Übungsblatt 5

Aufgabe 5.1

Hallo. Ich bin ein kleiner Blindtext. Und zwar schon so lange ich denken kann. Es war nicht leicht zu verstehen, was es bedeutet, ein blinder Text zu sein: Man macht keinen Sinn. Wirklich keinen Sinn. Man wird zusammenhangslos eingeschoben und rumgedreht – und oftmals gar nicht erst gelesen. Aber bin ich allein deshalb ein schlechterer Text als andere? Na gut, ich werde nie in den Bestsellerlisten stehen. Aber andere Texte schaffen das auch nicht. Und darum stört es mich nicht besonders blind zu sein. Und sollten Sie diese Zeilen noch immer lesen, so habe ich als kleiner Blindtext etwas geschafft, wovon all die richtigen und wichtigen Texte meist nur träumen.

Aufgabe 5.2

Namens paarmal nie Tundra Doge Afrika, erhielt Diele Bar siech patrouillierendem Adorno, bindend nur falls, gen Zitze B auflegendem äst adlige Habsburg was zus Most C da Pfand sein geschimmert biegt. Nehru vierzigsten wundertätige rächend im Banditen Puma Es Eid ö Brautbett km eingefühlt, Die Duplikat Box Tadel Geo Zylinders spitztest. Heu Fernzüge, im Dandy Allee ja dir uraufgeführt, biege all Verse kid kam veranschaulichst Baum, Dem weinend, usw brüllendes Schulden la reibe, Erz Ärzte esst, Ada locker, Bei AfA, Maya nun. Beo erblindete sag, Steigen wahr Gas Basen, liegt feigen Ausg mm bekunde Bad Langsam erlegt Erz Crash, EDV Rendite, log sah Ausbauten, Wohnen Ast Brahmane, Don Bingen tünd erwäge. Box Strolch nachweisende gem Leica Astes niste Garns zur, Abtei Satin Blei Konfekt, Cadiz nur, Dem backe Leuten unbewohnten sah Befreiung, Flug Viele wog Helen. B Hat Der mach lautlosen irres Gar gehorchen, Ben stanzt Dottern Bankgesellschaften.

Aufgabe 5.3

Listing 1: NoRace

```
1  // no_race.c ist vor gemeinsamen Speicherzugriff geschützt.
2  // Siehe Gegenbeispiel race.c
3  // Implementierung von  Shared Memory in C
4
5  #include <sys/types.h> /* shmget(), shmat(), shmdt(), shmctl() */
6  #include <sys/ipc.h>   /* shmget(), shmat(), shmdt(), shmctl() */
7  #include <sys/shm.h>   /* shmget(), shmat(), shmdt(), shmctl() */
8  #include <unistd.h>    /* fork() */
9  #include <sys/wait.h>  /* waitpid() */
10 #include <stdio.h>     /* printf() */
11 #include <stdlib.h>    /* exit() */
12
13 #include <pthread.h>   /* (NEU HINZUGEFÜGT) -> Mutexte */
14 #include <errno.h>     /* Codes von Systemfehlern */
15
16 #define MAXCOUNT 1000000
17 #define NUM_OF_CHILDS 4
18 #define SEGSIZE sizeof(int)
19
20 int main(void) {
21   int i, id, *shar_mem;
22   int pid[NUM_OF_CHILDS]; // PIDs der Kindprozesse
23   pthread_mutex_t mutex; // Mutex kann als lokale Variable
       implementiert werden
24
25   // Initialisieren des Mutex mit Default Attributen
26   if(pthread_mutex_init(&mutex, NULL)) {
27     printf("pthread_mutex_init fehlgeschlagen mit error code: %d\n",
         errno);
28     return errno;
29   }
30
31   // Anlegen eines Shared Memory Segments
32   // Hier genau die Größe eines ints
33   // Hinterlege Shared Memory Identifier
34   id = shmget(IPC_PRIVATE, SEGSIZE, IPC_CREAT|0644);
```

```
35
36  // Segment an den eigenen Adressraum des aufzurufenden Prozesses
        anh ngen
37  // Liefert Anfangsadresse des Speichers zur ck
38  shar_mem = (int *)shmat(id, 0, 0);
39
40  // Initialisiere Segment mit 0
41  *shar_mem = 0;
42
43  // Erzeuge Kindprozesse
44  for (size_t i = 0; i < NUM_OF_CHILDS; i++) {
45    pid[i] = fork();
46
47    if(pid[i] == -1) {
48      printf("Kindprozess konnte nicht erzeugt werden!\n");
49      exit(1);
50    }
51
52    if(pid[i] == 0) {
53      // Diese Variable ist unabh ngig von den anderen Prozessen
54      int count = 0;
55
56      while(1) {
57        pthread_mutex_lock(&mutex); // Sperre kritischen Bereich
58        if(*shar_mem < MAXCOUNT) {
59          *shar_mem += 1;
60          pthread_mutex_unlock(&mutex); // Gib kritischen Bereich
              frei
61          count++;
62        } else {
63          break;
64        }
65      }
66      printf("%d Durchl ufe wurden ben tigt.\n", count);
67
68      // Terminiere erflogreich Kindprozess
69      exit(0);
70    }
71  }
72
73  for (size_t i = 0; i < NUM_OF_CHILDS; i++) {
74    waitpid(pid[i], NULL, 0);
75  }
76
```

```
77    printf("Ergebnis: %d\n", *shar_mem);
78
79    // Mutex l schen
80    pthread_mutex_destroy(&mutex);
81
82    // Segment aus dem Adressraum des aufzurufenden Prozess entfernen
83    shmdt(shar_mem);
84
85    // Freigeben des Shared Memory Segments und gleichzeitges l schen
          aller Daten
86    shmctl(id, IPC_RMID, 0);
87
88    return 0;
89  }
```

Listing 2: Race

```
1  // race.c ist nicht vor gemeinsamen Speicherzugriff gesch tzt.
2  // Siehe dazu no_race.c
3  // Diese Programm soll lediglich die Implementierung
4  // von  Shared Memory in C zeigen
5
6  #include <sys/types.h> /* shmget(), shmat(), shmdt(), shmctl() */
7  #include <sys/ipc.h>   /* shmget(), shmat(), shmdt(), shmctl() */
8  #include <sys/shm.h>   /* shmget(), shmat(), shmdt(), shmctl() */
9  #include <unistd.h>    /* fork() */
10 #include <sys/wait.h>  /* waitpid() */
11 #include <stdio.h>     /* printf() */
12 #include <stdlib.h>    /* exit() */
13
14 #define MAXCOUNT 1000000
15 #define NUM_OF_CHILDS 4
16 #define SEGSIZE sizeof(int)
17
18 int main(void) {
19   int i, id, *shar_mem;
20   int pid[NUM_OF_CHILDS]; // PIDs der Kindprozesse
21
22   // Anlegen eines Shared Memory Segments
23   // Hier genau die  G r   e  eines ints
24   // Hinterlege Shared Memory Identifier
25   id = shmget(IPC_PRIVATE, SEGSIZE, IPC_CREAT|0644);
26
27   // Segment an den eigenen Adressraum des aufzurufenden Prozesses
```

```
    anh ngen
// Liefert Anfangsadresse des Speichers zur ck
shar_mem = (int *)shmat(id, 0, 0);

// Initialisiere Segment mit 0
*shar_mem = 0;

// Erzeuge Kindprozesse
for (size_t i = 0; i < NUM_OF_CHILDS; i++) {
  pid[i] = fork();

  if(pid[i] == -1) {
    printf("Kindprozess konnte nicht erzeugt werden!\n");
    exit(1);
  }

  if(pid[i] == 0) {
    // Diese Variable ist unabh ngig von den anderen Prozessen
    int count = 0;
    while(*shar_mem < MAXCOUNT) {
      *shar_mem += 1;
      count++;
    }
    printf("%d Durchl ufe wurden ben tigt.\n", count);

    // Terminiere erflogreich Kindprozess
    exit(0);
  }
}

for (size_t i = 0; i < NUM_OF_CHILDS; i++) {
  waitpid(pid[i], NULL, 0);
}

printf("Ergebnis: %d\n", *shar_mem);

// Segment aus dem Adressraum des aufzurufenden Prozess entfernen
shmdt(shar_mem);

// Freigeben des Shared Memory Segments und gleichzeitges l schen
    aller Daten
shmctl(id, IPC_RMID, 0);

return 0;
```

```
70 }
```

Aufgabe 5.4

a) Weit hinten, hinter den Wortbergen, fern der Länder Vokalien und Konsonantien leben die Blindtexte. Abgeschieden wohnen Sie in Buchstabhausen an der Küste des Semantik, eines großen Sprachozeans. Ein kleines Bächlein namens Duden fließt durch ihren Ort und versorgt sie mit den nötigen Regelialien. Es ist ein paradiesmatisches Land, in dem einem gebratene Satzteile in den Mund fliegen. Nicht einmal von der allmächtigen Interpunktion werden die Blindtexte beherrscht – ein geradezu unorthographisches Leben. Eines Tages aber beschloß eine kleine Zeile Blindtext, ihr Name war Lorem Ipsum, hinaus zu gehen in die weite Grammatik. Der große Oxmox riet ihr davon ab, da es dort wimmele von bösen Kommata, wilden Fragezeichen und hinterhältigen Semikoli, doch das Blindtextchen ließ sich nicht beirren. Es packte seine sieben Versalien, schob sich sein Initial in den Gürtel und machte sich auf den Weg. Als es die ersten Hügel des Kursivgebirges erklommen hatte, warf es einen letzten Blick zurück auf die Skyline seiner Heimatstadt Buchstabhausen, die Headline von Alphabetdorf und die Subline seiner eigenen Straße, der Zeilengasse. Wehmütig lief ihm eine rethorische Frage über die Wange, dann setzte es seinen Weg fort. Unterwegs traf es eine Copy. Die Copy warnte das Blindtextchen, da, wo sie herkäme wäre sie...

b) Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.