**homework 5**

钟赟

2016K8009915009

1. 使用Peterson算法

代码：

#include<stdio.h>

#include<pthread.h>

#include<sys/time.h>

#include<math.h>

#include<stdlib.h>

#define true 1

#define false 0

#define RUNTIMES 100

#define MAX 10000000

int flag[2];

int turn;

int index = 0;

int data[MAX];

//thread1

void \*thread1()

{

int i = 0;

while(i < MAX)

{

flag[0] = true;

turn = 1;

while(flag[1] == 1 && turn == 1);

//critical section

int count = 0;

for(count = 0; i < MAX && count < RUNTIMES; i += 2, count ++)

{

data[index] = i; //even ( i+1 for thread 2)

index ++;

}

flag[0] = false;

//reminder

}

}

//thread2

void \*thread2()

{

int i = 0;

while(i < MAX)

{

flag[1] = true;

turn = 0;

while(flag[0] == 1 && turn == 0);

//critical section

int count = 0;

for(count = 0; i < MAX && count < RUNTIMES; i += 2, count ++)

{

data[index] = i + 1; // odd

index ++;

}

flag[1] = false;

//reminder

}

}

int main()

{

pthread\_t tid1, tid2;

int i, err, diff, temp;

struct timeval start\_time, end\_time;

//flag[0] = flag[1] = false;

gettimeofday(&start\_time, NULL);

for(i = 0; i < 1000; i ++)

{

index = 0;

err = pthread\_create(&tid1, NULL, thread1, NULL);

if(err != 0)

{

printf("Create thread 1 failed!\n");

exit(0);

}

err = pthread\_create(&tid2, NULL, thread2, NULL);

if(err != 0)

{

printf("Create thread 2 failed!\n");

exit(0);

}

err = pthread\_join(tid1, NULL);

if(err != 0)

{

printf("Stop thread 1 failed!\n");

exit(0);

}

err = pthread\_join(tid2, NULL);

if(err != 0)

{

printf("Stop thread 2 failed!\n");

exit(0);

}

}

gettimeofday(&end\_time, NULL);

diff = abs(data[0] - data[1]);

for(i = 1; i < MAX && data[i]; i ++)

{

temp = abs(data[i] - data[i+1]);

diff = diff < temp ? temp : diff;

}

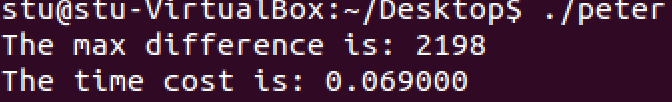
printf("Max abstraction of difference is %d.\n", diff);

printf("Runtime: %f.\n", difftime(end\_time.tv\_sec, start\_time.tv\_sec)/1000);

return 0;

}

测试结果：



1. 使用pthread\_mutex\_lock/unlock()函数

代码：

#include<stdio.h>

#include<pthread.h>

#include<sys/time.h>

#include<math.h>

#include<stdlib.h>

#define true 1

#define false 0

#define RUNTIMES 100

#define MAX 10000000

int index = 0;

int data[MAX];

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

void \*thread1()

{

int i = 0;

while(i < MAX)

{

pthread\_mutex\_lock(&mutex);

//critical section

int count = 0;

for(; i < MAX && count < RUNTIMES; i += 2, count ++)

{

data[index] = i; //even ( i+1 for thread 2)

index ++;

}

pthread\_mutex\_unlock(&mutex);

}

}

//thread2

void \*thread2()

{

int i = 0;

while(i < MAX)

{

pthread\_mutex\_lock(&mutex);

//critical section

int count = 0;

for(; i < MAX && count < RUNTIMES; i += 2, count ++)

{

data[index] = i + 1; // odd

index ++;

}

pthread\_mutex\_unlock(&mutex);

}

}

int main()

{

pthread\_t tid1, tid2;

int i, err, diff, temp;

struct timeval start\_time, end\_time;

gettimeofday(&start\_time, NULL);

for(i = 0; i < 1000; i ++)

{

index = 0;

err = pthread\_create(&tid1, NULL, thread1, NULL);

if(err != 0)

{

printf("Create thread 1 failed!\n");

exit(0);

}

err = pthread\_create(&tid2, NULL, thread2, NULL);

if(err != 0)

{

printf("Create thread 2 failed!\n");

exit(0);

}

err = pthread\_join(tid1, NULL);

if(err != 0)

{

printf("Stop thread 1 failed!\n");

exit(0);

}

err = pthread\_join(tid2, NULL);

if(err != 0)

{

printf("Stop thread 2 failed!\n");

exit(0);

}

}

gettimeofday(&end\_time, NULL);

diff = abs(data[0] - data[1]);

for(i = 1; i < MAX && data[i]; i ++)

{

temp = abs(data[i] - data[i+1]);

diff = diff < temp ? temp : diff;

}

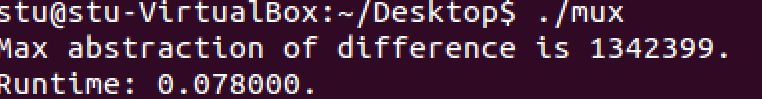
printf("Max abstraction of difference is %d.\n", diff);

printf("Runtime: %f.\n", difftime(end\_time.tv\_sec, start\_time.tv\_sec)/1000);

return 0;

}

测试结果：



1. 使用atomic\_add()函数

代码：

#include<stdio.h>

#include<pthread.h>

#include<sys/time.h>

#include<math.h>

#include<stdlib.h>

#define true 1

#define false 0

#define RUNTIMES 100

#define MAX 10000000

int index = 0;

int data[MAX];

void \*thread1()

{

int i, temp;

for(i = 0; i < MAX; i += 2)

{

temp = \_\_sync\_fetch\_and\_add(&index, 1);

data[temp] = i; //even ( i+1 for thread 2)

}

}

//thread2

void \*thread2()

{

int i, temp;

for(i = 0; i < MAX; i += 2)

{

temp = \_\_sync\_fetch\_and\_add(&index, 1);

data[temp] = i + 1; // odd

}

}

int main()

{

pthread\_t tid1, tid2;

int i, err, diff, temp;

struct timeval start\_time, end\_time;

gettimeofday(&start\_time, NULL);

for(i = 0; i < 1000; i ++)

{

index = 0;

err = pthread\_create(&tid1, NULL, thread1, NULL);

if(err != 0){

printf("Create thread 1 failed!\n");

exit(0);

}

err = pthread\_create(&tid2, NULL, thread2, NULL);

if(err != 0){

printf("Create thread 2 failed!\n");

exit(0);

}

err = pthread\_join(tid1, NULL);

if(err != 0){

printf("Stop thread 1 failed!\n");

exit(0);

}

err = pthread\_join(tid2, NULL);

if(err != 0){

printf("Stop thread 2 failed!\n");

exit(0);

}

}

gettimeofday(&end\_time, NULL);

diff = abs(data[0] - data[1]);

for(i = 1; i < MAX && data[i]; i ++)

{

temp = abs(data[i] - data[i+1]);

diff = diff < temp ? temp : diff;

}

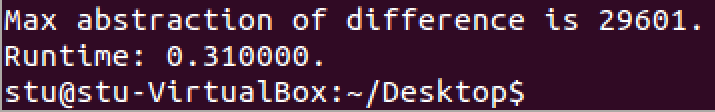
printf("Max abstraction of difference is %d.\n", diff);

printf("Runtime: %f.\n", difftime(end\_time.tv\_sec, start\_time.tv\_sec)/1000);

return 0;

}

执行结果：



从运行速度上来看，peterson算法与mutex函数差不多，比add\_atom函数快很多。从最大差值来看，三种方法相差很大，猜测可能与不同方法的调度时机有关。