

File I/O

Persistieren von Daten

Gastvorlesung - 18.01.10

Ralph Erdt

erd (bei) informatik.fh-wiesbaden.de

Ausflug: scanf()

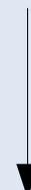
- Liest Werte ein
- Quelle: stdin (Tastatur)
- Syntax:
 - `#include <stdio.h>`
 - `scanf(Format, Ziel* [, Ziel* [..]])`
- Beispiele
 - `int i; scanf("%d", &i); //%c, %f, ...`
- Achtung! Unsicher, ist leicht zu zerstören.

Ausflug: scanf()

- Beispiel: Einlesen von Strings
 - `char* ziel; int x;`
 - `char buffer[100]; // oder; char* buffer = malloc(100);`
 - `scanf("%s", buffer);`
 - `for (x=0; buffer[x] != '\0'; x++);`
 - `ziel = (char*) malloc(x+1);`
 - `for (x=0; buffer[x] != '\0'; x++) ziel[x] = buffer[x];`
 - `ziel[x] = '\0';`

File I/O: Funktionsweise

- Öffnen
- Lesen/Schreiben
 - Zeiger auf eine Position
 - Zeiger wird bei einer Aktion weiterbewegt
 - Position wechseln
 - Position abfragen
 - EOF abfragen
- Schliessen



Hallo, \nwie geht es?\nMir geht es

The diagram shows a vertical arrow pointing downwards from the top of the text box to the first character 'H' of the string 'Hallo, \nwie geht es?\nMir geht es'. The text box has a light blue background and a thin black border.

Datei öffnen

- Syntax
 - `#include <stdio.h>`
 - `FILE* fopen(Name, Mode)`
- Beispiel
 - `FILE* fp = fopen("daten.dat", "r");`
- Mode:
 - Lesen: `r`
 - (Über)Schreiben: `w`
 - Anfügen: `a`
 - Binary (Windows): `b`

Fehlerabfrage

- Rückgabewert:
 - NULL= Fehler
- Anzeige:
 - `#include <stdio.h>`
 - `perror(Text)`
- Beispiel
 - `perror("Fehler beim öffnen der Datei");`
 - Fehler beim öffnen der Datei: File not found.

Datei schliessen

- Syntax
 - `fclose(FILE*)`
- Beispiel
 - `fclose(fp);`

Datei lesen: Textdateien

- Lesen:
 - `fscanf(FILE*, Format, Ziel* [, Ziel* [..]])`
- Beispiel:
 - `char c; fscanf(fp, "%c", &c);`
- Am Ende?
 - (boolean) `feof(FILE*)`
- Beispiel
 - `while (!feof(fp)) {..}`

Beispiel "cat"

Ausgabe: Ha

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

```
int main(void) {
```

```
    char c;
```

```
    FILE* fp = fopen("testcat.c", "r");
```

```
    if (fp == NULL) {
```

```
        perror("Datei kann nicht geöffnet werden");
```

```
        return 1;
```

```
    }
```

```
    while (!feof(fp)) {
```

```
        fscanf(fp, "%c", &c);
```

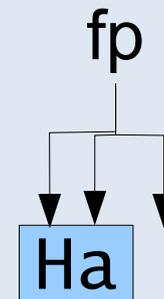
```
        printf("%c", c);
```

```
    }
```

```
    fclose(fp);
```

```
    return 0;
```

```
}
```



feof(fp): ~~false~~

C: ~~A~~

Datei schreiben

- Syntax:
 - `fprintf(FILE*, Format, [Var [, Var[, ..]]]);`
- Beispiel
 - `fprintf(fp, "Ergebnis: %d\n", erg);`

Weitere Textdateien Funktionen

- `getline()`
 - Liest eine komplette Zeile
- `fgetc/fgets`
 - Liest ein Zeichen/String
 - (`gets` – nicht verwenden)
- `fputc/fputs`
 - Schreibt ein Zeichen/String
- Sonstiges:
 - `atoi`
 - `sprintf`

Position setzen

- Position auslesen:
 - `long ftell(FILE*)`
- Position setzen:
 - `int fseek(FILE*, offset, whence);`
 - whence:
 - `SEEK_SET`
 - `SEEK_CUR`
 - `SEEK_END`
- Beispiel:
 - `fseek(fp, 0, SEEK_END) // geht ans Ende der Datei`

Binärdaten lesen/schreiben

- Lesen:
 - `size_t fread(void*, size, nmemb, FILE*);`
- Schreiben:
 - `size_t fwrite(void*, size, nmemb, FILE*);`

Beispiel: structs speichern

```
typedef struct {
    int nr;
    int a[10];
    char b[10];
} DATEN;

void schreiben(char* datei, DATEN* dat, int anz) {
    FILE* fp = fopen(datei, "w");
    if (fp == NULL) {
        perror("Fehler beim schreiben");
        exit(1);
    }
    int gesch = fwrite(dat, sizeof(DATEN), anz, fp);
    if (gesch != anz) {
        printf("Von %d wurden nur %d geschrieben!\n",
            anz, gesch);
    }
    fclose(fp);
}
```

Beispiel: structs speichern

```
DATEN* lesen(char* datei, int* anz) {
    DATEN* ret = NULL;
    *anz = 0;
    FILE* fp = fopen(datei, "r");
    if (fp == NULL) {
        perror("Fehler beim lesen"); exit(1);}
    // Anzahl enthaltener Elemente rausfinden
    if (fseek(fp, 0, SEEK_END) != 0) {
        perror("Fehler beim suchen"); exit(1);}
    *anz = ftell(fp) / sizeof(DATEN);
    if (fseek(fp, 0, SEEK_SET) != 0) {
        perror("Fehler beim suchen"); exit(1);}
    // lesen
    ret = (DATEN*) malloc(sizeof(DATEN) * *anz);
    int gelesen = fread(ret, sizeof(DATEN), *anz, fp);
    if (gelesen != *anz) {
        perror("Fehlerhafte Anzahl gelesen"); exit(1);}
    fclose(fp);
    return ret;
}
```

Beispiel: structs speichern

- Probleme:
 - Was passiert mit Zeigern?

```
struct X {  
    char* y;  
    struct X* naechster;  
}
```

Standard Streams

- `stdin`, `stdout`, `stderr`
- Normalerweise:
 - `stdin` = Tastatur
 - `stdout` / `stderr` = Terminal
- `printf("x");` == `fprintf(stdout, "x");`
- Umleiten: "`<`" "`>`" "`>>`" "`2>`" "`2>>`" "`|`"
 - `./a.out <eingabe.txt >Ausgabe.txt`
 - `grep "Fehler" Ausgabe.txt | less`

File I/O

- Themen:
 - scanf
 - Dateien: öffnen, schliessen, lesen, schreiben..
 - Streams
- Mehr:
 - Betriebssysteme (3. Semester)

Fragen?