Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Лабораторна робота №2

з дисципліни «Паралельне програмування -2. Програмування для паралельних систем»

Тема: **Win32.** Cемафори, мютекси, критичні секції, події.

Виконав

студент групи ІП-31

Кобилинський Дмитро

Київ 2016

**Тема:** Програмування для комп’ютерних систем зі спільною пам’яттю. Win32. Семафори, події, критичні секції, мютекси.

**Технічне завдання:** Розробити програму для розв’язання ПКС із ЛП (структура на рис. 1) математичної задачі: A =B(MO\*MKH) + α\*max(Z)\*E

**Засоби організації взаємодії процесів:** Семафори, події, критичні секції, мютекси.

A = B(MO\*MKH) + α\*max(Z)\*E

**ОП**

3

4

2

1

A,B, α MO,Z MK,E

**Виконання роботи:**

**Етап 1. Побудова паралельного алгоритму**

1. mi = max(ZH)
2. m = max (m, mi)
3. AH = B\*(MO\*MKH) + α\*m\*E

Спільний ресурс: m, B, MO,E, α

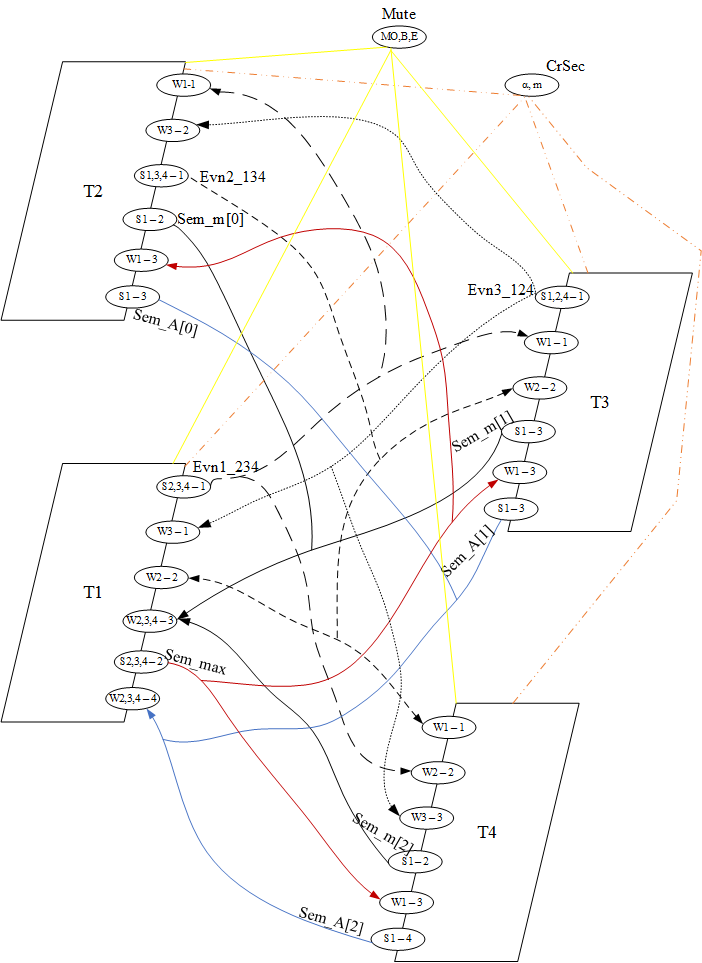
**Етап 2. Розроблення алгоритмів роботи кожного процесу**

|  |  |
| --- | --- |
| **Задача Т1** | ТС, КД |
| 1. Введення B, α |  |
| 2. Сигнал задачам Т2, Т3, Т4 про введення B , α | S2,3,4 – 1 |
| 3. Чекати на введення MO, Z у задачі Т2 | W2 – 1 |
| 4. Чекати на введення MK, E у задачі T3 | W3 – 2 |
| 5. Копіювати B1 = B, α1= α,E1=E,MO1=MO | КД |
| 6. Обчислення m1 = max(ZH) |  |
| 7. Обчислення m = max (m, m1) | КД |
| 8. Сигнал T2, T3, T4 про завершення обчислень m | S2,3,4 – 2 |
| 9. Чекати на завершення обчислень m в T2, T3, T4  10. Копіювання m1 = m | W2,3,4 – 3  КД |
|  |  |
| 11. Обчислення AH = B1(MO\*MKH) + α1\*m\*E1 |  |
| 12. Чекати на завершення обчислень MA в T2, T3, T4 | W2,3,4 – 4 |
| 13. Виведення A |  |
| **Задача T2** | ТС, КД |
| 1. Введення MO, Z |  |
| 2. Сигнал задачам Т1, Т3, Т4 про введення MO, Z | S1,3,4 – 1 |
| 3. Чекати на введення B, α у задачі T1 | W1 – 1 |
| 4. Чекати на введення MK,E у задачі T3 | W3 – 2 |
| 5. Копіювати B2 = B, α2= α,E2=E,MO2=MO | КД |
| 6. Обчислення m2 := max(ZH) |  |
| 7. Обчислення m: = max(m,m2) | КД |
| 8. Сигнал Т1 про про завершення обчислень m | S1 – 2 |
| 9. Чекати сигналу від T1 про завершення обчислень m | W1 –3 |
| 10. Копіювання m2: = m | КД |
| 11. Обчислення AH = B2(MO\*MKH) + α2\*m\*E2 |  |
| 12. Сигнал T1 про завершення обчислень MA | S1 – 3 |
| **Задача Т3** | ТС, КД |
| 1. Введення МК , E |  |
| 2. Сигнал задачам Т1, Т2, Т4 про введення МK, E | S1,2,4 – 1 |
| 3. Чекати на введення B, α у задачі T1 | W1 – 1 |
| 4. Чекати на введення MО,Z у задачі T2 | W3 – 2 |
| 5. Копіювати B3 = B, α3= α,E3=E,MO3=MO | КД |
| 6. Обчислення m3 := max(ZH) |  |
| 7. Обчислення m: = max(m,m3) | КД |
| 8. Сигнал Т1 про про завершення обчислень m | S1 – 2 |
| 9. Чекати сигналу від T1 про завершення обчислень m | W1 – 3 |
| 10. Копіювання m3: = m | КД |
| 11. Обчислення AH = B3(MO\*MKH) + α3\*m\*E3 |  |
| 12. Сигнал T1 про завершення обчислень MA | S1 – 3 |
| **Задача Т4** | ТС, КД |
| 1. Чекати на введення B, α у задачі T1 | W1 – 1 |
| 2. Чекати на введення MО,Z у задачі T2 | W2 – 2 |
| 3. Чекати на введення MK, E у задачі T3 | W3 – 3 |
| 4. Копіювати B4 = B, α4= α,E4=E,MO4=MO | КД |
| 5. Обчислення m4 := max(ZH) |  |
| 6. Обчислення m: = max(m,m4) | КД |
| 7. Сигнал Т1 про про завершення обчислень m | S1 – 1 |
| 8. Чекати сигналу від T1 про завершення обчислень m | W1 – 4 |
| 9. Копіювання m4: = m | КД |
| 10. Обчислення AH = B4(MO\*MKH) + α4\*m\*E4 |  |
| 11. Сигнал T1 про завершення обчислень MA | S1 – 2 |

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

На структурній схемі взаємодії задач уведено такі семафори

* Event1\_234 – подія для синхронізації із завершенням вводу в T1
* Event3\_124– подія для синхронізації із завершенням вводу в T3
* Event2\_134 – подія для синхронізації із завершенням вводу в T2
* CrSec – для доступу до спільного ресурсу b, C, MK
* Mute – для доступу до спільного ресурсу *a*
* Sem\_m[3] – семафор для синхронізації обчислень max в T2,T3,T4
* Sem\_max – для сигналізування кінця визначення max в Т1
* Sem\_A[3] – для синхронізації обчислень AH і виведення результату



**Етап 4. Лістинг коду**

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

--

-- Parallel programming --

-- Laboratory work #2. Win32 --

-- Task: A = B\*(MO\*MK)+alfa\*max(Z)\*E --

-- --

-- Author: Kobylynskiy Dmytro, group IP-31 --

-- Date: 24.03.2016 --

-- --

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

#include "stdafx.h"

#include <iostream>

#include <windows.h>

using namespace std;

typedef int\* vector;

typedef int\*\* matrix;

const int N = 4;

const int P = 4;

const int H = N / P;

int alfa, m;

vector Z = new int[N];

vector B = new int[N];

vector E = new int[N];

vector A = new int[N];

matrix MO = new vector[N],

MK = new vector[N];

HANDLE Evn1\_234, Evn2\_134, Evn3\_124, Evn4\_123,

Mute, Sem\_m[3], Sem\_A[3], Sem\_max;

CRITICAL\_SECTION CrSec;

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

void T1(){

int alfa1, m1, s;

vector E1 = new int[N];

matrix MO1 = new vector[N];

vector B1 = new int[N];

vector r = new int[N];

cout << "Process T1 started" << endl;

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

{

MO1[i] = new int[N];

};

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

{

B[i] = 1 ;r[i] = 0;

}

alfa = 1;

SetEvent(Evn1\_234);

WaitForSingleObject(Evn3\_124, INFINITE);

WaitForSingleObject(Evn2\_134, INFINITE);

WaitForSingleObject(Mute, INFINITE);

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

{

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

{

MO1[i][j] = MO[i][j];

}

}

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

{

B1[i] = B[i];

E1[i] = E[i];

}

ReleaseMutex(Mute);

EnterCriticalSection(&CrSec);

alfa1 = alfa;

LeaveCriticalSection(&CrSec);

m1 = 0;

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

{

if (Z[i]>m1){

m1 = Z[i];

}

}

EnterCriticalSection(&CrSec);

m = max(m, m1);

LeaveCriticalSection(&CrSec);

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

EnterCriticalSection(&CrSec);

m1 = m;

LeaveCriticalSection(&CrSec);

ReleaseSemaphore(Sem\_max, 1, NULL);

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

{

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

{

s = 0;

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

{

s += MK[i][k] \* MO1[k][j];

}

r[i] = r[i] + B1[i] \* s;

}

}

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

{

A[i] = r[i] + alfa1\*m1\*E1[i];

}

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

if (N<10)

{

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

{

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

}

cout << endl;

}

cout << "Process T1 finished" << endl;

}

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

void T2(){

int alfa1, m1, s;

vector E1 = new int[N];

vector r = new int[N];

matrix MO1 = new vector[N];

vector B1 = new int[N];

cout << "Process T2 started" << endl;

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

{

MO[i] = new int[N];

};

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

{

MO1[i] = new int[N];

};

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

{

Z[i] = 1;

r[i] = 0;

}

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

{

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

{

MO[i][j] = 1;

}

}

alfa = 1;

SetEvent(Evn2\_134);

WaitForSingleObject(Evn1\_234, INFINITE);

WaitForSingleObject(Evn3\_124, INFINITE);

WaitForSingleObject(Evn4\_123, INFINITE);

WaitForSingleObject(Mute, INFINITE);

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

{

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

{

MO1[i][j] = MO[i][j];

}

}

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

{

B1[i] = B[i];

E1[i] = E[i];

}

ReleaseMutex(Mute);

EnterCriticalSection(&CrSec);

alfa1 = alfa;

LeaveCriticalSection(&CrSec);

m1 = 0;

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

{

if (Z[i]>m1){

m1 = Z[i];

}

}

EnterCriticalSection(&CrSec);

m = max(m, m1);

LeaveCriticalSection(&CrSec);

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

WaitForSingleObject(Sem\_max, INFINITE);

EnterCriticalSection(&CrSec);

m1 = m;

LeaveCriticalSection(&CrSec);

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 += MK[i][k] \* MO1[k][j];

}

r[i] = r[i] + B1[i] \* s;

}

}

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

{

A[i] = r[i] + alfa1\*m1\*E1[i];

}

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

cout << "Process T2 finished" << endl;

}

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

void T3(){

int alfa1, m1, s;

vector E1 = new int[N];

matrix MO3 = new vector[N];

vector B1 = new int[N];

vector r = new int[N];

cout << "Process T3 started" << endl;

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

{

MK[i] = new int[N];

};

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

{

MO3[i] = new int[N];

};

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

{

E[i] = 1;

r[i] = 0;

}

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

{

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

{

MK[i][j] = 1;

}

}

SetEvent(Evn3\_124);

WaitForSingleObject(Evn1\_234, INFINITE);

WaitForSingleObject(Evn2\_134, INFINITE);

WaitForSingleObject(Mute, INFINITE);

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

{

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

{

MO3[i][j] = MO[i][j];

}

}

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

{

B1[i] = B[i];

E1[i] = B[i];

}

ReleaseMutex(Mute);

EnterCriticalSection(&CrSec);

alfa1 = alfa;

LeaveCriticalSection(&CrSec);

m1 = 0;

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

{

if (Z[i]>m1){

m1 = Z[i];

}

}

EnterCriticalSection(&CrSec);

m = max(m, m1);

LeaveCriticalSection(&CrSec);

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

WaitForSingleObject(Sem\_max, INFINITE);

EnterCriticalSection(&CrSec);

m1 = m;

LeaveCriticalSection(&CrSec);

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

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 += MK[i][k] \* MO3[k][j];

}

r[i] = r[i] + B1[i] \* s;

}

}

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

{

A[i] = r[i] + alfa1\*m1\*E1[i];

}

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

cout << "Process T3 finished" << endl;

}

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

void T4(){

int alfa1, m1, s;

vector E1 = new int[N];

matrix MO4 = new vector[N];

vector B1 = new int[N];

vector r = new int[N];

cout << "Process T4 started" << endl;

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

{

MO4[i] = new int[N];

r[i] = 0;

};

WaitForSingleObject(Evn2\_134, INFINITE);

WaitForSingleObject(Evn3\_124, INFINITE);

WaitForSingleObject(Evn1\_234, INFINITE);

WaitForSingleObject(Mute, INFINITE);

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

{

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

{

MO4[i][j] = MO[i][j];

}

}

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

{

B1[i] = B[i];

E1[i] = B[i];

}

ReleaseMutex(Mute);

EnterCriticalSection(&CrSec);

alfa1 = alfa;

LeaveCriticalSection(&CrSec);

m1 = 0;

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

{

if (Z[i]>m1){

m1 = Z[i];

}

}

EnterCriticalSection(&CrSec);

m = max(m, m1);

LeaveCriticalSection(&CrSec);

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

WaitForSingleObject(Sem\_max, INFINITE);

EnterCriticalSection(&CrSec);

m1 = m;

LeaveCriticalSection(&CrSec);

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 += MK[i][k] \* MO4[k][j];

}

r[i] = r[i] + B1[i] \* s;

}

}

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

{

A[i] = r[i] + alfa1\*m1\*E1[i];

}

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

cout << "Process T4 finished" << endl;

}

int main(int argc, char\* argv[])

{

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

Evn2\_134 = 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;

}