Пермский филиал федерального государственного автономного образовательного учреждения высшего образования   
«Национальный исследовательский университет   
«Высшая школа экономики»

*Факультет экономики, менеджмента и бизнес-информатики*

Чепоков Елизар Сергеевич

**ИМЕННОВАННЫЕ КАНАЛЫ (NAMED PIPES)**

*отчет по практической работе №1*

студента образовательной программы «Программная инженерия»

по направлению подготовки *09.03.04 Программная инженерия*

**Руководитель:**

доцент кафедры

информационных

технологий в бизнесе

Е. Б. Замятина

Пермь, 2021 год

**Оглавление**

[Требования к заданию 3](#_Toc88694340)

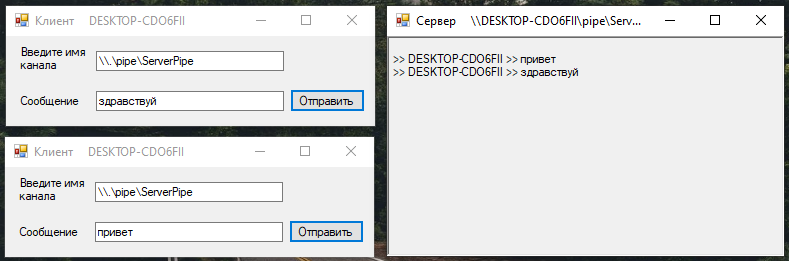
[Реализация задания 4](#_Toc88694341)

# **Требования к заданию**

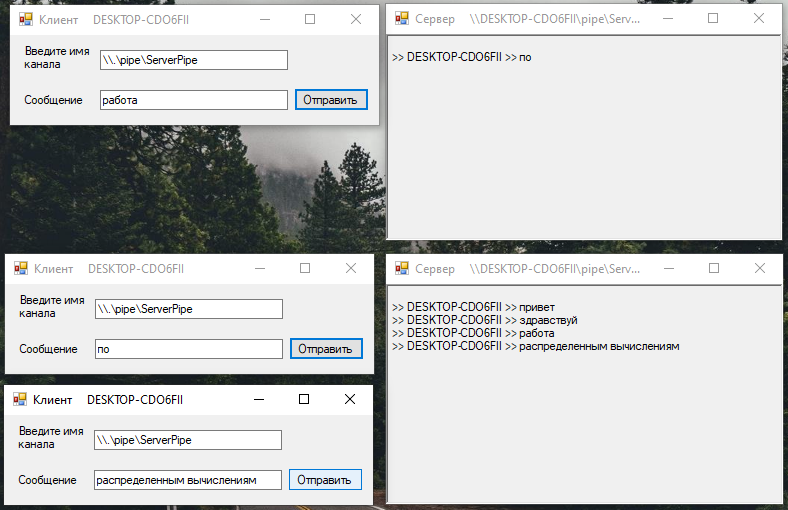
Выполнение задания включает в себя:

1. Изучите разработанное приложение. Запустите исполняемый файл сервера и клиента на одной машине, затем перенесите один из компонентов распределенного приложения на другой вычислительный узел (лучше на ноутбук, т.к. администратор ЛВС вуза может запретить обмен сообщения между различными узлами ЛВС) и попробуйте его запустить. Запустите несколько клиентов и попробуйте отправить сообщения одному серверу.
2. Запустите несколько серверов на одной машине. Отправьте им сообщения от нескольких клиентов. Объясните, почему приложение перестало работать.
3. Модифицируйте приложение так, чтобы существовала возможность на сервере идентифицировать клиентов не по имени вычислительного узла, а по нику/логину пользователя.
4. Модифицируйте приложение так, чтобы получился полноценный чат. Клиент может отправлять сообщения всем клиентам, участвующим в беседе. Для этого каждый клиент должен иметь возможность просмотра всех сообщений от всех клиентов, а сервер должен содержать список клиентов, которые хотят участвовать в беседе, чтобы каждый раз выполнять им рассылку сообщений.

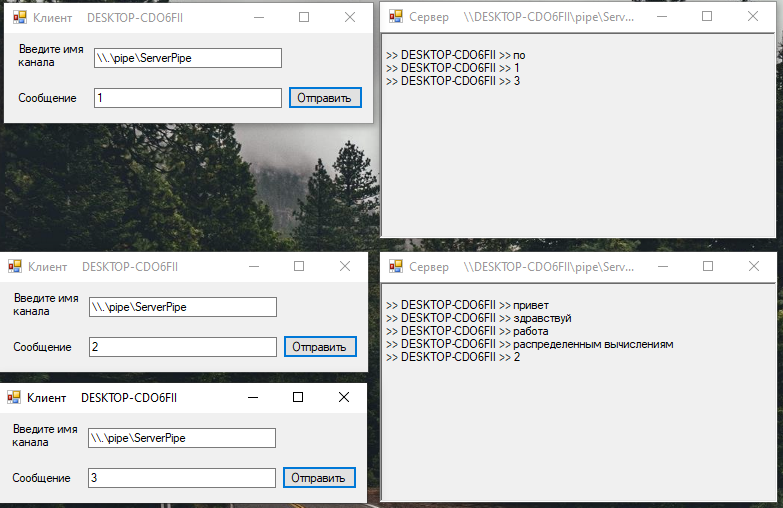
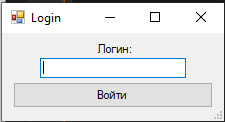
# **Реализация задания**

1. Запуск одного сервера и двух клиентов (Рисунок 1):

**Рисунок 1 – Работа программ**

1. Предполагаемого сбоя не произошло, но вывод сообщений остался некорректным. Каждый клиент подключается по одному имени канала, но так как названия каналов совпадают, то сообщения отправляются «как получится», предположительно на данный результат повлияла задержка в виде Thread sleep и сообщения отправились в разнобой, на рисунках 2 и 3 представлена работа чата, расположение клиентов не менялось. При отсутствии предполагается, что все сообщения поступили бы на самый последний запущенный сервер, тк он последний отправил запрос, либо на первый запущенный сервер, тк id приложения ниже.

**Рисунок 2 – Первая передача сообщений**

1. Добавление логина при входе (Рисунок 4):

**Рисунок 4 – Добавление логина при входе**

**Рисунок 3 – Вторая передача сообщений**

public partial class Login : Form

{

public string login;

public Login()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

if (tbLogin.Text == "")

{

MessageBox.Show("Введите логин");

return;

}

else

{

this.DialogResult = DialogResult.OK;

this.login = tbLogin.Text;

}

}

}

public frmMain()

{

Login form = new Login();

if (form.ShowDialog() == DialogResult.OK) login = form.login;

InitializeComponent();

lblShowLogin.Text = login;

// создание именованного канала

PipeHandle = DIS.Import.CreateNamedPipe(

"\\\\.\\pipe\\" + login,

DIS.Types.PIPE\_ACCESS\_DUPLEX,

DIS.Types.PIPE\_TYPE\_BYTE | DIS.Types.PIPE\_WAIT,

DIS.Types.PIPE\_UNLIMITED\_INSTANCES,

0,

1024,

DIS.Types.NMPWAIT\_WAIT\_FOREVER,

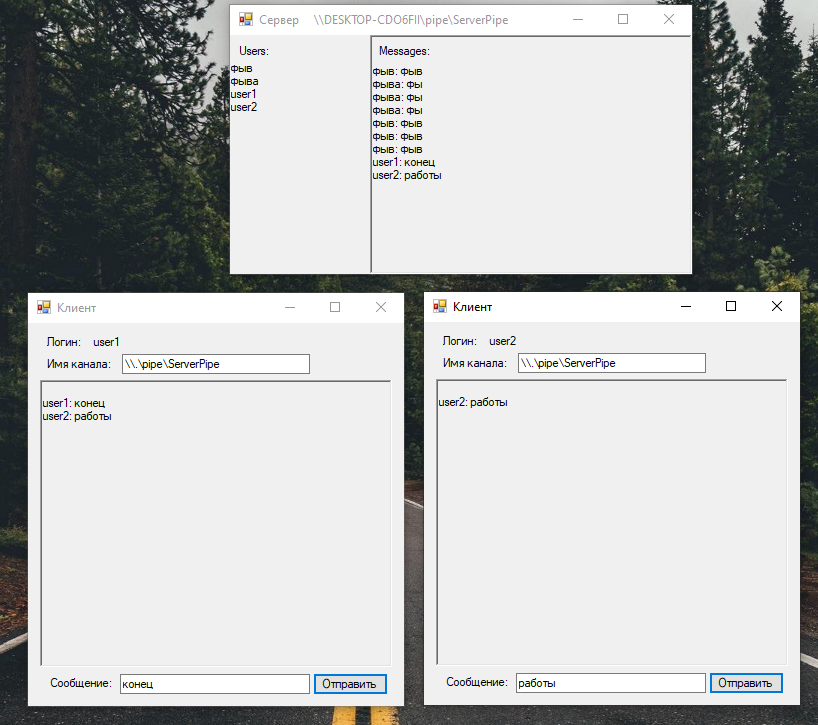
(uint)0);

// создание потока, отвечающего за работу с каналом

t = new Thread(ReceiveMessage);

t.Start();

}

1. Перемещаем сервер в клиент и переправляем обратно сообщения с сервера (Рисунок 5):

**Рисунок 5 – Пересылка сообщений**

namespace Pipes

{

public partial class frmMain : Form

{

private Int32 PipeHandle; // дескриптор канала

private string PipeName = "\\\\" + Dns.GetHostName() + "\\pipe\\ServerPipe"; // имя канала, Dns.GetHostName() - метод, возвращающий имя машины, на которой запущено приложение

private Thread t; // поток для обслуживания канала

private bool \_continue = true; // флаг, указывающий продолжается ли работа с каналом

private List<string> connectedUsers = new List<string>();

// конструктор формы

public frmMain()

{

InitializeComponent();

// создание именованного канала

PipeHandle = DIS.Import.CreateNamedPipe("\\\\.\\pipe\\ServerPipe",

DIS.Types.PIPE\_ACCESS\_DUPLEX,

DIS.Types.PIPE\_TYPE\_BYTE | DIS.Types.PIPE\_WAIT,

DIS.Types.PIPE\_UNLIMITED\_INSTANCES,

0,

1024,

DIS.Types.NMPWAIT\_WAIT\_FOREVER,

(uint)0);

// вывод имени канала в заголовок формы, чтобы можно было его использовать для ввода имени в форме клиента, запущенного на другом вычислительном узле

this.Text += " " + PipeName;

rtbUsers.Text += "\n";

rtbMessages.Text += "\n";

// создание потока, отвечающего за работу с каналом

t = new Thread(ReceiveMessage);

t.Start();

}

private void ReceiveMessage()

{

string msg = ""; // прочитанное сообщение

uint realBytesReaded = 0; // количество реально прочитанных из канала байтов

// входим в бесконечный цикл работы с каналом

while (\_continue)

{

if (DIS.Import.ConnectNamedPipe(PipeHandle, 0))

{

byte[] buf = new byte[1024]; // буфер прочитанных из канала байтов

DIS.Import.FlushFileBuffers(PipeHandle); // "принудительная" запись данных, расположенные в буфере операционной системы, в файл именованного канала

DIS.Import.ReadFile(PipeHandle, buf, 1024, ref realBytesReaded, 0); // считываем последовательность байтов из канала в буфер buff

msg = Encoding.Unicode.GetString(buf); // выполняем преобразование байтов в последовательность символов

string username = msg.Split(new string[] { ": " }, StringSplitOptions.None)[0];

rtbMessages.Invoke((MethodInvoker)delegate

{

if (msg != "")

{

rtbMessages.Text += "\n" + msg; // выводим полученное сообщение на форму

}

});

if (!connectedUsers.Contains(username))

{

connectedUsers.Add(username);

rtbMessages.Invoke((MethodInvoker)delegate { rtbUsers.Text += "\n" + username; });

}

DIS.Import.DisconnectNamedPipe(PipeHandle); // отключаемся от канала клиента

foreach (string user in connectedUsers)

{

string name = "\\\\.\\pipe\\" + user;

uint BytesWritten = 0; // количество реально записанных в канал байт

byte[] buff = Encoding.Unicode.GetBytes(msg); // выполняем преобразование сообщения (вместе с идентификатором машины) в последовательность байт

// открываем именованный канал, имя которого указано в поле tbPipe

int pipeSender = DIS.Import.CreateFile(name,

DIS.Types.EFileAccess.GenericWrite,

DIS.Types.EFileShare.Read,

0,

DIS.Types.ECreationDisposition.OpenExisting,

0,

0);

DIS.Import.WriteFile(pipeSender, buff, Convert.ToUInt32(buff.Length), ref BytesWritten, 0); // выполняем запись последовательности байт в канал

DIS.Import.CloseHandle(pipeSender); // закрываем дескриптор канала

}

Thread.Sleep(500); // приостанавливаем работу потока перед тем, как приcтупить к обслуживанию очередного клиента

}

}

}

private void SendMessage(string username, string message)

{

string name = "\\\\.\\pipe\\" + username;

uint BytesWritten = 0; // количество реально записанных в канал байт

byte[] buff = Encoding.Unicode.GetBytes(message); // выполняем преобразование сообщения (вместе с идентификатором машины) в последовательность байт

// открываем именованный канал, имя которого указано в поле tbPipe

int pipeSender = DIS.Import.CreateFile(name,

DIS.Types.EFileAccess.GenericWrite,

DIS.Types.EFileShare.Read,

0,

DIS.Types.ECreationDisposition.OpenExisting,

0,

0);

DIS.Import.WriteFile(pipeSender, buff, Convert.ToUInt32(buff.Length), ref BytesWritten, 0); // выполняем запись последовательности байт в канал

DIS.Import.CloseHandle(pipeSender); // закрываем дескриптор канала

}

private void frmMain\_FormClosing(object sender, FormClosingEventArgs e)

{

\_continue = false; // сообщаем, что работа с каналом завершена

if (t != null)

t.Abort(); // завершаем поток

if (PipeHandle != -1)

DIS.Import.CloseHandle(PipeHandle); // закрываем дескриптор канала

}

}

}

namespace Pipes

{

public partial class frmMain : Form

{

public static string login = "";

private Int32 PipeHandle; // дескриптор канала

private Thread t; // поток для обслуживания канала

private bool \_continue = true; // флаг, указывающий продолжается ли работа с каналом

// конструктор формы

public frmMain()

{

Login form = new Login();

if (form.ShowDialog() == DialogResult.OK) login = form.login;

InitializeComponent();

lblShowLogin.Text = login;

// создание именованного канала

PipeHandle = DIS.Import.CreateNamedPipe(

"\\\\.\\pipe\\" + login,

DIS.Types.PIPE\_ACCESS\_DUPLEX,

DIS.Types.PIPE\_TYPE\_BYTE | DIS.Types.PIPE\_WAIT,

DIS.Types.PIPE\_UNLIMITED\_INSTANCES,

0,

1024,

DIS.Types.NMPWAIT\_WAIT\_FOREVER,

(uint)0);

// создание потока, отвечающего за работу с каналом

t = new Thread(ReceiveMessage);

t.Start();

}

private void btnSend\_Click(object sender, EventArgs e)

{

if (tbMessage.Text == "")

{

MessageBox.Show("Введите сообщение");

return;

}

uint BytesWritten = 0; // количество реально записанных в канал байт

byte[] buff = Encoding.Unicode.GetBytes(login + ": " + tbMessage.Text); // выполняем преобразование сообщения (вместе с идентификатором машины) в последовательность байт

// открываем именованный канал, имя которого указано в поле tbPipe

int pipeSender = DIS.Import.CreateFile(tbPipe.Text,

DIS.Types.EFileAccess.GenericWrite,

DIS.Types.EFileShare.Read,

0,

DIS.Types.ECreationDisposition.OpenExisting,

0,

0);

DIS.Import.WriteFile(pipeSender, buff, Convert.ToUInt32(buff.Length), ref BytesWritten, 0); // выполняем запись последовательности байт в канал

DIS.Import.CloseHandle(pipeSender); // закрываем дескриптор канала

}

private void ReceiveMessage()

{

string msg = ""; // прочитанное сообщение

uint realBytesReaded = 0; // количество реально прочитанных из канала байтов

// входим в бесконечный цикл работы с каналом

while (\_continue)

{

if (DIS.Import.ConnectNamedPipe(PipeHandle, 0))

{

byte[] buff = new byte[1024]; // буфер прочитанных из канала байтов

DIS.Import.FlushFileBuffers(PipeHandle); // "принудительная" запись данных, расположенные в буфере операционной системы, в файл именованного канала

DIS.Import.ReadFile(PipeHandle, buff, 1024, ref realBytesReaded, 0); // считываем последовательность байтов из канала в буфер buff

msg = Encoding.Unicode.GetString(buff); // выполняем преобразование байтов в последовательность символов

rtbMessages.Invoke((MethodInvoker)delegate

{

if (msg != "")

rtbMessages.Text += "\n" + msg; // выводим полученное сообщение на форму

});

DIS.Import.DisconnectNamedPipe(PipeHandle); // отключаемся от канала клиента

Thread.Sleep(500); // приостанавливаем работу потока перед тем, как приcтупить к обслуживанию очередного клиента

}

}

}

private void frmMain\_FormClosing(object sender, FormClosingEventArgs e)

{

\_continue = false; // сообщаем, что работа с каналом завершена

if (t != null)

t.Abort(); // завершаем поток

if (PipeHandle != -1)

DIS.Import.CloseHandle(PipeHandle); // закрываем дескриптор канала

}

}

}