Zum Umgang mit Dateien in C:

- Für das **Schreiben** von Daten in formatierte (d.h.: in ASCII-) Dateien mit **`fprintf()`** gibt es voreingestellte (default) Formate, die für eine Vielfalt von Anwendungen genügen. Die am häufigsten verwendeten sind **`%d`** (für **`int`**), **`%f`** (für **`float`**) und **`%s`** (für **`char`**-Zeichenketten). Soll eine konkrete Ausgabeform eingesetzt werden (z.B. zur Erzeugung von Tabellen, zur Übertragung einer bestimmten Anzahl von Nachkommastellen o.ä.), so kann (wie bei **`printf()`**) zwischen dem Prozentzeichen (**%**) und dem Formatcode (**`d`**, **`f`**, **`s`** etc.) eine Formatangabe erfolgen, bestehend aus einer Zahl für die Feldweite und einem Punkt mit einer Zahl für die Genauigkeit. Sie hat also die Form:

  %< Feldweite>.<Genauigkeit><Formatcode>

- Unter Feldweite (auch: Feldbreite, Feldlänge) versteht man die Mindestzahl von Stellen (inkl. Vorzeichen), die für die Ausgabe (z.B. im Sinne einer Tabelle) reserviert werden. Kürzere Zahlen werden mit Leerzeichen aufgefüllt; wird der Feldweite eine Null (**`0`**) vorgesetzt, so werden statt dessen Nullen (der Zahl oder dem String) vorausgeschickt. Längere Ausgaben werden durch entsprechend mehr Zeichen dargestellt (und verlassen ggf. ein vorgesehenes Tabellenschema). Ein vorangestelltes Minuszeichen (**`-`**) bewirkt jeweils linksbündige Ausgabe (Voreinstellung ist rechtsbündig).

- Die Genauigkeit ist hilfreich vor allem bei der Ausgabe von **`float`**; sie gibt die Anzahl der auszugebenden Nachkomma-Ziffern vor. Bei **`int`**-Werten sind es die Minimal-Anzahl von Ziffern, ggf. aufgefüllt mit führenden Nullen (unabhängig vom eingestellten Füllzeichen, s.o.); wird als Genauigkeit für **`int`** null angegeben, wird der Wert ´0´ nicht ausgegeben. Bei Ausgabe von Zeichenketten (Strings) bedeuten Feldweite die max. reservierte Anzahl auszugebender Zeichen (d.h., ggf. vorhandener Rest-Text wird ignoriert), Genauigkeit die exakte Anzahl auszugebender Zeichen (evtl. mit Leerzeichen oder Nullen aufgefüllt, s.o.).

- Feldweite und Genauigkeit können auch variabel gehalten werden, indem an ihrer Stelle ein Stern (**`*`**) eingestellt wird. Dann muß in der **`fprintf()`**-Anweisung für jeden Stern eine **`int`**-Variable, ein Makro oder eine Zahl vorgesehen sein. Werden sie (beide oder einzeln) ausgelassen (wobei die Anwesenheit bzw. Lage des Punktes über ihre Bedeutung entscheidet), so wird das voreingestellte Format verwendet.

  Beispiel:

```
#define WIDE 6
int     ii=1;
fprintf (filnam,"\"%-*.*sfor%*d\" %3.1f\n", WIDE, WIDE-1,
                                    "Diners", ii+1, ii, (float)ii+1);
```

  Das Ergebnis ist die Ausgabe:

```
"Diner for 1" 2.0
```

- Das **Lesen** aus ASCII-Dateien erfolgt vorzugsweise mit der Anweisung **`fscanf()`**; diese liest so viele Datenobjekte aus, wie in ihrer Formatangabe vorgesehen sind und legt die Werte in den bereitgestellten Argumenten ab, die alle Zeigervariablen sein müssen.

- Die Formatcodes für **`fscanf()`** sind größtenteils die gleichen wie jene für **`fprintf()`**. Für Zahlen braucht / sollte man keine Feldweite und Genauigkeit angeben, damit Zahlen vollständig (bis zum nächsten Trennzeichen, z.B. Leerzeichen) ausgelesen werden.

Bei Zeichenketten (%s) dagegen, für die, naturgemäß, keine Trennzeichen definiert sind, ist die Feldweite nützlich; sie gibt an, wie viele Zeichen (maximal) gelesen werden sollen.

- „Weiße“ Zeichen (Leerzeichen, Tabulatoren, Zeilenwechsel) dienen als Trennzeichen, wenn mehrere Zahlenwerte hintereinander in der Datei stehen; sie werden beim Lesen übergangen. Explizit gelesen werden sie nur, wenn die gelesenen Daten als Zeichenketten (%s) interpretiert werden. Interpunktionszeichen (Kommata, Semikola) müssen im Format explizit vorgesehen sein (z.B.: **`fscanf (filnam, "%f ; %f", &x,&y);`**), sonst führen sie zu Fehlern.
- Interessant für manche Aufgaben ist, daß ein Stern (**`*`**) zwischen dem %-Zeichen und dem Datentyp (vergleichbar der variablen Feldweite bei **`fprintf()`**) bei **`scanf()`** bewirkt, daß die gelesenen Daten nicht in die bereitgestellte Variable übertragen werden.
- Über die Eingabe eines sog. **Scanset** (Suchmenge) kann man in eckigen Klammern (**`[]`**) eine Gruppe oder einen Bereich von Zeichen definieren, die als einzige zu berücksichtigen – bzw. bei vorangestelltem Dachzeichen (**`^`**) nicht zu berücksichtigen – sind; d.h.:

  `fscanf (filnam,"%*[^A-Z]s", string);`

  bedeutet (nach Definition einer Zeigervariablen **`string`**), daß aus der Datei (Datenstrom) **`filnam`** eine Zeichenkette – z.B.: **`"abcABCdefDEF"`**) solange ausgelesen werden soll, bis der erste Großbuchstabe (A-Z) gefunden wird (aber ohne diesen auszulesen, wegen **`^`** – z.B. hier: "**`abc`**"). Die ausgelesenen Zeichen sollen dann aber nicht in die (zuvor definierte Zeigervariable) **`string`** gespeichert werden (was durch **`*`** angezeigt wird); nur der Dateizeiger wird (vor den ersten auszulesenden Großbuchstaben) weiter positioniert.
- Die Anweisung

  `fgets (string, LENGTH, filnam);`

  liest aus der formatierten Datei eine ganze Zeile aus (bzw. ihren Rest, falls der Dateizeiger nicht am Zeilenanfang steht), bis einschließlich **`"\n"`** oder **`EOF`**, bzw. maximal die Anzahl von Zeichen, die durch **`LENGTH`** (als **`#define`** oder als Zahl) angegeben wird. Der Rückgabewert von **`fgets()`** ist die Adresse des gelesenen Strings (**`char *`**), andernfalls (bei Fehler oder **`EOF`**) ein **`NULL`**-Pointer.
- Die Datenübertragung vom Arbeitsspeicher in die Datei erfolgt über einen Puffer (Zwischenspeicher); dessen physische Leerung wird vom Betriebssystem gesteuert (und ausgelöst, wenn der Puffer voll ist, das Programm beendet werden soll o.ä.); **`fflush()`** sorgt nach **`fprintf()`** für eine Leerung des Datenpuffers zu einer vom Programmablauf bestimmten Zeit; dies ist z.B. wichtig, wenn Geschriebenes sofort wieder gelesen werden soll. Ein Aufruf von **`fflush()`** nach **`fscanf()`** ist nicht vorgesehen, sorgt aber (Compiler-abhängig, z.B. bei MS-VS) für korrekte Positionierung des Dateizeigers.
- Es ist zu beachten, daß beim Wechsel zwischen Datei-Lesen und -Schreiben die Plazierung des Dateizeigers nicht mehr zur Verfügung steht; sie kann wiedergewonnen werden durch Aufruf von **`fseek()`** oder **`rewind()`**. (Man beachte dabei die Warnung der MS-VS-Hilfe: „For streams opened in text mode, **`fseek`** has limited use, because carriage return–linefeed translations can cause **`fseek`** to produce unexpected results.“)

Nähreres zu allen obigen Anweisungen findet sich in der einschlägigen Literatur (s. Vorlesungsfolien) und in der On-Line-Hilfe der Entwicklungsumgebung.