Freitagsrunde präsentiert: C Kurs – Vortrag 9

Tag 5: Bibliotheken, Häufige Fehler, Literatur

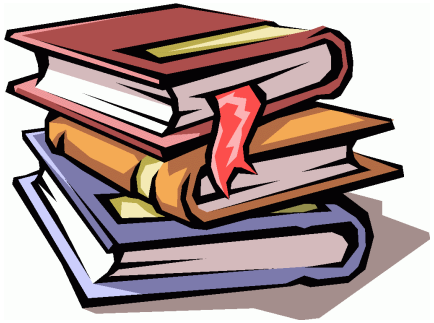
Martin Kresse

21.9.2010

Inhalt

1. Nützliche Bibliotheken

- **Sammlung von Programmfunktionen**
- **Lösung wiederkehrender Probleme**
- **Abstraktion von Low-Level-Hardware- oder OS-Programmierung**
- **Statische oder dynamische Bindung (Linking)**
  - Statisch: Bibliotheksfunktionen wird mit Programmcode zu einer Datei verknüpft
  - Dynamisch: Bibliothek liegt separat vor (z.B. .dll/.so/.dylib-Dateien)
- **Vorteile:**
  - Zeitersparnis
  - Fehlerreduktion
  - Erhöhung der Portabilität
  - Unabhängige Pflege



1.1. Standard C Library

- **Seit 1989 durch ANSI-C Standard definiert**
- **Kein fester Bestandteil der Sprache C**
- **Implementierung meist compilerabhängig**
  - Linux: glibc, uClibc, dietlibc
  - Windows: Microsoft-Visual C++-Laufzeitumgebung (msvcrt.dll)
- **C99 Library umfasst 482 Funktionen (Vergleich: Java SE 6 API hat über 21.000 Funktionen)**
- **Verwendung durch Einbindung der Header-Dateien:**

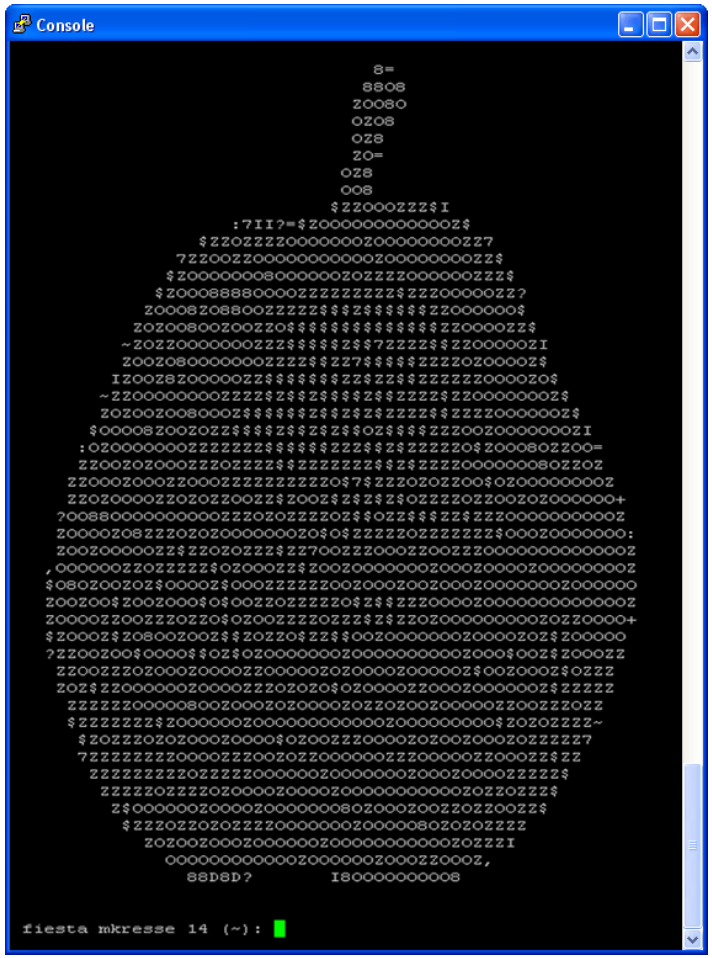
```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define PI 3.14159265
5
6  int main(int argc, char** argv) {
7          printf("sin(30°) = %f", sin(30 * PI/180));
   ...
```

1.1. Standard C Library

- **Inhalte (Auszug):**
  - stdio.h — Ein- und Ausgabe
    `printf()`, `scanf()`, `fgets()`, `fopen()`
  - string.h — Manipulation von Zeichenketten
    `strcpy()`, `strlen()`, `strcmp()`, `memcpy()`
  - stdlib.h — Speicherverwaltung, Zahlenkonvertierung
    `malloc()`, `atoi()`, `rand()`, `qsort()`
  - math.h — Mathematische Funktionen
    `sin()`, `cos()`, `log()`, `pow()`, `sqrt()`
  - stdbool.h (C99) — Definiert den Datentyp `bool`
  - complex.h (C99) — Operationen für Komplexe Zahlen
    `imag()`, `polar()`

→ http://www.cplusplus.com/reference/clibrary

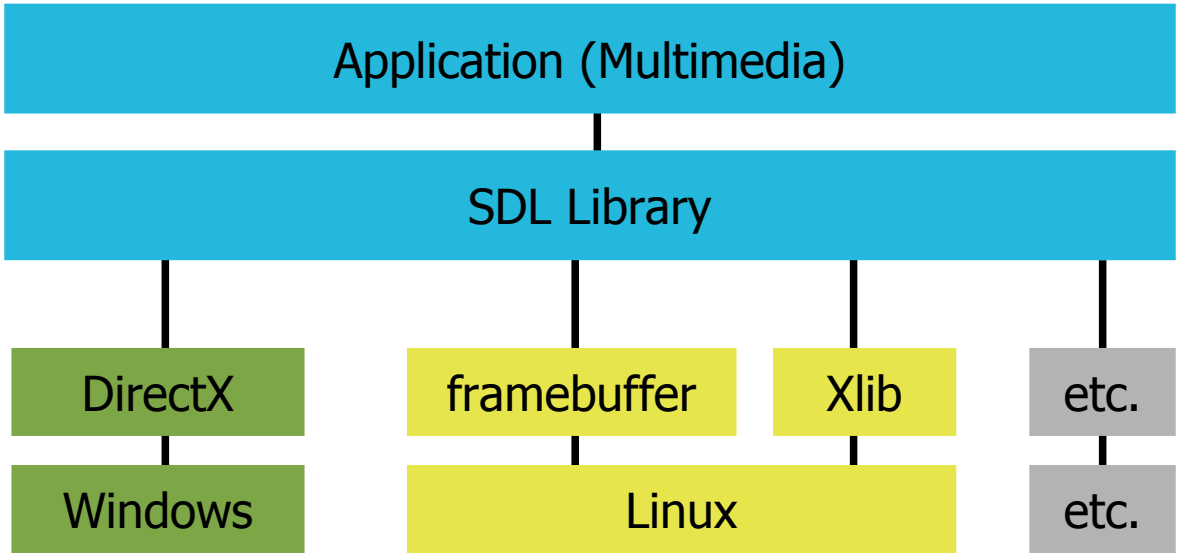
1.2. Simple DirectMedia Layer



1.2. Simple DirectMedia Layer

- Plattformunabhängige Multimediabibliothek (Linux, Windows, MacOS, BSD...)
- Low-Level-Zugriff auf 2D-/3D-Videoschnittstellen, Audiohardware, Eingabegeräte, Multithreading, ...
- Freie Software / Open Source (LGPL)
- Verfügbar unter → http://www.libsdl.org/

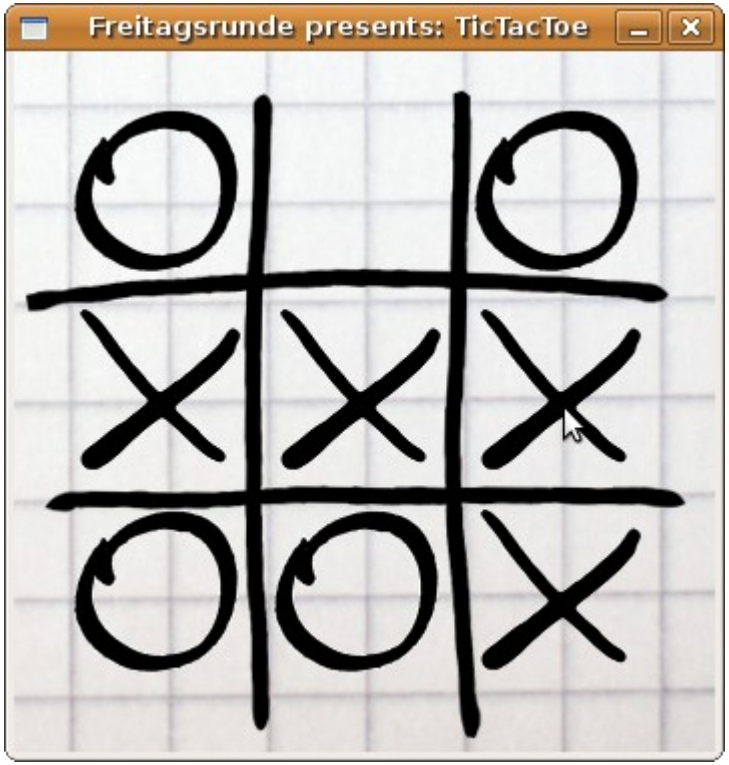
1.2. Simple DirectMedia Layer



- **Dokumentation**
  - API-Dokumentation:
    → http://www.libsdl.org/cgi/docwiki.cgi/SDL_API
  - Umfangreiches Tutorial:
    → http://lazyfoo.net/SDL_tutorials
  - Weitere Links unter:
    → http://www.libsdl.de/docs.htm

1.2. Simple DirectMedia Library

Live-Demo



2. Top 10 – die beliebtesten Fehler

Top 10 – die beliebtesten Fehler

1. Vergleich mit = statt ==

```
1  int antwort = 17;
2
3  if (antwort = 42) {
4      printf("Antwort ist 42.");
5  } else if (antwort = 17) {
6      printf("Antwort ist 17.");
7  }
```

```
Antwort ist 42.
```

- Zuweisung mit anschließendem Vergleich auf ≠ 0
  ⇒ In if-Klauseln kein = verwenden
  oder Yoda-Conditions: `if (42 == antwort)`

Top 10 – die beliebtesten Fehler

2. Nicht initialisierte Variablen / Datenstrukturen

```
int undefined;
printf("Und die Zusatzzahl lautet: %d", undefined);
```

- Keine automatische Initialisierung auf Defaultwerte
  ⇒ Jede Variable manuell initialisieren

Top 10 – die beliebtesten Fehler

3. Falscher Typ in Verbindung mit Zeigern

```
1  int* ptr = anotherPtr;
2  int i = 88;
3
4  ptr = i;        ✗ Inkompatible Typen -> error
5  *ptr = i;       ✓ Ziel des Zeigers mit Wert von i belegen
6  ptr = &i;       ✓ Zeiger auf Adresse von i „umbiegen“
7  ptr = 42;       ✗ Wahrscheinlich falsch -> warning
8  *ptr = 42;      ✓ Ziel des Zeigers mit 42 belegen
9
10 printf("Wert: %d", ptr);    ✗ Falsch -> keine warning
11 printf("Wert: %d", &ptr);   ✗ Falsch -> keine warning
12 printf("Wert: %d", *ptr);   ✓ Wert des Ziels von ptr ausgeben
```

```
Wert: xxxxxxxx
Wert: yyyyyyyy
Wert: 42
```

Top 10 – die beliebtesten Fehler

4. Überschreiten von Arraygrenzen / buffer overflow

```
printf("Letztes Argument: %s", argv[argc]);
```
✗ Overflow

- Keine automatische Indexvalidierung
  ⇒ Manuelle Absicherung, z.B. in Loops:
  `assert(i<size);`

```
1  int isAdmin = 0;
2  char small_buffer[10], buffer[100];
3  scanf("%s", small_buffer);
4  strcpy(buffer, small_buffer);
```
✗ potentieller Overflow
✗ potentieller Overflow

- Puffer ausreichend groß dimensionieren
- Pufferbegrenzende Funktionen verwenden, z.B.:

```
scanf("%9s", small_buffer);
strncpy(buffer, small_buffer, sizeof(buffer) - 1);
buffer[sizeof(buffer)-1] = '\0';  /* !!!!! */
```

Top 10 – die beliebtesten Fehler

5. Ungültige Zeiger (Null, undefiniert, freigegeben)

- Zeiger immer initialisieren, notfalls mit 0 (`NULL`)
- Ergebnis von `malloc()` überprüfen
- Zeiger auf freigegebenen Speicher auf 0 setzen

6. Keine Stringverkettung mit +

- +-Operator ist nicht überladen
  ⇒ `printf()` oder `sprintf()` verwenden

Top 10 – die beliebtesten Fehler

7. Rückgabe von Zeigern auf Stackvariablen

```
1  int * get_array() {
2      int array[] = {1, 2, 3};
3      return array;
4  }
```

✗ außerhalb von get_array() ungültig

- Zeiger auf lokale, auf dem Stack definierte Variablen dürfen nicht an übergeordnete Funktion zurückgegeben werden
  ⇒ Stattdessen `malloc()` verwenden

8. Memory leaks

- Für jedes `malloc()` komplementäres `free()` vorsehen
- In dynamischen Datenstrukturen für jedes Element separat `free()` aufrufen

Top 10 – die beliebtesten Fehler

9. Überraschende Defines

```
1  #define SUMME 10+10
2  printf("Ergebnis: %d", SUMME*2);
```

- `SUMME` wird ersetzt durch: `10+10*2` → `10+(10*2)`
- Ergebnis: 30 statt 40
- Ausdrücke in `#define` immer klammern
- Keine Semikolons oder geschweifte Klammern

Top 10 – die beliebtesten Fehler

10. Arraylängen und sizeof()

Java:

```
1 int zahlen[] = {1, 2, 3};
2 System.out.printf("Länge: %d", zahlen.length);
```

C:

```
1 int zahlen_1[] = {1, 2, 3};
2 int *zahlen_2 = {1, 2, 3};
3 printf("Länge: %d", sizeof(zahlen_1));
4 printf("Länge: %d", sizeof(zahlen_2));
```

- sizeof() berechnet die Größe von Datentypen:
  - Anzahl von Bytes um eine Variable zu speichern
- Größe eines Pointers auf 32 bit System ist immer 4
- Entspricht nicht immer der Länge eines Arrays

```
int length = sizeof(zahlen_1) / sizeof(zahlen_1[0]);
```

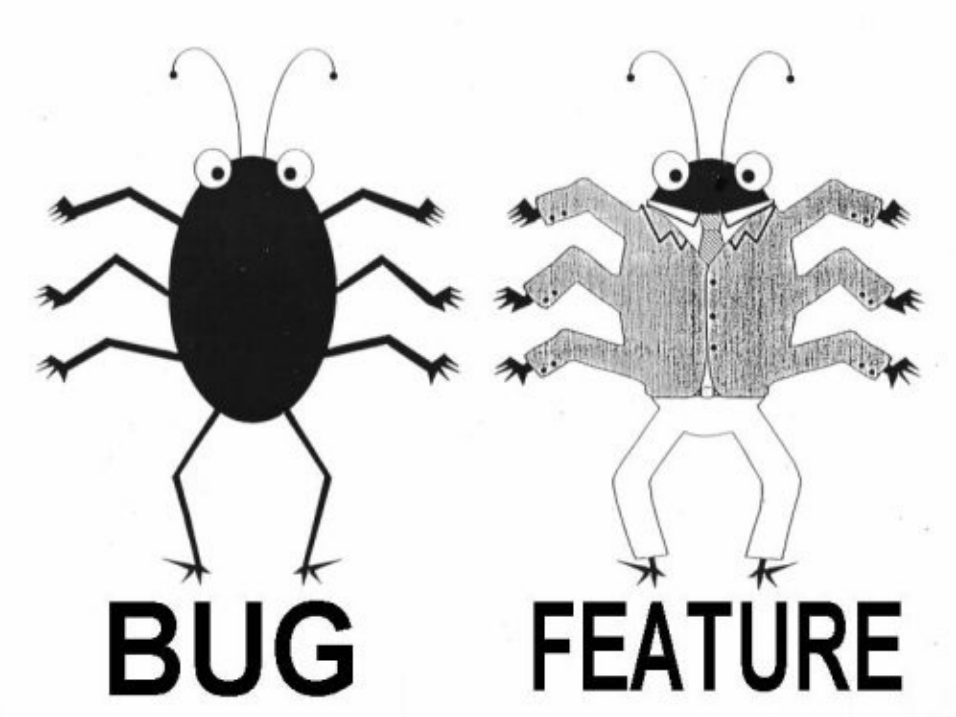
Top 10 – die beliebtesten Fehler

11. Off by one: Fehlendes Nullbyte am Stringende

```
char *copy_str = malloc( strlen(orig_str) + 1);
```

- Beim Kopieren von Strings Platz für Nullbyte lassen

Außerdem...



kerryosborne.oracle-guy.com/2009/03

Last but not least

- **Wenn alles nichts hilft...**
  - Code innerhalb der Gruppe gegenseitig erklären (Peer-Review / Walkthrough)
  - Zeitlicher Abstand oder eine „Mütze Schlaf“
  - Rewrite des fraglichen Codes
- **Besser nicht...**
  - Wahllos Code ändern bis etwas funktioniert

```
x = compute(y);
/* compute() doesn't work for y == 17, so fix it */
if (y == 17)
    x = 25.15;
```

3.1. Literatur – elektronisch (1)

- **Thinking in C – A Flash Based Audio-Visual Seminar**
  - Bruce Eckel
    → http://www.mindviewinc.com/CDs/ThinkingInC
    - Flash-basierter Multimedia-Kurs (englisch)
- **Einführung in ANSI-C**
  - Prof. Dr. Peter Baeumle-Courth
    → http://info.baeumle.com/ansic.html
    - An Kernighan und Richie angelehntes Script
    - Übersichtlich und gut strukturiert
- **Wikibooks: C-Programmierung**
  Einsteigerkurs in das Programmieren mit ANSI C
  - → http://de.wikibooks.org/wiki/C-Programmierung
    - Kollaboratives Werk
    - In verschiedenen Sprachen und als PDF erhältlich

3.1. Literatur – elektronisch (2)

- **UNIX manual-Pages**
  - Suche nach Stichworten via apropos-Kommando, z.B.:

```
(~): apropos printf
```

  - Zugriff auf manual-Pages von System- und Bibliotheks-aufrufen unter der UNIX-Systemen via man-Kommando:

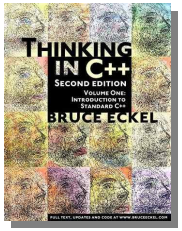
```
(~): man -s 3c printf
```

  - Online verfügbare man-Archive:
    - Man-Pages der Debian-Linux-Distribution:
      → http://manpages.debian.net
    - Umfangreiches man-Page-Archiv:
      → http://linuxmanpages.com

3.1. Literatur – elektronisch (3)

- **C Standard Library**
  - Online-Referenz der Standard C Library unter:
    → http://www.cplusplus.com/reference/clibrary

- **Falls euch C langweilt, lernt C++, denn:**
  - C++ erlaubt objektorientiertes Programmieren
  - C++ ist performant
  - Es gibt umfangreiche Bibliotheken für C++
  - Einblicke in (in Java verborgene) Details
  - Kostenloses ebook: Thinking in C++, Bruce Eckel
    - → http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html

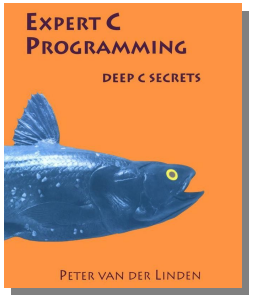


3.2. Literatur – auf Papier (1)

- **C Programmieren von Anfang an**
  - Helmut Erlenkötter;
    2. Auflage 1999; Rowohlt Taschenbuch; ISBN 3499600749

    - Leicht verständlich, für Programmiereinsteiger geeignet
    - Viele Beispiele, Aufgaben am Kapitelende
    - 1. Platz der Amazon-Verkaufs-Charts

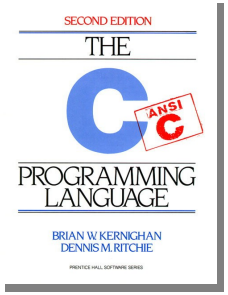
- **Expert C Programming**
  - Peter van der Linden;
    1. Edition 1994; Prentice Hall; ISBN 0131774298



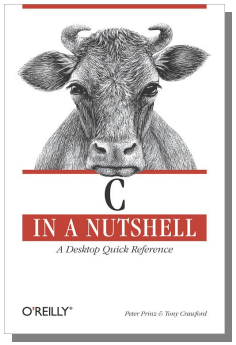
    - Detailiertes Hintergrundwissen, best practises
    - Humorvoller Stil, viele Anektdoten
    - Für erfahrene C Programmierer geeignet
    - Nur auf Englisch erhältlich

3.2. Literatur – auf Papier (2)

- **The C Programming Language**
  - Brian W. Kernighan, Dennis M. Ritchie;
    2. Edition 1988; Prentice Hall; ISBN 0131103628



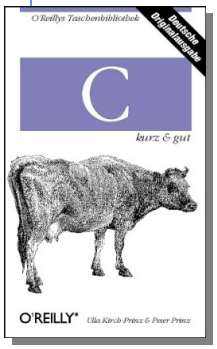
    - **DAS** Standardwerk, gut als Nachschlagewerk geeignet
    - Für Programmieranfänger weniger geeignet
    - Letzte Auflage von von 1988, kein C99
    - 274 Seiten, enthält Referenz der C Standardbibliothek
- **C in a Nutshell**
  - Peter Prinz, Tony Crawford;
    1. Auflage 2006; O'Reilly; ISBN 3897213443



    - Sehr aktuell, basiert auf ISO/IEC 9899:1999 (C99)
    - Umfangreich auf über 600 Seiten
    - Umfasst C, Standardbibliothek, GNU-Tools (gcc, gdb, make)
    - Nicht unbedingt für Einsteiger geeignet

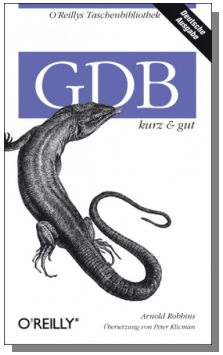
3.2. Literatur – auf Papier (3)

- **C – Kurz & gut**
  - Peter Prinz, Ulla Kirch-Prinz



  1. Auflage 2002; O'Reilly; ISBN 3897212382
    - Nachschlagewerk für Sprache C und Standardbibliothek
    - Preiswert erhältlich

- **GDB – Kurz & gut**
  - Arnold Robbins



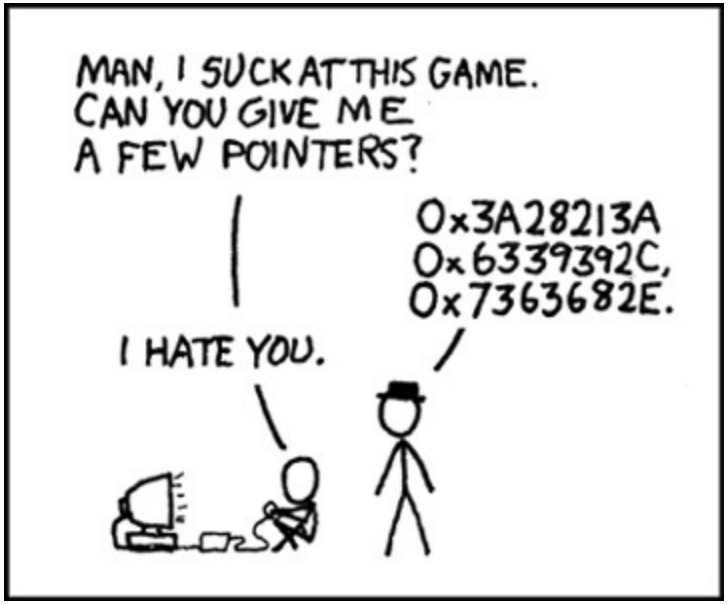
  1. Auflage 2005; O'Reilly; ISBN 3897215128
    - Nachschlagewerk für den GNU Debugger (GDB)
    - Preiswert erhältlich

Fragen?

[Bitte vergesst nicht den Feedbackbogen...]

Das wars...

Vielen Dank für
Eure Aufmerksamkeit!



www.xkcd.com/138