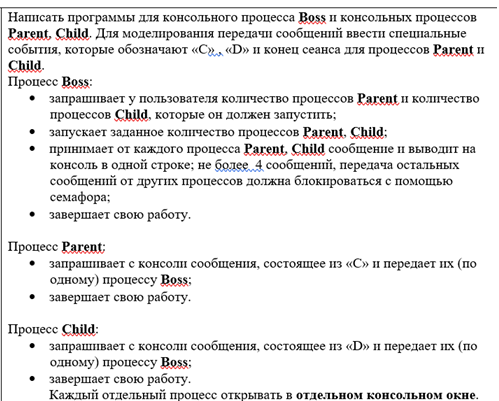
Лабораторная работа №4

Синхронизация процессов и потоков при помощи событий и семафоров в ОС Windows

Самусев Д. А. АСОИ-181

**Цель:** Научиться использовать функции Win32 API предназначенные для синхронизации процессов и потоков в ОС Windows.



using System; //СЕРВЕР (BOSS)

using System.Threading;

using System.Diagnostics;

namespace lab4

{

class Program

{

public static Semaphore sem = new Semaphore(4, 4);

class MyThread1

{

public Thread Thrd;

public MyThread1()

{

Thrd = new Thread(this.Run);

Thrd.Start();

}

void Run()

{

Console.Write("C");

Thread.Sleep(5000);

// Освободить семафор

sem.Release();

}

}

class MyThread2

{

public Thread Thrd;

public MyThread2()

{

Thrd = new Thread(this.Run);

Thrd.Start();

}

void Run()

{

Console.Write("D");

Thread.Sleep(5000);

// Освободить семафор

sem.Release();

}

}

static void Main(string[] args)

{

int n, m, counter;

Console.WriteLine("Введите количество Parent процессов:");

n = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите количество Child процессов:");

m = Convert.ToInt32(Console.ReadLine());

counter = n + m;

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

Process.Start(@"E:\3 курс\Лаб СПО (2 часть)\для прог\lab4\lab4\_2\bin\Debug\lab4\_2.exe");

for (int j=0;j<m;j++)

Process.Start(@"E:\3 курс\Лаб СПО (2 часть)\для прог\lab4\lab4\_3\bin\Debug\lab4\_3.exe");

//Создаем два события с ручным сбросом

EventWaitHandle CEvent = new EventWaitHandle(false, EventResetMode.ManualReset, "CEvent");

EventWaitHandle DEvent = new EventWaitHandle(false, EventResetMode.ManualReset, "DEvent");

EventWaitHandle EndClientEvent = new EventWaitHandle(false, EventResetMode.ManualReset, "EndClientEvent");

Console.WriteLine("Ожидание клиентов...");

while (true)

{

//Устанавливаем режим ожидания одного из двух событий

int eventIndex = WaitHandle.WaitAny(new WaitHandle[] { CEvent, DEvent, EndClientEvent });

if (eventIndex == 0)

{

CEvent.Reset();

sem.WaitOne();

MyThread1 newThread = new MyThread1();

}

else if (eventIndex == 1)

{

DEvent.Reset();

sem.WaitOne();

MyThread2 newThread = new MyThread2();

}

else if (eventIndex==2)

{

counter--;

if (counter == 0)

break;

EndClientEvent.Reset();

}

}

}

}

}

using System; //КЛИЕНТ (PARENT)

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

namespace lab4\_2

{

class Program

{

static void Main(string[] args)

{

//Открываем созданные Сервером события

EventWaitHandle CEvent = EventWaitHandle.OpenExisting("CEvent");

EventWaitHandle EndClientEvent = EventWaitHandle.OpenExisting("EndClientEvent");

Console.WriteLine("Введите сообщение состоящие из С для передачи серверу");

while (true)

{

string message = Console.ReadLine();

if (message == "C")

{

CEvent.Set();

}

else

if (message == "")

{

EndClientEvent.Set();

break;

}

}

}

}

}

using System; //КЛИЕНТ (CHILD)

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

namespace lab4\_3

{

class Program

{

static void Main(string[] args)

{

EventWaitHandle DEvent = EventWaitHandle.OpenExisting("DEvent");

EventWaitHandle EndClientEvent = EventWaitHandle.OpenExisting("EndClientEvent");

Console.WriteLine("Введите сообщение состоящие из D для передачи серверу");

while (true)

{

string message = Console.ReadLine();

if (message == "D")

{

DEvent.Set();

}

else

if (message == "")

{

EndClientEvent.Set();

break;

}

}

}

}

}

