# Kompaktkurs C-Programmierung

26. 03. 2007

Franz Schenk

# Organisatorisches

- Vorlesung
- Accountvergabe
- Skriptverkauf
- Übungen:
  zwei Blöcke (13:15-14:45 / 14:45-16:15)
  - Gruppenübung
  - Rechnerübung
- Klausur
- Ferien: Karfreitag - Ostermontag

# C: Geschichtliches

- 1969/1973 Entwicklung von C um UNIX zu implementieren (v.a. Ken Thompson und Dennis Ritchie)
- 1978: Ritchie und Brian Kernighan veröffentlichen die erste Auflage von *The C Programming Language* (K&R)
- 1983 beginn der ANSI Standardisierung
  1989: ANSI-C 89
- 1990 ISO-C 90
- 1999: C99 (ISO C 99)

# [Hello World] Programmiersprachen

```
        .file   "hello.c"
        .section        .rodata
.LC0:
        .string "hello"
        .text
.globl main
        .type   main, @function
main:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $8, %esp
        andl    $-16, %esp
        movl    $0, %eax
        addl    $15, %eax
        addl    $15, %eax
        shrl    $4, %eax
        sall    $4, %eax
        subl    %eax, %esp
        subl    $12, %esp
        pushl   $.LC0
        call    puts
        addl    $16, %esp
        leave
        ret
        .size   main, .-main
```

## Unterschiedliche Sprachkonzepte

```
/* Hello World in C */

#include <stdio.h>

main()
{
        printf("hello\n");
}
```

```
/* Hello World in perl */

print("hello\n");
```

# [Hello World] Programmiersprachen

Ziele der Entwickler von C:

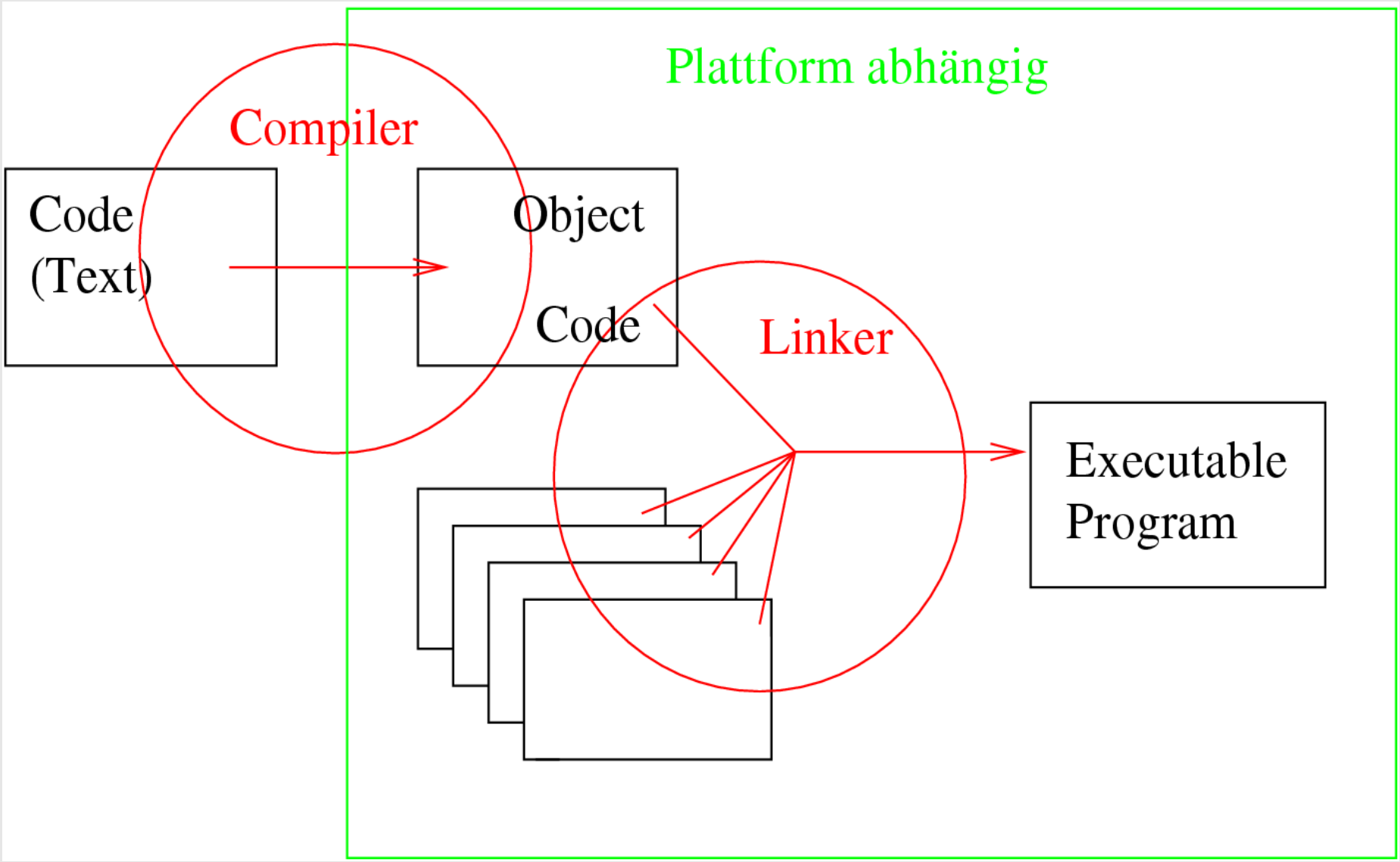
- maschinennahe Programmierung ohne direkt Maschinenbefehle (Assembler) zu benutzen

- damit: Übertagbarkeit von Code auf beliebige Maschinen

# [Hello World] Schritte der Programmentwicklung

- Werkzeuge:
  Editor
  Compiler (Übersetzer) / Linker
- Methodik:
  Quelltexterstellung
  Compilierung
    u.U. Fehlersuche und das ganze noch mal

# [Hello World] Programmwerdung



# [Hello World]  „Hello World“

```
/* Das allererste Programm */

#include <stdio.h>

int main( )
{


    printf( “Hello World!\n“);


return 0;

}
```

Kommentarzeile

Einbinden einer Bibliothek von Funktionen

Hauptfunktion, die beim Programmaufruf gestartet wird.

Anweisung: Hello World! ausgeben (mit Zeilenumbruch)

Beenden der (Haupt)Funktion, Rückreichung eines Wertes

# [Hello World] Ingredienzien eines Programms

- /* Das allererste Programm */

- Kommentare sind für den Compilationsprozess belanglos.
- Sie erhöhen aber entschieden die Lesbarkeit, vor allem damit die Möglichkeit Code weiterzureichen. Aber auch der eigenen Erinnerung sehr dienlich!
- Kommentare sollen immer so viel wie möglich verwendet werden und werden immer viel zu wenig verwendet.

# [Hello World] Ingredienzien eines Programms

- #include <stdio.h>

- Definiert, daß die Standardbibliothek für Ein- und Ausgabe verwendet werden soll.
- Damit müssen nicht alle Funktionen im eigenen Programm geschrieben werden, sondern können eingebunden werden.
- Eigene oder vorgefertigte Bibliotheken
- Wichtige Information für Präprozessor und Linker

# [Hello World] Ingredienzien eines Programms

```
int main(){
}
```

- Die Funktionsdefinition
- main: Der Funktionsname. Diese ist eine spezielle Funktion, welche beim Programmstart aufgerufen wird.
- { } Alles, was in diesem Klammerpaar eingeschlossen wird, gehört zur Funktion.
- int: Rückgabewert der Funktion, den sie an den Aufrufer zurückliefert (hier: ein integer-Wert)
- () in den Klammern könnte definiert sein, was der Funktion übergeben wird.

# [Hello World] Ingredienzien eines Programms

- printf(“Hello World!\n“);

- printf() Funktionsaufruf (Funktion ist in der stdio -Bibliothek enthalten)
- (.....) Innerhalb der Klammern stehen ein oder mehrere Parameter, welche der Funktion übergeben werden.
- Mehrere Parameter würden durch Kommata getrennt. z.B. einekleineFunktion(param1,param2)
- ; Wichtig: Jede Befehlszeile wird damit abgeschlossen.
- \n Mitteilung, dass die Ausgabefunktion einen Zeilenumbruch einfügen soll.

# [Hello World] Richtlinien für schönen und guten Code

```
#include "stdio.h"
#define e 3
#define g (e/e)
#define h ((g+e)/2)
#define f (e-g-h)
#define j (e*e-g)
#define k (j-h)
#define l(x) tab2[x]/h
#define m(n,a) ((n&(a))==(a))

long tab1[]={ 989L,5L,26L,0L,88319L,123L,0L,9367L };
int tab2[]={ 4,6,10,14,22,26,34,38,46,58,62,74,82,86 };

main(m1,s) char *s; {
    int a,b,c,d,o[k],n=(int)s;
    if(m1==1){ char b[2*j+f-g]; main(l(h+e)+h+e,b); printf(b); }
    else switch(m1-=h){
 case f:
     a=(b=(c=(d=g)<<g)<<g)<<g;
     return(m(n,a|c)|m(n,b)|m(n,a|d)|m(n,c|d));
 case h:
     for(a=f;a<j;++a)if(tab1[a]&&!(tab1[a]%((long)l(n))))return(a);
 case g:
     if(n<h)return(g);
     if(n<j){n-=g;c='D';o[f]=h;o[g]=f;}
     else{c='\r'-'\b';n-=j-g;o[f]=o[g]=g;}
     if((b=n)>=e)for(b=g<<g;b<n;++b)o[b]=o[b-h]+o[b-g]+c;
     return(o[b-g]%n+k-h);
 default:
     if(m1-=e) main(m1-g+e+h,s+g); else *(s+g)=f;
     for(*s=a=f;a<e;) *s=(*s<<e)|main(h+a++,(char *)m1);
 }
}
```

# [Hello World] ... mehr über guten Code

- Wozu guter Code?
  > andere sollen ihn lesen können
  > man selbst soll ihn (schnell) lesen können
  > andere und man selbst sollen verstehen, was der Code bewirkt.

- Daher:
  > Kommentieren
  > Kommentieren
  > Kommentieren
  > Strukturieren
  > Lesbarkeit erhöhen: Leerzeichen, Leerzeilen
  > Einfache Lösungen

# [Hello World] ... noch mehr über Stil

```
#include <stdio.h>
int l;int main(int o,char **O,
int I){char c,*D=O[1];if(o>0){
for(l=0;D[l                ];D[l
++]-=10){D   [l++]-=120;D[l]-=
110;while   (!main(0,O,l))D[l]
+=   20;   putchar((D[l]+1032)
/20   )   ;}putchar(10);}else{
c=o+     (D[I]+82)%10-(I>l/2)*
(D[I-l+I]+72)/10-9;D[I]+=I<0?0
:!(o=main(c/10,O,I-1))*((c+999
)%10-(D[I]+92)%10);}return o;}
```

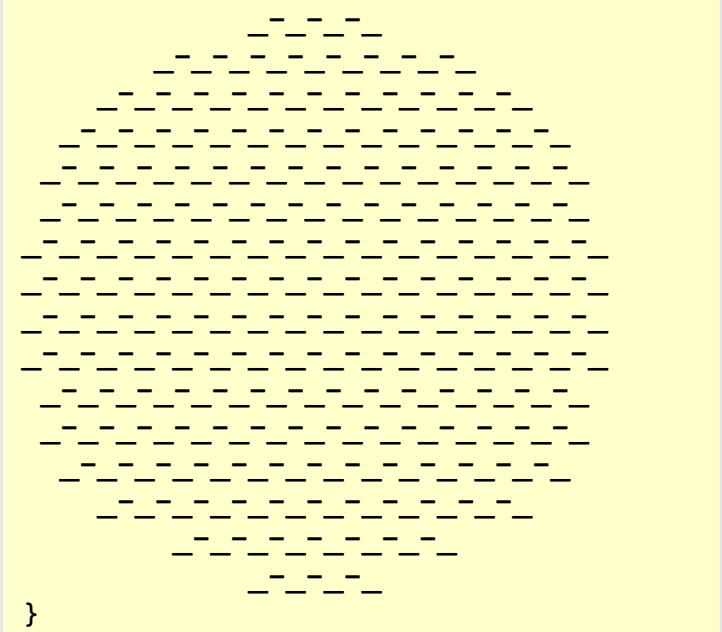
- Ein gutes oder schlechtes Beispiel für:

  - selbst-dokumentierenden Code?

  - leicht zu wartenden Code

# [Hello World] ... mehr dubiose Beispiele

```
#define _ -F<00||--F-OO--;
int F=00,OO=00;
main(){F_OO();printf("%1.3f\n",4.*-
F/OO/OO);}F_OO()
{
```



```
}
```

- Code verschleiern als Spiel: obfuscation

  viele schöne Beispiele:

  http://www.de.ioccc.org

# [Hello World] Vom Code zum Programm

- gcc -Wall -o hallo hallo.c

- gcc gnu compiler collection
  Compiler und linker
- -Wall Warnungen werden ausgegeben
  (noch besser: -ansi -pedantic)
  Es werden nicht nur Fehler gemeldet, welche den Sprachstandard verletzen, sondern auch Warnungen ausgegeben.
- -o hallo erzeugt als ausführbares Programm die Datei hallo aus dem Quellcode in Datei hallo.c
- Die Endung .c ist nicht zwingend, hilft aber vielen Editoren zu erkennen, welche Sprache vorliegt.

# [Hello World] Compilationsschritte

- Lexikalische Analyse
  -> Tokenerzeugung
- Präprozessierung
- Syntaktische Analyse
- Semantische Analyse

# [Hello World] Programmausführung

- hallo
  ```
  Hello World!
  ```
- Funktioniert oft nicht.
- ls -al hallo
  ```
  -r--r--r-- 1 schenk schenk 6.9K 2007-02-28 15:49 hallo
  ```
- Möglichkeit 1: unzulängliche Rechte. Abhilfe mit chmod
- echo $PATH
- Möglichkeit 2: Die Systemvariable PATH ist zeigt nicht auf das aktuelle Verzeichnis.
  Abhilfe: ./hallo
  oder: export PATH=$PATH:.

# [Hello World] Hello World, improved

- Noch mehr Hello World

```
#include <stdio.h>

int main()
{
    int zaehler = 0;

    while (zaehler < 10)
    {
        printf(“Hello World!\n“);
        zaehler = zaehler + 1;
    }

    return 0;
}
```

# [Hello World] Hello World, nummeriert

- Hello World, mit Zahlen

```
#include <stdio.h>

int main()
{
    int zaehler = 0;

    while (zaehler < 10)
    {
        printf(“Hello World %d!\n“, zaehler);
        zaehler = zaehler + 1;
    }

    return 0;
}
```

# [Hello World] Hello World, manchmal

- Hello World, mit geraden Zahlen

```
#include <stdio.h>

int main()
{
    int zaehler = 0;

    while (zaehler < 10)
    {
        if ( zaehler%2 == 0)
        {
            printf(“Hello World %d!\n“, zaehler);
        }
        zaehler = zaehler + 1;
    }

    return 0;
}
```

# [Hello World]  Hello World, manchmal, ganz genau

- Hello World, nur mit geraden Zahlen

```
#include <stdio.h>

int main()
{
    int zaehler = 0;

    while (zaehler < 10)
    {
        if ( zaehler%2 == 0 && zaehler != 0)
        {
            printf(“Hello World %d!\n“, zaehler);
        }
        zaehler = zaehler + 1;
    }

    return 0;
}
```