Aufgabenblatt 7
Systemprogrammierung (SoSe 2019)

Institut: Beuth Hochschule für Technik Berlin
Dozent: Prof. Dr. Christian Forler
Url: `https://lms.beuth-hochschule.de/`
Email: `cforler(at)beuth-hochschule.de`

Erstellen Sie ein **Makefile** mit denen sich alle Ihre C-Programme bauen lassen.

Aufgabe 1 (2 Punkte) Zugriffskontrollmatrix
Angenommen Gegeben ist ein Computersystem mit drei Benutzern: Alice, Bob, und Carol. Alice ist der Eigentümer des Programms *calculator.exe* welche Bob und Carol lesen und ausführen können. Bob ist der Eigentümer einer Datei *manuskript.txt*. Als Lektorin kann Carol diese Datei lesen und modifiziern. Weiterhin ist Carol Eigentümer der Datei *tagebuch.docx*.

Erstellen Sie eine Zugriffskontrollmatrix.

Aufgabe 2 (3 Punkte) Dateien Anlegen
Legen Sie die folgenden Dateien mit den entsprechenden Zugriffsrechten welches als 3-Tupel (Rechte des Eigentümers, Rechte der Gruppe, Rechte vom Rest) angegeben werden.

a) Datei `foo`: $(\{r, w, x\}, \{w\}, \emptyset)$

b) Datei `bar.txt`: $(\{\ r, w\}, \{r, w\}, \{r\})$

c) Datei `baz`: $(\{r, x\}, \{r, x\}, \{x\})$

Aufgabe 3 (4 Punkte) Anhängen von Dateien
Erstellen Sie ein Programm `append.c`, welches als Kommandozeilenparameter zwei Dateinamen übergeben werden. Verwenden Sie ausschließlich elementare I/O-Funktionen, um den Inhalt der zuerst angegeben Datei an die zweite Datei anzuhängen. Dies bedeutet, daß sie keine keine Funktionen verwenden sollen, die in der Header-Datei `stdio.h` deklariert sind.

Vergessen Sie nicht eine Usage-Meldung auszugeben, falls der Benutzer mehr oder weniger als zwei Parameter angibt.

Beispielaufrufe:

- `$ ./append foo.txt bar.txt`
- ```
  $ ./append foo.txt
  Usage: append <src-file> <dst-file>
  ```
  ```

Aufgabe 4 (4 Punkte) Dateien erstellen
Erstellen Sie mittels dem Systemcall `open()` die folgenden Dateien mit den angegebenen Zugriffsrechten.

- Datei: `foo.bin`; Zugriffsrechte: `rwxrwx--x`
- Datei: `bar.txt`; Zugriffsrechte: `rw-rw-r--`
- Datei: `fnord`; Zugriffsrechte: `r-xr-x---`
- Datei: `foobar`; Zugriffsrechte: `rw-r-----`

Aufgabe 5 (4 Punkte) Dateideskriptoren Duplizieren
Angenommen ein Prozess führt die folgenden Zeilen Code aus.

```
fd1 = open("foo.txt", flags);
fd2 = dup(fd1);
fd3 = open("foo.txt", flags);
fd4 = dup(fd3);
```

a) Zeichen Sie das durch die Aufrufe resultierende Skizze der Prozess-, Datei- und v-Node-Tabelle.

b) Schreiben Sie ein Programm, um Ihre Vermutung zu überprüfen. Hatten Sie mit Ihrer Vermutung recht?

Hinweis: Bei dup() handelt es sich um einen Systemcall.

Aufgabe 6 (4 Punkte) Dateiinhalt

```
#include <fcntl.h>
#include <unistd.h>

int main() {
  int i = 0x12345678;
  int fd = open("int.bin", O_CREAT | O_WRONLY, 0664);
  write(fd,&i,sizeof(int));
}
```

Geben Sie den Inhalt der Datei *int.bin*, nach Ausführung des obigen Programms, byteweise in Hex-Code an.

Aufgabe 7 (4 Punkte) Rückwärts Ausgeben
Erstellen Sie ein Programm `reverse.c` welches als Kommandozeilenparameter einen Dateinamen übergeben bekommt. Es soll die Datei zeilenweise einlesen und die Reihenfolge der Zeichen pro Zeile umgedrehter Reihenfolge ausgeben. Geben Sie eine Usage-Meldung aus, falls die Anzahl der übergeben Kommandozeilenparameter nicht stimmt.
Verwenden Sie ausschließlich die Funktionen `fopen()`, `fdfopen()` und `getline()`, um die Datei einzulesen und `write()`, um den Dateiinhalt auszugeben. Schreiben Sie wieterhin eine Funktion `void reverse(char *msg)` welche die Reihenfolge der Zeichenkette `msg` umdreht.

Beispiel:

```
$ cat foo.txt
Hallo Welt
Test123
$ ./reverse foo.txt
tleW ollaH
321tseT
```