НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №2

з дисципліни **«**Програмування паралельних комп’ютерних систем**»**

Виконав:

студент 3 курсу

ФІОТ гр. ІО-43

Крут Владислав

Перевірив:

Корочкін О. В.

Київ – 2017 р.

**Тема:** Програмування для комп’ютерних систем зі спільною пам’яттю. Бібліотека Win32

**Технічне завдання:** Розробити програму для розв’язання ПКС із СП математичної задачі: MA = d \* (MO \* MK) + min(Z) \* MR

Бібліотека: Win32.

Засоби організації взаємодії: семафори, мютекси, критичні секції, події

Засоби взаємодії: семафори.

C:\Users\Lilu\Downloads\PPKS_2.png

Структурна схема ПКС

**Виконання роботи:**

**Етап 1. Побудова паралельного алгоритму**

1. mi = min(ZH), i =
2. m = min(mi, m)
3. MAH = d\*MOH\*MK + m\*MRH

Спільний ресурс: m, d, MK

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

|  |  |
| --- | --- |
| **Задача Т1** | ТС, КД |
| 1. Введення d |  |
| 2. Сигнал задачам Т2, Т3, Т4 про введення d | S2,3,4 – 1 |
| 3. Чекати на введення MO, MR у задачі Т3 | W3 – 1 |
| 4. Чекати на введення Z, MK у задачі T4 | W4 – 2 |
| 5. Копіювати MK1 := MK | КД |
| 6. Копіювати d1:=d | КД |
| 7. Обчислення m1 := min(ZH) |  |
| 8. Обчислення m: = min(m, m1) | КД |
| 9. Чекати на завершення обчислень m в T2, T3, T4 | W2,3,4 – 3 |
| 10. Копіювання m1: = m | КД |
| 11. Сигнал T2, T3, T4 про завершення обчислень m | S2,3,4 – 2 |
| 12. Обчислення MAH = d1\*MOH\*MK1 + m1\*MRH |  |
| 13. Чекати на завершення обчислень MA в T2, T3, T4 | W2,3,4 – 4 |
| 14. Виведення MA |  |
| **Задача T2** | ТС, КД |
| 1. Чекати на введення d у задачі T1 | W1 – 1 |
| 2. Чекати на введення MO, MR у задачі T3 | W3 – 2 |
| 3. Чекати на введення Z у задачі T4 | W4 – 3 |
| 4. Копіювати MK2:= MK |  |
| 5. Копіювати d2:=d | КД |
| 6. Обчислення m2 := min(ZH) |  |
| 7. Обчислення m: = min(m,m2) | КД |
| 8. Сигнал Т1 про про завершення обчислень m | S1 – 2 |
| 9. Чекати сигналу від T1 про завершення обчислень m | W1 – 4 |
| 10. Копіювання m2: = m | КД |
| 11. Обчислення MAH = d2\*MOH\*MK2 + m2\*MRH |  |
| 12. Сигнал T1 про завершення обчислень MA | S1 – 3 |
| **Задача Т3** | ТС, КД |
| 1. Введення МO, MR |  |
| 2. Сигнал задачам Т1, Т3, Т4 про введення МO, MR | S1,3,4 – 1 |
| 3. Чекати на введення d у задачі T1 | W1 – 1 |
| 4. Чекати на введення Z, MK у задачі T4 | W4 – 2 |
| 5. Копіювати MK3:= MK | КД |
| 6. Копіювати d3:= d | КД |
| 7. Обчислення m3 := min(ZH) |  |
| 8. Обчислення m: = min(m,m3) | КД |
| 9. Сигнал Т1 про про завершення обчислень m | S1 – 2 |
| 10. Чекати сигналу від T1 про завершення обчислень m | W1 – 3 |
| 11. Копіювання m3: = m | КД |
| 12. Обчислення MAH = d3\*MOH\*MK3 + m3\*MRH |  |
| 13. Сигнал T1 про завершення обчислень MA | S1 – 3 |
| **Задача Т4** | ТС, КД |
| 1. Введення MK, Z |  |
| 2. Сигнал задачам T1, T2, T3 про введення MK, Z | S1,2,3 – 1 |
| 3. Чекати на введення d у задачі T1 | W1 – 1 |
| 4. Чекати на введення MO, MR у задачі T3 | W3 – 2 |
| 5. Копіювати MK4:= MK | КД |
| 6. Копіювати d4:= d | КД |
| 7. Обчислення m4 := min(ZH) |  |
| 8. Обчислення m: = min(m,m4) | КД |
| 9. Сигнал Т1 про про завершення обчислень m | S1 – 2 |
| 10. Чекати сигналу від T1 про завершення обчислень m | W1 – 3 |
| 11. Копіювання m4: = m | КД |
| 12. Обчислення MAH = d4\*MOH\*MK4 + m4\*MRH |  |
| 13. Сигнал T1 про завершення обчислень MA | S1 – 3 |

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

На структурній схемі взаємодії задач уведено такі умовні позначення

* Evn1\_234 – подія для синхронізації із завершенням вводу в T1
* Evn3\_124 – подія для синхронізації із завершенням вводу в T3
* Evn4\_123 – подія для синхронізації із завершенням вводу в T4
* CrSec – для доступу до спільного ресурсу d, m
* Mute – для доступу до спільного ресурсу MC
* Sem\_m[3], Sem\_max – для синхронізації обчислень максимуму вектора Z
* Sem\_MA[3] – для синхронізації решти обчислень і виведення результату

C:\Users\Lilu\Downloads\PPKS_2_Scheme.png

Структурна схема взаємодії задач

Етап 4. Розроблення програми

/\*------------------------------------------------------------------

-- --

-- Parallel and Distributed Computing --

-- Laboratory work #2. Win32 --

-- --

-- --

-- Task: MA = d\*MO\*MK + min(Z)\*MR --

-- --

-- Author: Krut Vladyslav, group IO-43 --

-- Date: 09.04.2017 --

-- --

------------------------------------------------------------------ \*/

#include "stdafx.h"

#include <iostream>

#include <windows.h>

using namespace std;

typedef int\* vector;

typedef int\*\* matrix;

const int N = 1000;

const int P = 4;

const int H = N / P;

int d, m;

vector Z = new int[N];

matrix MA = new vector[N],

MO = new vector[N],

MK = new vector[N],

MR = new vector[N];

HANDLE Evn1\_234, Evn3\_124, Evn4\_123,

Mute, Sem\_m[3], Sem\_MA[3], Sem\_min;

CRITICAL\_SECTION CrSec;

//-----------------------------------------T1--------------------------------------------

void T1(){

int d1, m1, s;

matrix MK1 = new vector[N];

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

{

MK1[i] = new int[N];

};

cout << "Thread T1 started" << endl;

//1. Введення d

d = 1;

//2. Сигнал задачам Т2, Т3, Т4 про введення d

SetEvent(Evn1\_234);

//3. Чекати на введення MO, MR у задачі Т3

WaitForSingleObject(Evn3\_124, INFINITE);

//4. Чекати на введення MK, Z у задачі T4

WaitForSingleObject(Evn4\_123, INFINITE);

//5. Копіювати MK1 := MK

WaitForSingleObject(Mute, INFINITE);

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

{

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

{

MK1[i][j] = MK[i][j];

}

}

ReleaseMutex(Mute);

//6. Копіювання d

EnterCriticalSection(&CrSec);

d1 = d;

LeaveCriticalSection(&CrSec);

//7. Обчислення m1 := min(ZH)

m1 = 0;

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

{

if (Z[i]<m1){

m1 = Z[i];

}

}

//8. Обчислення m: = min(m, m1)

EnterCriticalSection(&CrSec);

m = min(m, m1);

LeaveCriticalSection(&CrSec);

//9. Чекати на завершення обчислень m в T2, T3, T4

WaitForMultipleObjects(3, Sem\_m, TRUE, INFINITE);

//10. Копіювання m1: = m

EnterCriticalSection(&CrSec);

m1 = m;

LeaveCriticalSection(&CrSec);

//11. Сигнал T2, T3, T4 про завершення обчислень m

ReleaseSemaphore(Sem\_min, 1, NULL);

//12. Обчислення MAH = d1\*MOH\*MK1 + m1\*MRH

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

{

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

{

s = 0;

for (int k = 0; k < N; k++)

{

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

}

MA[i][j] = s\*d1 + m1\*MO[i][j];

}

}

//13. Чекати на завершення обчислень MA в T2, T3, T4

WaitForMultipleObjects(3, Sem\_MA, TRUE, INFINITE);

//14. Виведення MA

if (N<10)

{

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

{

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

{

cout << MA[i][j];

}

cout << endl;

}

}

cout << "Thread T1 finished" << endl;

}

//-------------------------------------------T2-------------------------------------------

void T2(){

int d2, s, m2;

matrix MK2 = new vector[N];

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

{

MK2[i] = new int[N];

};

cout << "Thread T2 started" << endl;

//1. Чекати на введення d у задачі T1

WaitForSingleObject(Evn1\_234, INFINITE);

//2. Чекати на введення MO, MR у задачі T3

WaitForSingleObject(Evn3\_124, INFINITE);

//3. Чекати на введення MK, Z у задачі T4

WaitForSingleObject(Evn4\_123, INFINITE);

//4. Копіювати MK2:= MK

WaitForSingleObject(Mute, INFINITE);

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

{

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

{

MK2[i][j] = MK[i][j];

}

}

ReleaseMutex(Mute);

//5. Копіювання d

EnterCriticalSection(&CrSec);

d2 = d;

LeaveCriticalSection(&CrSec);

//6. Обчислення m2 := min(ZH)

m2 = 0;

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

{

if (Z[i]<m2){

m2 = Z[i];

}

}

//7. Обчислення m: = min(m,m2)

EnterCriticalSection(&CrSec);

m = min(m, m2);

LeaveCriticalSection(&CrSec);

//8. Сигнал Т1 про про завершення обчислень m

ReleaseSemaphore(Sem\_m[0], 1, NULL);

//9. Чекати сигналу від T1 про завершення обчислень m

WaitForSingleObject(Sem\_min, INFINITE);

//10. Копіювання m2: = m

EnterCriticalSection(&CrSec);

m2 = m;

LeaveCriticalSection(&CrSec);

//11. Обчислення MAH = d2\*MOH\*MK2 + m2\*MRH

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

{

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

{

s = 0;

for (int k = 0; k < N; k++)

{

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

}

MA[i][j] = s\*d2 + m2\*MO[i][j];

}

}

//12. Сигнал T1 про завершення обчислень MA

ReleaseSemaphore(Sem\_MA[0], 1, NULL);

cout << "Thread T2 finished" << endl;

}

//----------------------------------------T3----------------------------------------------

void T3() {

int d3, s, m3;

matrix MK3 = new vector[N];

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

{

MK3[i] = new int[N];

MA[i] = new int[N];

};

cout << "Thread T3 started" << endl;

//1. Введення MO,MR

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

{

MO[i] = new int[N];

MR[i] = new int[N];

};

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

{

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

{

MO[i][j] = 1;

MR[i][j] = 1;

}

}

//2. Сигнал задачам Т1, Т3, Т4 про введення МO, MR

SetEvent(Evn3\_124);

//3. Чекати на введення d у задачі T1

WaitForSingleObject(Evn3\_124, INFINITE);

//4. Чекати на введення MR, Z у задачі T4

WaitForSingleObject(Evn4\_123, INFINITE);

//5. Копіювати MK3:= MK

WaitForSingleObject(Mute, INFINITE);

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

{

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

{

MK3[i][j] = MK[i][j];

}

}

ReleaseMutex(Mute);

//6. Копіювання d

EnterCriticalSection(&CrSec);

d3 = d;

LeaveCriticalSection(&CrSec);

//7. Обчислення m3 := min(ZH)

m3 = 0;

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

{

if (Z[i]<m3){

m3 = Z[i];

}

}

//8. Обчислення m: = min(m,m3)

EnterCriticalSection(&CrSec);

m = min(m, m3);

LeaveCriticalSection(&CrSec);

//9. Сигнал Т1 про про завершення обчислень m

ReleaseSemaphore(Sem\_m[1], 1, NULL);

//10. Чекати сигналу від T1 про завершення обчислень m

WaitForSingleObject(Sem\_min, INFINITE);

//11. Копіювання m3: = m

EnterCriticalSection(&CrSec);

m3 = m;

LeaveCriticalSection(&CrSec);

//12. Обчислення MAH = d3\*MOH\*MK3 + m3\*MRH

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

{

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

{

s = 0;

for (int k = 0; k < N; k++)

{

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

}

MA[i][j] = s\*d3 + m3\*MO[i][j];

}

}

//13. Сигнал T1 про завершення обчислень MA

ReleaseSemaphore(Sem\_MA[1], 1, NULL);

cout << "Thread T3 finished" << endl;

}

//---------------------------------------T4-------------------------------------------------

void T4(){

int d4, s, m4;

matrix MK4 = new vector[N];

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

{

MK4[i] = new int[N];

};

cout << "Thread T4 started" << endl;

//1. Введення MK, Z

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

{

Z[i] = 1;

MK[i] = new int[N];

}

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

{

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

{

MK[i][j] = 1;

}

}

//2. Сигнал задачам T1, T2, T3 про введення MK, Z

SetEvent(Evn4\_123);

//3. Чекати на введення d у задачі T1

WaitForSingleObject(Evn1\_234, INFINITE);

//4. Чекати на введення MO, MR у задачі T3

WaitForSingleObject(Evn3\_124, INFINITE);

//5. Копіювати MK4:= MK

WaitForSingleObject(Mute, INFINITE);

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

{

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

{

MK4[i][j] = MK[i][j];

}

}

ReleaseMutex(Mute);

//6. Копіювання d

EnterCriticalSection(&CrSec);

d4 = d;

LeaveCriticalSection(&CrSec);

//7. Обчислення m4 := min(ZH)

m4 = 0;

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

{

if (Z[i]<m4){

m4 = Z[i];

}

}

//8. Обчислення m: = min(m,m4)

EnterCriticalSection(&CrSec);

m = min(m, m4);

LeaveCriticalSection(&CrSec);

//9. Сигнал Т1 про про завершення обчислень m

ReleaseSemaphore(Sem\_m[2], 1, NULL);

//10. Чекати сигналу від T1 про завершення обчислень m

WaitForSingleObject(Sem\_min, INFINITE);

//11. Копіювання m4: = m

EnterCriticalSection(&CrSec);

m4 = m;

LeaveCriticalSection(&CrSec);

//12. Обчислення MAH = d4\*MOH\*MK4 + m4\*MRH

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

{

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

{

s = 0;

for (int k = 0; k < N; k++)

{

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

}

MA[i][j] = s\*d4 + m4\*MO[i][j];

}

}

//13. Сигнал T1 про завершення обчислень MA

ReleaseSemaphore(Sem\_MA[2], 1, NULL);

cout << "Thread T4 finished" << endl;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

cout << "Lab 2 started" << endl;

Evn4\_123 = CreateEvent(NULL, TRUE, FALSE, NULL);

Evn3\_124 = CreateEvent(NULL, TRUE, FALSE, NULL);

Evn1\_234 = CreateEvent(NULL, TRUE, FALSE, NULL);

InitializeCriticalSection(&CrSec);

Mute = CreateMutex(NULL, FALSE, NULL);

Sem\_m[0] = CreateSemaphore(NULL, 0, 1, NULL);

Sem\_m[1] = CreateSemaphore(NULL, 0, 1, NULL);

Sem\_m[2] = CreateSemaphore(NULL, 0, 1, NULL);

DWORD Tid1, Tid2, Tid3, Tid4;

HANDLE threads[] =

{

CreateThread(NULL, NULL, (LPTHREAD\_START\_ROUTINE)T1, NULL, NULL, &Tid1),

CreateThread(NULL, NULL, (LPTHREAD\_START\_ROUTINE)T2, NULL, NULL, &Tid2),

CreateThread(NULL, NULL, (LPTHREAD\_START\_ROUTINE)T3, NULL, NULL, &Tid3),

CreateThread(NULL, NULL, (LPTHREAD\_START\_ROUTINE)T4, NULL, NULL, &Tid4)

};

WaitForMultipleObjects(4, threads, true, INFINITE);

CloseHandle(threads[0]);

CloseHandle(threads[1]);

CloseHandle(threads[2]);

CloseHandle(threads[3]);

cout << "Lab2 ended" << endl;

char key;

cin >> key;

return 0;

}