第 2 回演習 (11 月 9 日)

坪井正太郎（101830245）

2020 年 11 月 23 日

1　問題 1

以下のようなプログラムを作成した。

ソースコード 1　SemWrite.c

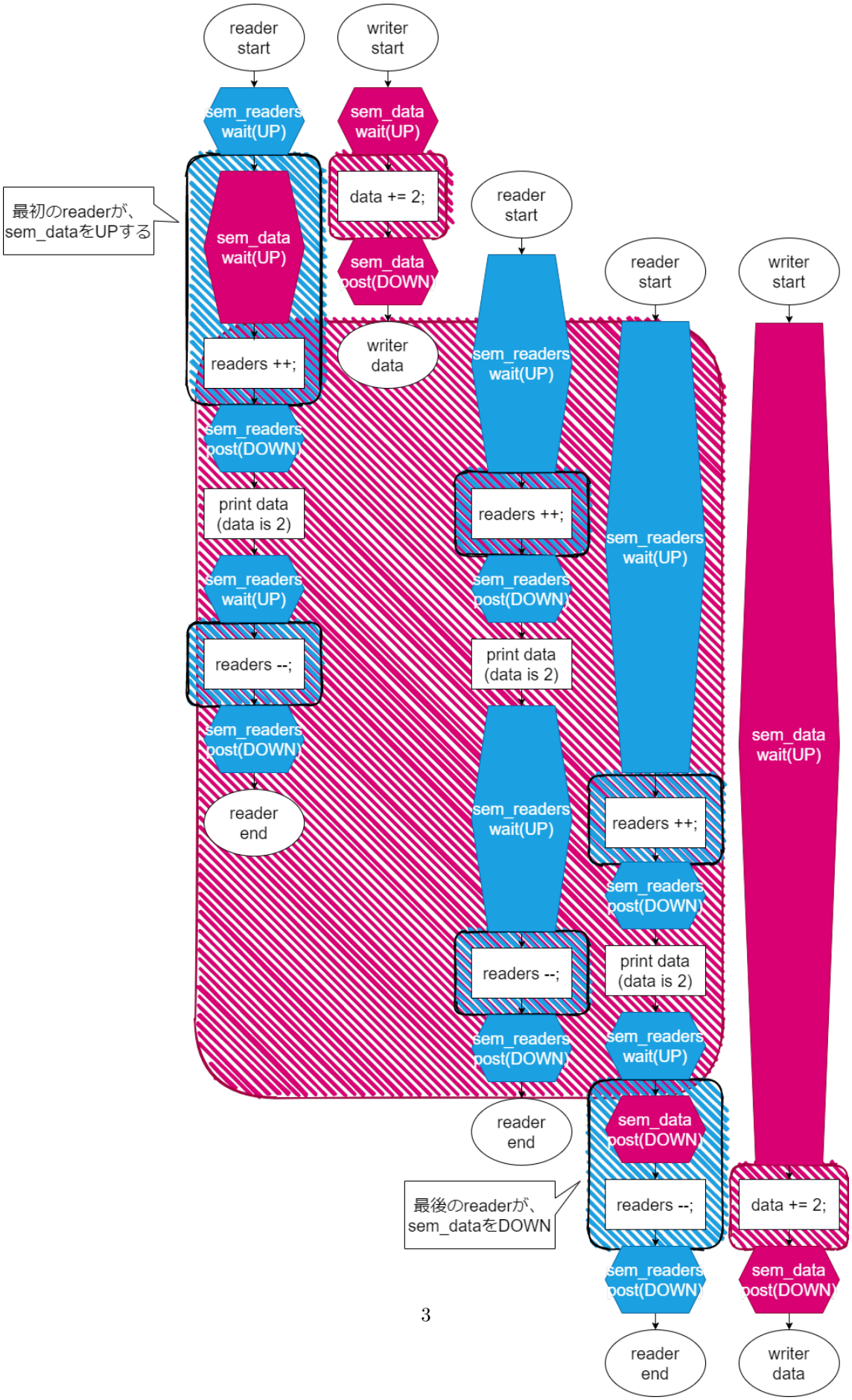
```
1  #include <pthread.h>
2  #include <semaphore.h>
3  #include <stdio.h>
4
5  #define THREADS 10
6
7  sem_t sem_data, sem_readers;
8  int readers = 0;
9  int data = 0;
10
11 void *writer(void *arg) {
12   int f;
13   f = ((int)arg);
14   sem_wait(&sem_data);
15   data = data + 2;
16   printf("Data writen by the writer%d is %d\n", f, data);
17   // sleep(2);
18   sem_post(&sem_data);
19 }
20
21 void *reader(void *arg) {
22   int f;
23   f = ((int)arg);
24
25   // readers アクセス部
26   sem_wait(&sem_readers);
27   if (readers == 0) {
28     // reader の代表が書き込みロック
29     sem_wait(&sem_data);
```

```
30    }
31    readers++;
32    sem_post(&sem_readers);
33
34    printf("Data read by the reader%d is %d\n", f, data);
35    // sleep(2);
36
37    // readers アクセス部
38    sem_wait(&sem_readers);
39    if (readers == 1) {
40      // 最後にアクセスしたreader がロック解除
41      sem_post(&sem_data);
42    }
43    readers--;
44    sem_post(&sem_readers);
45  }
46
47  main() {
48    int i, b;
49    pthread_t wtid[THREADS], rtid[THREADS];
50    sem_init(&sem_data, 0, 1);
51    sem_init(&sem_readers, 0, 1);
52    for (i = 0; i <= THREADS - 1; i++) {
53      pthread_create(&wtid[i], NULL, writer, (void *)i);
54      pthread_create(&rtid[i], NULL, reader, (void *)i);
55    }
56
57    for (i = 0; i <= THREADS - 1; i++) {
58      pthread_join(wtid[i], NULL);
59      pthread_join(rtid[i], NULL);
60    }
61  }
```

readers 変数で、現在 data にアクセスしている reader をカウントしている。また、カウントアップ時の排他処理のために、sem_readersd でスレッド間の同期を制御している。reader は、readers を参照して、最初に data を読む reader が sem_data の UP を、最後の reader が DOWN を行なっている。

以下に、フローチャートを示す。main 内でスレッドが作成されてからの図で、3 つの reader と 2 つの writer が並列に動作している。見やすさのために、readers を参照しての分岐は省略している。(コメントの部分)

また、セマフォの上げ下げは六角形で表し、対応する上げ下げの間を、斜線で囲っている。sem_data に対応する部分は赤、sem_readers に対応する部分は青になっている。斜線部はロックされていて、横に同じ色が来ないことがわかる。


reader start
writer start
sem_readers wait(UP)
sem_data wait(UP)
data += 2;
sem_data post(DOWN)
最初のreaderが、sem_dataをUPする
sem_data wait(UP)
writer data
readers ++;
sem_readers post(DOWN)
print data (data is 2)
sem_readers wait(UP)
readers --;
sem_readers post(DOWN)
reader end
reader start
sem_readers wait(UP)
readers ++;
sem_readers post(DOWN)
print data (data is 2)
sem_readers wait(UP)
readers --;
sem_readers post(DOWN)
reader end
reader start
sem_readers wait(UP)
readers ++;
sem_readers post(DOWN)
print data (data is 2)
sem_readers wait(UP)
sem_data post(DOWN)
readers --;
sem_readers post(DOWN)
reader end
writer start
sem_data wait(UP)
data += 2;
sem_data post(DOWN)
writer data
最後のreaderが、sem_dataをDOWN

実行結果は、以下のようになった。

ソースコード 2　実行結果

```
$ gcc -pthread SemWrite.c
$ ./a.out
Data writen by the writer0 is 2
Data writen by the writer2 is 4
Data writen by the writer1 is 6
Data writen by the writer3 is 8
Data read by the reader0 is 8
Data read by the reader1 is 8
Data read by the reader3 is 8
Data read by the reader4 is 8
Data read by the reader2 is 8
Data writen by the writer4 is 10
Data writen by the writer5 is 12
Data read by the reader5 is 12
Data writen by the writer6 is 14
Data read by the reader6 is 14
Data writen by the writer7 is 16
Data read by the reader7 is 16
Data read by the reader8 is 16
Data writen by the writer9 is 18
Data read by the reader9 is 18
Data writen by the writer8 is 20
```

2　問題 2

すべての哲学者が同時にステップ 1 を実行すると、誰もステップ 2 に進むことができない。全員フォークを置くこともできないので、デッドロックになる。

2.1　回避するための手順

以下の手順を、各哲学者が独立して実行する。

1. 両隣の哲学者が手を挙げていなければ、手を挙げることができる (同時に挙げたときは、左の哲学者が手を挙げていたら下げる)
2. 手を挙げることができ、両方のフォークが揃っていれば手順 1 にしたがって食事する (フォークが揃っていなければ、手を挙げたまま待つ)

この手順で、フォークを双方確保できることが保証できる。優先度によって、ループして上げ下げが行われる可能性があるが、独立して実行されるので、必ず終息し、デッドロックにはならない。