Beispiel / Übung:

- Erstellen Sie ein kurzes, plattform-unabhängiges C-Programm („Konsolenanwendung“), das sich in die Reihe bereits gestarteter Kopien einordnet, sich nach 20 sec (Rechenzeit) abmeldet und sich nach weiteren 20 sec selbsttätig beendet.

  (Für Hedonist/inn/en: wahlweise vorzeitige Ausführung des jeweiligen Schrittes durch **<CR>** und Beendigung durch **<ESC>** über die Funktion **`_kbhit()`**.) `(Memo.exe)`
- Benutzen Sie bei der Programmierung ausschließlich Anweisungen aus dem Sprachumfang von C (d.h. u.a.: aktives Warten). Zur Simulation eines gemeinsamen Speicherbereichs können Sie eine formatierte Datei verwenden. Sie können sich dabei am Programmbeispiel für die Zeitmessung bei R/W-Operationen orientieren.
- Bauen Sie Ihr Programm so aus, daß es die benötigte Datei im aktuellen Verzeichnis einrichtet, (erst:) wenn es die erwartete Verzeichnisstruktur (**`"../dat/"`**) nicht vorfindet. `(MemoTest.exe)`
- Weitere Hinweise und Hilfen: s. Übungsblatt.

Übersicht: Programmiersprache C

Öffnen / Schließen einer (formatierten oder binären) Datei:

- **`FILE *fopen ( const char *filename, const char *mode );`**
- **`int fclose (FILE *fileP);`**`//Fehler=>Rueckgabe!=0`

Zulässig als **`*mode`**: **`r|w|a [+] [b]`** (Reihenfolge beliebig)

`r` : vorhandene Datei nur zum Lesen öffnen

`r+`: vorhandene Datei zum Lesen und Schreiben öffnen

`w` : neue Datei nur zum Schreiben erzeugen

`w+`: neue Datei zum Schreiben und Lesen erzeugen

`a` : Datei öffnen (ggf. erzeugen), um Daten anzuhängen

`a+`: Datei öffnen (ggf. erzeugen), um Daten anzuhängen und beliebig zu lesen (**`"a"`**/**`"a+"`**:Schreibzugriff ab Dateiende)

`b` : Daten-Übertragung in binärer Form: schneller (Felder, Pixel, Strukturen) , genauer (**`float`**); nicht editierbar

`(MemoTest.exe)`

Übersicht: Programmiersprache C

Formatiertes Schreiben / Lesen bei geöffneter Datei:

- `int fprintf(FILE *fileP, const char *format [, var1, ...]);`
- `int fscanf (FILE *fileP, const char *format [, &var1, ...]);`

`FILE *fileP` : Zeiger auf Struktur mit Datei-Zustand

Lesen ein Zeichen aus (geöffneter) Datei (vgl. **`getchar`**):

- `int getc (FILE *fileP);//auch:fgetc(fileP);`

Lesen bis zum nächsten **`'\n'`**(inkl.), max. **`nChar`** Zeichen:

- `char *fgets (char *line, int nChar, FILE *fileP);`

(Rückgabewerte: ASCII-Wert bzw. **`line`**–Startadresse)

Übersicht: Programmiersprache C

C-Funktionen zur binären Datenübertragung:

- `size_t fwrite (const void *buffer, size_t size, size_t count, FILE *fileP);`
- `size_t fread (void *buffer, size_t size, size_t count, FILE *fileP);`

Parameter:

`*buffer` :	Speicher-Adresse der / für die Daten
`count` :	Anzahl zu übertragender Datenobjekte
`size` :	Größe der Datenobjekte in Byte (**`sizeof()`**)
`size_t` :	**`unsigned int`** (definiert in **`<stddef.h>`**)
`*fileP` :	Filepointer

Wichtige Aspekte beim Umgang mit Dateien:

- Datenübertragung in Datei erfolgt gepuffert und vom BS gesteuert, physisch ggf. erst bei Programm-Beendigung.

  Abhilfe, z.B. um Geschriebenes lesen zu können:

  ```
  int fflush(FILE *fileP); //Fehler=>Rueckgabe!=0
  ```

  veranlaßt Leerung des Datenpuffers vom Programm aus.

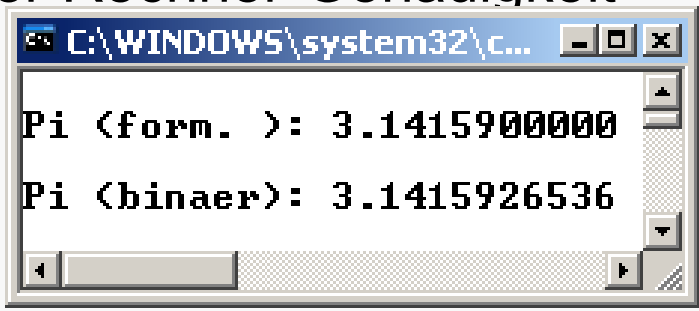
- Bei Wechsel zwischen Datei-Lesen und -Schreiben steht d. Plazierung des Dateizeigers nicht mehr zur Verfügung; Zeiger-Positionierung an Datei-Anfang:

  ```
  void rewind(FILE *fileP);
  ```

  (Gezielte Positionierung mit **`fseek()`** unsicher: s.Notiz).

Übersicht: Programmiersprache C

Beispiel: Speicherung von π in maximaler Rechner-Genauigkeit




```
int main(void) /*Aus: Pi4File.c*/
{ FILE  *memo;
  double Pi=4*atan(1.),dummy=0.;
  memo=fopen("PiFile.txt","w+");
  fprintf(memo,"%7.5lf",Pi);fflush(memo);rewind(memo);
  fscanf(memo,"%lf",&dummy);                rewind(memo);
  printf ("Pi (form.) : %12.10lf \n\r",  dummy);
  fclose (memo); /* ------------------------------ */
  memo=fopen("PiFile.txt","w+b");
  fwrite(&Pi,  sizeof(double), 1, memo);
                          fflush(memo);  rewind(memo);
  fread(&dummy,sizeof(double), 1, memo); rewind(memo);
  printf ("Pi (binaer): %12.10lf \n\r",  dummy);
  fclose (memo); _getch();  return (0); }
```

Beispiel / Übung:


```
/*              MemoTest.c (gekuerzt)                    */
int main(void)
{ int j1=0, j2=WDHLG;
  FILE *memo;
  clock_t tj1, tj2;

 printf ("Aris Christidis:\n\r");
 printf ("Formatierte Datei (fscanf/fprintf):\n\r");
 _getch(); tj1 = clock();
 if ((memo=fopen("../dat/Memo.txt","r+"))==NULL &&
     (memo=fopen("../dat/Memo.txt","w+"))==NULL)
    return(-1);
 for (j1=0; j1<WDHLG; j1++)
 { fprintf(memo,"%10d", j1); fflush(memo); rewind(memo);
   fscanf(memo,"%10d", &j2); rewind(memo);
 }
 tj2 = clock(); fclose (memo);
 printf ("Zeit fuer %d RW-Operationen: %.2lf msec\n\n\r",\
             j1, ((double)(tj2-tj1)/CLOCKS_PER_SEC)*1000);
 /*...*/ _getch(); return(0);
}
```

```
#include <stdio.h>
#include <conio.h>
#include <time.h>
#define WDHLG 10000
```

Anmerkungen zu `MemoTest.c`:

- C-Konvention: „Kurzschluß-Auswertung“ von **`if`**-Abfragen (Ggs.: „vollständige Auswertung“). Bedingungen werden nur solange ausgewertet, wie sich ihr Ergebnis ändern kann, d.h.:

  in **`||`**-Verknüpfgn bis zum ersten Ausdruck mit Ergebnis **`!=0`**
  in **`&&`**-Verknüpfgn bis zum ersten Ausdruck **`=0`** (bzw. **`NULL`**).

- Unterbrechungszeichen für Zeilenwechsel innerhalb eines Ausdrucks: „**`\`**“ (bei Parameter-Aufzählung unnötig)