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

*Факультет экономики, менеджмента и бизнес-информатики*

Чепоков Елизар Сергеевич

**МЭЙЛСЛОТЫ (MAILSLOTS)**

*отчет по практическому заданию №2*

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

**Содержание**

[Задание 1 3](#_Toc94786411)

[Задание 2 4](#_Toc94786412)

[Задание 3 5](#_Toc94786413)

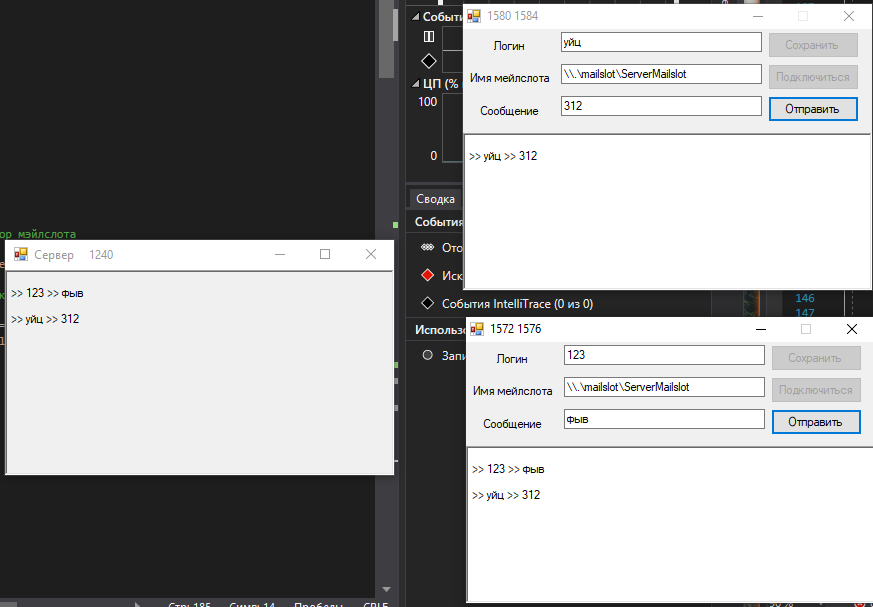
[Задание 4 7](#_Toc94786414)

[Задание 5 11](#_Toc94786415)

# Задание 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.

**Решение**

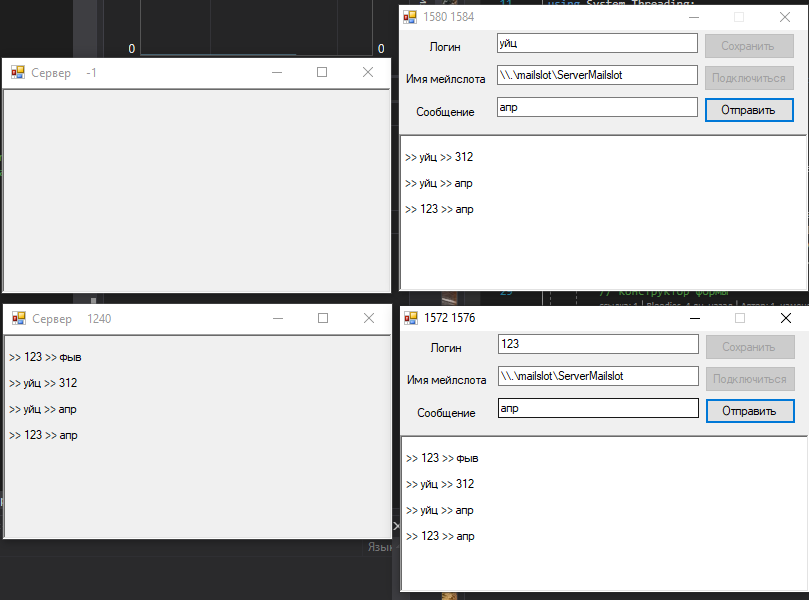
После запуска одного сервера и двух клиентов, ошибок при отправке сообщений не возникло. Ошибок не возникло как при передаче сообщений с помощью одного узла, так и с помощью нескольких (рис. 1.1).

**Рисунок 1.1 Передача сообщений с двух клиентов одному серверу**

# Задание 2

Запустите несколько серверов на одной машине. Отправьте им сообщения от нескольких клиентов. Объясните, почему приложение перестало работать.

**Решение**

При запуске нескольких серверов на одной машине, передача сообщений происходит некорректно. Сообщения приходят лишь на первый запущенный сервер. При этом ошибок, которые бы привели к краху приложения не возникло. (рис.2.1) 

***Рисунок 2.1 Передача сообщений с двух клиентов нескольким серверам***

# Задание 3

Модифицируйте приложение так, чтобы существовала возможность на сервере идентифицировать клиентов не по имени вычислительного узла, а по нику/логину пользователя.

**Решение**

Чтобы была возможность идентифицировать клиентов по никнейму, на форму клиента было добавлено поле ввода никнейма и внесены соответствующие изменения в коде клиента.

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

{

if (!string.IsNullOrEmpty(tbLogin.Text))

{

\_guid = new Guid();

\_userName = tbLogin.Text;

this.Text = \_userName;

btnSave.Enabled = false;

try

{

// открываем мэйлслот с название нашего сервера

\_mailServerSlot = DIS.Import.CreateFile

(@"\\.\mailslot\ServerMailslot",

DIS.Types.EFileAccess.GenericWrite,

DIS.Types.EFileShare.Read,

0,

DIS.Types.ECreationDisposition.OpenExisting,

0,

0);

if (\_mailServerSlot != -1)

{

btnSend.Enabled = true;

// создание собсбтвенного мейлслота для нашего приема сообщений

\_ourOwnClientMailSlot =

DIS.Import.CreateMailslot(

"\\\\.\\mailslot\\" + \_userName,

0,

DIS.Types.MAILSLOT\_WAIT\_FOREVER,

0);

this.Text = \_ourOwnClientMailSlot.ToString() + " " + \_mailServerSlot.ToString();

\_thredForReceivingMessages = new Thread(ReceiveMessage);

\_thredForReceivingMessages.Start();

sendMessage(\_userName + ":" + \_ourOwnClientMailSlot + "\_NAME\_");

}

else

MessageBox.Show("Не удалось подключиться к мейлслоту");

}

catch

{

MessageBox.Show("Не удалось подключиться к мейлслоту");

}

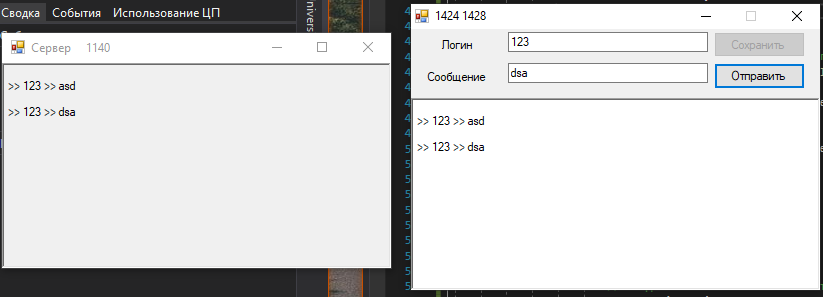
}

else

{

MessageBox.Show("Empty Name");

}

} 

***Рисунок 3.3 Работа приложения с никнеймами***

# Задание 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.

**Решение**

Необходимо добавить клиентам возможность сервера, получать сообщения.

Листинг:

private void ReceiveMessage()

{

string msg = ""; // прочитанное сообщение

int MailslotSize = 0; // максимальный размер сообщения

int lpNextSize = 0; // размер следующего сообщения

int MessageCount = 0; // количество сообщений в мэйлслоте

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

try

{

// входим в бесконечный цикл работы с мэйлслотом

while (\_continue)

{

// получаем информацию о состоянии мэйлслота

DIS.Import.GetMailslotInfo(\_ourServiceMailSlot,

MailslotSize,

ref lpNextSize,

ref MessageCount,

0);

// если есть сообщения в мэйлслоте, то обрабатываем каждое из них

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

{

byte[] buff = new byte[1024];

DIS.Import.FlushFileBuffers(\_ourServiceMailSlot); // "принудительная" запись данных, расположенные в буфере операционной системы, в файл мэйлслота

DIS.Import.ReadFile(\_ourServiceMailSlot,

buff,

1024,

ref realBytesReaded,

0); // считываем последовательность байтов из мэйлслота в буфер buff

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

if (\_previousMessage.Equals(msg))

continue;

bool isMessageToSend = true;

if (string.IsNullOrEmpty(msg))

break;

if (msg.Contains("\_NAME\_"))

{

\_clients.Add(msg.Split('\_')[0]);

string s = msg.Split('\_')[0].Split(':')[0];

int number = int.Parse(msg.Split('\_')[0].Split(':')[1]);

if (storageOfMailslot.ContainsKey(s))

;

else

storageOfMailslot.Add(s, number);

isMessageToSend = false;

}

if (msg.Contains("\_DELETE\_"))

{

\_clients.Remove(msg.Split('\_')[0]);

isMessageToSend = false;

}

if (msg.Contains("\_NO\_"))

{

isMessageToSend = false;

}

if (isMessageToSend)

{

rtbMessages.Invoke((MethodInvoker)delegate

{

if (msg != "")

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

});

foreach (string \_connection in storageOfMailslot.Keys)

{

//отправить сообщение обратно

int \_mailServerSlot = DIS.Import.CreateFile

(@"\\.\mailslot\" + \_connection,

DIS.Types.EFileAccess.GenericWrite,

DIS.Types.EFileShare.Read,

0,

DIS.Types.ECreationDisposition.OpenExisting,

0,

0);

if (\_mailServerSlot != -1)

{

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

byte[] buff1 = Encoding.Unicode.GetBytes(msg); // выполняем преобразование сообщения (вместе с идентификатором машины) в последовательность байт

DIS.Import.WriteFile(\_mailServerSlot,

buff1,

Convert.ToUInt32(buff1.Length),

ref BytesWritten,

0);

}

}

// SendMessageToAllClients("\n >> " + msg+ "\n");

}// приостанавливаем работу потока перед тем, как приcтупить к обслуживанию очередного клиента

\_previousMessage = msg;

}

Thread.Sleep(500);

}

}

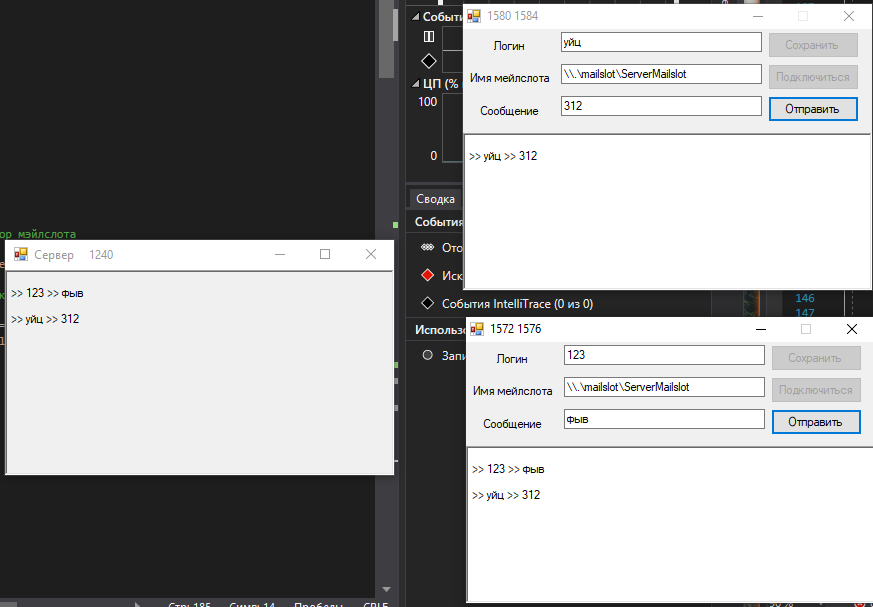
catch (ThreadInterruptedException ex)

{

MessageBox.Show(ex.Message);

}

}



**Рисунок 4.3 Пример работы программы**

# Задание 5

Модифицируйте приложение так, чтобы клиенты, желающие принять участие в беседе, находили сервер в сети с помощью широковещательного запроса, а не через ввод имени машины, на которой расположен сервер, в форме клиента.

try

{

// открываем мэйлслот с название нашего сервера

\_mailServerSlot = DIS.Import.CreateFile

(@"\\.\mailslot\ServerMailslot",

DIS.Types.EFileAccess.GenericWrite,

DIS.Types.EFileShare.Read,

0,

DIS.Types.ECreationDisposition.OpenExisting,

0,

0);

if (\_mailServerSlot != -1)

{

btnSend.Enabled = true;

// создание собсбтвенного мейлслота для нашего приема сообщений

\_ourOwnClientMailSlot =

DIS.Import.CreateMailslot(

"\\\\.\\mailslot\\" + \_userName,

0,

DIS.Types.MAILSLOT\_WAIT\_FOREVER,

0);

this.Text = \_ourOwnClientMailSlot.ToString() + " " + \_mailServerSlot.ToString();

\_thredForReceivingMessages = new Thread(ReceiveMessage);

\_thredForReceivingMessages.Start();

sendMessage(\_userName + ":" + \_ourOwnClientMailSlot + "\_NAME\_");

}

else

MessageBox.Show("Не удалось подключиться к мейлслоту");

}

catch

{

MessageBox.Show("Не удалось подключиться к мейлслоту");

}