# Министерство Образования Республики Беларусь

Учреждение образования

«Гомельский государственный технический университет

имени П.О.Сухого»

Кафедра: «Информатика»

по курсу: «Операционные системы и среды»

Лабораторная работа №7

«Синхронизация процессов»

Выполнил: студент группы ИП-31

Земчёнок Вадим Геннадьевич

Допуск к защите: Проверил: старший преподаватель

Дата защиты: Самовендюк Николай Владимирович

Гомель 2020 г.

**Цель:** изучить типовые механизмы синхронизации процессов

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

Таблица 1 Варианты заданий

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

using System;

using System.Text;

using System.Threading;

namespace ConsoleApp1

{

class Program

{

private static StringBuilder buffer { get; set; }

private static StringBuilder bufferMutex { get; set; }

private static readonly string[] mats = new string[] { "123", "456", "789", "abc", "def", "ghi"};

private static readonly int idleInSecondsX10 = 15;

static Mutex mutex = new Mutex();

static int x = 0;

static void Main(string[] args)

{

Random rnd = new Random();

buffer = new StringBuilder("");

Console.WriteLine("Without mutex:");

Thread first = new Thread(WriteDigitsToBuffer) { Name = "1st"};

Thread second = new Thread(WriteLettersToBuffer) { Name = "2nd"};

first.Start();

second.Start();

Thread.Sleep(idleInSecondsX10 \* 400);

Console.WriteLine("RESULT AFTER {0} seconds: {1}\n", idleInSecondsX10 \* 0.4, buffer);

bufferMutex = new StringBuilder("");

Console.WriteLine("With mutex:");

Thread third = new Thread(WriteDigitsToBufferMutex) { Name = "3rd" };

Thread fourth = new Thread(WriteLettersToBufferMutex) { Name = "4th" };

third.Start();

fourth.Start();

Thread.Sleep(idleInSecondsX10\*400);

Console.WriteLine("RESULT AFTER {0} seconds: {1}\n", idleInSecondsX10 \* 0.4, bufferMutex);

Console.ReadLine();

}

public static void WriteDigitsToBuffer()

{

Random rnd = new Random();

int timeToSleep = rnd.Next(1, idleInSecondsX10) \* 100;

Console.WriteLine(Thread.CurrentThread.Name + " thread started its work");

for(int a = 0; a < 3; a++)

{

buffer.Append(mats[a]);

Console.WriteLine(string.Format("{0} thread wrote {1} and now sleeps for {2} s", Thread.CurrentThread.Name, mats[a], timeToSleep));

Thread.Sleep(timeToSleep);

}

Console.WriteLine(Thread.CurrentThread.Name + " thread finished its work");

}

public static void WriteDigitsToBufferMutex()

{

Random rnd = new Random();

int timeToSleep = rnd.Next(1,idleInSecondsX10) \* 100;

mutex.WaitOne();

Console.WriteLine(Thread.CurrentThread.Name + " thread started its work with locking mutex");

for (int a = 0; a < 3; a++)

{

bufferMutex.Append(mats[a]);

Console.WriteLine(string.Format("{0} thread wrote {1} and now sleeps for {2} s", Thread.CurrentThread.Name, mats[a], timeToSleep));

Thread.Sleep(timeToSleep);

}

mutex.ReleaseMutex();

Console.WriteLine(Thread.CurrentThread.Name + " thread finished its work with releasing mutex");

}

public static void WriteLettersToBuffer()

{

Random rnd = new Random();

int timeToSleep = rnd.Next(1, idleInSecondsX10) \* 100;

Console.WriteLine(Thread.CurrentThread.Name + " thread started its work");

for (int a = 3; a < 6; a++)

{

buffer.Append(mats[a]);

Console.WriteLine(string.Format("{0} thread wrote {1} and now sleeps for {2} s", Thread.CurrentThread.Name, mats[a], timeToSleep));

Thread.Sleep(timeToSleep);

}

Console.WriteLine(Thread.CurrentThread.Name + " thread finished its work");

}

public static void WriteLettersToBufferMutex()

{

Random rnd = new Random();

int timeToSleep = rnd.Next(idleInSecondsX10) \* 100;

mutex.WaitOne();

Console.WriteLine(Thread.CurrentThread.Name + " thread started its work with locking mutex");

for (int a = 3; a < 6; a++)

{

bufferMutex.Append(mats[a]);

Console.WriteLine(string.Format("{0} thread wrote {1} and now sleeps for {2} s", Thread.CurrentThread.Name, mats[a], timeToSleep));

Thread.Sleep(timeToSleep);

}

Console.WriteLine(Thread.CurrentThread.Name + " thread finished its work with releasing mutex");

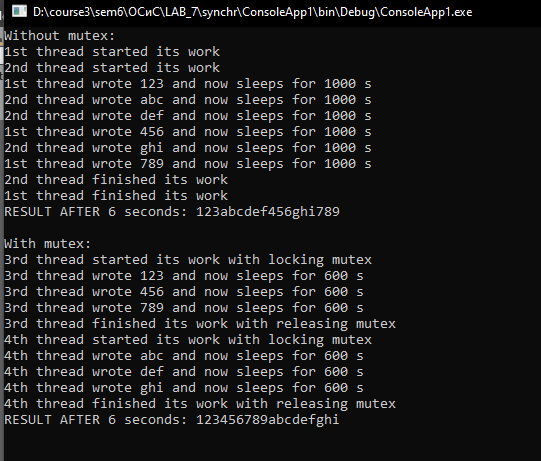
mutex.ReleaseMutex();

}

}

}

**Результаты выполнения**

****

**Вывод:** я успешно изучил типовые механизмы синхронизации процессов