**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

по дисциплине **«Операционные системы и среды**»

на тему: «**Синхронизация процессов**»

Выполнил: ст. гр. ИП-31

Пикун Я. И.

Принял: ст. преподаватель

Самовендюк Н.В.

Гомель 2023

**Цель:** изучить типовые механизмы синхронизации процессов.

**Задание.**

1. Разработать многопоточное приложение с использованием минимум двух потоков и различных средств синхронизации.

|  |  |  |
| --- | --- | --- |
| 22. | Два потока увеличивают значение общей переменной | Мьютексы |

2. Время входа в критическую секцию для каждого потока генерировать случайным образом.

3. В процессе работы приложение в консоль должна выводится информация о состоянии потока (работа в некритической секции, работа в критической секции, время входа и выхода из критической секции).

4. Убедиться в результативности применения средств синхронизации потоков, сравнив результаты работы программ с использованием и без использования средств синхронизации.

**Листинг Program.cs:**

using System.Diagnostics;

namespace lab7

{

public class Program

{

public static int number = 0;

public static void Main(string[] args)

{

Thread thread1 = new Thread(new ParameterizedThreadStart(WithoutSync.WithoutMutex));

Thread thread2 = new Thread(new ParameterizedThreadStart(WithoutSync.WithoutMutex));

Stopwatch stopWatch = new Stopwatch();

stopWatch.Start();

thread1.Start(1);

thread2.Start(2);

thread1.Join();

thread2.Join();

stopWatch.Stop();

Console.WriteLine();

Console.WriteLine($"Время работы без мьютекса: {stopWatch.ElapsedMilliseconds} мс.");

Console.WriteLine();

thread1 = new Thread(new ParameterizedThreadStart(WithSync.WithMutex));

thread2 = new Thread(new ParameterizedThreadStart(WithSync.WithMutex));

stopWatch = new Stopwatch();

stopWatch.Start();

thread1.Start(1);

thread2.Start(2);

thread1.Join();

thread2.Join();

stopWatch.Stop();

Console.WriteLine();

Console.WriteLine($"Время работы с мьютексом: {stopWatch.ElapsedMilliseconds} мс.");

Console.WriteLine();

}

}

}

**Листинг WithoutSync.cs:**

namespace lab7

{

internal class WithoutSync

{

public static void WithoutMutex(object obj)

{

var random = new Random();

Thread.Sleep(random.Next(50, 1000));

int i = (int)obj;

Console.WriteLine($"Поток {i} вошёл в крит. секцию.");

Program.number++;

Console.WriteLine($"Значение общей переменной: {Program.number}");

Console.WriteLine($"Поток {i} вышел из крит. секции.");

}

}

}

**Листинг WithSync.cs:**

namespace lab7

{

internal class WithSync

{

public static void WithMutex(object obj)

{

var random = new Random();

Thread.Sleep(random.Next(50, 1000));

int i = (int)obj;

var mutex = new Mutex();

mutex.WaitOne();

Console.WriteLine($"Поток {i} вошёл в крит. секцию.");

Program.number++;

Console.WriteLine($"Значение общей переменной: {Program.number}");

mutex.ReleaseMutex();

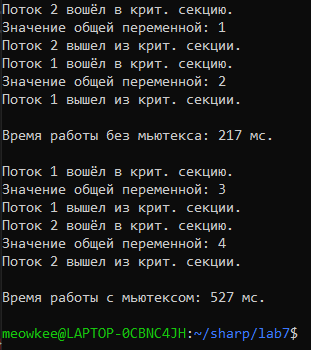
Console.WriteLine($"Поток {i} вышел из крит. секции.");

}

}

}

**Результат выполнения:**



**Вывод:** Я, выполнив задание, изучил типовые механизмы синхронизации процессов.