using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Thtread\_P2

{

class MySemaphore

{

private Semaphore sem=null;

private static MemoryStream ms = null;

public MySemaphore(Semaphore sem)

{

this.sem = sem;

}

public void Read(object state)

{

ManualResetEventSlim me=state as ManualResetEventSlim;

Console.WriteLine("Read enter");

StreamReader sr=new StreamReader("1.txt",Encoding.Default);

ms = new MemoryStream(Encoding.Default.GetBytes(sr.ReadToEnd()));

Console.WriteLine("Read exit");

me.Set();

sem.Release(2);

}

public void ShowConsole(object state)

{

sem.WaitOne();

Console.WriteLine("Console enter");

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

while (tr.Peek()>0)

{

Console.WriteLine(tr.ReadLine());

}

Console.WriteLine("Console exit");

ms=new MemoryStream(ms.ToArray());

tr.Close();

sem.Release();

}

public void WriteFile(object state)

{

sem.WaitOne();

MemoryStream msTmp=new MemoryStream(ms.ToArray());

Console.WriteLine("Write enter");

FileStream fs=new FileStream("test.txt",FileMode.Create,FileAccess.Write);

TextReader tr = new StreamReader(msTmp, Encoding.Default);

StreamWriter tw = new StreamWriter(fs);

while (tr.Peek()>0)

{

tw.WriteLine(tr.ReadLine());

}

Console.WriteLine("Write exit");

tr.Close();

tw.Close();

sem.Release();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Thtread\_P2

{

class Program

{

static void Main(string[] args)

{

MySemaphore sem=new MySemaphore(new Semaphore(0,2));

ManualResetEventSlim ms=new ManualResetEventSlim(false);

ThreadPool.QueueUserWorkItem(new WaitCallback(sem.Read),ms);

ThreadPool.QueueUserWorkItem(new WaitCallback(sem.ShowConsole),ms);

ThreadPool.QueueUserWorkItem(new WaitCallback(sem.WriteFile));

Console.ReadKey();

}

}

}



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace EventThread

{

class Program

{

const int count = 5;

private volatile static bool f = true;

static AutoResetEvent mn = new AutoResetEvent(false);

static EventWaitHandle eh = new EventWaitHandle(false, EventResetMode.ManualReset, "myEH");

static void Main(string[] args)

{

RegisteredWaitHandle wh = ThreadPool.RegisterWaitForSingleObject(eh, (state, @out) =>mn.Set(),null,Timeout.Infinite, false);

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

{

ThreadPool.QueueUserWorkItem(CallBack);

}

while (true)

{

if (Console.ReadKey().Key != ConsoleKey.Spacebar)

{

eh.Set();

eh.Reset();

}

else

{

wh.Unregister(eh);

eh.Reset();

break;

}

}

wh.Unregister(eh);

Console.ReadKey();

}

private static void CallBack(object state)

{

Console.WriteLine("tread WaitOne" + Thread.CurrentThread.ManagedThreadId);

mn.WaitOne();

Console.WriteLine("tread start" + Thread.CurrentThread.ManagedThreadId);

}

}

}



using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Net.Mime;

using System.Security.AccessControl;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace MutexOneProcess

{

class Program

{

private static Mutex mut;

static void Main(string[] args)

{

if (Mutex.TryOpenExisting("MUTEX", out mut))

{

Console.WriteLine("Solution Close");

Process.GetCurrentProcess().Kill();

}

else

{

mut=new Mutex(false, "MUTEX");

}

Console.WriteLine("Solution Run");

new Thread(Start).Start();

}

private static void Start()

{

while (true)

{

mut.WaitOne();

Console.WriteLine("Curent Thread - {0}", Thread.CurrentThread.ManagedThreadId);

Console.ReadKey();

mut.ReleaseMutex();

}

}

}

}