

Barry Linnert

Nichtsequentielle und verteilte Programmierung, SS2021

Übung 2

Tutor: Florian Alex
Tutorium 3

Rui Zhao, William Djalal, Simeon Vasilev

2. Mai 2021

1 Kritischer Abschnitt in C

(10 Punkte)

Implementieren Sie in C eine Simulation zweier Autos, die regelmäßig wiederkehrend eine Brücke überqueren wollen, die jedoch für nur ein Auto hinreichend breit ist. Implementieren Sie die Autos als eigenständige POSIX-Threads. Die Simulation soll Unfälle, die eintreten, wenn beide Fahrzeuge die Brücke gleichzeitig befahren erkennen und melden. Beachten Sie, dass die Überquerung der Brücke selbst auch Zeit in Anspruch nimmt. Die Autos sollen die Brücke mindestens 100.000 mal überqueren. Dokumentieren Sie Ihr Programm und stellen Sie immer die Ausgaben des jeweiligen Programms zur Verfügung.

```
1 #include <pthread.h>
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <unistd.h>
5
6
7
8 #define NUM_THREADS 2
9
10 int crash = 0;
11
12 int car[2] = {0}; // 1: on bridge , 0: not on bridge
13
14 void *cross_bridge (void *threadid)
15 {
16     long tid;
17     tid = (long) threadid;
18
19     for (int i = 0; i < 100000; i++)
20     {
21         car[tid] = 1;
22
23         if ((car[1-tid] + car[tid]) == 2)
24         {
25             crash++;
26         }
27
28         usleep(1);
29         car[tid] = 0;
30     }
31
32     pthread_exit (NULL);
33 }
```

```

35 int main (/*int argc, char *argv[]*/)
36 {
37     pthread_t threads[NUM_THREADS];
38     int rc;
39     long t;
40     for (t=0; t < NUM_THREADS; t++) {
41         printf ("In main: creating thread %ld\n", t);
42         rc = pthread_create (&threads[t], NULL, cross_bridge, (void *)t);
43         if (rc) {
44             printf ("ERROR; return code from pthread_create () is %d\n", rc);
45             exit (-1);
46         }
47     }
48
49     // joining threads
50     for (long t = 0; t < 2; t++) {
51         pthread_join (threads[t], NULL);
52     }
53
54     printf("numbers of crashes :%d\n", crash);
55
56     /* Last thing that main() should do */
57     pthread_exit(NULL);
58 }

```

Wie habt ihr das Problem modelliert? Was sind eure Beobachtungen?

8/10

2 Sicherung des kritischen Abschnitts in C

(8 Punkte)

Erweitern Sie Ihre Simulation so, dass das Befahren der Brücke als kritischer Abschnitt gesichert ist. Nutzen Sie hierzu die in der Vorlesung vorgestellten Verfahren ohne Betriebssystems- und Hardware-Unterstützung.

```

1  #include <pthread.h>
2  #include <stdlib.h>
3  #include <stdio.h>
4  #include <unistd.h>
5
6
7  #define TRUE 1
8  #define FALSE 0
9
10 #define NUM_THREADS 2
11
12 int crash = 0;
13
14 int car[2] = {0};      // 1: on bridge , 0: not on bridge
15
16 int flag[2] = {0};    // 'Petersons Algorithm
17 int turn = 0;
18
19 void *cross_bridge (void *threadid)
20 {
21     long tid;
22     tid = (long) threadid;
23
24     for (int i = 0; i < 100000; i++)
25     {
26         flag[tid] = TRUE;
27         turn = 1 - tid;
28         while(flag[1-tid] && turn == (1 - tid));
29
30         car[tid] = 1;    //critical section
31         if ((car[1-tid] + car[tid]) == 2)
32         {
33             crash++;

```

Ihr könnt hier ein Lock aus der Vorlesung verwenden

???

```

34 }
35 usleep(1);
36 car[tid] = 0;

38 flag[tid] = FALSE;
39 }

40
41 // return crash;
42 pthread_exit (NULL);
43 }

45 int main (/*int argc, char *argv[]*/)
46 {
47     pthread_t threads[NUM_THREADS];
48     int rc;
49     long t;
50     for (t=0; t < NUM_THREADS; t++) {
51         printf ("In main: creating thread %ld\n", t);
52         rc = pthread_create (&threads[t], NULL, cross_bridge, (void *)t);
53         if (rc) {
54             printf ("ERROR; return code from pthread_create () is %d\n", rc);
55             exit (-1);
56         }
57     }

58
59     // joining threads
60     for (long t = 0; t < 2; t++) {
61         pthread_join (threads[t], NULL);
62     }

64     printf("numbers of crashes :%d\n", crash);

66     /* Last thing that main() should do */
67     pthread_exit(NULL);
68 }

```

Ist der Abschnitt gesichert? Wieso funktioniert euer Lock?

6/8

3 Erweiterung der Anzahl der Threads in C

(8 Punkte)

Erweitern Sie Ihre Lösung, sodass eine beliebige, aber feste Anzahl von Autos die Brücke überqueren will. Sichern Sie auch hier die Überquerung mit den in der Vorlesung vorgestellten Verfahren ohne Betriebssystems- und Hardware-Unterstützung.

```

1 #include <pthread.h>
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <unistd.h>

7 #define TRUE 1
8 #define FALSE 0

10 #define NUM_THREADS 5

12 int crash = 0;

14 int car[NUM_THREADS] = {0};    // 1: on bridge , 0: not on bridge

16 int level[NUM_THREADS] = {0};  // init with 0
17 int last[NUM_THREADS] = {0};   // init with 0

19 int unlock (long tid) {
20     level[tid] = 0;
21     return 0;
22 }

```

```

24 int lock (long tid) {    // Multiple Locks with Mutual Access - Peterson
25     int i, j, loop;

27     for (i = 1; i < NUM_THREADS; i++) {
28         level[tid] = i;
29         last[i] = tid;
30         loop = 1;
31         while ((loop)&&(last[i] == tid)) {
32             j = 0;
33             loop = 0;
34             while ((j < NUM_THREADS) && ((level[j] < i)|| (j == tid)))
35                 j++;
36             if (j < NUM_THREADS)
37                 loop = 1;
38         }
39     }
40     return 0;
41 }

44 void *cross_bridge (void *threadid)
45 {
46     long tid;
47     tid = (long) threadid;

49     for (int i = 0; i < 100000; i++)
50     {
51         lock(tid);

53         car[tid] = 1;        //critical section

55         int sum = 0;        // how many cars on the bridge
56         for (int i = 0; i < NUM_THREADS; i++)
57         {
58             sum = sum + car[i];
59         }

61         if (sum > 1)        // if more than one car, crash
62         {
63             crash++;
64         }
65         usleep(1);
66         car[tid] = 0;


68         unlock(tid);

70     }

72     // return crash;
73     pthread_exit (NULL);
74 }

77 int main (/*int argc, char *argv[]*/)
78 {
79     pthread_t threads[NUM_THREADS];
80     int rc;
81     long t;
82     for (t=0; t < NUM_THREADS; t++) {
83         printf ("In main: creating thread %ld\n", t);
84         rc = pthread_create (&threads[t], NULL, cross_bridge, (void *)t);
85         if (rc) {
86             printf ("ERROR; return code from pthread_create () is %d\n", rc);
87             exit (-1);
88         }
89     }
90 }

```



```

91 // joining threads
92 for (long t = 0; t < 2; t++) {
93     pthread_join (threads[t], NULL);
94 }
95
96 printf("numbers of crashes :%d\n", crash);
97
98 /* Last thing that main() should do */
99 pthread_exit(NULL);
100 }

```

8/8

4 Bewertung der Ansätze

(4 Punkte)

Testen Sie Ihre Lösung(en) hinreichend. Beschreiben Sie möglicherweise auftretende Effekte auf Basis des Maschinen-/Ausführungsmodells. Schlagen Sie geeignete Maßnahmen zur Behebung vor.

Beim Kompilieren wird eine Warnung angezeigt:

warning: implicit declaration of function 'usleep'; did you mean 'sleep'?

Der Grund dafür ist, usleep() wurde seit POSIX-2008 entfernt.

Es wurde versucht, die POSIX-Version usw. zu ändern, dies führt jedoch zu weiteren Fehlern.

Eine Möglichkeit, diese Warnung zu beseitigen, besteht darin, nanosleep() anstelle von usleep() zu verwenden.

```

zhaor@6P00NT1-88WD3YW:~/alp4/u2$ ./a1
In main: creating thread 0
In main: creating thread 1
numbers of crashes :199122
zhaor@6P00NT1-88WD3YW:~/alp4/u2$ ./a1
In main: creating thread 0
In main: creating thread 1
numbers of crashes :199033
zhaor@6P00NT1-88WD3YW:~/alp4/u2$ ./a1
In main: creating thread 0
In main: creating thread 1
numbers of crashes :198123
zhaor@6P00NT1-88WD3YW:~/alp4/u2$ ./a1
In main: creating thread 0
In main: creating thread 1
numbers of crashes :199098

```

Abbildung 1: Test Screen A1

Hier wurde eine gewisse Vereinfachung vorgenommen, d.h. es wird nicht zwischen den beiden Ufern des Flusses unterschieden. Autos werden kollidieren, wenn sie sich gleichzeitig auf der Brücke befinden.

Da keine kritischen Abschnitte festgelegt ist, lässt sich Folgendes beobachten: Es gibt sehr viele Fälle, in denen zwei Threads gleichzeitig in Betrieb sind, d. h. zwei Autos fahren gleichzeitig auf die Brücke und kollidieren.

```

zhaor@6P00NT1-88WD3YW:~/alp4/u2$ ./a2
In main: creating thread 0
In main: creating thread 1
numbers of crashes :0

```

Abbildung 2: Test Screen A2

Hier wird Peterson's Algorithmus (original) verwendet, der gut funktioniert, wenn es nicht mehr als 2 Threads gibt, und der sicherstellt, dass sich die beiden Threads nicht gegenseitig stören, d.h. die Autos teilen sich eine gemeinsame Brücke und kollidieren nicht.[1]

```
zhaor@6P00NT1-88WD3YW:~/alp4/u2$ ./a3
In main: creating thread 0
In main: creating thread 1
In main: creating thread 2
In main: creating thread 3
In main: creating thread 4
numbers of crashes :0
```

Abbildung 3: Test Screen A3

Der in der VL beschriebene erweiterte Peterson-Algorithmus (Multiple Locks with Mutual Access) wird hier verwendet, um sicherzustellen, dass sich mehrere Threads nicht gegenseitig stören. [2]

Literatur

- [1] <https://zh.wikipedia.org/wiki/Peterson%E7%AE%97%E6%B3%95>
- [2] http://www.inf.fu-berlin.de/inst/ag-se/teaching/V-NSEQ-2021/05_Parallelization.pdf