

I Ein-/Ausgabe

I.1 Überblick

- E-/A-Funktionalität nicht Teil der Programmiersprache
- Realisierung durch "normale" Funktionen
 - Bestandteil der Standard-Funktionsbibliothek
 - einfache Programmierschnittstelle
 - effizient
 - portabel
 - betriebssystemnah
- Funktionsumfang
 - Öffnen/Schließen von Dateien
 - Lesen/Schreiben von Zeichen, Zeilen oder beliebigen Datenblöcken
 - Formatierte Ein-/Ausgabe

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.1

Reproduktion jeder Art oder Vervielfältigung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.2 Standard Ein-/Ausgabe (2)

- Pipes
 - ◆ die Standardausgabe eines Programms kann mit der Standardeingabe eines anderen Programms verbunden werden
 - Aufruf
`prog1 | prog2`
- ! Die Umlenkung von Standard-E/A-Kanäle ist für die aufgerufenen Programme völlig unsichtbar
- automatische Pufferung
 - ◆ Eingabe von der Tastatur wird normalerweise vom Betriebssystem zeilenweise zwischengespeichert und erst bei einem **NEWLINE**-Zeichen (`'\n'`) an das Programm übergeben!

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.3

Reproduktion jeder Art oder Vervielfältigung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.2 Standard Ein-/Ausgabe

1

- Jedes C-Programm erhält beim Start automatisch 3 E-/A-Kanäle:
 - ◆ `stdin` Standardeingabe
 - normalerweise mit der Tastatur verbunden
 - Dateiende (**EOF**) wird durch Eingabe von **CTRL-D** am Zeilenanfang signalisiert
 - bei Programmaufruf in der Shell auf Datei umlenkbar
`prog <eingabedatei`
(bei Erreichen des Dateiendes wird **EOF** signalisiert)
 - ◆ `stdout` Standardausgabe
 - normalerweise mit dem Bildschirm (bzw. dem Fenster, in dem das Programm gestartet wurde) verbunden
 - bei Programmaufruf in der Shell auf Datei umlenkbar
`prog >ausgabedatei`
 - ◆ `stderr` Ausgabekanal für Fehlermeldungen
 - normalerweise ebenfalls mit Bildschirm verbunden

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.2

Reproduktion jeder Art oder Vervielfältigung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.3 Öffnen und Schließen von Dateien

- Neben den Standard-E/A-Kanälen kann ein Programm selbst weitere E/A-Kanäle öffnen
 - Zugriff auf Dateien
- Öffnen eines E/A-Kanals
 - Funktion `fopen`
 - Prototyp:

```
FILE *fopen(char *name, char *mode);
```

<code>name</code>	Pfadname der zu öffnenden Datei
<code>mode</code>	Art, wie die Datei geöffnet werden soll
<code>"r"</code>	zum Lesen
<code>"w"</code>	zum Schreiben
<code>"a"</code>	append: Öffnen zum Schreiben am Dateiende
<code>"rw"</code>	zum Lesen und Schreiben

 - Ergebnis von `fopen`:
Zeiger auf einen Datentyp `FILE`, der einen Dateikanal beschreibt
im Fehlerfall wird ein `NULL`-Zeiger geliefert

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.4

Reproduktion jeder Art oder Vervielfältigung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.3 Öffnen und Schließen von Dateien (2)

■ Beispiel:

```
#include <stdio.h>

main(int argc, char *argv[]) {
    FILE *eingabe;

    if (argv[1] == NULL) {
        fprintf(stderr, "keine Eingabedatei angegeben\n");
        exit(1);          /* Programm abbrechen */
    }

    if ((eingabe = fopen(argv[1], "r")) == NULL) {
        /* eingabe konnte nicht geöffnet werden */
        perror(argv[1]);  /* Fehlermeldung ausgeben */
        exit(1);          /* Programm abbrechen */
    }

    ... /* Programm kann jetzt von eingabe lesen */
}
```

■ Schließen eines E/A-Kanals

```
int fclose(FILE *fp)
```

➤ schließt E/A-Kanal `fp`

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.5

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.4 Zeichenweise Lesen und Schreiben (2)

■ Beispiel: copy-Programm

Aufruf: `copy Quelldatei Zieldatei`

```
#include <stdio.h>
                                Teil 1: Aufrufargumente
                                auswerten
main(int argc, char *argv[]) {
    FILE *quelle;
    FILE *ziel;
    int c;                        /* gerade kopiertes Zeichen */

    if (argc < 3) {
        fprintf(stderr, "Aufruf: copy Quelle Ziel");
        exit(1);                  /* Programm abbrechen */
    }

    if ((quelle = fopen(argv[1], "r")) == NULL) {
        perror(argv[1]);          /* Fehlermeldung ausgeben */
        exit(1);                  /* Programm abbrechen */
    }

    if ((ziel = fopen(argv[2], "w")) == NULL) {
        perror(argv[2]);          /* Fehlermeldung ausgeben */
        exit(1);                  /* Programm abbrechen */
    }

    /* ... */
}
```

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.7

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.4 Zeichenweise Lesen und Schreiben

■ Lesen eines einzelnen Zeichens

◆ von der Standardeingabe

```
int getchar( )
```

➤ lesen das nächste Zeichen

➤ geben das gelesene Zeichen als `int`-Wert zurück

➤ geben bei Eingabe von `CTRL-D` bzw. am Ende der Datei `EOF` als Ergebnis zurück

◆ von einem Dateikanal

```
int getc(FILE *fp )
```

■ Schreiben eines einzelnen Zeichens

◆ auf die Standardausgabe

```
int putchar(int c)
```

➤ schreiben das im Parameter `c` übergebene Zeichen

➤ geben gleichzeitig das geschriebene Zeichen als Ergebnis zurück

◆ auf einen Dateikanal

```
int putc(int c, FILE *fp )
```

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.6

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.4 Zeichenweise Lesen und Schreiben (2)

... Beispiel: copy-Programm

— Fortsetzung

```
/* ... */
                                Teil 2: kopieren
while ( (c = getc(quelle)) != EOF ) {
    putc(c, ziel);
}

fclose(quelle);
fclose(ziel);
}
```

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.8

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1.5 Zeilenweise Lesen und Schreiben

- Lesen einer Zeile von einem Dateikanal

```
char *gets(char *s)
```

- liest Zeichen von der Standardeingabe in das Feld `s` bis entweder `'\n'` oder `EOF` gelesen wurde
 - `'\n'` wird entfernt und `s` wird mit `'\0'` abgeschlossen
 - gibt bei `EOF` oder Fehler `NULL` zurück, sonst `s`
- !!! die Länge des Feldes `s` wird nicht überprüft
- wenn bis zu einem `'\n'` zu viele Zeichen kommen, wird unkontrolliert Speicher überschrieben
 - besonders gefährlich, wenn die Standardeingabe auf eine Internetverbindung umgelenkt ist
 - durch trickreiche Eingabe kann Programm mißbraucht werden
 - Angriffspunkt für Hacker

!!! Funktion `gets()` NICHT BENUTZEN

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.9

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1.5 Zeilenweise Lesen und Schreiben (3)

- ... Beispiel: copy-Programm aus 1.4 (erster Teil identisch)
— Fortsetzung mit zeilenweise Kopieren

```
/* ... */                                     Teil 2: kopieren
{ /* neuen Block beginnen, um Feld definieren zu können */
#define BUFSIZE 1024                          /* Größe des Zeilenpufferspeichers */
char buffer[BUFSIZE]; /* Feld für Zeilenpuffer anlegen */
while ( fgets(buffer, BUFSIZE, quelle) != EOF ) {
    fputs(buffer, ziel);
}
fclose(quelle);
fclose(ziel);
}
```

1.6 Kopieren beliebiger Blöcke

- Funktionen `fread` und `fwrite` (bei Bedarf siehe Manual)

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.11

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1.5 Zeilenweise Lesen und Schreiben (2)

- Lesen einer Zeile von der Standardeingabe

```
char *fgets(char *s, int n, FILE *fp)
```

- liest Zeichen von Dateikanal `fp` in das Feld `s` bis entweder `n-1` Zeichen gelesen wurden oder `'\n'` oder `EOF` gelesen wurde
- `s` wird mit `'\0'` abgeschlossen (`'\n'` wird nicht entfernt)
- gibt bei `EOF` oder Fehler `NULL` zurück, sonst `s`
- für `fp` kann `stdin` eingesetzt werden, um von der Standardeingabe zu lesen

- Schreiben einer Zeile

```
int fputs(char *s, FILE *fp)
```

- schreibt die Zeichen im Feld `s` auf Dateikanal `fp`
- für `fp` kann auch `stdout` oder `stderr` eingesetzt werden
- als Ergebnis wird die Anzahl der geschriebenen Zeichen geliefert

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.10

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1.7 Formatierte Ausgabe

1 Bibliotheksfunktionen — Prototypen (Schnittstelle)

```
int printf(char *format, /* Parameter */ ... );
int fprintf(FILE *fp, char *format, /* Parameter */ ... );
int sprintf(char *s, char *format, /* Parameter */ ... );
int snprintf(char *s, int n, char *format, /* Parameter */ ... );
```

- Die statt ... angegebenen Parameter werden entsprechend der Angaben im `format`-String ausgegeben
 - bei `printf` auf der Standardausgabe
 - bei `fprintf` auf dem Dateikanal `fp` (für `fp` kann auch `stdout` oder `stderr` eingesetzt werden)
 - `sprintf` schreibt die Ausgabe in das `char`-Feld `s` (achtet dabei aber nicht auf das Feldende -> potentielle Sicherheitsprobleme!)
 - `snprintf` arbeitet analog, schreibt aber maximal nur `n` Zeichen (`n` sollte natürlich nicht größer als die Feldgröße sein)

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.12

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Formatangaben

1.7 Formatierte Ausgabe

- Zeichen im `format`-String können verschiedene Bedeutung haben
 - normale Zeichen: werden einfach auf die Ausgabe kopiert
 - Escape-Zeichen: z. B. `\n` oder `\t`, werden durch die entsprechenden Zeichen (hier Zeilenvorschub bzw. Tabulator) bei der Ausgabe ersetzt
 - Format-Anweisungen: beginnen mit `%`-Zeichen und beschreiben, wie der dazugehörige Parameter in der Liste nach dem `format`-String aufbereitet werden soll
- Format-Anweisungen
 - `%d, %i` `int` Parameter als Dezimalzahl ausgeben
 - `%f` `float` oder `double` Parameter wird als Fließkommazahl (z. B. 271.456789) ausgegeben
 - `%e` `float` oder `double` Parameter wird als Fließkommazahl in 10er-Potenz-Schreibweise (z. B. 2.714567e+02) ausgegeben
 - `%c` `char`-Parameter wird als einzelnes Zeichen ausgegeben
 - `%s` `char`-Feld wird ausgegeben, bis `'\0'` erreicht ist

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.13

Reproduktion jeder Art oder Vervielfältigung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Bearbeitung der Eingabe-Daten

1.8 Formatierte Eingabe

- *White space* (Space, Tabulator oder Newline `\n`) bildet jeweils die Grenze zwischen Daten, die interpretiert werden
 - *white space* wird in beliebiger Menge einfach überlesen
 - Ausnahme: bei Format-Anweisung `%c` wird auch *white space* eingelesen
- Alle anderen Daten in der Eingabe müssen zum `format`-String passen oder die Interpretation der Eingabe wird abgebrochen
 - wenn im `format`-String normale Zeichen angegeben sind, müssen diese exakt so in der Eingabe auftauchen
 - wenn im `format`-String eine Format-Anweisung (`%...`) angegeben ist, muß in der Eingabe etwas hierauf passendes auftauchen
 - ➔ diese Daten werden dann in den entsprechenden Typ konvertiert und über den zugehörigen Zeiger-Parameter der Variablen zugewiesen
- Die `scanf`-Funktionen liefern als Ergebnis die Zahl der erfolgreich an die Parameter zugewiesenen Werte

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.15

Reproduktion jeder Art oder Vervielfältigung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1.8 Formatierte Eingabe

1.8 Formatierte Eingabe

1 Bibliotheksfunktionen — Prototypen (Schnittstelle)

```
int scanf(char *format, /* Parameter */ ...);  
int fscanf(FILE *fp, char *format, /* Parameter */ ...);  
int sscanf(char *s, const char *format, /* Parameter */ ...);
```

- Die Funktionen lesen Zeichen von `stdin` (`scanf`), `fp` (`fscanf`) bzw. aus dem `char`-Feld `s`.
- `format` gibt an, welche Daten hiervon extrahiert und in welchen Datentyp konvertiert werden sollen
- Die folgenden Parameter sind Zeiger auf Variablen der passenden Datentypen (bzw. `char`-Felder bei Format `%s`), in die die Resultate eingetragen werden
- relativ komplexe Funktionalität, hier nur Kurzüberblick für Details siehe Manual-Seiten

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.14

Reproduktion jeder Art oder Vervielfältigung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Format-Anweisungen

1.8 Formatierte Eingabe

<code>%d</code>	<code>int</code>
<code>%hd</code>	<code>short</code>
<code>%ld</code>	<code>long int</code>
<code>%lld</code>	<code>long long int</code>
<code>%f</code>	<code>float</code>
<code>%lf</code>	<code>double</code>
<code>%Lf</code>	<code>long double</code>
analog auch <code>%e</code> oder <code>%g</code>	
<code>%c</code>	<code>char</code>
<code>%s</code>	String, wird automatisch mit <code>'\0'</code> abgeschl.

- nach `%` kann eine Zahl folgen, die die maximale Feldbreite angibt
 - `%3d` = 3 Ziffern lesen
 - `%5c` = 5 char lesen (Parameter muß dann Zeiger auf `char`-Feld sein)
 - `%5c` überträgt exakt 5 char (hängt aber kein `'\0'` an!)
 - `%5s` liest max. 5 char (bis *white space*) und hängt `'\0'` an
- Beispiele:

```
int a, b, c, d, n;  
char s1[20]="XXXXXX", s2[20];  
n = scanf("%d %2d %3d %5c %s %d",  
          &a, &b, &c, s1, s2, &d);  
  
Eingabe: 12 1234567 sowas humm  
Ergebnis: n=5, a=12, b=12, c=345  
           s1="67 soX", s2="was"
```

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-EA.doc 1998-07-22 14.21

I.16

Reproduktion jeder Art oder Vervielfältigung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.