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»

ВИКОНАВ:

студент ІІІ курсу ФІОТ

групи ІО-51

Обозний Д.М.

Залікова – 5112

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:

MA = MB \* max( Z ) + MO \* MK \* min( S )\* d

Мова програмування: C++.

Засоби організації взаємодії процесів: семафори мови Ада з пакета Ada.Synchronous\_Task\_Control.



Рис. 1 Структурна схема ПКС

**Виконання роботи:**

**Етап 1. Побудова паралельного алгоритму**

MAH = MBH \* max( ZH ) + MO \* MKH \* min( SH ) \* d , де ZH – Н елементів вектора Z.

1. Визначити maxi = max( ZH ); max = max(maxi) та   
   mini = min( SH ); min = min(mini)
2. Обчислити MAH = MBH \* max+ MO \* MKH \* min \* d

Спільні ресурси: d, max, min

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

|  |  |  |
| --- | --- | --- |
|  | **Задача Т1** | Точки синхронізації |
| 1. | Уведення MK, S |  |
| 2. | **Сигнал** задачам Т2, T3, T4 про введення MK та S | -- S2/3/4,1 |
| 3. | **Чекати** на уведення данних в задачі Т3 | -- W3,1 |
| 4. | **Чекати** на уведення данних в задачі Т4 | -- W4,1 |
| 5. | Обчислення max = max( ZH ) та min = min( SH ) | -- КД |
| 6. | **Сигнал** задачам Т2, T3, T4 про завершення обчислень max та min | -- S2/3/4,2 |
| 7. | **Чекати** завершення обчислень max та min Т2 | -- W2,1 |
| 8. | **Чекати** завершення обчислень max та min Т3 | -- W3,2 |
| 9. | **Чекати** завершення обчислень max та min Т4 | -- W4,2 |
| 10. | Копіювати max1 = max | -- КД |
| 11. | Копіювати min1 = min | -- КД |
| 12. | Копіювати MO1 = MO | -- КД |
| 13. | Копіювати d1 = d | -- КД |
| 14. | Обчислення MAH = MBH \* max1+ MO \* MKH \* min1 \* d1 |  |
| 15. | **Сигнал** задачі Т4 про завершення обчислень MAH в Т1 | -- S4,3 |

|  |  |  |
| --- | --- | --- |
|  | **Задача Т2** | Точки синхронізації |
| 3. | **Чекати** на уведення данних в задачі Т1 | -- W1,1 |
| 3. | **Чекати** на уведення данних в задачі Т3 | -- W3,1 |
| 4. | **Чекати** на уведення данних в задачі Т4 | -- W4,1 |
| 5. | Обчислення max = max( ZH ) та min = min( SH ) | -- КД |
| 6. | **Сигнал** задачам Т1, T3, T4 про завершення обчислень max та min | -- S1/3/4,2 |
| 7. | **Чекати** завершення обчислень max та min Т1 | -- W1,2 |
| 8. | **Чекати** завершення обчислень max та min Т3 | -- W3,2 |
| 9. | **Чекати** завершення обчислень max та min Т4 | -- W4,2 |
| 10. | Копіювати max2 = max | -- КД |
| 11. | Копіювати min2 = min | -- КД |
| 12. | Копіювати MO2 = MO | -- КД |
| 13. | Копіювати d2 = d | -- КД |
| 11. | Обчислення MAH = MBH \* max2+ MO \* MKH \* min2 \* d2 |  |
| 12. | **Сигнал** задачі Т4 про завершення обчислень MAH в Т1 | -- S4,3 |

|  |  |  |
| --- | --- | --- |
|  | **Задача Т3** | Точки синхронізації |
| 1. | Уведення MO, Z |  |
| 2. | **Сигнал** задачам Т1, T2, T4 про введення MO, Z | -- S1/2/4,1 |
| 3. | **Чекати** на уведення данних в задачі Т1 | -- W1,1 |
| 4. | **Чекати** на уведення данних в задачі Т4 | -- W4,1 |
| 5. | Обчислення max = max( ZH ) та min = min( SH ) | -- КД |
| 6. | **Сигнал** задачам Т1, T2, T4 про завершення обчислень max та min | -- S1/2/4,2 |
| 7. | **Чекати** завершення обчислень max та min Т1 | -- W1,2 |
| 8. | **Чекати** завершення обчислень max та min Т2 | -- W2,1 |
| 9. | **Чекати** завершення обчислень max та min Т4 | -- W4,2 |
| 10. | Копіювати max3 = max | -- КД |
| 11. | Копіювати min3 = min | -- КД |
| 12. | Копіювати MO3 = MO | -- КД |
| 13. | Копіювати d3 = d | -- КД |
| 14. | Обчислення MAH = MBH \* max3+ MO \* MKH \* min3 \* d3 |  |
| 15. | **Сигнал** задачі Т4 про завершення обчислень MAH в Т1 | -- S4,3 |

|  |  |  |
| --- | --- | --- |
|  | **Задача Т4** | Точки синхронізації |
| 1. | Уведення MB, d |  |
| 2. | **Сигнал** задачам Т1, T2, T3 про введення MB, d | -- S1/2/3,1 |
| 3. | **Чекати** на уведення данних в задачі Т1 | -- W1,1 |
| 4. | **Чекати** на уведення данних в задачі Т3 | -- W3,1 |
| 5. | Обчислення max = max( ZH ) та min = min( SH ) | -- КД |
| 6. | **Сигнал** задачам Т1, T2, T3 про завершення обчислень max та min | -- S1/2/3,2 |
| 7. | **Чекати** завершення обчислень max та min Т1 | -- W1,2 |
| 8. | **Чекати** завершення обчислень max та min Т2 | -- W2,1 |
| 9. | **Чекати** завершення обчислень max та min Т3 | -- W3,2 |
| 10. | Копіювати max4 = max | -- КД |
| 11. | Копіювати min4 = min | -- КД |
| 12. | Копіювати MO3 = MO | -- КД |
| 13. | Копіювати d3 = d | -- КД |
| 14. | Обчислення MAH = MBH \* max4 + MO \* MKH \* min4 \* d4 |  |
| 15. | **Чекати** завершення обчислень MAH в Т1 | -- W1,3 |
| 16. | **Чекати** завершення обчислень MAH в Т2 | -- W2,2 |
| 17. | **Чекати** завершення обчислень MAH в Т3 | -- W3,3 |
| 18. | Виведення результату MА |  |

**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, T3, T4;
* SEvent[3] – для синхронізації з завершення обчислень min та max в T1, T2, T3, T4;
* OEvent[3] – для синхронізації з завершення обчислень в T1, T2, T3 і виведення результатів в T4;
* Eve – для керування доступом до спільного ресурсу MO;

Мютекс: Mut - для керування доступом до спільного ресурсу min;

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

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

**Етап 4. Розробка програми**

/\*

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

Variant: MA = MB \* max (Z) + MO \* MK \* min (S) \* d

Created: 18.02.2018 8:52 PM

Author: Obozniy Dmitriy IO-51

\*/

//The specifications of main program Lab2

#include "stdafx.h"

#include "Data.h"

#include <iostream>

#include <cstdlib>

#include <windows.h>

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]; //Events for min/max Calc Sync

HANDLE OEvent[3]; //Events for Output Sync

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

HANDLE Mut; //Mutex for shared resource min

HANDLE Sem; //Semaphore for shared resource d

HANDLE Eve; //Event for shared resource MO

int n = 20; // size

int P = 4; //processors count

int H = n / P; //subsize

//shared resources and global variables

int min = INT\_MAX;

int max = INT\_MIN;

int d;

int\* Z = new int[n];

int\* S = new int[n];

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

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

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

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

//Main method of Lab2

int main(void)

{

cout << "Lab4 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.

\*/

Data(n,P);

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

{

MA[i] = new int[n];

MB[i] = new int[n];

MO[i] = new int[n];

MK[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, CREATE\_SUSPENDED, &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);

//Starting thread C

ResumeThread(hThreadC);

//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);

OEvent[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

Mut = CreateMutex(0, 0, 0);

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

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

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

Eve = CreateEvent(0, 0, 1, 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(OEvent[i]);

}

CloseHandle(SEvent[3]);

CloseHandle(Eve);

CloseHandle(Sem);

CloseHandle(Mut);

DeleteCriticalSection(&CS);

cout << "Lab4 finished" << endl;

system("pause");

}

//Thread function TF1

void TF1(void) {

int d1;

int max1 = INT\_MAX;

int min1 = INT\_MIN;

int\*\* MO1;

cout << "TF1 started" << endl;

if (n < 1)

{

S = Vector\_Input("S");

MK = Matrix\_Input("MK");

}

else {

S = Vector\_Input1(1);

MK = Matrix\_Input1(1);

}

//Input sync

SetEvent(IEvent[0]);

WaitForMultipleObjects(3, IEvent, TRUE, INFINITE);

//Calculating min and max

Func1(Z, S, max1, min1, 0);

WaitForSingleObject(Mut, INFINITE);

if (min1 < min) {

min = min1;

}

ReleaseMutex(Mut);

EnterCriticalSection(&CS);

if (max1 > max) {

max = max1;

}

LeaveCriticalSection(&CS);

//End of calc min/max sync

SetEvent(SEvent[0]);

WaitForMultipleObjects(4, SEvent, TRUE, INFINITE);

//Coping shared resource

WaitForSingleObject(Sem, INFINITE);

d1 = d;

ReleaseSemaphore(Sem, 1, 0);

EnterCriticalSection(&CS);

max1 = max;

LeaveCriticalSection(&CS);

WaitForSingleObject(Mut, INFINITE);

min1 = min;

ReleaseMutex(Mut);

WaitForSingleObject(Eve, INFINITE);

MO1 = MO;

SetEvent(Eve);

//Calculating MA

Func2(MA, MB, MO1, MK, d1, max1, min1, 0);

//Output sync

SetEvent(OEvent[0]);

cout << "TF1 finished" << endl;

}

//Thread function TF2

void TF2(void) {

int d2;

int max2 = INT\_MAX;

int min2 = INT\_MIN;

int\*\* MO2;

cout << "TF2 started" << endl;

//Input sync

WaitForMultipleObjects(3, IEvent, TRUE, INFINITE);

//Calculating min and max

Func1(Z, S, max2, min2, 1);

WaitForSingleObject(Mut, INFINITE);

if (min2 < min) {

min = min2;

}

ReleaseMutex(Mut);

EnterCriticalSection(&CS);

if (max2 > max) {

max = max2;

}

LeaveCriticalSection(&CS);

//End of calc min/max sync

SetEvent(SEvent[1]);

WaitForMultipleObjects(4, SEvent, TRUE, INFINITE);

//Coping shared resource

WaitForSingleObject(Sem, INFINITE);

d2 = d;

ReleaseSemaphore(Sem, 1, 0);

EnterCriticalSection(&CS);

max2 = max;

LeaveCriticalSection(&CS);

WaitForSingleObject(Mut, INFINITE);

min2 = min;

ReleaseMutex(Mut);

WaitForSingleObject(Eve, INFINITE);

MO2 = MO;

SetEvent(Eve);

//Calculating MA

Func2(MA, MB, MO2, MK, d2, max2, min2, 1);

//Output sync

SetEvent(OEvent[1]);

cout << "TF2 finished" << endl;

}

//Thread function TF3

void TF3(void) {

int d3;

int max3 = INT\_MAX;

int min3 = INT\_MIN;

int\*\* MO3;

cout << "TF3 started" << endl;

if (n < 1)

{

Z = Vector\_Input("Z");

MO = Matrix\_Input("MO");

}

else {

Z = Vector\_Input1(1);

MO = Matrix\_Input1(1);

}

//Input sync

SetEvent(IEvent[1]);

WaitForMultipleObjects(3, IEvent, TRUE, INFINITE);

//Calculating min and max

Func1(Z, S, max3, min3, 2);

WaitForSingleObject(Mut, INFINITE);

if (min3 < min) {

min = min3;

}

ReleaseMutex(Mut);

EnterCriticalSection(&CS);

if (max3 > max) {

max = max3;

}

LeaveCriticalSection(&CS);

//End of calc min/max sync

SetEvent(SEvent[2]);

WaitForMultipleObjects(4, SEvent, TRUE, INFINITE);

//Coping shared resource

WaitForSingleObject(Sem, INFINITE);

d3 = d;

ReleaseSemaphore(Sem, 1, 0);

EnterCriticalSection(&CS);

max3 = max;

LeaveCriticalSection(&CS);

WaitForSingleObject(Mut, INFINITE);

min3 = min;

ReleaseMutex(Mut);

WaitForSingleObject(Eve, INFINITE);

MO3 = MO;

SetEvent(Eve);

//Calculating MA

Func2(MA, MB, MO3, MK, d3, max3, min3, 2);

//Output sync

SetEvent(OEvent[2]);

cout << "TF3 finished" << endl;

}

//Thread function TF4

void TF4(void) {

int d4;

int max4 = INT\_MAX;

int min4 = INT\_MIN;

int\*\* MO4;

cout << "TF4 started" << endl;

d = 1;

if (n < 1)

{

MB = Matrix\_Input("MB");

}

else {

MB = Matrix\_Input1(1);

MA = Matrix\_Input1(1);

}

//Input sync

SetEvent(IEvent[2]);

WaitForMultipleObjects(3, IEvent, TRUE, INFINITE);

//Calculating min and max

Func1(Z, S, max4, min4, 3);

WaitForSingleObject(Mut, INFINITE);

if (min4 < min) {

min = min4;

}

ReleaseMutex(Mut);

EnterCriticalSection(&CS);

if (max4 > max) {

max = max4;

}

LeaveCriticalSection(&CS);

//End of calc min/max sync

SetEvent(SEvent[3]);

WaitForMultipleObjects(4, SEvent, TRUE, INFINITE);

//Coping shared resource

WaitForSingleObject(Sem, INFINITE);

d4 = d;

ReleaseSemaphore(Sem, 1, 0);

EnterCriticalSection(&CS);

max4 = max;

LeaveCriticalSection(&CS);

WaitForSingleObject(Mut, INFINITE);

min4 = min;

ReleaseMutex(Mut);

WaitForSingleObject(Eve, INFINITE);

MO4 = MO;

SetEvent(Eve);

//Calculating MA

Func2(MA, MB, MO4, MK, d4, max4, min4, 3);

//Output sync

WaitForMultipleObjects(3, OEvent, TRUE, INFINITE);

//Output MA

if (n <= 20) {

cout << "MA = \n";

Matrix\_Output(MA);

}

cout << "TF4 finished" << endl;

}