2020K8009907032
唐嘉良
作业 5
5.1 写一个两线程程序，两线程同时向一个数组分别写入 1000 万以内的奇数和偶数，写入过程中两个线程共用一个偏移量 index，代码逻辑如下所示。写完后打印出数组相邻两个数的最大绝对差值。

```
int MAX=10000000;
index = 0
//thread1
for(i=0;i<MAX;i+=2) {
data[index] = i; //even ( i+1 for thread 2)
index++;
}
//thread2
for(i=0;i<MAX;i+=2) {
data[index] = i+1; //odd
index++;
}
```

请分别按下列方法完成一个不会丢失数据的程序：

1) 请用 Peterson 算法实现上述功能；

2) 请学习了解 pthread_mutex_lock/unlock()函数，并实现上述功能；

3) 请学习了解 atomic_add()(_sync_fetch_and_add()for gcc 4.1+) 函数，并实现上述功能。

提交：

1. 说明你所写程序中的临界区（注意：每次进入临界区之后，执行 200 次操作后离开临界区。）
2. 提供上述三种方法的源代码，运行结果截图(即，数组相邻两个数的最大绝对差值)
3. 请找一个双核系统测试三种方法中完成数组写入时，各自所需的执行时间，不用提供计算绝对差值的时间。

答：1）Peterson 算法源代码如下：（临界区为注释//critical section 后的一段代码，以空行为结束分界）

```c
#include<time.h>
#include<stdio.h>
#include<pthread.h>
#define MAXLEN 10000000
long indexx, array[MAXLEN];
int turn,flag[2]={0,1};

void* thread1(void* arg){
    int j=0;
    do{
        flag[0]=1;
        turn=1;
        while(flag[1] && turn == 1);
        //critical section
        for(int i=0;i<200;i++){
            array[indexx++]=j;
            j+=2;
        }

        flag[0]=0;
        if(j>=MAXLEN) break;
    }while(1);
}
```

```c
void* thread2(void* arg){
    int j=0;
    do{
        flag[1]=1;
        turn=0;
        while(flag[0] && turn == 0);
        //critical section
        for(int i=0;i<200;i++){
            array[indexx++]=j+1;
            j+=2;
        }

        flag[1]=0;
        if(j>=MAXLEN) break;
    }while(1);
}

int sub(){
    long sub =0,p;
    for(long i=0;i<MAXLEN-1;i++){
        long asub;
        if(array[i+1]>array[i]){
```

```
    for(long i=0;i<MAXLEN-1;i++){
        long asub;
        if(array[i+1]>array[i]){
            asub=array[i+1]-array[i];
        }
        else asub = array[i]-array[i+1];
        if(asub>sub){
            sub=asub;
            p=i;
        }
    }
    printf("max sub is %ld\n",sub);
    return sub;
}

int main(){
    pthread_t thread[2];
    struct timespec start,end;
    unsigned long int timeuse;
    clock_gettime(CLOCK_REALTIME,&start);

    pthread_create(thread,NULL,(void*)thread1,NULL);
    pthread_create(thread+1,NULL,(void*)thread2,NULL);
    pthread_join(thread[0],NULL);
```

```
    pthread_create(thread,NULL,(void*)thread1,NULL);
    pthread_create(thread+1,NULL,(void*)thread2,NULL);
    pthread_join(thread[0],NULL);
    pthread_join(thread[1],NULL);

    clock_gettime(CLOCK_REALTIME,&end);
    timeuse=((end.tv_sec-start.tv_sec)*1000+(end.tv_nsec-start.tv_nsec)/1000000);
    printf("time = %ld ms\n",timeuse);
    sub();
    return 0;
}
```

运行结果截图：

```
ubuntu@ubuntu:~/Desktop$ gcc hw.c -lpthread -o hw
ubuntu@ubuntu:~/Desktop$ ./hw
time = 89 ms
max sub is 397
ubuntu@ubuntu:~/Desktop$ gcc hw.c -lpthread -o hw
ubuntu@ubuntu:~/Desktop$ ./hw
time = 104 ms
max sub is 397
ubuntu@ubuntu:~/Desktop$ ./hw
time = 113 ms
max sub is 397
ubuntu@ubuntu:~/Desktop$ ./hw
time = 127 ms
max sub is 397
```

2）方法二代码如下：（临界区为注释//critical section 后的一段代码，以空行为结束分界）

```
9   void* thread1(void* arg){
10      int j=0;
11      do{
12          /*flag[0]=1;
13          turn=1;
14          while(flag[1] && turn == 1);*/
15          pthread_mutex_lock(&data_mutex);
16          //critical section
17          for(int i=0;i<200;i++){
18              array[indexx++]=j;
19              j+=2;
20          }
21
22          //flag[0]=0;
23          pthread_mutex_unlock(&data_mutex);
24          if(j>=MAXLEN) break;
25      }while(1);
26  }
```

```
27  void* thread2(void* arg){
28      int j=0;
29      do{
30          /*flag[1]=1;
31          turn=0;
32          while(flag[0] && turn == 0);*/
33          pthread_mutex_lock(&data_mutex);
34          //critical section
35          for(int i=0;i<200;i++){
36              array[indexx++]=j+1;
37              j+=2;
38          }
39
40          pthread_mutex_unlock(&data_mutex);
41          //flag[1]=0;
42          if(j>=MAXLEN) break;
43      }while(1);
44  }
```

其余代码同方法一。

运行结果如下：

```
ubuntu@ubuntu:~/Desktop$ gcc hw.c -lpthread -o hw
ubuntu@ubuntu:~/Desktop$ ./hw
time = 53 ms
max sub is 1673597
ubuntu@ubuntu:~/Desktop$ ./hw
time = 44 ms
max sub is 5404797
ubuntu@ubuntu:~/Desktop$ ./hw
time = 43 ms
max sub is 1712797
ubuntu@ubuntu:~/Desktop$
```

3）方法三代码如下：（临界区为注释//critical section 后的一段代码，以空行为结束分界）

```c
#include<time.h>
#include<stdio.h>
#include<pthread.h>
#define MAXLEN 10000000
long indexx=0, array[MAXLEN];
int turn,flag[2]={0,1};
//pthread_mutex_t data_mutex;

void* thread1(void* arg){
    int j=0;
    do{
        /*flag[0]=1;
        turn=1;
        while(flag[1] && turn == 1);*/
        //pthread_mutex_lock(&data_mutex);

        //critical section
        long startdex = __sync_fetch_and_add(&indexx, 200);

        for(int i=0;i<200;i++){
            array[startdex + i]=j;
            j+=2;
        }

        //flag[0]=0;
        //pthread_mutex_unlock(&data_mutex);
        if(j>=MAXLEN) break;
    }while(1);
}
void* thread2(void* arg){
    int j=0;
    do{
        /*flag[1]=1;
        turn=0;
        while(flag[0] && turn == 0);*/
        //pthread_mutex_lock(&data_mutex);

        //critical section
        long startdex = __sync_fetch_and_add(&indexx, 200);

        for(int i=0;i<200;i++){
            array[startdex + i]=j+1;
            j+=2;
        }

        //pthread_mutex_unlock(&data_mutex);
        //flag[1]=0;
        if(j>=MAXLEN) break;
    }while(1);
}

```

其余代码同方法一。

运行结果如下：

```
ubuntu@ubuntu:~/Desktop$ ./hw
time = 68 ms
max sub is 2557997
ubuntu@ubuntu:~/Desktop$ ./hw
time = 78 ms
max sub is 511997
ubuntu@ubuntu:~/Desktop$ ./hw
time = 67 ms
max sub is 2985597
```

5.2 现有一个长度为 5 的整数数组，假设需要写一个两线程程序，其中，线程 1 负责往数组中写入 5 个随机数（1 到 20 范围内的随机整数），写完这 5 个数后，线程 2 负责从数组中读取这 5 个数，并求和。该过程循环执行 5 次。注意：每次循环开始时，线程 1 都重新写入 5 个数。请思考：

1）上述过程能否通过 pthread_mutex_lock/unlock 函数实现？如果可以，请写出相应的源代码，并运行程序，打印出每次循环计算的求和值；如果无法实现，请分析并说明原因。

提交：实现题述功能的源代码和打印结果，或者无法实现的原因分析说明。

答：在允许使用 flag 这类标记变量的时候，能。源代码如下：

```c
#include<time.h>
#include<stdio.h>
#include<pthread.h>
#include<stdlib.h>
#define LEN 5
#define OPTIMES 5
long array[LEN];
int flag = 0;//0 means turn is write, 1 means turn is read
pthread_mutex_t data_mutex;

void* thread1(void* arg){
    int j=0;
    do{
        if(!flag){
            pthread_mutex_lock(&data_mutex);

            for(int i=0;i<LEN;i++){
                array[i]=rand()%20+1;//1-20
            }
            j++;
            flag = 1;

            pthread_mutex_unlock(&data_mutex);
        }
    }while(j<OPTIMES);
}
```

```c
void* thread2(void* arg){
    int j=0;
    do{
        if(flag){
            pthread_mutex_lock(&data_mutex);

            int sum=0;
            for(int i=0;i<LEN;i++){
                sum+=array[i];
            }
            printf("sum is %d\n",sum);
            j++;
            flag = 0;

            pthread_mutex_unlock(&data_mutex);
        }
    }while(j<OPTIMES);
}
```

```c

int main(){
    pthread_t thread[2];
    struct timespec start,end;
    unsigned long int timeuse;
    clock_gettime(CLOCK_REALTIME,&start);

    pthread_create(thread,NULL,(void*)thread1,NULL);
    pthread_create(thread+1,NULL,(void*)thread2,NULL);
    pthread_join(thread[0],NULL);
    pthread_join(thread[1],NULL);

    return 0;
}

```

运行结果截图如下：

```
ubuntu@ubuntu:~/Desktop$ ./hw
sum is 59
sum is 48
sum is 46
sum is 45
sum is 42
```

在不允许使用标记变量，仅仅可以使用互斥锁的时候，不能。因为同一个锁只能保证本线程操作的原子性，却没有对线程之间的交互执行有任何要求。