Organisatorisches

- Übungsleiter: Karsten Otto (otto@inf.fu-berlin.de)
- Homepage: http://www.inf.fu-berlin.de/lehre/SS04/SySi/
- Aufgaben
  - Montags im Netz
  - Vorbesprechung Dienstag/Mittwoch in den Übungen
  - Abgabe am darauf folgenden Donnerstag vor der Vorlesung
  - Bearbeitung in 2er Gruppen
  - Bei Abgabe bitte Namen und Übungsgruppe (A/B) angeben
  - Pro Aufgabenblatt müssen 60% der Punkte erreicht werden
  - N-1 Aufgabenblätter müssen erfolgreich bearbeitet werden
  - Alle N Aufgabenblätter gehen in die Übungsnote ein

C Programmierung – Alles klar?

```
int a[1817];main(z,p,q,r){
for(p=80;q+p-80;p-=2*a[p])
for(z=9;z--;)q=3&(r=time(0)+r*57)/7,
q=q?q-1?q-2?1-p%79?-1:0:p%79-77?
1:0:p<1659?79:0:p>158?-79:0,
q?!a[p+q*2]?a[p+=a[p+=q]=q]=q:0:0;
for(;q++-1817;)
printf(q%79?"%c":"%c\n"," #"[!a[q-1]]);}
```

Keine Panik!

Einfacheres Beispiel

```
#include <stdio.h>

void strmcpy(char* dest, char* src, int m)
{
    while (m-- && (*dest++ = *src++));
}

int main(int argc, char **argv)
{
    char buf[10];
    strmcpy(buf, argv[1], 10);
    printf("%s\n", buf);
}
```

C

- B. W. Kernighan, D. M. Ritchie:
  Programmieren in C, Zweite Ausgabe, C. Hanser

- keine Objektorientierung
- keine Garbage Collection, manuelle Speicherverwaltung
- kein Speicherschutz
- Zeiger-Arithmetik

Typen

- **`char`** (1 byte), **`short`** (2 bytes), **`int`** (4 bytes), **`float`** (4 bytes), **`double`** (8 bytes), ...
- Aufzählungstypen: **`enum boolean {YES, NO};`**
- Felder: **`char str[80]`**
- Strukturen
  `struct Punkt {int x; int y;};`
  `struct Punkt p1;`
- Umbenennung
  `typedef struct Punkt punkt_t;`
  `punkt_t p2;`
- Unions
  `union wert {int i, float f};`

Quelltext-Struktur

```
#include <string.h>
#include "myheader.h"

#define MAX 100

extern int countall;
static int step = 1;

int getCount() {
    static int count = 0;
    count += step;
    countall += step;
    return count;
}

int main(int argc, char** argv) { … }
```

- Importe
- Konstanten
- Variablen-Deklarationen
  - zu Beginn (jedes Blocks)
  - aus anderem Modul: **`extern`**
  - Modul-lokal: **`static`**
  - Funktionsaufruf-überdauernd: **`static`**
- Hauptprogramm

Zeiger

- Verweis auf Ort im Speicher

```
int *iptr;
int i = 1;
iptr = &i;     /* Adressoperator */
*(iptr) = 2;   /* Dereferenzierung */
```

- Zeiger auf Strukturen

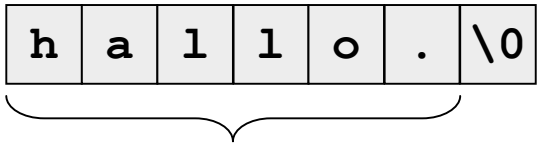
```
struct Punkt {
    int x; int y;
};
struct Punkt *p1 = … ;
```

`(*p1).x` ist das gleiche wie `p1->x`

Zeichenketten

- sind nullterminiert:

```
char str[] = "hallo."
```




```
strlen(str) =
```

- Zugriff

```
str[1] = 'e';
*str = 'H';
*(str+5) = '!';              /* Zeiger-Arithmetik */
memset(str, 'X', 5);
```

- Hilfsfunktionen

```
#include <string.h> /* strlen,strncpy,memset,… */
```

Dynamische Speicherverwaltung

- Hilfsfunktionen

```
#include <stdlib.h> /* malloc,realloc,free,… */
```

- Beschaffung von Speicher fester Grösse

```
punkt_t* p = (punkt_t*) malloc ( sizeof(punkt_t));
             ↑typecast            ↑Datentyp-Grösse
free(p);
```

- Verwendung mit Feldern / Strings

```
char* str = (char*) malloc( sizeof(char) * 10);
str = (char*) realloc( str, sizeof(char) * 20);
free(str);
```

Ein-/Ausgabe

- Schreiben auf die Standardausgabe

```
int printf ( char *format, ... );

name = "Max"; zahl = 4711;
printf( "Name: %s, Zahl: %d\n", name, zahl );
```

- Lesen von der Standardeingabe

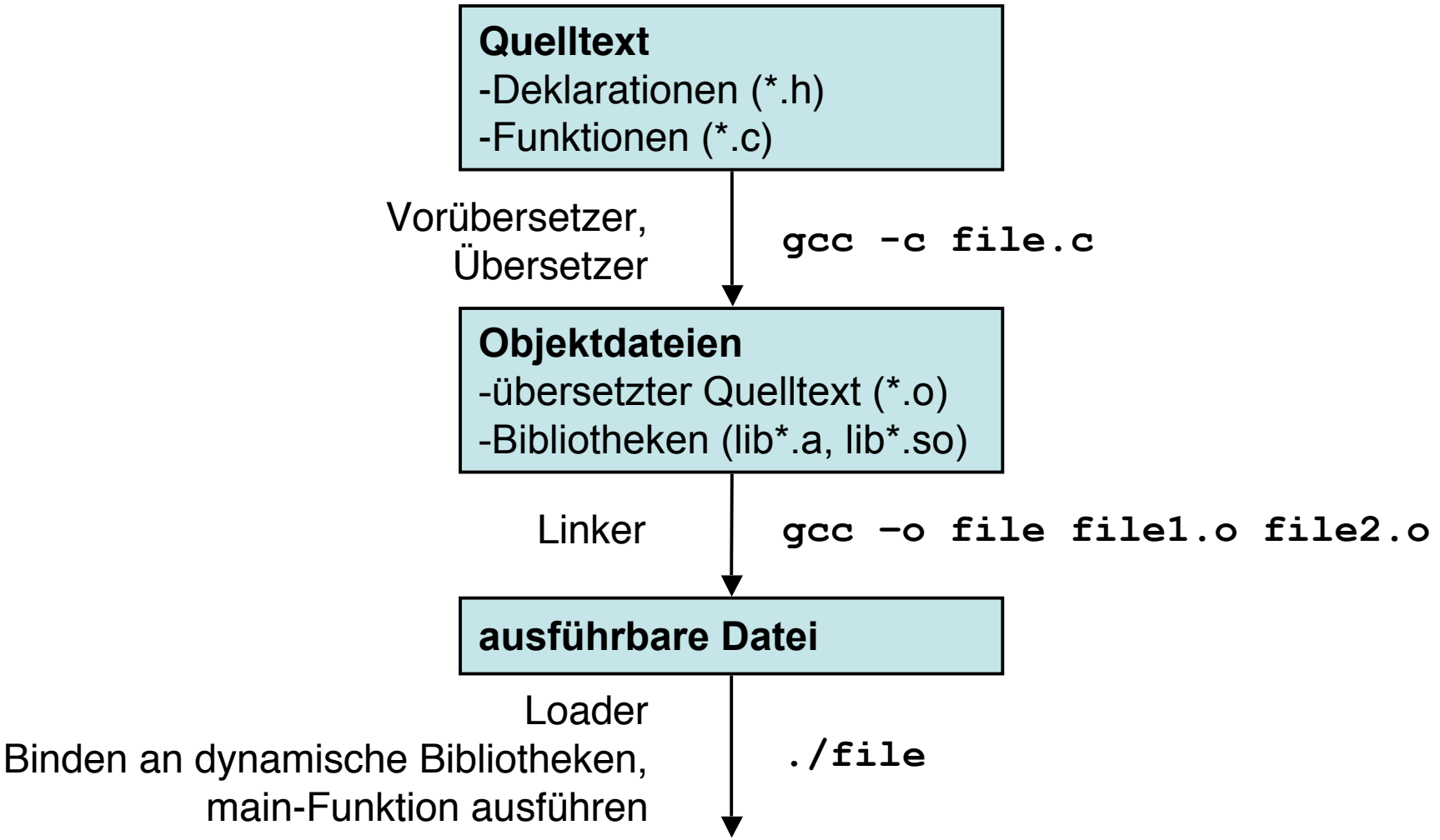
```
int scanf( char *format, ... );

char name[80];
int alter;
scanf( "Name: %80s  Alter: %d", name, &alter );
```

Dokumentation

- API-Dokumentation: Man(ual) Pages
  - Sektion 2: Systemaufrufe (**`open`**, **`close`**, …)
  - Sektion 3: Bibliotheksfunktionen:
    `man [2,3] <funktion>`
    `man -k <Schlüsselwort>`
    `(= apropos <Schlüsselwort>)`

Übersetzungsvorgang



Makefiles

- Syntax (-->| bedeutet TAB)

```
target: dependency1 dependency2 ...
 -->| command
```

- Beispiel: main.c util.c defs.h

```
CFLAGS = -ansi -pedantic -Wall
myapp: main.o util.o
    gcc -o myapp $(CFLAGS) main.o util.o
main.o: main.c defs.h
    gcc -c $(CFLAGS) main.c
util.o: util.c defs.h
    gcc -c $(CFLAGS) util.c
clean:
    rm *.o
```

http://web.mit.edu/sipb-iap/unixsoftdev/www/makefiles.html

Zurück zum Beispiel

```
#include <stdio.h>

void strmcpy(char* dest, char* src, int m)
{
    while (m-- && (*dest++ = *src++));
}

int main(int argc, char **argv)
{
    char buf[10];
    strmcpy(buf, argv[1], 10);
    printf("%s\n", buf);
}
```

Zum besseren Verständnis…

```
void strmcpy(char* dest, char* src, int m) {
  while (m-- && (*dest++ = *src++));
}

void strmcpy(char* dest, char* src, int m){
  while (m-- > 0 && (*dest++ = *src++) != '\0');
}

void strmcpy(char* dest, char* src, int m){
  while (m-- > 0 && *src != '\0') {
    *dest++ = *src++;
  }
  if (m >= 0) *dest = '\0';
}
```

… Abkürzungen auflösen

```
void strmcpy(char* dest, char* src, int m) {
  while (m-- > 0 && *src != '\0') {
    *dest++ = *src++;
  }
  if (m >= 0) *dest = '\0';
}
void strmcpy(char* dest, char* src, int m) {
    while (m > 0 && *src != '\0') {
        *dest = *src;
        dest++;
        src++;
        m--;
    }
    if (m > 0) *dest = '\0';
}
```