МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний авіаційний університет

Лабораторна робота № 2

з дисципліни ««Високопродуктивні обчислювальні системи»

Тема: «Засоби синхронізації процесів: семафори»

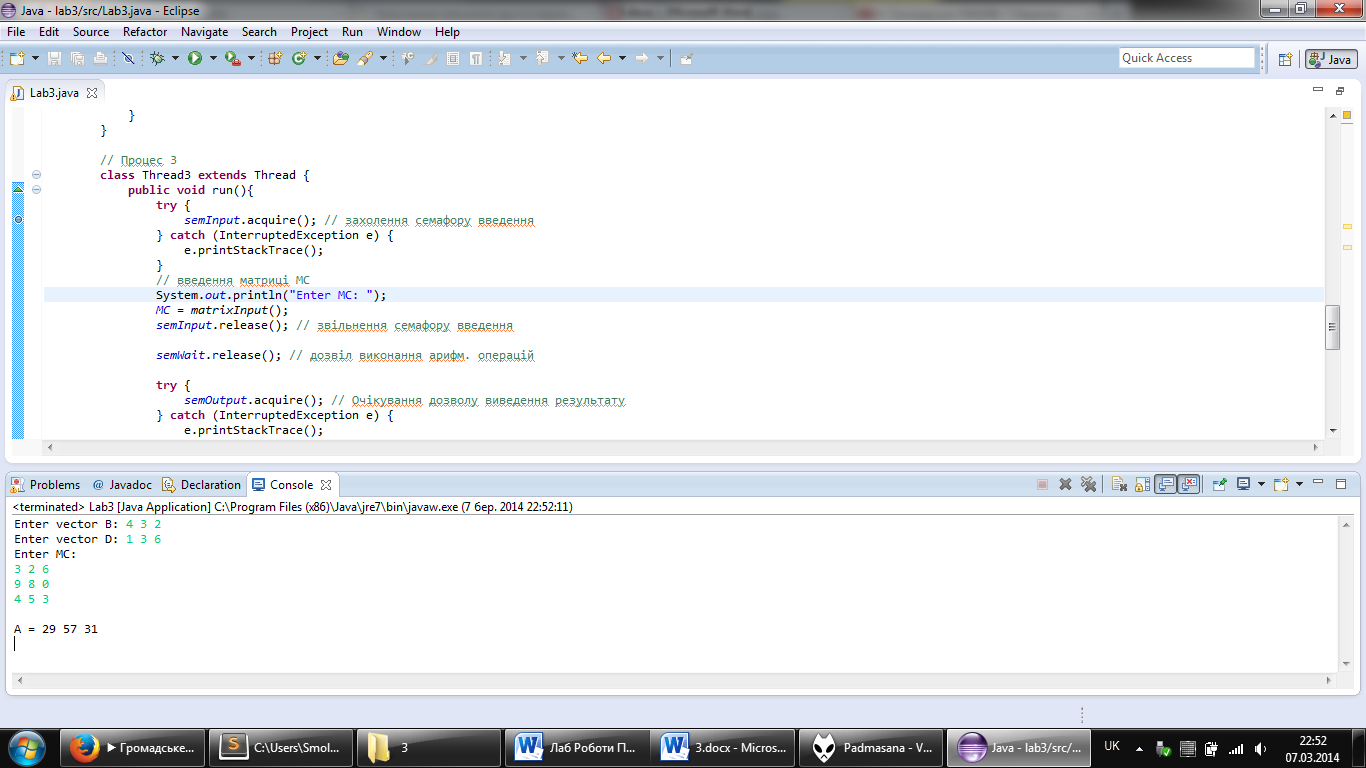
Київ – 2015

**Завдання:**

Розробити програму для ВПОС, яка включає4 процесори і два пристрої введення-виведення. Математична задача ―обчислення виразу A = B\* MC – D. Введення векторів B і D виконується в другому процесорі, матриці МС – в третьому процесорі, виведення результату А – в третьому процесорі.

|  |  |  |  |
| --- | --- | --- | --- |
| **№ варіанту** | Операція | **Номера процесору ВПОС (4 процесори)** | |
| Для введення | Для виведення |
|  | A = B\* MC – D | B,D – 2  MC - 3 | 3 |

**Скріншот роботи програми:**



**Лістинг коду:**

**import** java.util.Scanner;

**import** java.util.concurrent.Semaphore;

**public class** Lab3 {

**public static** Semaphore*semWait* = **new**Semaphore(-1); //семафор для очікування введення всіх змінних

**public static** Semaphore*semInput* = **new**Semaphore(1); // семафор для вводу змінних

**public static** Semaphore*semOutput* = **new**Semaphore(0); // семафор для очікування виведення результату

**final static int** *N* = 3; // константа розмірності для матриць та векторів

// Спільні змінні АКА загальна пам'ять

**public static int** [] *A*, *B*, *D*;

**public static int** [][] *MC*;

// Метод для множення матриці на вектор

**public static int**[] matrixVectorMul(**int** MA[][], **int** A[]){

**int**[] x = **new int**[*N*];

**int** tmp;

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

tmp = 0;

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

tmp = tmp + MA[i][j] \* A[j];

}

x[i] = tmp;

}

**return** x;

}

// Метод для віднімання двох векторів

**public static int**[] vectorSub(**int** a[], **int** b[]) {

**int**[] x = **new int**[*N*];

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

x[i] = a[i] - b[i];

}

**return** x;

}

// Метод для виведення вектора

**public static void** vectorOutput(**int** a[]){

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

System.*out*.print(a[i] + " ");

}

System.*out*.println();

}

// Метод для вводу вектора розміром N

**public static int**[] vectorInput(){

Scannersc = **new** Scanner(System.*in*);

**int**[] A = **new int**[*N*];

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

A[i] = sc.nextInt();

}

**return** A;

}

// Метод для вводу матриці розміром NxN

**public static int**[][] matrixInput(){

Scannersc = **new** Scanner(System.*in*);

**int**[][] MA = **new int**[*N*][*N*];

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

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

MA[i][j] = sc.nextInt();

}

}

**return** MA;

}

**public static void** main(String[] args) {

// Процес 2

**class** Thread2 **extends**Thread {

**public void** run(){

**try** {

*semInput*.acquire(); // захолення семафору введення

} **catch** (InterruptedException e) {

e.printStackTrace();

}

// Введення векторів B і D

System.*out*.print("Entervector B: ");

*B* = *vectorInput*();

System.*out*.print("Entervector D: ");

*D* = *vectorInput*();

*semInput*.release(); // звільнення семафору введення

*semWait*.release(); // дозвіл виконання арифм. операцій

}

}

// Процес 3

**class** Thread3 **extends**Thread {

**publi cvoid** run(){

**try** {

*semInput*.acquire(); // захолення семафору введення

} **catch** (InterruptedException e) {

e.printStackTrace();

}

// введення матриці МС

System.*out*.println("Enter MC: ");

*MC* = *matrixInput*();

*semInput*.release(); // звільнення семафору введення

*semWait*.release(); // дозвіл виконання арифм. операцій

**try** {

*semOutput*.acquire(); // Очікування дозволу виведення результату

} **catch** (InterruptedException e) {

e.printStackTrace();

}

// виведення результату

System.*out*.print("\nA = ");

*vectorOutput*(*A*);

}

}

// Процес 4

**class** Thread4 **extends**Thread {

**public void** run(){

**try** {

*semWait*.acquire(); // Очікування двох дозволів для виконання арифм. операцій

*A* = *vectorSub*(*matrixVectorMul*(*MC*, *B*), *D*); // Обчислення арифм. виразу

*semOutput*.release(); // Дозвіл виводу результату

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

// Створення об'єктів тредів

Thread2 T2 = **new** Thread2();

Thread3 T3 = **new** Thread3();

Thread4 T4 = **new** Thread4();

// Запуск тредів

T2.start();

T3.start();

T4.start();

}

}

Отже, одним зі способів синхронізації процесів в мові Java є використання семафорів, які доступні в пакеті java.util.concurrent.Semaphore. Семафор ― це лічильник, початковий стан якого задається цілим числом, яке може бути як від’ємним так і додатнім. Об’єкт семафору має 2 основні методи:

1. s.acquire() ― перевіряє значення лічильника семафора і блокує подальше виконання процесу, якщо це значення <= 0.Якщо значення лічильника семафору позитивне (> 0), то метод зменшує його значення та дозволяє процесу продовжити виконання.
2. s.release() ― збільшуєзначення лічильника семафора.

В даній роботі було використано 3 прийоми роботи з семафорами:

1. Використання семафору для контролю доступу до критичної ділянки коду, яка використовує загальний ресурс. Одночасна робота з загальним ресурсом дозволяється лише одному процесу. Приклад (початкове значення semInput=1):

*semInput*.acquire(); // захолення семафору введення

// введення матриці МС

System.*out*.println("Enter MC: ");

*MC* = *matrixInput*();

*semInput*.release(); // звільнення семафору введення

1. Використання семафору для створення бар’єру, подолання якого одним або більше процесами дає сигнал для початку роботи процесу, що очікує. Такий підхід можна використовувати коли, наприклад, процес А залежить від даних, які надаються процесами Б і В. Процес А блокується доти, доки процеси Б і В не подолають бар’єр, тим самим сигналізуючи процесу А,що необхідні дані для продовження його виконаннястали доступними. Приклад (початкове значення semWait=-1):

Процес А:

*semWait*.acquire(); // Очікування двох дозволів для виконання арифм. операцій

*A* = *vectorSub*(*matrixVectorMul*(*MC*, *B*), *D*); // Обчислення арифм. виразу

Процес Б:

*MC* = *matrixInput*(); // введення матриці МС

*semWait*.release(); // 1й дозвіл виконання арифм. операцій

Процес В:

// Введення векторів B і D

*B* = *vectorInput*();

*D* = *vectorInput*();

*semWait*.release(); // 2й дозвіл виконання арифм. операцій

1. Відправка сигналу від процесу А процесу Б. Процес Б блокується доти, доки не отримає сигнал про дозвіл продовження виконання. Приклад (початкове значення semOutput=0):

Процес А:

*A* = *vectorSub*(*matrixVectorMul*(*MC*, *B*), *D*); // Обчислення арифм. виразу

*semOutput*.release(); // Сигнал виводу результату

Процес Б:

*semOutput*.acquire(); // Очікування сигналу виведення результату

*vectorOutput*(*A*);// виведення результату