Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 2

з дисципліни «Програмування для паралельних комп'ютерних систем»

на тему «Програмування для комп’ютерних систем зі спільною

пам’яттю. WinAPI. Семафори, мютекси, події, критичні секції»

Виконала:

студентка 3 курсу ФІОТ

групи ІО-51

Поштацька К. В.

Залікова – 5115

ПЕРЕВІРИВ:

Доцент, к.т.н.

Корочкін О.В.

Київ – 2018

**Тема:** «Програмування для комп’ютерних систем зі спільною пам’яттю. WinAPI. Семафори, мютекси, події, критичні секції»

**Технічне завдання:** Розробити програму для розв’язання ПКС із СП (структура на рис. 1) математичної задачі:

A = d\* \*(MO\*MK) – e\* Sort()

Мова програмування: C++.



**Виконання роботи:**

**Етап 1. Побудова паралельного алгоритму**

1. ZHc= Sort(ZH), H= N/P
2. Z2Hc = Sort\*( ZHc ,ZHc)
3. Z = Sort\*( Z2Hc ,Z2Hc)
4. A = d\* \*(MOH\*MK) – e\* ZH

Спільні ресурси: d, B, MK, e

**Етап 2. Розроблення алгоритмів роботи кожного процесу**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Задача Т1** | | Точки синхронізації |
| 1. | Уведення e, z | |  |
| 2. | **Сигнал** задачам Т2, T3, T4 про введення e та z | | -- S2/3/4,1 |
| 3. | **Чекати** на уведення даних в задачі Т2, T4 | | -- W2/4,1 |
| 5. | Обчислення ZHc = Sort(ZH) | |  |
| 6. | **Сигнал** Т3 про завершення сортування ZHc | | -- S3,2 |
| 7. | **Чекати** на завершення сортування ZHc в задачах Т2, Т4 | | -- W2/4,2 |
| 7. | **Чекати** на завершення сортування ZHc в задачах Т3 | | -- W3,1 |
| 8. | Z2Hc = Sort\*( ZHc ,ZHc) | |  |
| 9. | **Чекати** на завершення сортування Z2Hc в задачі Т3 | | -- W3,2 |
| 10. | Z = Sort\*( Z2Hc ,Z2Hc) | |  |
| 12. | **Сигнал** задачам Т2, T3, T4 про завершення сортування Z | | -- S2/3/4,2 |
| 16. | Копіювати d1 =d | | -- КД |
| 17. | Копіювати B1=B | | -- КД |
| 18. | Копіювати MK1=MK | | -- КД |
| 19. | Копіювати e1=e | | -- КД |
| 20. | Обчислення AH = d\* \*(MO\*MKH) – e\* Sort(H) | |  |
| 21. | **Чекати** завершення обчислень AH в Т2, T3, Т4 | | -- W2/3/4,3 |
| 24. | Виведення результату А | |  |
|  | |

|  |  |  |
| --- | --- | --- |
|  | **Задача Т2** | Точки синхронізації |
| 1. | Уведення B, MK |  |
| 2. | **Сигнал** Т1, Т3, Т4 про введення В, МК | -- S1/3/4,1 |
| 3. | **Чекати** на уведення даних в задачі Т1, T4 | -- W1/4,1 |
| 5. | Обчислення ZH = Sort(ZH) |  |
| 6. | **Сигнал** задачам Т1, T3 про завершення обчислень | -- S1/3,2 |
| 7. | **Чекати** завершення обчислень Т1 | -- W1,2 |
| 10. | Копіювати d1 =d | -- КД |
| 11. | Копіювати B1=B | -- КД |
| 12. | Копіювати MK1=MK | -- КД |
| 13. | Копіювати e1=e | -- КД |
| 11. | Обчислення AH = d\* \*(MO\*MKH) – e\* Sort( H) |  |
| 12. | **Сигнал** задачі Т1 про завершення обчислень MAH | -- S1,3 |

|  |  |  |
| --- | --- | --- |
|  | **Задача Т3** | Точки синхронізації |
| 1. | **Чекати** на уведення даних в задачі Т1, T2, T4 | -- W1/2/4,1 |
| 4. | Обчислення ZH = Sort(ZH) | -- КД |
| 5. | **Сигнал** задачі Т1 про завершення сортування ZHc | -- S1,1 |
| 7. | **Чекати** на завершення сортування ZHc в задачах Т1,Т2,Т4 | -- W1/2/4,2 |
| 8. | Z2Hc = Sort\*( ZHc ,ZHc) |  |
| 9. | **Сигнал** задачі Т1 про завершення сортування Z2Hc | -- S1,2 |
| 7. | **Чекати** завершення обчислень Т1 | -- W1,3 |
| 9. | Копіювати d1 =d | -- КД |
| 10. | Копіювати B1=B | -- КД |
| 11. | Копіювати MK1=MK | -- КД |
| 12. | Копіювати e1=e | -- КД |
| 13. | Обчислення AH = d\* \*(MO\*MKH) – e\* Sort() |  |
| 14. | **Сигнал** задачі Т1 про завершення обчислень AH | -- S1,3 |

|  |  |  |
| --- | --- | --- |
|  | **Задача Т4** | Точки синхронізації |
| 1. | Уведення d, MO |  |
| 2. | **Сигнал** задачам Т1, T2, T3 про введення d, MO | -- S1/2/3,1 |
| 3. | **Чекати** на уведення даних в задачі Т1, T2 | -- W1/2,1 |
| 5. | Обчислення zH = Sort(zH) |  |
| 6. | **Сигнал** задачам Т1, T3 про завершення обчислень | -- S1/3,2 |
| 7. | **Чекати** завершення обчислень Т1 | -- W1,2 |
| 8. | Копіювати d1 =d | -- КД |
| 9. | Копіювати B1=B | -- КД |
| 10. | Копіювати MK1=MK | -- КД |
| 11. | Копіювати e1=e | -- КД |
| 12. | Обчислення AH = d\* \*(MO\*MKH) – e\* Sort() |  |
| 13. | **Сигнал** задачі Т1 про завершення обчислень AH | -- S1,3 |

**Етап 3. Розроблення структурної схеми взаємодії задач**

На структурній схемі взаємодії задач (рис. 2) задіяні такі події:

* IEvent[3] – для синхронізації з завершення вводу даних в T1, T2, T4;
* SEvent[4] – для синхронізації з завершення сортування ZH T2, T3, T4;
* SortE1 – сигнал задачі Т3 обчислень Z2Hc в Т1
* SortE – сигнал задачам T2, T3, T4 про завершення сортування Z злиттям;

Мютекс: Mut1 - для керування доступом до спільного ресурсу e;

Критична секція: СS - для керування доступом до спільного ресурсу B;

Семафор: Sem1 - для керування доступом до спільного ресурсу d;

Sem2 - для керування доступом до спільного ресурсу MK;

OSem – сигнал задачі Т1 з завершення обчислень AH в T2, T3, T4;

**Етап 4. Розробка програми**

/\*

Laboratory work on the topic "WinAPI. Semaphores, Mutexes, Events, Critical sections"

Variant: A = d\*B(MO\*MK)-e\*Sort(z)

Author: Poshtatska Kateryna IO-51

\*/

//The specifications of main program Lab2

#include "stdafx.h"

#include <iostream>

#include <cstdlib>

#include <windows.h>

#include <ctime>

using namespace std;

void TF1(void);

void TF2(void);

void TF3(void);

void TF4(void);

HANDLE IEvent[3]; //Events for Input Sync

HANDLE SEvent[4];

HANDLE SortE1 = CreateEvent(0, 1, 0, 0); //Event for Sort Sync

HANDLE SortE = CreateEvent(0, 1, 0, 0); //Event for Sort Sync

HANDLE OSem; //Semaphore for Output Sync

CRITICAL\_SECTION CS; //Critical section for shared resource

HANDLE Mut1; //Mutex for shared resource

HANDLE Sem1; //Semaphore for shared resource

HANDLE Sem2; //Semaphore for shared resource

int n = 4; // size

int P = 4; //processors count

int H = n / P; //subsize

//shared resources and global variables

int d;

int e;

int\* A = new int[n];

int\* B = new int[n];

int\* Z = new int[n];

int\*\* MO = new int\*[n];

int\*\* MK = new int\*[n];

int\*\* MR = new int\*[n];

//Main method of Lab2

int main(void)

{

cout << "Lab2 started" << endl;

DWORD TidA, TidB, TidC, TidD;

HANDLE hThreadA, hThreadB, hThreadC, hThreadD;

/\*

Variable n using for rhe size of all matrix and vectors in program.

data is an exeplar of class Data with parametrs n and p.

\*/

for (int i = 0; i < n; i++)

{

MO[i] = new int[n];

MK[i] = new int[n];

MR[i] = new int[n];

}

//Announcement of threads with their parametrs. Getting their handles

hThreadA = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)TF1, NULL, 0, &TidA);

hThreadB = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)TF2, NULL, 0, &TidB);

hThreadC = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)TF3, NULL, 0, &TidC);

hThreadD = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)TF4, NULL, 0, &TidD);

//Setting priority of threads

SetThreadPriority(hThreadA, THREAD\_PRIORITY\_LOWEST);

SetThreadPriority(hThreadB, THREAD\_PRIORITY\_NORMAL);

SetThreadPriority(hThreadC, THREAD\_PRIORITY\_HIGHEST);

SetThreadPriority(hThreadD, THREAD\_PRIORITY\_HIGHEST);

//Announcement of events with their parametrs. Getting their handles

for (int i = 0; i < 3; i++) {

IEvent[i] = CreateEvent(0, 1, 0, 0);

SEvent[i] = CreateEvent(0, 1, 0, 0);

}

SEvent[3] = CreateEvent(0, 1, 0, 0);

//initialize of critical section

InitializeCriticalSection(&CS);

//Announcement of mutex with it's parametrs. Getting it's handle

Mut1 = CreateMutex(0, 0, 0);

//Announcement of semaphore with it's parametrs. Getting it's handle

Sem1 = CreateSemaphore(0, 1, 1, 0);

Sem2 = CreateSemaphore(0, 1, 1, 0);

OSem = CreateSemaphore(0, 0, 3, 0);

//Waiting the end of running thread

WaitForSingleObject(hThreadA, INFINITE);

WaitForSingleObject(hThreadB, INFINITE);

WaitForSingleObject(hThreadC, INFINITE);

WaitForSingleObject(hThreadD, INFINITE);

//Closing the descriptors

CloseHandle(hThreadA);

CloseHandle(hThreadB);

CloseHandle(hThreadC);

CloseHandle(hThreadD);

for (int i = 0; i < 3; i++) {

CloseHandle(IEvent[i]);

CloseHandle(SEvent[i]);

}

CloseHandle(SEvent[3]);

CloseHandle(SortE);

CloseHandle(SortE1);

CloseHandle(Sem1);

CloseHandle(Sem2);

CloseHandle(OSem);

CloseHandle(Mut1);

DeleteCriticalSection(&CS);

cout << "Lab2 finished" << endl;

system("pause");

}

//Thread function TF1

void TF1(void) {

int d1;

int e1;

int\* B1;

int\* X = Z;

int\*\* MK1;

cout << "TF1 started" << endl;

//srand((unsigned)time(NULL));

for (int i = 0; i < n; i++) {

Z[i] = 1;

}

Z[0] = -1;

Z[1] = 1;

Z[2] = 2;

Z[3] = 3;

e = 1;

//Input sync

SetEvent(IEvent[0]);

WaitForMultipleObjects(3, IEvent, TRUE, INFINITE);

bool Finished;

int Temp = 0;

for (int i = 0; i < H; i++) {

Finished = true;

for (int j = 1; j < (H - i); j++) {

if (Z[j - 1] > Z[j]) {

Temp = Z[j - 1];

Z[j - 1] = Z[j];

Z[j] = Temp;

Finished = false;

}

}

if (Finished) {

break;

}

}

SetEvent(SEvent[0]);

WaitForMultipleObjects(4, SEvent, TRUE, INFINITE);

int num = 0;

int\* C = new int[2] {0, H};

for (int i = 0; i < n/2; i++) {

Temp = INT\_MAX;

for (int j = 0; j < 2; j++) {

if ((C[j] - H\*j)<H) {

if (Z[C[j]] < Temp) {

num = j;

Temp = Z[C[j]];

}

}

}

X[i] = Temp;

C[num]++;

}

for (int i = 0; i < n/2; i++) {

Z[i] = X[i];

}

WaitForSingleObject(SortE1, INFINITE);

num = 0;

C = new int[P] {0, 2 \* H};

for (int i = 0; i < n; i++) {

Temp = INT\_MAX;

for (int j = 0; j < P; j++) {

if ((C[j] - 2\*H \* j)<2\*H) {

if (Z[C[j]] < Temp) {

num = j;

Temp = Z[C[j]];

}

}

}

X[i] = Temp;

C[num]++;

}

for (int i = 0; i < n; i++) {

Z[i] = X[i];

}

SetEvent(SortE);

//Coping shared resource

WaitForSingleObject(Sem1, INFINITE);

d1 = d;

ReleaseSemaphore(Sem1, 1, 0);

WaitForSingleObject(Sem2, INFINITE);

MK1 = MK;

ReleaseSemaphore(Sem2, 1, 0);

WaitForSingleObject(Mut1, INFINITE);

e1 = e;

ReleaseMutex(Mut1);

EnterCriticalSection(&CS);

B1 = B;

LeaveCriticalSection(&CS);

for (int i = 0; i < H; i++) {

for (int j = 0; j < n; j++) {

MR[i][j] = 0;

for (int k = 0; k < n; k++) {

MR[i][j] += MO[i][k] \* MK1[k][j];

}

MR[i][j] \*= d1;

}

}

for (int i = 0; i < H; i++)

{

A[i] = 0;

for (int j = 0; j < n; j++)

{

A[i] += MR[i][j] \* B1[j];

}

A[i] -= e1 \* Z[i];

}

//Output sync

WaitForSingleObject(OSem, INFINITE);

WaitForSingleObject(OSem, INFINITE);

WaitForSingleObject(OSem, INFINITE);

cout << "A = " << endl;

for (int i = 0; i < n; i++) {

cout << A[i] << " ";

}

cout << endl;

cout << "\nTF1 finished" << endl;

}

//Thread function TF2

void TF2(void) {

int d2;

int e2;

int\* B2;

int\*\* MK2;

cout << "TF2 started" << endl;

for (int i = 0; i < n; i++) {

B[i] = 1;

for (int j = 0; j < n; j++) {

MK[i][j] = 1;

}

}

//Input sync

SetEvent(IEvent[1]);

WaitForMultipleObjects(3, IEvent, TRUE, INFINITE);

bool Finished;

int Temp = 0;

for (int i = 0; i < H; i++) {

Finished = true;

for (int j = 1; j<(H - i); j++) {

if (Z[j - 1 + H] > Z[j + H]) {

Temp = Z[j - 1 + H];

Z[j - 1 + H] = Z[j + H];

Z[j + H] = Temp;

Finished = false;

}

}

if (Finished) {

break;

}

}

SetEvent(SEvent[1]);

WaitForSingleObject(SortE, INFINITE);

//Coping shared resource

WaitForSingleObject(Sem1, INFINITE);

d2 = d;

ReleaseSemaphore(Sem1, 1, 0);

WaitForSingleObject(Sem2, INFINITE);

MK2 = MK;

ReleaseSemaphore(Sem2, 1, 0);

WaitForSingleObject(Mut1, INFINITE);

e2 = e;

ReleaseMutex(Mut1);

EnterCriticalSection(&CS);

B2 = B;

LeaveCriticalSection(&CS);

for (int i = H; i < 2 \* H; i++) {

for (int j = 0; j < n; j++) {

MR[i][j] = 0;

for (int k = 0; k < n; k++) {

MR[i][j] += MO[i][k] \* MK2[k][j];

}

MR[i][j] \*= d2;

}

}

for (int i = H; i < 2 \* H; i++)

{

A[i] = 0;

for (int j = 0; j < n; j++)

{

A[i] += MR[i][j] \* B2[j];

}

A[i] -= e2 \* Z[i];

}

//Output sync

ReleaseSemaphore(OSem, 1, 0);

cout << "TF2 finished" << endl;

}

//Thread function TF3

void TF3(void) {

int d3;

int e3;

int\* B3;

int\*\* MK3;

cout << "TF3 started" << endl;

//Input sync

WaitForMultipleObjects(3, IEvent, TRUE, INFINITE);

bool Finished;

int Temp = 0;

for (int i = 0; i < H; i++) {

Finished = true;

for (int j = 1; j < (H - i); j++) {

if (Z[j - 1 + 2 \* H] > Z[j + 2 \* H]) {

Temp = Z[j - 1 + 2 \* H];

Z[j - 1 + 2 \* H] = Z[j + 2 \* H];

Z[j + 2 \* H] = Temp;

Finished = false;

}

}

if (Finished) {

break;

}

}

SetEvent(SEvent[2]);

WaitForMultipleObjects(4, SEvent, TRUE, INFINITE);

int\* X = new int[n];

int num = 0;

int\* C = new int[2]{ 2\*H, 3\*H };

for (int i = n / 2; i < n; i++) {

Temp = INT\_MAX;

for (int j = 0; j < 2; j++) {

if ((C[j] - H \* (j+1))<H) {

if (Z[C[j]] < Temp) {

num = j;

Temp = Z[C[j]];

}

}

}

X[i] = Temp;

C[num]++;

}

for (int i = n / 2; i < n; i++) {

Z[i] = X[i];

}

SetEvent(SortE1);

WaitForSingleObject(SortE, INFINITE);

//Coping shared resource

WaitForSingleObject(Sem1, INFINITE);

d3 = d;

ReleaseSemaphore(Sem1, 1, 0);

WaitForSingleObject(Sem2, INFINITE);

MK3 = MK;

ReleaseSemaphore(Sem2, 1, 0);

WaitForSingleObject(Mut1, INFINITE);

e3 = e;

ReleaseMutex(Mut1);

EnterCriticalSection(&CS);

B3 = B;

LeaveCriticalSection(&CS);

for (int i = 2 \* H; i < 3 \* H; i++) {

for (int j = 0; j < n; j++) {

MR[i][j] = 0;

for (int k = 0; k < n; k++) {

MR[i][j] += MO[i][k] \* MK3[k][j];

}

MR[i][j] \*= d3;

}

}

for (int i = 2 \* H; i < 3 \* H; i++)

{

A[i] = 0;

for (int j = 0; j < n; j++)

{

A[i] += MR[i][j] \* B3[j];

}

A[i] -= e3 \* Z[i];

}

//Output sync

ReleaseSemaphore(OSem, 1, 0);

cout << "TF3 finished" << endl;

}

//Thread function TF4

void TF4(void) {

int d4;

int e4;

int\* B4;

int\*\* MK4;

cout << "TF4 started" << endl;

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

MO[i][j] = 1;

}

}

d = 1;

//Input sync

SetEvent(IEvent[2]);

WaitForMultipleObjects(3, IEvent, TRUE, INFINITE);

bool Finished;

int Temp = 0;

for (int i = 0; i < H; i++) {

Finished = true;

for (int j = 1; j < (H - i); j++) {

if (Z[j - 1 + 3 \* H] > Z[j + 3 \* H]) {

Temp = Z[j - 1 + 3 \* H];

Z[j - 1 + 3 \* H] = Z[j + 3 \* H];

Z[j + 3 \* H] = Temp;

Finished = false;

}

}

if (Finished) {

break;

}

}

SetEvent(SEvent[3]);

WaitForSingleObject(SortE, INFINITE);

//Coping shared resource

WaitForSingleObject(Sem1, INFINITE);

d4 = d;

ReleaseSemaphore(Sem1, 1, 0);

WaitForSingleObject(Sem2, INFINITE);

MK4 = MK;

ReleaseSemaphore(Sem2, 1, 0);

WaitForSingleObject(Mut1, INFINITE);

e4 = e;

ReleaseMutex(Mut1);

EnterCriticalSection(&CS);

B4 = B;

LeaveCriticalSection(&CS);

for (int i = 3 \* H; i < 4 \* H; i++) {

for (int j = 0; j < n; j++) {

MR[i][j] = 0;

for (int k = 0; k < n; k++) {

MR[i][j] += MO[i][k] \* MK4[k][j];

}

MR[i][j] \*= d4;

}

}

for (int i = 3 \* H; i < 4 \* H; i++)

{

A[i] = 0;

for (int j = 0; j < n; j++)

{

A[i] += MR[i][j] \* B4[j];

}

A[i] -= e4 \* Z[i];

}

//Output sync

ReleaseSemaphore(OSem, 1, 0);

cout << "TF4 finished" << endl;

}