7. Dateizugriff

Entwickeln Sie ein Kopierprogramm `swfcp`, das (ähnlich wie `cp`) Dateien kopiert. Zur Vereinfachung soll es nur zwei Aufrufargumente auswerten und als Namen der Quell- und Zieldateien interpretieren. Es soll am Ende eine Statistik ausgeben und dabei mitteilen, wie viele Zeichen gelesen und wie viele geschrieben wurden:

```
student@swfdebian:~$ swfcp input.txt output.txt
  345 bytes read from input.txt
  345 bytes written to output.txt
```

Das folgende Programm (`swfcp.c`) wertet die ersten beiden Aufrufparameter aus und erzeugt die Statistik; der interessante Teil (das Kopieren) fehlt aber noch:

```
#include <stdio.h>                  #include <sys/stat.h>
#include <stdlib.h>                 #include <fcntl.h>
#include <sys/types.h>              #include <errno.h>

int main (int argc, char* argv[]) {
  if (argc != 3) {
    printf ("%s: Falsche Zahl Aufrufargumente\n", argv[0]);
    exit (1);                       // Fehler!
  }

  int total_read    = 0;
  int total_written = 0;

  char *inputfilename  = argv[1];    // 1. Argument: Eingabedatei
  char *outputfilename = argv[2];    // 2. Argument: Ausgabedatei

  // The magic happens here

  printf ("%5d bytes read from %s\n",  total_read,    inputfilename);
  printf ("%5d bytes written to %s\n", total_written, outputfilename);
}
```

(Diese Datei können Sie von der Kursseite oder aus dem Moodle herunterladen.)

Weitere Hinweise zur Implementierung:

- Beschreibungen der benötigten Funktionen (die ihrerseits Syscalls verwenden) finden Sie im Skript (ab S. 204); Details verraten auch die Manpages; da es teilweise mehrere Manpages mit demselben Namen gibt, geben Sie an, auf welches Handbuch (Nr. 2) zugegriffen werden soll:

  ```
  man 2 write
  ```

- Lesen Sie die Quelldatei *blockweise* in einer Schleife ein; als Blockgröße verwenden Sie 4096.
- Werten Sie die Anzahl der gelesenen Zeichen bzw. geschriebenen Zeichen aus, um `total_read` und `total_written` zu aktualisieren.
- Prüfen Sie auf alle Fehler, die im Programmverlauf auftreten können. Funktionen geben i.d.R. den Wert `-1` zurück, wenn ein Fehler aufgetreten ist – den Fehlergrund erfahren Sie dann über einen Blick in errno. Lesen Sie die Manpage dazu (`man errno`). Beachten Sie, dass `errno` durch „gelungene“ Aufrufe von Bibliotheksfunktionen auf `0` zurück gesetzt wird, weswegen der folgende Code-Block nicht funktionieren wird:

  ```
  int ret = some_function();
  if (ret == -1) {
    printf ("DEBUG: Fehler %d\n", errno);     // printf() setzt errno auf 0
    if (errno == EEXIST) {
      printf ("DEBUG: Datei existiert schon.\n"); exit (1);
    }
  }
  ```