OS_Assignment6

Assign	
tag	homework
姓名	周鹏宇
学号	2019K8009929039

第一个题目是之前的作业，故不赘述

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，过程中共用一个偏移量，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以及数组各元素的总和(注意:使用uint64)。

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) 学习了解 pthread_mutex_lock/unlock()函数的功能，并实现上述程序功能。

a. peterson算法

```
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <stdatomic.h>
#define M 10000000
int data[M];
atomic_int myindex;
volatile int flag[2];
volatile int turn;

void *thread_1(void *arg)
{
  for (int i = 0; i < M;) {
    flag[1] = 1;
    turn = 0;
    while (flag[0] && turn == 0) ;
    for (int j = 0; j < 100; j++) {
      data[myindex] = i + 1;
      myindex++;
      i += 2;
    }
    flag[1] = 0;
  }
  return NULL;
}

void *thread_0(void *arg)
{
```

```
  for (int i = 0; i < M;) {
    flag[0] = 1;
    turn = 1;
    while (flag[1] && turn == 1) ;
    for (int j = 0; j < 100; j++) {
      data[myindex] = i;
      myindex++;
      i += 2;
    }
    flag[0] = 0;
  }
  return NULL;
}

int main(void)
{
  unsigned long tick1, tick2;
  pthread_t thread0, thread1;
  pthread_create(&thread0, NULL, thread_0, NULL);
  pthread_create(&thread1, NULL, thread_1, NULL);
  pthread_join(thread0, NULL);
  pthread_join(thread1, NULL);

  int max = 0;

  for (int i = 0; i < M - 1; i++) {
    int diff = abs(data[i] - data[i + 1]);
    if (diff > max) {
      maxDiff = diff;
    }
  }

  unsigned long long int sum = 0;

  for (int i = 0; i < M; i++)
    sum += data[i];

  printf("Sum = %llu\nMax diff is: %d\n", sum, max);
  return 0;
}
```

运行结果如下：

```
Sum = 49999995000000
Max diff is: 2997
./beauty  1.02s user 0.02s system 189% cpu 0.549 total
```

b. lock

```
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
```

```
#include <stdio.h>
#include <stdatomic.h>

#define M 10000000
int data[M];
atomic_int myindex;
pthread_mutex_t the_mutex;

void *thread_1(void *arg)
{
  for (int i = 0; i < M;) {
    pthread_mutex_lock(&the_mutex);
    for (int j = 0; j < 100; j++) {
      data[myindex] = i + 1;
      myindex++;
      i += 2;
    }
    pthread_mutex_unlock(&the_mutex);
  }
  return NULL;
}

void *thread_0(void *arg)
{
  for (int i = 0; i < M;) {
    pthread_mutex_lock(&the_mutex);
    for (int j = 0; j < 100; j++) {
      data[myindex] = i;
      myindex++;
      i += 2;
    }
    pthread_mutex_unlock(&the_mutex);
  }
  return NULL;
}

int main(void)
{
  unsigned long tick1, tick2;
  pthread_t thread0, thread1;

  pthread_create(&thread0, NULL, thread_0, NULL);
  pthread_create(&thread1, NULL, thread_1, NULL);
  pthread_join(thread0, NULL);
  pthread_join(thread1, NULL);

  int max = 0;

  for (int i = 0; i < M - 1; i++) {
    int diff = abs(data[i] - data[i + 1]);
    if (diff > max)
      max = diff;
  }

  unsigned long long int sum = 0;

  for (int i = 0; i < M; i++)
    sum += data[i];
```

```
  printf("Sum = %llu\nMax diff is: %d\n", sum, max);

  return 0;
}
```

运行结果为：

```
Sum = 49999995000000
Max diff is: 1057
./beauty  0.03s user 0.02s system 114% cpu 0.040 total
```

简单总结

- peterson算法始终比lock函数开销大（由运行时间即可看出）
- 双核运行Peterson会出现问题，原因是核间同步并未纳入算法考量，解决方案有：
  - 引入stdatomic.h头文件（参考《21世纪C语言》），用atomic_int声明变量，可以保证对此变量操作的原子性（甚至不需要写Peterson了）
  - 每次操作前用嵌入式汇编刷内存（思路来自舍友，不过这个开销很大）