

C-Programmierung

Dateisysteme

- ❑ Eine **Datei** ist eine Sammlung logischer Dateneinheiten
- ❑ **Dateisysteme** speichern z.B. Daten, Code, Programme dauerhaft in Dateien
- ❑ Dateisysteme leisten
 - Abstraktion von Hintergrundspeichern, z.B. Platten, CD-ROM, USB, Bandlaufwerke, ...
 - Einheitliche Schnittstelle

- ☐ **Name:** Symbolischer Name, vom Benutzer les- und interpretierbar
- ☐ **Größe:** Länge der Datei (Bytes, Blocks, ...)
- ☐ **Zeitstempel:** Zeitpunkt der Erstellung, letzte Modifikation, ...
- ☐ **Rechte:** Zugriffsrechte
- ☐ **Eigentümer:** Identifikation
- ☐ ...
- ☐ Beispiel:

```
anja% ls -l .
```

```
...
```

```
drwxr-xr-x 10 anja user    2048 2014-08-21 12:41 progintro
-rw-r--r--  1 anja user    1047 2014-09-16 15:56 hello.c
```

```
...
```

Operationen auf Dateien

- ☐ Erzeugen (**create**)
- ☐ Schreiben (**write**)
- ☐ Lesen (**read**)
- ☐ Löschen (**delete**)
- ☐ Öffnen/Schließen einer Datei (**open/close**)
- ☐ ...

- ❑ Ein **Verzeichnis** ist eine Sammlung von Dateien und Verzeichnissen
- ❑ Verzeichnisattribute: Ähnlich wie Dateiattribute:
 - Name, Größe, Datum des letzten Updates, Eigentümer, Rechte
 - ...
- ❑ Beispiel:

```
anja% ls -l .
```

```
...
```

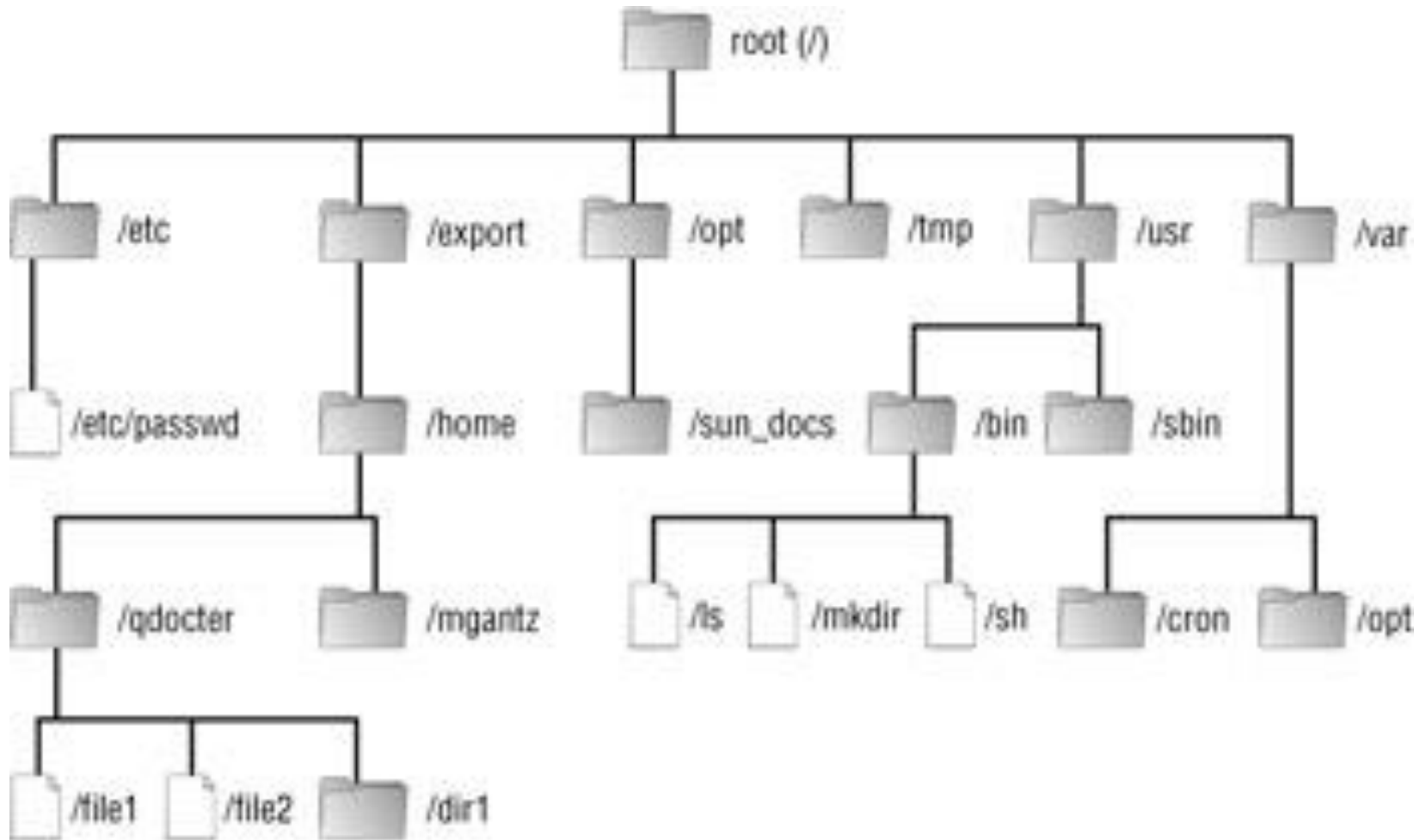
```
drwxr-xr-x 10 anja user    2048 2014-08-21 12:41 progintro
-rw-r--r--  1 anja user    1047 2014-09-16 15:56 hello.c
```

Operationen auf Verzeichnissen

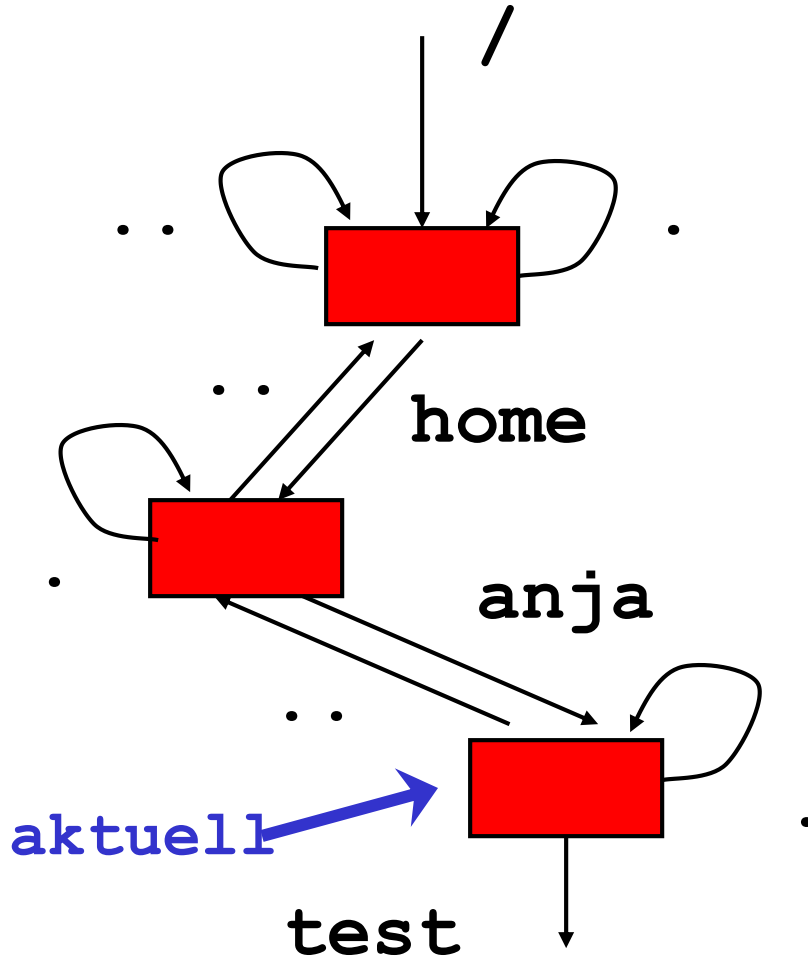
- ☐ Suchen einer Datei, Verzeichnisses
- ☐ Erzeugen einer Datei, Verzeichnisses
- ☐ Löschen einer Datei, Verzeichnisses
- ☐ Umbenennen von Dateien, Verzeichnisses
- ☐ Auslesen der Einträge
- ☐ Durchlaufen des Dateisystems

- ☐ Wichtige Unix Befehle:
 - ls, mv, cp, mkdir, rm, rmdir, find
- ☐ Details:
 - man <Befehl>

Verzeichnisse haben Baumstruktur



Unix Pfadnamen



- ☐ Benannt sind die Verbindungen zwischen Dateien und Katalogen
- ☐ Rückverweise: `".."`
- ☐ Selbstverweise: `"."`
- ☐ Verschiedene Pfade für selbe Datei, Verzeichnisse
- ☐ Aktuelles Verzeichnis:
 - `/home/anja/test/`
 - `/home/anja/test/.`
 - `/home/anja/test/../test`

Arbeiten mit Dateien in C

Dateien – Formatierte Ausgabe: `fprintf`

Aufruf: `fprintf(FILE *stream, fmt, args)`

□ `fprintf()` wie `printf` jedoch mit Dateien / Streams, d.h. konvertiert und gibt die Parameter `args` unter Kontrolle des Formatstrings `fmt` auf `stream` aus

□ Jedes laufende C-Programm (= Prozess) hat voreingestellt drei Kanäle für Ein-/Ausgabe:

- **stdin** Standardeingabe, meist Tastatur
- **stdout** Standardausgabe, meist Bildschirm
- **stderr** Standardfehlerausgabe, meist Bildschirm

Dateien – Formatierte Ausgabe: fprintf

Aufruf: `fprintf(FILE *stream, fmt, args)`

❑ `fprintf()` wie `printf` jedoch mit Dateien / Streams, d.h. konvertiert und gibt die Parameter `args` unter Kontrolle des Formatstrings `fmt` auf `stream` aus

❑ Beispiel C Streams: `stdout`, `stdin`

❑ Beispiele: `// fprintf(stdout, ...) entspricht printf`

- `fprintf(stdout, "Hello world\n");`
- `fprintf(stdout, "Wert von i: %d\n", i);`
- `fprintf(stdout, "a(%d)+b(%d) ist: %d\n",
a, b, a+b);`

Dateien – Formatierte Eingabe: fscanf

Aufruf: `fscanf(FILE *stream, fmt, args)`

❑ `fscanf()` liest von `stream` und versucht die Eingabe unter Kontrolle des Formatstrings `fmt` auf die Parameter `args` abzubilden

❑ Beispiele: `// fscanf(stdin, ...) entspricht scanf`

➤ `int a, b; fscanf(stdin, "%d %d", &a, &b);`

➤ `float x; fscanf(stdin, "%f", &x);`

➤ `char a; fscanf(stdin, "%c", &a);`

Öffnen von Dateien: fopen

Aufruf: `FILE *stream fopen(path, mode)`

❑ `fopen()` öffnet die Datei `path` im Modus `mode`

❑ Beispiele:

- `FILE *file_pointer_in, *file_pointer_out;`
- `file_pointer_in = fopen("./datei", "r");`
`// öffnet Datei „datei“ zum Lesen`
- `file_pointer_out = fopen("./datei", "w");`
`// öffnet „datei“ zum Schreiben`
- `file_pointer_out = fopen("./datei", "a");`
`// öffnet „datei“ zum Schreiben mittels Anhängen`

Schließen von Dateien: `fclose`

Aufruf: `fclose(FILE *stream)`

- ❑ `fclose()` schließt den Stream `stream`
- ❑ Fehlerfreies Beenden der Operationen
- ❑ Gibt Ressourcen frei, u.a. im Betriebssystem

❑ Beispiel:

```
➤ fclose(file_pointer_in) ;  
   //schließt den Stream file_pointer_in
```


Lesen von Datei Ausgabe auf Stdout

```
// Konvention: fp == file_pointer
FILE *fp = fopen("datei.txt", "r");
int a, b;
while (!feof(fp)) {
    fscanf(fp, "%d %d\n", &a, &b);
    printf("%d %d\n", a, b);
}
fclose(fp);
```

- ❑ **feof()** testet den Stream auf EOF (End of File)
- ❑ **Hinweis: Fehlerbehandlung fehlt ist aber notwendig!!!**

Fehlerbehandlung mit perror

```
#include <errno.h>
FILE *fp = fopen("datei.txt", "r");
int a, b;
if (fp == NULL ) {
    perror("Datei oeffnen Fehler");
    return 1; }
while (!feof(fp)) {
    fscanf(fp, "%d %d\n", &a, &b);
    printf("%d %d\n", a, b); }
fclose(fp);
```

❑ **void perror(msg)**

- Gibt die letzte Systemfehlermeldung auf **stderr** aus

Lesen von Datei Ausgabe in Datei

```
FILE *fpin = fopen("datei_in", "r");  
FILE *fpout = fopen("datei_out", "a");  
int a, b;  
while (!feof(fpin)) {  
    fscanf(fpin, "%d %d\n", &a, &b);  
    fprintf(fpout, "%d + %d = %d\n", a, b, a+b);  
}  
fclose(fpin); fclose(fpout);
```

❑ Hinweis: Fehlerbehandlung fehlt ist aber notwendig!!!

Problem: Fehlerhaften Eingabedaten

- ❑ Eingabedaten entsprechend nicht unbedingt den Erwartungen
- ❑ Z.B. anstelle einer Zahl ein String
- ❑ Deshalb immer überprüfen, ob das Lesen erfolgreich war!
- ❑ Kann bei scanf, fscanf problematisch sein.
- ❑ Lösung:
 - fgets zum lesen einer Zeile
 - gefolgt von sscanf zum Konvertieren des Strings in die Token

Dateien – Formatierte Eingabe: fscanf

Aufruf:

```
char *fgets(char *buf, int n, FILE *stream)
```

❑ `fgets()` liest bis zum ersten Newline „\n“ von `stream`
dabei aber maximal `n-1` Zeichen und fügt `\0` hinzu

❑ Beispiele:

```
➤ char buf[200]; int a, b;  
  char *h = fgets(buf, 200, stdin);  
  if (h==0) {printf(„error in fgets“);exit(1);}  
  int ret = sscanf(buf, „%d %d“, &a, &b);  
  if (ret == 2) {  
      printf(„read %d %d\n“, a, b);  
  } else { printf(„error reading two ints\n“);}
```

Dateien – Formatierte Eingabe: fscanf

```
char buf[200];  
int a, b;  
char *h = fgets(buf, 200, stdin);  
if (h==0) {  
    printf („error in fgets“);  
    exit(1);  
}  
int ret = sscanf(buf, „%d %d“, &a, &b);  
if (ret == 2) {  
    printf („read %d %d\n“, a, b);  
} else {  
    printf („error reading two ints\n“);  
}
```

Dateien – Formatierte Eingabe: `fscanf`

Aufruf:

```
char *fgets(char *buf, int n, FILE *stream)
```

- ❑ `fgets()` liest bis zum ersten Newline „`\n`“ von `stream`
dabei aber maximal `n-1` Zeichen und fügt `\0` hinzu

- ❑ Finger weg von `gets(char *buf)`

`gets` liest unabhängig von der Größe des Buffers bis zum ersten Newline „`\n`“ von `stream`. **Buffer Overflow Gefahr!**

□ Jedes laufende C-Programm (= Prozess) hat voreingestellt drei Kanäle für Ein-/Ausgabe:

- **stdin** Standardeingabe, meist Tastatur
- **stdout** Standardausgabe, meist Bildschirm
- **stderr** Standardfehlerausgabe, meist Bildschirm

□ Die Standardkanäle sind umlenkbar:

```
$ ./meinprog < InFile  
$ ./meinprog > OutFile
```


- ❑ Die Standardkanäle sind umlenkbar:

```
$ ./meinprog < InFile
```

```
$ ./meinprog > OutFile
```

- ❑ Die Standardkanäle sind kombinierbar:

```
$ ./meinprog < InFile > OutFile
```

- ❑ Unix Tools:

```
cat, sort, more, less, grep, wc, tail, cut,  
sed, split, uniq
```

- ❑

```
$ ./meinprog1 | sort > OutFile
```

Ausgabe von ./meinprog1 als Eingabe für sort verwenden

C-Kurs Kompensationsaufgabe

Ausgabe: Mittwoch 15.12

Abgabe: Sonntag 08.01 18:00

- ❑ Für alle, die die Portfoliopunkte vom C-Kurs nicht übernehmen wollen
 - Hauptsächlich für Studierende die am C-Kurs nicht teilnehmen konnten
 - Warum:
Wiederholbarkeit des C-Kurses

- ❑ Entscheidung für die Kompensationsaufgabe
 - Anmeldung bis zum 5.1
 - Sonst werden automatisch die Portfoliopunkte vom C-Kurs übernommen

Hinweise zur C-Kurs Kompensationsaufgabe

- ❑ Es handelt sich um ein selbstständig zu bearbeitendes Projekt
 - Hinweis: Es wird auf Plagiate geprüft!
 - Keine Gruppenabgabe!
 - Keine speziellen Tutorien oder Rechnerübung!

- ❑ Während der vorlesungsfreien Zeit wird es nur eingeschränkte Unterstützung des Introprogteams geben
 - Nutzen Sie das ISIS Forum