

using System;

using System.Threading;

// Lock - не принимает типов значений, а только ссылочные.

namespace monitor

{

class Program

{

static private int counter = 0;

// Нельзя использовать объекты блокировки структурного типа.

// block - не может быть структурным.

static private object block = new object();

static private void Function1()

{

lock (block)

{

Console.WriteLine("1 Enter ");

for (int i = 0; i < 50; ++i)

{

Monitor.Wait(block, Timeout.Infinite);

Console.Write("1 ");

Console.WriteLine("{0} из потока {1}", ++counter, Thread.CurrentThread.GetHashCode());

Monitor.Pulse(block);

}

}

}

static private void Function2()

{

lock (block)

{

Console.WriteLine("2 Enter ");

for (int i = 0; i < 50; ++i)

{

Monitor.Wait(block, Timeout.Infinite);

Console.Write("2 ");

Console.WriteLine("{0} из потока {1}", ++counter, Thread.CurrentThread.GetHashCode());

Monitor.Pulse(block);

}

}

}

static private void Function3()

{

lock (block)

{

Console.WriteLine("3 Enter ");

for (int i = 0; i < 50; ++i)

{

Monitor.Pulse(block);

Monitor.Wait(block, Timeout.Infinite);

Console.Write("3 ");

Console.WriteLine("{0} из потока {1}", ++counter, Thread.CurrentThread.GetHashCode());

}

}

}

static void Main()

{

var t1 = new Thread(Function1);

t1.Name = "Thread1";

var t2 = new Thread(Function2);

t2.Name = "Thread2";

var t3 = new Thread(Function3);

t3.Name = "Thread3";

Thread[] threads = { t1, t2, t3 };

foreach (Thread t in threads)

{

t.Start();

}

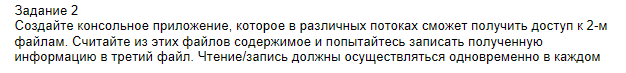
// Задержка.

Console.ReadKey();

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace ReadWreate

{

class FileOperations

{

MemoryStream ms = new MemoryStream();

static FileStream fs = new FileStream("3.txt", FileMode.Create, FileAccess.Write, FileShare.Write);

public String File { get; set; }

public void Read(object obj)

{

FileStream fs = new FileStream(File, FileMode.Open, FileAccess.Read);

ShowConsole(fs, obj);

fs.Position = 0;

fs.CopyTo(ms);

fs.Close();

}

public void Write(object obj)

{

StreamWriter sw = new StreamWriter(fs,Encoding.Default);

TextReader t = new StreamReader(ms, Encoding.Default);

lock (obj)

{

ms.Position = 0;

while (t.Peek()!=-1)

{

Monitor.Pulse(obj);

Monitor.Wait(obj,10,true);

//byte[] arr = new byte[200];

//ms.Read(arr, 0, arr.Length);

//sw.WriteLine(Encoding.Default.GetString(arr));

sw.WriteLine(t.ReadLine());

sw.Flush();

//Console.WriteLine(Encoding.Default.GetString(arr));

}

}

}

private void ShowConsole(Stream fs, object obj)

{

Random r=new Random();

StreamReader sr = new StreamReader(fs,Encoding.Default);

lock (obj)

{

Console.WriteLine(File);

while (!sr.EndOfStream)

{

Monitor.Pulse(obj);

Monitor.Wait(obj, 10,true);

Console.WriteLine(sr.ReadLine());

}

}

Console.WriteLine(File + " exit");

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace ReadWreate

{

class Program

{

static void Main(string[] args)

{

FileOperations f1=new FileOperations(){File = "1.txt"};

FileOperations f2 = new FileOperations() { File = "2.txt" };

object obj = new object();

Thread[] th = new[] { new Thread(f1.Read), new Thread(f2.Read)};

RunThread(th, obj);

WaitThread(th);

th = new[] { new Thread(f1.Write), new Thread(f2.Write) };

RunThread(th, obj);

Console.ReadKey();

}

private static void RunThread(Thread[] th, object obj)

{

foreach (Thread thread in th)

{

thread.Start(obj);

}

}

private static void WaitThread(Thread[] th)

{

foreach (Thread thread in th)

{

thread.Join();

}

}

}

}