

Programmieren in C

Grundlagen

thoto

/dev/tal e.V.

7. März 2013

Agenda für Heute

- 1 Einführung
 - Organisatorisches
 - Was ist C und was ist Programmierung?
- 2 Das erste Programm
 - Vor dem Programmieren
 - Wie funktioniert das?
 - Hello World
 - Bessere Hello World ...

Organisatorisches

Wo bin ich hier?

- /dev/tal e.V., Mirker Bahnhof

Wo bin ich hier?

- /dev/tal e.V., Mirker Bahnhof
- Kurs: jeden Samstag um 1400

Wo bin ich hier?

- /dev/tal e.V., Mirker Bahnhof
- Kurs: jeden Samstag um 1400
- Alle sind hier freiwillig!
- regelmäßiges Erscheinen hilft Zusammenhänge zu verstehen!

Wer ist der da?

- thoto (Thorsten M.), Student
- /dev/tal-Gründungsmitglied

Wer ist der da?

- thoto (Thorsten M.), Student
- /dev/tal-Gründungsmitglied

eMail thoto@devtal.de

Wer ist der da?

- thoto (Thorsten M.), Student
- /dev/tal-Gründungsmitglied

eMail thoto@devtal.de

Jabber thoto@devtal.de / [thotowpt@jabber.ccc.de](jabber:thotowpt@ccc.de)

Und was passiert mit uns?

- Einsteigerkurs
- Programmieranfänger und Umsteiger
- Arbeit mit grundlegenden Werkzeugen
- keine „Profi-“Werkzeuge

Und was passiert mit uns?

- Einsteigerkurs
 - Programmieranfänger und Umsteiger
 - Arbeit mit grundlegenden Werkzeugen
 - keine „Profi-“Werkzeuge
- ⇒ Wissen, was passiert!

Was brauche ich?

- Uralte Möhre

Was brauche ich?

- Uralte Möhre (Computer!)

Was brauche ich?

- Uralte Möhre (Computer!)
 - Bitte: Linux
- ⇒ Rechner stehen zur Verfügung.
- Am besten Debian basiert

Was brauche ich?

- Uralte Möhre (Computer!)
 - Bitte: Linux
- ⇒ Rechner stehen zur Verfügung.
- Am besten Debian basiert
 - Mehr dazu später

Was ist C?

Was ist denn eigentlich Programmierung?

- Programme sind Kochrezepte
- Bleiben im Speicher des Rechners
- Laufen „von oben nach unten“ ab.

Beispiel:

Programm

Wir brauchen Heizöl, Benzin, eine Weinflasche und Stoff.

Zwei Drittel Heizöl einfüllen, ein drittel Benzin dazu. Mischen.
Stoff dicht reinstopfen.

Bei Anwendung Stoff anzünden und schnell wegwerfen.

Beispiel:

Programm

Deklaration:

Wir brauchen Heizöl, Benzin, eine Weinflasche und Stoff.

Implementierung:

Zwei Drittel Heizöl einfüllen, ein drittel Benzin dazu. Mischen.
Stoff dicht reinstopfen.

Ausführung:

Bei Anwendung Stoff anzünden und schnell wegwerfen.

Beispiel:

Programm

Deklaration:

Wir brauchen Heizöl, Benzin, eine Weinflasche und Stoff.

Typisierung: Heizöl, Benzin sind Flüssigkeiten

Weinflasche ist Flasche und Stoff ist Material

Implementierung:

Zwei Drittel Heizöl einfüllen, ein drittel Benzin dazu. Mischen.

Stoff dicht reinstopfen.

Ausführung:

Bei Anwendung Stoff anzünden und schnell wegwerfen.

Wie funktioniert mein Computer?

- „Von Neumann Maschine“

Wie funktioniert mein Computer?

- „Von Neumann Maschine“
⇒ großer Speicher

Wie funktioniert mein Computer?

- „Von Neumann Maschine“
⇒ großer Speicher
aber verschiedene Geschwindigkeiten

Wie funktioniert mein Computer?

- „Von Neumann Maschine“
 - ⇒ großer Speicher
aber verschiedene Geschwindigkeiten
 - ⇒ Rechen- und Steuerwerk

Okay. Und was ist C?

- Sprache
- „Einfach“
- Grundlage von UNIX ⇒ Einfache Integration
- Systemnah, dennoch bequem

Wo wird C verwendet?

- Mikrocontroller

Wo wird C verwendet?

- Mikrocontroller
- eingebettete Systeme

Wo wird C verwendet?

- Mikrocontroller
- eingebettete Systeme
- Heimrechner

Wo wird C verwendet?

- Mikrocontroller
- eingebettete Systeme
- Heimrechner
- Server

Wo wird C verwendet?

- Mikrocontroller
- eingebettete Systeme
- Heimrechner
- Server
- Großrechner

Wo wird C verwendet?

- Mikrocontroller
- eingebettete Systeme
- Heimrechner
- Server
- Großrechner
- Mainframe

Wo wird C verwendet?

- Mikrocontroller
- eingebettete Systeme
- Heimrechner
- Server
- Großrechner
- Mainframe
- Supercomputer

Warum ausgerechnet C?

- vglw. einfache Sprache
- sehr weit verbreitet
- Allzweck-Sprache
- sehr effizient

Aber C++...?

- C kein Vorgänger sondern Untermenge
- Linux nutzt immernoch C

Ja, und, wie jetzt?

Realisierung, aber erstmal:

Vor dem Programmieren

Okay, lets fetz.

Was brauchen wir?

```
apt-get install gcc make
```

```
apt-get install vim-gtk vim vim-scripts vim-doc
```

```
apt-get install ttf-dejavu exuberant-ctags ctags
```

Und was war das jetzt?

- `gcc` Compiler: Das, was C in ein Programm verwandelt
- `make` Bauwerkzeug
- `vim` Texteditor (dazu später mehr)
- `ctags` Hilfsumgebung (cool!)

Schön und gut ... aber was heißt das?

oder:

Wie funktioniert das?

Wie funktioniert das?

Compiler I

- GCC: GNU C Compiler
- übersetzt den Code in Maschinsprache
- Syntaxfehler: Grammatik falsch

Wie funktioniert das?

Compiler I

- GCC: GNU C Compiler
- übersetzt den Code in Maschinsprache
- Syntaxfehler: Grammatik falsch

Komm mal bei die Oma.

⇒ Prädikat? Subjekt? Objekt?

Compiler II

Komponenten:

- Präprozessor

Compiler II

Komponenten:

- Präprozessor
- Compiler

Wie funktioniert das?

Compiler II

Komponenten:

- Präprozessor
- Compiler
- Linker/Binder

Make

- Baut automatisch Projekt
- Definition der Dateien und Ziele
- besonders bei großen Projekten verwendet

Make

- Baut automatisch Projekt
- Definition der Dateien und Ziele
- besonders bei großen Projekten verwendet
- Verzeichnisorientiert

Make

- Baut automatisch Projekt
- Definition der Dateien und Ziele
- besonders bei großen Projekten verwendet
- Verzeichnisorientiert
- `make <ziel>`

Make

- Baut automatisch Projekt
- Definition der Dateien und Ziele
- besonders bei großen Projekten verwendet
- Verzeichnisorientiert
- `make <ziel>`
- Warum? Einfach in VIM integrierbar

VIM

- komfortabler Texteditor

Wie funktioniert das?

VIM

- komfortabler Texteditor
- Komplette Tastaturorientiert!

Wie funktioniert das?

VIM

- komfortabler Texteditor
- Komplette Tastaturorientiert!
- GVIM auch mit Maus bedienbar

Wie funktioniert das?

VIM

- komfortabler Texteditor
- Komplette Tastaturorientierung!
- GVIM auch mit Maus bedienbar
- Einfache Integration von Make

Wie funktioniert das?

VIM

- komfortabler Texteditor
- Komplette Tastaturorientiert!
- GVIM auch mit Maus bedienbar
- Einfach Integration von Make
- eigentlich fast IDE!

VIM

- komfortabler Texteditor
 - Komplette Tastaturorientiert!
 - GVIM auch mit Maus bedienbar
 - Einfach Integration von Make
 - eigentlich fast IDE!
- ⇒ Sehr bequem!

CTAGS

IDE-like

Und wie geht das jetzt wirklich?

Und wann geht's mit C los?

oder:

Hello World

Hello World

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

```
int main(int argc, char** argv){  
printf("Hello World!\n");  
return 0;  
}
```


Kompilieren auf die manuelle Weise

```
$ gcc -Wall -g -O0 -c helloworld.c  
$ gcc helloworld.o -g -o helloworld
```

Hello World

Ausführung

```
$ ./foo  
Hello World!
```

Makefile

```
CFLAGS = -Wall -O0 -g
```

```
LFLAGS = -g
```

```
all: helloworld
```

```
helloworld.o:
```

```
helloworld: helloworld.o
```

Aufruf des Programmes

```
int main(int argc, char** argv){  
  ...  
  return <irgendwas>;  
}
```

Beim jedem Start d. Programmes:
Ausführung main().

Das Programm an sich

```
printf("Hello World!\n");
```

- printf ist Befehl zum Drucken
- "Hello World!\n" ist Ausgabe
- \n ist Zeilenumbruch (Ausprobieren!)

fortgeschrittenes Hello World!

```
#include<stdio.h>

int main(int argc, char** argv){
char helloworld[]="Hello World\n"; //$
printf(helloworld);
return 0;
}
```

- helloworld: Platzhalter
- char: Typ
- []: ⇒ später

NEU: noch besser!

```
int sprache;  
// sprache=1; //Deutsch  
sprache=2; // Englisch  
  
if(sprache==1){  
printf("Hallo Welt");  
}else{  
printf("Hello World");  
}  
return 0;
```

NEU: noch besser!

- `//`: Kommentar bis Zeilenende
- `int`: Typ – Zahl
- `if`(Bedingung): Wenn Bedingung stimmt:
Ausführung geschweiften Klammern
- `else`: Wenn nicht: Ausführung
geschweiften Klammern (`else`)

NEU: noch besser!

- `//`: Kommentar bis Zeilenende
- `int`: Typ – Zahl
- `if(Bedingung)`: Wenn Bedingung stimmt:
Ausführung geschweiften Klammern
- `else`: Wenn nicht: Ausführung
geschweiften Klammern (`else`)
- *Wichtig hier*: Trennung von Deklaration und Zuweisung!