad aufgenommen). Einen anderen Namen für die ausführbare Datei erhält man mit der Option `-o`

```
gcc HalloWelt.c -o HalloWelt
```

5.2 Interne Details beim Compilieren (∗)

Der leicht geänderte Aufruf zum Compilieren

```
gcc -v HalloWelt.c
```

erzeugt eine längere Bilschirmausgabe, welche mehrere Phasen des Compilierens anzeigt. Im folgenden einige Tipps, wie man sich diese einzelnen Phasen anschauen kann, um den Ablauf besser zu verstehen:

a) Präprozessing:
Headerfiles (*.h) werden zur Quelldatei hinzugefügt (+ Makrodefinitionen, bedingte Compilierung)

```
gcc -E HalloWelt.c > HalloWelt.E
```

Der Zusatz > HalloWelt.E lenkt die Bildschirmausgabe in die Datei HalloWelt.E. Diese Datei HalloWelt.E kann mit einem Editor angesehen werden und ist eine lange C-Quelltextdatei.

b) Übersetzen in Assemblercode:
Hier wird eine Quelltextdatei in der (prozessorspezifischen) Programmiersprache Assembler erzeugt.

```
gcc -S HalloWelt.c
```

Die entstandene Datei HalloWelt.s kann mit dem Editor angesehen werden.

c) Objektcode erzeugen:
Nunmehr wird eine Datei erzeugt, welche die direkten Steuerbefehle, d.h. Zahlen, für den Prozessor beinhaltet.

```
gcc -c HalloWelt.c
```

Die Ansicht dieser Datei mit einem normalen Texteditor liefert eine unverständliche Zeichenfolge. Einblicke in die Struktur vom Objektcodedateien können mit Hilfe eines Monitors (auch ein Editor Programm) erfolgen.

```
hexedit HalloWelt.o
```

(Nur falls das Programm hexedit oder ein anderer Monitor installiert ist.)

d) Linken:
Verbinden aller Objektdateien und notwendigen Bibliotheken zum ausführbaren Programm *Dateiname.out* (*Dateiname.exe* unter Windows).

```
gcc -o Dateiname HalloWelt.c
```