**Министерство Образования Республики Беларусь**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

Кафедра «Информатика»

**Лабораторная работа № 7**

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

на тему:«**Синхронизация процессов**»

Выполнил: ст. гр. ИП-32

Кардаш Е.В.

Принял: ст. преподаватель

Самовендюк Н.В.

Гомель 2021

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

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

|  |  |  |
| --- | --- | --- |
| № варианта | Разделяемый ресурс | Механизм синхронизации |
| 7 | Два потока с различной частотой считывают и записывают данные в общий буфер памяти | мьютексы |

**Листинг программы:**

**Файл Program.cs:**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Threading;

namespace Lab7

{

public class Program

{

private static Mutex mutex = new Mutex();

public static List<string> listTextLines = new List<string>

{

"Hello",

};

static void Main(string[] args)

{

int k = 0;

while (k < 2)

{

Thread thread1 = new Thread(new ParameterizedThreadStart(WithoutSynchronization.ReadWithoutMutex));

Thread thread2 = new Thread(new ParameterizedThreadStart(WithoutSynchronization.ReadWithoutMutex));

Stopwatch stopWatch1 = new Stopwatch();

stopWatch1.Start();

thread1.Start(1);

thread2.Start(2);

thread1.Join();

thread2.Join();

stopWatch1.Stop();

Console.WriteLine();

Console.WriteLine($"Время работы без мьютекса:{stopWatch1.ElapsedMilliseconds} мс.");

Console.WriteLine();

k++;

}

Thread thread3 = new Thread(new ParameterizedThreadStart(WithSynchronization.ReadWithMutex));

Thread thread4 = new Thread(new ParameterizedThreadStart(WithSynchronization.ReadWithMutex));

Stopwatch stopWatch2 = new Stopwatch();

stopWatch2.Start();

thread3.Start(1);

thread4.Start(2);

thread3.Join();

thread4.Join();

stopWatch2.Stop();

Console.WriteLine();

Console.WriteLine($"Время работы без мьютекса:{stopWatch2.ElapsedMilliseconds} мс.");

Console.WriteLine();

}

}

}

**Файл WithoutSynchronization.cs:**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Threading;

namespace Lab7

{

class WithoutSynchronization

{

public static void ReadWithoutMutex(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.listTextLines.AddRange(Program.listTextLines);

Console.WriteLine($"Содержимое буфера памяти:\n{string.Join('\n', Program.listTextLines)}\n");

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

}

}

}

**Файл WithSynchronization.cs:**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Threading;

namespace Lab7

{

class WithSynchronization

{

public static void ReadWithMutex(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 мьютекс.");

Program.listTextLines.AddRange(Program.listTextLines);

Console.WriteLine($"Содержимое буфера памяти:\n{string.Join('\n', Program.listTextLines)}\n");

mutex.ReleaseMutex();

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

}

}

}

**Результат выполнения:**

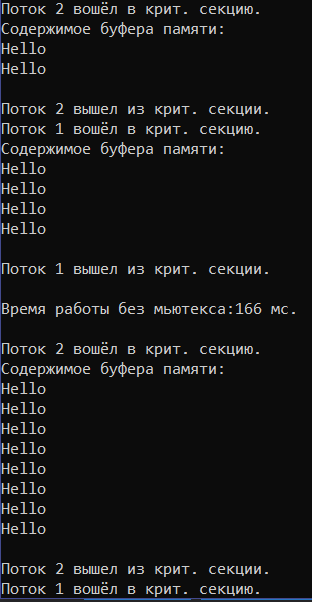


Рисунок 1 − Результат выполнения программы

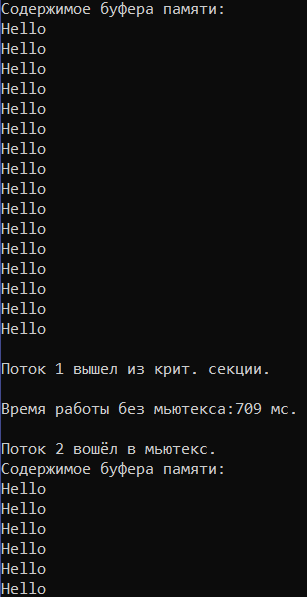


Рисунок 2 − Результат выполнения программы

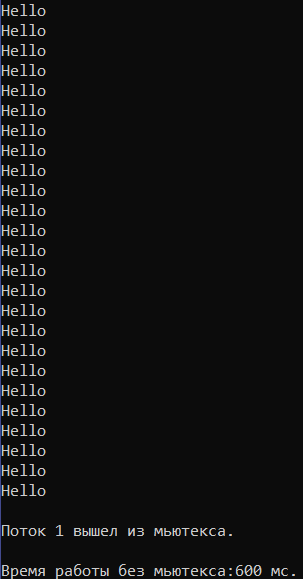


Рисунок 3 − Результат выполнения программы

**Вывод:** в ходе данной лабораторной работы были изучены типовые механизмы синхронизации процессов.