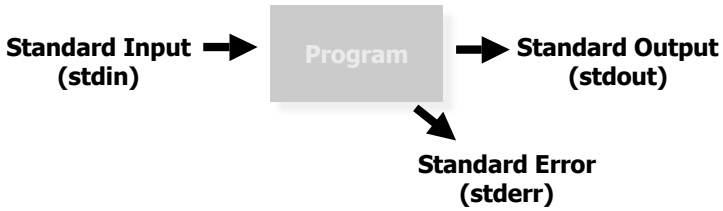



Grundlagen der Programmierung in C
File I/O

Wintersemester 2005/2006
G. Zachmann
Clausthal University, Germany
zach@in.tu-clausthal.de


Erinnerung: stdin / stdout / stderr

- Jeder Prozeß hat stdin, stdout, stderr



- Normalerweise verbunden mit Tastatur bzw. Terminalfenster
- Oder durch Redirection verbunden mit Files!

```
% program parameters < infile > outfile
```

  - `stderr` kommt weiterhin im Terminal raus

Ausgabe auf stdout

- Einzelnes Zeichen:

```
putchar('x');
char c; ...; putchar(c);
```

- Eine Zeile:

```
puts("text");
puts( zeiger-auf-string );
```

- Formatiert:

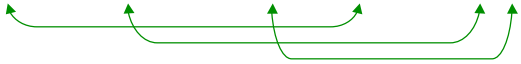
```
printf( "format-string", arg1, arg2, … );
```

- Befinden sich in der sog. libc (C standard library, glibc); wird immer automatisch dazugelinkt

Der Format-String

- Normale Zeichen und mit % eingeleitete *Formatierungsanweisung*
- Korrespondenz zwischen %-Anweisung und Argumenten:

```
printf( "Blub %· Bla %· Blubber …", arg1, arg2, … );
```



- Häufige %-Formatangaben:

%-Angabe	Formatierung
%d	**int**
%u	**unsigned int**
%f	**float oder double**
%c	**char**
%s	**string**
%x	**Hexadezimal-Zahl**

New-line

- Codiert durch spezielles Zeichen: `\n`
- Beispiele:

```
printf("Bla   %d\n", x );
printf("blub    %f\n"
       "blubber %f\n", f1, f2 );
```


Lesen von stdin

- Einzelnes Zeichen

```
char c; ...; c = getchar();
```

- Ganze Zeile

```
gets( zeiger-auf-string );
```

- Formatiert

```
scanf( "format-string", arg1, arg2, … );
```

  - Formatstring (fast) wie bei printf
  - Achtung: am Schluß niemals `\n` lesen wollen!

Beispiele zu `scanf`

- 3 Floats einlesen:

```
printf("Geben Sie den Vektor ein: ");
float v1, v2, v3;
scanf( "%f %f %f", &v1, &v2, &v3 );
```

- Space im Formatstring matcht beliebig viele Whitespaces!
- Verwendung:
  - Von Tastatur einlesen:

```
% program
Geben Sie den Vektor ein: □
```

  - Mit I/O-Redirection aus File lesen:

```
% program < vector.txt
Geben Sie den Vektor ein:
%
```


Interaktives Beispiel zu printf / scanf

- ...

- Werte lesen bis Input leer:
  - Man-Page: scanf liefert EOF falls kein Input mehr da
  - Siehe Man-Page für weitere Fälle!

```
int i;
int success = scanf("%d", &i );
while ( success == 1 && success != EOF )
{
    // do something ...
    success = scanf("%d", &i );
}
```

Sorgt dafür, daß die gelesene Zeichenkette als Int interpretiert wird.