Лекция 14 Работа с каналами в языке C#

В лекции использован материал книги Побегайло А.П. Системное программирование в Windows и книги Албахари Д. и Албахари Б. C# 3.0 Справочник.

14.1 Каналы в языке C#

Как мы отмечали в предыдущей лекции, обмен данными между процессами в системе Windows осуществляется через каналы. Естественно в языке C# имеется специальный класс, реализующий технологию передачи данных через канал – это класс PipeStream (дословно поток-канал).

Класс PipeStream предоставляет набор методов, реализующих взаимодействие процессов с помощью протоколов (правил обмена данными) каналов в системе Windows.

Существу два вида каналов в языке C# это анонимный и именованный каналы.

Именованный канал предполагает создание объекта канал с заданием ему имени для работы в сети Windows. Основное его назначение передача данных между процессами (обычно их называют клиентом и сервером), работающими на разных компьютерах.

Анонимный канал предназначен для передачи данных между процессами, работающими на одном компьютере, что позволяет обходиться без транспортного протокола.

Класс PipeStream состоит из четырех подклассов. Два первых подкласса обеспечивают работу анонимного канала (AnonymousPipeServerStream и AnonymousPipeClientStream), а два вторых обеспечивают работу именованного канала (NamePipeServerStream и NamePipeClientStream).

Знакомство с работой каналов начнем с именованного канала передачи данных.

14.2 Именованные каналы

Именованным каналом называется объект языка C#, который обеспечивает передачу данных между процессами в локальной сети.

Обычно именованные каналы используются для передачи данных между процессами, выполняющимися на компьютерах в одной локальной сети, но они могут использоваться для передачи данных между процессами на одном компьютере.

К сожалению, администрация корпоративной сети нашего университета запрещает все «несанкционированные» передачи между компьютерами в сети, поэтому передачи будут организованы между процессами на одном компьютере.

Процесс, который создает именованный канал, называется сервером именованного канала. Процессы, которые связываются с именованным каналом, называются клиентами именованного канала.

Перечислим основные характеристики именованных каналов:

- канал имеет имя, которое используется клиентами для связи с именованным каналом;
- передача данных в канале может осуществляться как потоком, так и сообщениями;
- обмен данными может быть как синхронным, так и асинхронным.

Теперь приведем порядок работы с именованными каналами, который и будет использоваться в дальнейшем:

- создание именованного канала сервером;
- ожидание сервера подключения клиента к экземпляру именованного канала;
- создание именованного канала клиентом;
- соединение клиента с экземпляром именованного канала;
- обмен данными по именованному каналу;
- закрытие именованного канала клиентом и сервером.

Рассмотрим подробнее перечисленные пункты работы с именованными каналами.

Именованные каналы создаются процессом-сервером при помощи конструктора NamedPipeServerStream, который имеет 10 различных вариантов записи:

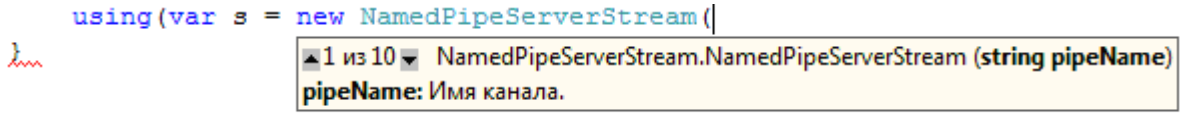

Рисунок 14.1 Варианты создания объекта именованного канала

Все варианты создания объекта именованного канала можно посмотреть в справке Visual Studio. Мы рассмотрим один, имеющий максимальное количество параметров:

NamedPipeServerStream
(String,
PipeDirection,
Int32,
PipeTransmissionMode,
PipeOptions,
Int32,
Int32,
PipeSecurity,
HandleInheritability,
PipeAccessRights)

Первый параметр предложенного варианта конструктора содержит имя канала. Это имя канала будет использоваться и сервером и клиентом в локальной сети, поэтому оно должно быть уникальным.

Второй параметр PipeDirection, определяет направление передачи данных. Возможны варианты In, InOut и Out. Именованные каналы, по умолчанию, являются двунаправленными. Серверу и клиенту разрешено передавать или читать данные в канал. Для этого сервер и клиент «должны договориться» о каком-нибудь протоколе, координирующем их действия (например, обмен сообщениями поочередно), чтобы исключить одновременного обращения к каналу.

Третий параметр определяет максимальное число экземпляров сервера с одинаковыми именами, которым могут подключаться клиенты. В нашем случае будет существовать одна пара сервер – клиент.

Параметр PipeTransmissionMode определяет режим передачи данных, например, PipeTransmissionMode.Message определяет передачу сообщениями.

Параметр PipeOptions определяет способ создания или открытия канала.

Следующие параметры задают размеры входного и выходного буферов обмена данными канала. Обычно они определяются по умолчанию.

Режим безопасности работы канала определяется параметром PipeSecurity. У вас будет дисциплина «Защита информации», в которой будут рассмотрены значения подобных параметров объектов.

Параметр HandleInheritability определяет, может ли базовый дескриптор канала наследоваться «дочерними» процессами.

Параметр PipeAccessRights определяет права доступа к каналу.

Таким образом, конструктор рассматриваемого варианта создает и инициализирует новый экземпляр класса NamedPipeServerStream с заданным именем канала, направлением канала, максимальным количеством экземпляров сервера, режимом передачи, параметрами канала, рекомендуемыми размерами входного и выходного буферов, режимом безопасности канала, режимом наследования и правами доступа к каналу.

После создания именованного канала сервер ждет подключения клиента с помощью метода WaitForConnection();.

Процесс клиента в языке C# также создает объект именованного канала с помощью конструктора, имеющего 8 вариантов записей.

Все варианты создания объекта именованного канала процессом клиентом можно посмотреть в справке Visual Studio. Мы рассмотрим один, имеющий максимальное количество параметров:

```
NamedPipeClientStream
(String,
String,
PipeDirection,
PipeOptions,
TokenImpersonationLevel,
```

HandleInheritability)

Как сказано в Help (F1) среды данная запись инициализирует новый экземпляр класса NamePipeClientStream с заданными именами канала и сервера, направлением передачи данных в канале, параметрами канала, уровнем безопасности и режимом наследования.

Следующими этапами по протоколу работы именованного канала клиент должен соединиться с экземпляром именованного канала. Далее выполняется обмен данными между сервером и клиентом. По окончании обмена осуществляется закрытие именованного канала клиентом и сервером.

В качестве учебного примера рассмотрим работу именованного канала, через который сервер передает клиенту массив байтов, сформированный случайным образом. Клиент сортирует его и возвращает серверу. Сервер распечатывает полученный массив. Задача реализована двумя программами – основной программой (сервером) и дополнительной программой (клиентом). Исходный код основной программы:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO.Pipes;

namespace ConsoleApplication1
{
 class Program
 {
  static void Main(string[] args)
  {
   byte[] a = new byte[10];
   using (var s = new NamedPipeServerStream("Pipe_lab6"))
   {
    Random rnd = new Random();
    Console.WriteLine("Массив создан: ");
    for (int i = 0; i < 10; i++)
    {
     a[i] = (byte)(rnd.Next() % 101);
     Console.Write(a[i] + "\t");
    }
    Console.WriteLine();

    s.WaitForConnection();
    for (int i = 0; i < 10; i++)
         s.WriteByte(a[i]);
    for (int i = 0; i < 10; i++)
         a[i] = (byte)(s.ReadByte());
    Console.WriteLine("Массив после сортировки: ");
    for (int i = 0; i < 10; i++)
         Console.Write(a[i] + "\t");
```

```
     Console.WriteLine();
    }
    Console.WriteLine("Сеанс сервера закончен");
    Console.ReadLine();
   }
  }
}
```

Исходный код дополнительной программы:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO.Pipes;

namespace ConsoleApplication2
{
 class Program
 {
  static void Main(string[] args)
  {
   byte[] a = new byte[10];
   using (var s = new NamedPipeClientStream("Pipe_lab6"))
   {
    Console.WriteLine("Начинает работать клиент");
    byte b;
    s.Connect();
    for (int i = 0; i < 10; i++)
        a[i] = (byte)(s.ReadByte());
    Console.WriteLine("Полученный массив байтов:");
    for (int i = 0; i < 10; i++)
        Console.Write(a[i] + "\t");
    Console.WriteLine();
    for (int i = 0; i < 9; i++)
     for (int j = i + 1; j < 10; j++)
      if (a[i] < a[j])
          { b = a[i]; a[i] = a[j]; a[j] = b; }
    Console.WriteLine("Массив после сортировки:");
    for (int i = 0; i < 10; i++)
        Console.Write(a[i] + "\t");
    Console.WriteLine();

    for (int i = 0; i < 10; i++)
        s.WriteByte(a[i]);
   }
   Console.WriteLine("Сеанс клиента закончен");
   Console.ReadLine();
  }
 }
}
```

Работа программы представлена на рисунке 14.2.

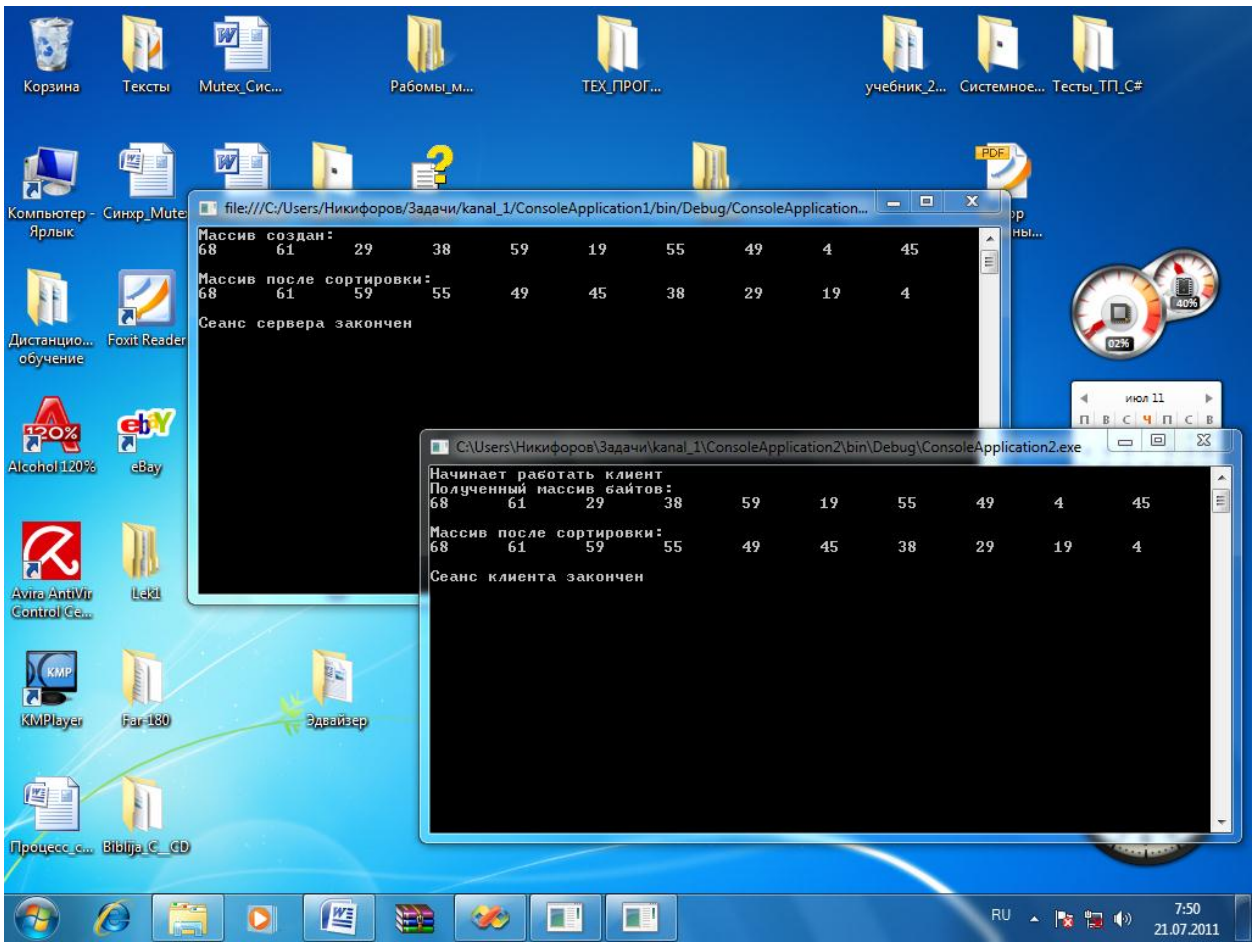

Рисунок 14.2 – Работа именованного канала при передаче массива байтов

14.3 Использование именованного канала для передачи сообщений

Рассмотрим пример программ, в которых сервер и клиент поочередно обмениваются текстовыми сообщениями. Процесс обмена прекращается, когда один из участников отправит сообщение “ДОМОЙ”.

Исходный код сервера:

```
using System;
using System.Threading;
using System.Diagnostics;
using System.IO;
using System.IO.Pipes;

namespace Server_Name_1
{
 class Program
 {
  static void Main(string[] args)
  {
   string clientfileExe = "Klient_Name_1";
   Process p = Process.Start(clientfileExe);
   // Создание именованного канала сервером.
   NamedPipeServerStream pipestream = new
NamedPipeServerStream("Kanal_Name_1");
```

```
    // Ждем соединения с клиентом.
    Console.WriteLine("Ждем соединения с клиентом");
    pipestream.WaitForConnection();

    StreamReader reader = new StreamReader(pipestream);
    StreamWriter writer = new StreamWriter(pipestream);
    string ss = "";
    int fl = 0;
    while (ss != "ДОМОЙ")
    {
     if (fl == 0)
     {
      Console.WriteLine("Ваше сообщение ?");
      ss = Console.ReadLine();
      writer.WriteLine(ss);
      writer.Flush();
      fl = 1;
     }
     else
     {
      ss = reader.ReadLine();
      Console.WriteLine("Получено сообщение от клиента : " + ss);
      fl = 0;
     }
    }
    Console.WriteLine("Сеанс связи закончен");
    Console.ReadLine();
    // Закрываем канал.
    pipestream.Close();
   }
  }
}
```

Исходный код клиента:

```
using System;
using System.Threading;
using System.Diagnostics;
using System.IO;
using System.IO.Pipes;

namespace Klient_Name_1
{
 class Program
 {
  static void Main(string[] args)
  {
   string ss;
   //Создаем именованный канал клиентом.
   NamedPipeClientStream pipestream = new
NamedPipeClientStream("Kanal_Name_1");
   //Соединяемся с каналом.
   pipestream.Connect();

   StreamReader reader = new StreamReader(pipestream);
```

```
    StreamWriter writer = new StreamWriter(pipestream);
    int fl = 0;
    do
    {
     if (fl == 0)
     {
      ss = reader.ReadLine();
      Console.WriteLine("Получено сообщение от сервера: " + ss);
      fl = 1;
     }
     else
     {
      Console.WriteLine("Сообщение от клиента ?");
      ss = Console.ReadLine();
      writer.WriteLine(ss);
      writer.Flush();
      fl = 0;
     }
    }
    while (ss != "ДОМОЙ");
    Console.WriteLine("Сеанс связи закончен");
    Console.ReadLine();
    // Закрываем канал.
    pipestream.Close();
   }
  }
}
```

Для нормальной работы программ файл `Klient_Name_1.exe`, должен находиться в папке с файлом `Server_Name_1.exe`.

Работа программ:

Созданный канал открыт для чтения и записи, поэтому для исключения ситуации одновременного обращения к каналу и сервером и клиентом использован протокол поочередного доступа к каналу, который реализуется с помощью специальной переменной «флага» fl и операторов условного перехода.

Для передачи текстовых сообщений по именованному каналу использованы потоковые адаптеры:

```
StreamReader reader = new StreamReader(pipestream);
StreamWriter writer = new StreamWriter(pipestream);
```

которые будут рассмотрены в следующей лекции.

Работа программы представлена копией экрана консольных приложений сервера и клиента, представленной на рисунке 14.3.

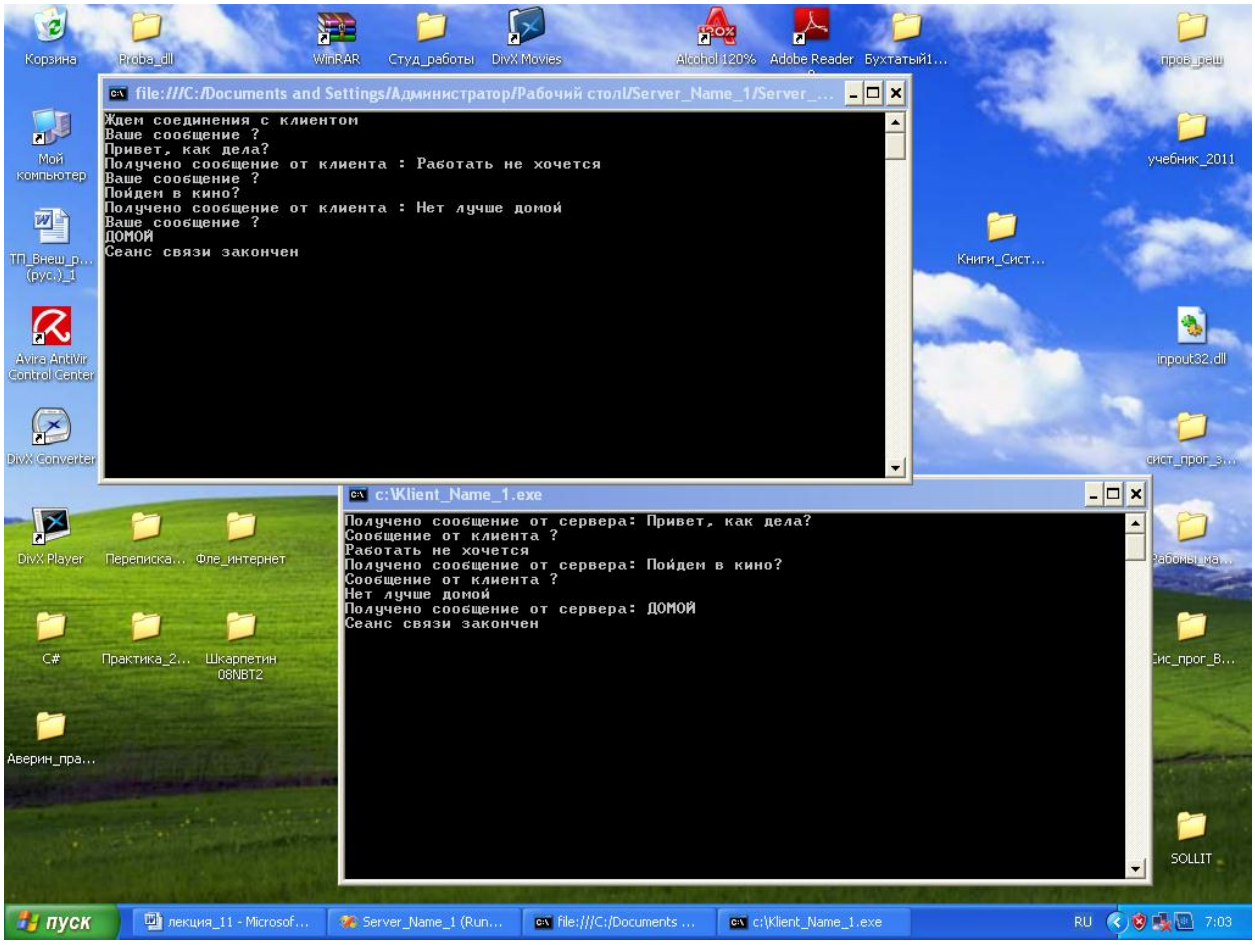

Рисунок 14.3 – Работа именованного канала при передаче сообщений