Technische Informatik 2 | WS 2016/17 | C-01
Tutor: Marc/Bingbin

Habermann, Paul
Köster, Joschka
Rohde, Florian

Übungsblatt 6

1	2	3	4	5	$\sum$

Lösungsvorschlag
Abgabe: 12.12.2016

Aufgabe 1

Listing 1: mycp.c

```
10 // Check the files size to be abled to read 100 percent
11 size_t getFilesize(const char* filename) {
12    struct stat st;
13    stat(filename, &st);
14    return st.st_size;
15 }
```

Zuerst benötigen wir eine Hilfsmethode, die uns die Größe der Quelldatei extrahiert. Diese benötigen wir, damit wir wissen, wie lange wir kopieren müssen.

Listing 2: mycp.c

```
18 int main(int argc, char **argv){
19    int fd_in, fd_out;
20    size_t offset=0;
21    void *data;
22    static size_t pagesize = sysconf(_SC_PAGESIZE);
23
24    if (argc != 3) {
25      std::cerr << "Usage: " << argv[0] << " source destination" << std::endl;
26      return -1;
27    }
28
29    if ((fd_in = open(argv[1],O_RDONLY)) == -1){
30      perror("open");
31      return -1;
32    }
33
34    if ((fd_out = open(argv[2],O_WRONLY|O_CREAT|O_TRUNC,S_IRUSR|S_IWUSR)) == -1){
35      close(fd_in);
36      perror("open");
37      return -1;
38    }
```

Die Main-Methode ist größtenteils äquivalent zur Vorgabe, wir haben hier noch eine Variable pagesize eingefügt.

Listing 3: mycp.c

```
41   size_t filesize = getFilesize(argv[1]);
42
43   while(offset < filesize) {
44    if ((filesize-offset) < pagesize) {
45          pagesize = filesize-offset;
46       }
```

Unsere `while`-Schleife läuft nun über unser (als 0 initialisiertes) Offset, welches quasi die Daten darstellt, die bereits kopiert wurden. Danach fangen wir direkt den Fall ab, dass die verbleibenden Daten kleiner sind als pagesize (4096). In diesem Falle kopieren wir nicht erneut 4096B sondern nur die verbleibenden. Wir können so mit ungeraden Dateigrößen umgehen.

Listing 4: mycp.c

```
50       data=mmap(NULL, pagesize, PROT_READ, MAP_PRIVATE | MAP_POPULATE, fd_in, offset);
51       if (data == MAP_FAILED) perror("Mapping failed while mapping");
52          write(fd_out, data, pagesize);
53
54       int closed = munmap(data,pagesize);
55          if (closed != 0) perror("unMapping failed");
56
57      offset += pagesize;
58
59    }
```

Schließlich laden wir die nächsten $`pagesize` B mittels mmap und speichern den zurückzugebenen Pointer in data. Wir prüfen hier auf Fehler und liefern diesen im Zweifelsfall zurück. Danach schreiben wir mittels write die in data enthaltenen Daten in die Datei, die über fd_out verlinkt ist. Beim unmappen läuft es äquivalent zum mappen, schließlich erhöhen wir noch offset um pagesize. Fertig.

Aufgabe 1.a Vergleich mit Vorgabe

Wir haben einige Vergleiche angestellt. Hier einer davon (cache natürlich geleert):

```
x10->time ./original testdateien/data_100mb.test 100mbOLD.test

real 0m4.937s
user 0m0.040s
sys 0m0.584s

x10->time ./mycp testdateien/100mb_kompressable.test 100mb-2.test

real 0m4.468s
user 0m0.012s
sys 0m0.516s
```

Aufgabe 1.b Tests

```
x10->ls -la testdateien/
total 243013
drwxr-xr-x 2 frohde stud 12 Dec 8 11:49 .
drwxr-xr-x 3 frohde stud 13 Dec 10 11:10 ..
-rw-r--r-- 1 frohde stud 104857600 Feb 17 2015 100mb_kompressable.test
-rw-r--r-- 1 frohde stud 2048 Dec 8 11:09 2048.B
-rw-r--r-- 1 frohde stud 3333 Dec 8 11:05 3333.B
-rw-r--r-- 1 frohde stud 333333 Dec 8 11:05 333333.B
-rw-r--r-- 1 frohde stud 340992 Dec 8 11:05 333K.B
-rw-r--r-- 1 frohde stud 4096 Dec 8 11:09 4096.B
-rw-r--r-- 1 frohde stud 8192 Dec 8 11:09 8192.B
-rw-r--r-- 1 frohde stud 104857600 Feb 13 2015 data_100mb.test
-rw-r--r-- 1 frohde stud 52428800 Feb 13 2015 data_50mb.test
-rw-r--r-- 1 frohde stud 0 Dec 8 11:49 empty.test
```

Aufgabe 1.b.1 Dateien mit Nullbytes

Wir haben mithilfe von Windows 7 und fsutils ein paar Testdateien erzeugt, die Nullbytes enthalten.

```
x10->time ./mycp testdateien/8192.B nullbytetest00.test

real 0m0.088s
user 0m0.000s
sys 0m0.000s

x10->ls -la
total 191819
drwxr-xr-x 3 frohde stud 13 Dec 10 11:10 .
drwxr-xr-x 4 frohde stud 12 Dec 8 11:18 ..
-rw-------- 1 frohde stud 8192 Dec 9 13:48 .0test0
-rw-r--r-- 1 frohde stud 10 Dec 8 09:54 .gitignore
-rw-------- 1 frohde stud 104857600 Dec 9 13:44 0test
-rw-------- 1 frohde stud 104857600 Dec 9 13:44 110test
-rw-r--r-- 1 frohde stud 240 Dec 8 11:20 Makefile
-rwxr-xr-x 1 frohde stud 38040 Dec 9 13:47 mycp
-rw-r--r-- 1 frohde stud 1603 Dec 9 13:47 mycp.cc
******** -rw-------- 1 frohde stud 8192 Dec 10 11:10 nullbytetest00.test ***********
-rwxr-xr-x 1 frohde stud 31075 Dec 8 11:27 original
-rw-r--r-- 1 frohde stud 1093 Dec 8 11:19 original.cc
drwxr-xr-x 2 frohde stud 12 Dec 8 11:49 testdateien
```

Wir haben dies mit mindestens 6 verschieden großen Dateien gemacht, verzichten aber darauf, jeden einzelnen Fall zu dokumentieren.

Aufgabe 1.b.2 Leere Dateien

```
x10->time ./mycp testdateien/empty.test empty.test

real 0m0.105s
user 0m0.000s
sys 0m0.000s
x10->ls -la
total 191819
drwxr-xr-x 3 frohde stud 14 Dec 10 11:14 .
drwxr-xr-x 4 frohde stud 12 Dec 8 11:18 ..
-rw------- 1 frohde stud 8192 Dec 9 13:48 .0test0
-rw-r--r-- 1 frohde stud 10 Dec 8 09:54 .gitignore
-rw------- 1 frohde stud 104857600 Dec 9 13:44 0test
-rw------- 1 frohde stud 104857600 Dec 9 13:44 110test
-rw-r--r-- 1 frohde stud 240 Dec 8 11:20 Makefile
******* -rw------- 1 frohde stud 0 Dec 10 11:14 empty.test **********
-rwxr-xr-x 1 frohde stud 38040 Dec 9 13:47 mycp
-rw-r--r-- 1 frohde stud 1603 Dec 9 13:47 mycp.cc
-rw------- 1 frohde stud 8192 Dec 10 11:10 nullbytetest00.test
-rwxr-xr-x 1 frohde stud 31075 Dec 8 11:27 original
-rw-r--r-- 1 frohde stud 1093 Dec 8 11:19 original.cc
drwxr-xr-x 2 frohde stud 12 Dec 8 11:49 testdateien
```

Aufgabe 1.b.3 Ungerade Länge

```
x10->time ./mycp testdateien/3333.B ungerade.test

real 0m0.156s
user 0m0.000s
sys 0m0.004s
x10->ls -la
total 191820
drwxr-xr-x 3 frohde stud 15 Dec 10 11:16 .
drwxr-xr-x 4 frohde stud 12 Dec 8 11:18 ..
-rw------- 1 frohde stud 8192 Dec 9 13:48 .0test0
-rw-r--r-- 1 frohde stud 10 Dec 8 09:54 .gitignore
-rw------- 1 frohde stud 104857600 Dec 9 13:44 0test
-rw------- 1 frohde stud 104857600 Dec 9 13:44 110testls
-rw-r--r-- 1 frohde stud 240 Dec 8 11:20 Makefile
-rw------- 1 frohde stud 0 Dec 10 11:14 empty.test
-rwxr-xr-x 1 frohde stud 38040 Dec 9 13:47 mycp
-rw-r--r-- 1 frohde stud 1603 Dec 9 13:47 mycp.cc
-rw------- 1 frohde stud 8192 Dec 10 11:10 nullbytetest00.test
-rwxr-xr-x 1 frohde stud 31075 Dec 8 11:27 original
-rw-r--r-- 1 frohde stud 1093 Dec 8 11:19 original.cc
drwxr-xr-x 2 frohde stud 12 Dec 8 11:49 testdateien
******* -rw------- 1 frohde stud 3333 Dec 10 11:16 ungerade.test **********
```

Aufgabe 1.b.4 100MB

```
x10->time ./mycp testdateien/100mb_kompressable.test 100mbkomprimiert.test

real 0m4.468s
user 0m0.012s
sys 0m0.516s
x10->ls -la
total 191820
drwxr-xr-x 3 frohde stud 16 Dec 10 11:17 .
drwxr-xr-x 4 frohde stud 12 Dec 8 11:18 ..
-rw------- 1 frohde stud 8192 Dec 9 13:48 .0test0
-rw-r--r-- 1 frohde stud 10 Dec 8 09:54 .gitignore
-rw------- 1 frohde stud 104857600 Dec 9 13:44 0test
***** -rw------- 1 frohde stud 104857600 Dec 10 11:17 100mbkomprimiert.test *****
-rw------- 1 frohde stud 104857600 Dec 9 13:44 110test
-rw-r--r-- 1 frohde stud 240 Dec 8 11:20 Makefile
-rw------- 1 frohde stud 0 Dec 10 11:14 empty.test
-rwxr-xr-x 1 frohde stud 38040 Dec 9 13:47 mycp
-rw-r--r-- 1 frohde stud 1603 Dec 9 13:47 mycp.cc
-rw------- 1 frohde stud 8192 Dec 10 11:10 nullbytetest00.test
-rwxr-xr-x 1 frohde stud 31075 Dec 8 11:27 original
-rw-r--r-- 1 frohde stud 1093 Dec 8 11:19 original.cc
drwxr-xr-x 2 frohde stud 12 Dec 8 11:49 testdateien
-rw------- 1 frohde stud 3333 Dec 10 11:16 ungerade.test
```

Aufgabe 1.b.5 weitere Tests

Unzählige weitere Tests, wie “Gesicht über die Tastatur”, haben wir auch ausgeführt....

```
x10->./mycp
Usage: ./mycp source destination

x10->./mycp jfndgkjdsahbgklrfdahgb ngklbfdsdhgbklfdsjhgn
open: No such file or directory

x10->./mycp aujdsfnbkjdfhgb
Usage: ./mycp source destination

x10->./mycp original mycp
open: Text file busy

...
```

Aufgabe 2

Quellenangabe: Wir haben zur Lösung dieser Aufgabe eine alte Abgabe der Gruppe 6 (Sven Kreuz, Florian Rohde) aus TI-2 im WS 2014/15 zu rate gezogen!

Für diese Aufgabe haben wir die folgende Ausgangssituation gegeben:

- Zeit pro Spur (t_{Spur}) = $\frac{6000ms}{7200Umdrehungen} = 0{,}8\overline{3}$ ms.
- Byte pro Spur (V_{Spur}) = $512 \cdot 1200 = 614.400$ Byte.
- Byte pro Oberfläche ($V_{Oberfl.}$) = 614.400 Byte $\cdot$ 100.000 = 61.440.000.000 Byte.
- $V_{Datei} = 139.586.400$ Byte.
- $V_{Sek.} = 512$ Byte.

Aufgabe 2.a Best-case

Lesezeit = $t_{Spur} \cdot \frac{V_{Datei}}{V_{Spur}} + (\#Spurwechsel) \cdot 4ms$.

Lesezeit = $0, 8\overline{3} * 227, 19140625 + 228 \cdot 4ms =$<u>1.669,304657207813 ms</u>.

Die Anzahl der Spurwechsel ist gleich der Anzahl der Spuren $\frac{V_{Datei}}{V_{Spur}}$, aufgerundet, da wir aus der Ruheposition beginnen und auf jede Spur wechseln müssen.
Die Datenrate berechnen wir, indem wir die Dateigröße durch die Lesezeit teilen:

$\frac{139.586.400Byte}{1.669{,}304657207813ms} = 83.619, 48755Byte/ms = 0, 0797457MiB/ms =$ <u>79,7457 MiB/s</u>.

Aufgabe 2.b Worst-case

Worst-case Szenario: Jeder Sektor liegt auf einer anderen Spur. Da wir 272.630 Sektoren für die Datei benötigen, ergibt sich folgende Rechnung:

Lesezeit = $272.630 \cdot 4ms + \frac{0{,}8\overline{3}ms}{1200} \cdot 272.630 =$<u>1.090.709,32631 ms</u>.

Daraus ergibt sich die Berechnung der Datenrate:

$\frac{139586400Byte}{1.090.709{,}32631ms} = 127, 977634951Byte/ms = 0, 12497815913KiB/ms =$<u>124,97815913 KiB/s</u>.

Aufgabe 2.c Worst-case light

Wenn wir die Blockgröße auf 4096 Byte erhöhen, benötigen wir "nur" noch 34.079 Blöcke.
Daraus folgt die Rechnung:

Lesezeit = $34.079 \cdot 4ms + \frac{0{,}8\overline{3}ms}{1200} \cdot 34.079 =$<u>136.339,665971 ms</u>.
Datenrate = $\frac{139586400Byte}{136.339{,}665971ms} = 1023, 8135689Byte/ms = 0.99981793837KiB/ms =$<u>999,81793837 KiB/s</u>.

Aufgabe 3

Hilfreiche Quelle:Wir haben zur Lösung dieser Aufgabe eine alte Abgabe für den Rechenweg benutzt.(WS2015/16, E-Bei Jesse/1, Prien, Kirschner, Habermann)

a

$$\frac{1B}{10000\frac{b}{s}} = 0,0001s$$
$$\frac{0LWM}{10000} = 0$$
$$0,005s - 0 = 0,005$$
$$0,0001 + 0,005s = 0,0051$$

1 Byte a 0,0051 sek

$$1 * (\frac{1}{0,0051}) = 196,078$$

Wir besitzen also eine Datenrate von $196\frac{b}{s}$

b

$$\frac{1024B}{10000\frac{b}{s}} = 0,1024s$$
$$\frac{0LWM}{10000} = 0$$
$$0,005s - 0 = 0,005$$
$$0,1024 + 0,005s = 0,1074$$

1 Byte a 0,1074 sek

$$1024 * (\frac{1}{0,1074}) = 9534,450$$

Wir besitzen also eine Datenrate von $9534\frac{b}{s}$

c

$$\frac{1024B}{10000\frac{b}{s}} = 0,1024s$$
$$\frac{25LWM}{10000} = 0,0025$$
$$0,005s - 0,0025 = 0,0025$$
$$0,1024 + 0,0025s = 0,1049$$

1 Byte a 0,1049 sek

$$1024 * (\frac{1}{0,1049}) = 9761,6777884$$

Wir besitzen also eine Datenrate von $9762\frac{b}{s}$

d

$10000 * 0,005 = 50$
Die Lower Watermark muss also bei mindestens 50 Bytes liegen. Somit könnten durchgehend Bytes verarbeitet werden und die Netto Datenrate ist erreicht.