FH-OÖ Hagenberg/HSD
Betriebssysteme 3, WS 2025
Übung 4

Name:	**Marco Söllinger**	Aufwand in h:	**4**
Mat.Nr:	**s2410306011**	Punkte:	
Übungsgruppe:	**Gruppe 1**	korrigiert:	

Aufgabe 1: Elementare E/A-Funktionen

Schreiben Sie ein C-Programm `append.c`, das zwei Dateinamen auf der Kommandozeile erwartet und dann unter Verwendung der elementaren E/A-Funktionen (`stdio.h` und `stdlib.h` dürfen nicht inkludiert werden!) den Inhalt der zuerst angegebenen Datei an die zweite Datei anhängt.

- Verwenden Sie zum Lesen und Schreiben einen Puffer der Größe 512 Bytes.
- Geben Sie am Ende auf der Standardausgabe die Anzahl der angehängten Bytes aus.
- Führen Sie weiters eine entsprechende Fehlerbehandlung durch, überprüfen Sie die übergebenen Argumente und geben Sie entsprechende Fehlertexte auf die Standardfehlerausgabe (standard error) aus.
- Existiert die zweite Datei nicht, so wird sie neu erzeugt!

Hinweis:
Mit dem Befehl `truncate -s [Groesse] [Datei]` kann die Groesse einer Datei geändert werden. Die Groesse kann in Byte angegeben werden, oder man verwendet Groessenangabe wie K (Kilobyte), M (Megabyte), G (Gigabyte).

Beispiel:

```
truncate -s 1M datei.txt //Datei mit 1 MByte wird angelegt.
```

Allgemeine Hinweise: Legen Sie bei der Erstellung Ihrer Übung großen Wert auf eine **saubere Strukturierung** und auf eine **sorgfältige Ausarbeitung!** Dokumentieren Sie alle Schnittstellen und versehen Sie Ihre Algorithmen an entscheidenden Stellen ausführlich mit Kommentaren! Testen Sie ihre Implementierungen ausführlich! Geben Sie den **Testoutput** mit ab!

1

Beispiel 1

In dieser Aufgabe soll ein Programm erstellt werden, welches den Inhalt einer Textdatei in eine andere Textdatei kopiert werden.
Dabei muessen die speziall Faelle beachtet werden:
- Die Quelldatei existiert nicht.
- Die Zieldatei existiert nicht(erstellen).
- Argumente Anzahl stimmt nicht.
- Quelldatei und Zieldatei sind identisch.

Die Implementierung darf nur direkte Systemaufrufe verwenden (open, read, write, close, ...).
Mit der Ausnahme von der string.h Bibliothek fuer String Operationen.

1.1 Code

append.c

```c
#include <fcntl.h>
#include <stddef.h>
#include <string.h>
#include <sys/stat.h>
#include <unistd.h>

#define _STR_HELPER(x) #x
#define _STR(x) _STR_HELPER(x)

#define ERR_LOG(msg)                                                          \
  do {                                                                        \
    const char *_m = (msg);                                                   \
    size_t _len = strlen(_m);                                                 \
    write(STDERR_FILENO, __FILE__, sizeof(__FILE__) - 1);                     \
    write(STDERR_FILENO, ":", 1);                                             \
    write(STDERR_FILENO, _STR(__LINE__), sizeof(_STR(__LINE__)) - 1);         \
    write(STDERR_FILENO, ": ", 2);                                            \
    write(STDERR_FILENO, _m, _len);                                           \
  } while (0)

#define ERR_LOG_WITHOUT_LINE(msg)                                             \
  do {                                                                        \
    const char *_m = (msg);                                                   \
    size_t _len = strlen(_m);                                                 \
    write(STDERR_FILENO, _m, _len);                                           \
  } while (0)

static short const EXIT_FAILURE = 1;
static short const EXIT_SUCCESS = 0;

static char const *const USAGE_CMD =
    "useage: ./append.c <inputfile> <outputfile>\n";
static char const *const ERR_OPEN = "error in open: ";
static char const *const ERR_CLOSE = "error in close: ";
static char const *const ERR_WRITE = "error in write";
static char const *const ERR_READ = "error in read";
static char const *const ERR_BUFFER_OVERFLOW = "error: buffer overflow\n";
static char const *const ERR_INPUT_OUTPUT_SAME =
    "error: input and output file must be different\n";

static char const *const OUTPUT_MESSAGE = "Total bytes written: ";

static size_t const BUFFER_SIZE = 512;

static int convertToString(unsigned int x, char *buf, size_t bufsize) {
  char tmp[32];
  size_t n = 0;

  if (bufsize == 0)
    return -1;

  do {
    tmp[n++] = (char)('0' + (x % 10));
```

```
54      x /= 10;
55    } while (x != 0);
56
57    if (n + 1 > bufsize)
58      return -1;
59
60    // Reverse into output
61    for (size_t i = 0; i < n; ++i) {
62      buf[i] = tmp[n - 1 - i];
63    }
64    buf[n] = '\0';
65
66    return (int)n;
67  }
68
69  int main(int argc, char *argv[]) {
70    if (argc != 3) {
71      ERR_LOG(USAGE_CMD);
72      return EXIT_FAILURE;
73    }
74
75    if (strcmp(argv[1], argv[2]) == 0) {
76      ERR_LOG(ERR_INPUT_OUTPUT_SAME);
77      return EXIT_FAILURE;
78    }
79
80    char buffer[BUFFER_SIZE];
81    unsigned int totalBytesWritten = 0;
82
83    umask(0);
84
85    // ---------- Open Files ---------
86
87    int readfile = open(argv[1], O_RDONLY);
88    if (readfile == -1) {
89      ERR_LOG(ERR_OPEN);
90      ERR_LOG_WITHOUT_LINE(argv[1]);
91      ERR_LOG_WITHOUT_LINE("\n");
92      close(readfile); // keine neue Fehlerbehandlung
93      return EXIT_FAILURE;
94    }
95
96    int writefile =
97        open(argv[2], O_WRONLY | O_CREAT | O_APPEND, S_IRUSR | S_IWUSR);
98    if (writefile == -1) {
99      ERR_LOG(ERR_OPEN);
100     ERR_LOG_WITHOUT_LINE(argv[2]);
101     ERR_LOG_WITHOUT_LINE("\n");
102     close(readfile);  // keine neue Fehlerbehandlung
103     close(writefile); // keine neue Fehlerbehandlung
104     return EXIT_FAILURE;
105   }
106
107   // ---------- Read from input and write to output ----------
108
109   ssize_t bytesRead;
110   do {
111     bytesRead = read(readfile, buffer, BUFFER_SIZE);
112     if (bytesRead == -1) {
113       ERR_LOG(ERR_READ);
114       ERR_LOG_WITHOUT_LINE("\n");
115       close(readfile);  // keine neue Fehlerbehandlung
116       close(writefile); // keine neue Fehlerbehandlung
117       return EXIT_FAILURE;
118     }
119
120     ssize_t bytesWritten = write(writefile, buffer, bytesRead);
121     if (bytesWritten == -1) {
122       ERR_LOG(ERR_WRITE);
123       ERR_LOG_WITHOUT_LINE("\n");
124       close(readfile);
125       close(writefile);
126       return EXIT_FAILURE;
127     }
128     totalBytesWritten += (unsigned int)bytesWritten;
129   } while ((size_t)bytesRead == BUFFER_SIZE);
```

```
130
131   // ---------- Close Files ---------
132
133   if (close(readfile) == -1) {
134     ERR_LOG(ERR_CLOSE);
135     ERR_LOG_WITHOUT_LINE(argv[1]);
136     ERR_LOG_WITHOUT_LINE("\n");
137     close(writefile); // keine neue Fehlerbehandlung
138     return EXIT_FAILURE;
139   }
140
141   if (close(writefile) == -1) {
142     ERR_LOG_WITHOUT_LINE(ERR_CLOSE);
143     ERR_LOG_WITHOUT_LINE(argv[2]);
144     ERR_LOG("\n");
145     return -1;
146   }
147
148   char byteCountStr[32];
149   int written =
150       convertToString(totalBytesWritten, byteCountStr, sizeof(byteCountStr));
151   if (written == -1) {
152     ERR_LOG(ERR_BUFFER_OVERFLOW);
153     return EXIT_FAILURE;
154   }
155   write(STDOUT_FILENO, OUTPUT_MESSAGE, strlen(OUTPUT_MESSAGE));
156   write(STDOUT_FILENO, byteCountStr, (size_t)written);
157   write(STDOUT_FILENO, "\n", 1);
158
159   return EXIT_SUCCESS;
160 }
```

1.2 Test

Zum Testen wurde ein bash script erstellt, welches die verschiedenen Faelle testet.

test.sh

```
1  #!/usr/bin/env bash
2
3  gcc -o append append.c -Wall -Wextra
4
5  echo "Run with no arguments:"
6
7  ./append
8
9  echo "########################################"
10 echo "Run with one argument:"
11
12 ./append test1.txt
13
14 echo "########################################"
15 echo "Run with three argument:"
16
17 ./append test1.txt test2.txt test3.txt
18
19 echo "########################################"
20 echo "Run with non existent input file:"
21
22 ./append nonExist.txt output.txt
23
24 echo "########################################"
25 echo "Run with a file without content (Input: test1.txt | Output: empty.txt):"
26 rm test1.txt
27 touch test1.txt #create empty input file
28 rm empty.txt
29 touch empty.txt
30
31 ./append test1.txt empty.txt
32
33 echo "########################################"
34 echo "Run with indentical Input and Output file (Input: test1.txt | Output: test1.txt):"
35 # test1.txt already exists from previous step
```

```

./append test1.txt test1.txt

echo "########################################"
echo "Run with existent output file (Input: test2.txt | Output: output2.txt):"
rm test2.txt
echo -e "Line 1\nLine 2\nLine 3" >test2.txt
rm output2.txt
touch output2.txt #create empty output file

./append test2.txt output2.txt

echo "########################################"
echo "Run with existent output file and appent (Input: test3.txt | Output: output3.txt):"
rm test3.txt
echo -e "First Line\nSecond Line\nThird Line" >test3.txt
rm output3.txt
echo "Already inside" >output3.txt

./append test3.txt output3.txt

echo "########################################"
echo "Run with nonexistent output file (Input: test4.txt | Output: output4.txt):"
rm test4.txt
echo -e "Alpha\nBeta\nGamma" >test4.txt
rm output4.txt #ensure output file does not exist

./append test4.txt output4.txt

echo "########################################"
echo "Run with large output file (Input: test5.txt | Output: output5.txt):"
rm test5.txt

# print 1000 lines with 10 characters each to test5.txt
for ((i = 0; i < 1000; i++)); do
  echo "0123456789" >>test5.txt # print 11 characters per line
done

rm output5.txt

./append test5.txt output5.txt
```

Terminal Output

```
flashfish@fedora-4 ~/D/R/F/B/Uebung04 (main)> ./test.sh
Run with no arguments:
append.c:71: useage: ./append.c <inputfile> <outputfile>
########################################
Run with one argument:
append.c:71: useage: ./append.c <inputfile> <outputfile>
########################################
Run with three argument:
append.c:71: useage: ./append.c <inputfile> <outputfile>
########################################
Run with non existent input file:
append.c:89: error in open: nonExist.txt
########################################
Run with a file without content (Input: test1.txt | Output: empty.txt):
Total bytes written: 0
########################################
Run with indentical Input and Output file (Input: test1.txt | Output: test1.txt):
append.c:76: error: input and output file must be different
########################################
Run with existent output file (Input: test2.txt | Output: output2.txt):
Total bytes written: 21
########################################
Run with existent output file and appent (Input: test3.txt | Output: output3.txt):
Total bytes written: 34
########################################
Run with nonexistent output file (Input: test4.txt | Output: output4.txt):
Total bytes written: 17
########################################
Run with large output file (Input: test5.txt | Output: output5.txt):
Total bytes written: 11000
```

test1.txt

```
```

empty.txt

```
```

test2.txt

```
Line 1
Line 2
Line 3
```

output2.txt

```
Line 1
Line 2
Line 3
```

test3.txt

```
First Line
Second Line
Third Line
```

output3.txt

```
Already inside
First Line
Second Line
Third Line
```

test4.txt

```
Alpha
Beta
Gamma
```

output4.txt

```
Alpha
Beta
Gamma
```

Test5.txt und Output5.txt wurde nicht angehaengt, da es sich um eine grosse Datei handelt (11.000 Zeichen).