Министерство образования и науки РФ

ФГБОУ ВПО «Тамбовский государственный технический университет»

Кафедра «»

Лабораторная работа №6

по дисциплине «Операционные системы»

«Межпроцессное взаимодействие»

Выполнил: студент гр..

Проверил:.

Тамбов, 20

***Цели и задачи****.*

Изучение механизмов межпроцессного взаимодействия (InterProcess Communication) в Windows NT. Получение практических навыков по использованию Win32 API для программирования механизмов IPC.

Задание: Разработать программу, реализующую обмен текстовыми сообщениями между несколькими процессами (Чат) при помощи механизмов межпроцессного взаимодействия. Способ межпроцессного взаимодействия выбирается исходя из особенностей поставленной задачи.

***Решение задачи****.*

Каналы передачи данных Mailslot:

простой способ организации передачи данных между различными процессами, основанный на использовании датаграммных каналов Mailslot.

Каналы Mailslot позволяют выполнять одностороннюю передачу данных от одного или нескольких клиентов к одному или нескольким серверам. Главная особенность каналов Mailslot заключается в том, что они, в отличие от других средств, позволяют передавать данные в широковещательном режиме. Это означает, что на компьютере или в сети могут работать несколько серверных процессов, способных получать сообщения через каналы Mailslot. При этом один клиентский процесс может посылать сообщения сразу всем этим серверным процессам.

*С помощью каналов Pipe вы не сможете передавать данные в широковещательном режиме, так как только два процесса могут создать канал типа Pipe.*

**Создание канала Mailslot**

Канал Mailslot создается серверным процессом с помощью специально предназначенной для этого функции **CreateMailslot**. После создания серверный процесс получает идентификатор канала Mailslot. Пользуясь этим идентификатором, сервер может читать сообщения, посылаемые в канал клиентскими процессами. Однако сервер не может выполнять над каналом Mailslot операцию записи, так как этот канал предназначен только для односторонней передачи данных - от клиента к серверу.

Прототип функции **CreateMailslot**:

HANDLE **CreateMailslot**(

LPCTSTR **lpName**, // адрес имени канала Mailslot

DWORD **nMaxMsgSize**, // максимальный размер сообщения

DWORD **lReadTimeout**, // время ожидания для чтения

LPSECURITY\_ATTRIBUTES **lpSecurityAttributes**); // адрес

// структуры защиты

Через параметр **lpName** передается адрес строки символов с именем канала Mailslot. Эта строка имеет следующий вид:

\\.\mailslot\[Путь]ИмяКанала

В этом имени путь является необязательной компонентой.

**Открытие канала Mailslot**

Прежде чем приступить к работе с каналом Mailslot, клиентский процесс должен его открыть. Для выполнения этой операции следует использовать функцию **CreateFile.**

**Запись сообщений в канал Mailslot**

Запись сообщений в канал Mailslot выполняет клиентский процесс, вызывая для этого функцию **WriteFile**. Этой функции необходимо передать идентификатор канала Mailslot, полученный от функции CreateFile.

**Чтение сообщений из канала Mailslot**

Серверный процесс может читать сообщения из созданного им канала Mailslot при помощи функции **ReadFile.**

**Определение состояния канала Mailslot**

Серверный процесс может определить текущее состояние канала Mailslot по его идентификатору с помощью функции **GetMailslotInfo**. Прототип этой функции:

BOOL **GetMailslotInfo**(

HANDLE **hMailslot**, // идентификатор канала Mailslot

LPDWORD **lpMaxMessageSize**, // адрес максимального размера

// сообщения

LPDWORD **lpNextSize**, // адрес размера следующего сообщения

LPDWORD **lpMessageCount**, // адрес количества сообщений

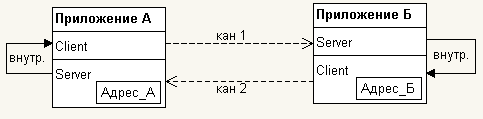
LPDWORD **lpReadTimeout**); // адрес времени ожидания

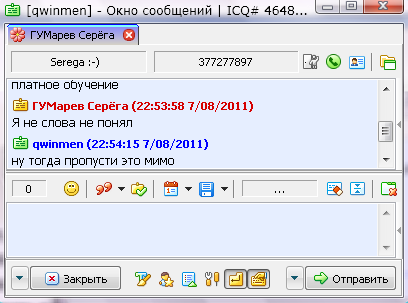
Через параметр hMailslot функции передается идентификатор канала Mailslot, состояние которого необходимо определить.

В случае успешного завершения функция GetMailslotInfo возвращает значение TRUE, а при ошибке - FALSE. Код ошибки можно получить, вызвав функцию GetLastError.

Итого: есть возможность создать сервер, а к ниму подключить клиента. Сообщения возможны только в одну сторону, от клиента на сервер.

Делаем финт ушами, а точней вот что:

т.е. сделаем коннект по логике типо «Кольцо». Логика не нарушена, по каналам 1 и 2 идет обмен сообщений только в одну сторону и только от клиента на сервер. На руку и то, что этих ящиков можно создать аж целую кучу, а тут – всего два и канал обмена готов. Внутренний обмен подразумевает некое текстовое поле, или контейнер, куда будет выводится текст готовый к отправке на другой сервер, и текст который будет принят от противоположной стороны. Некое подобие аськи, её и повторим в будущем приложении. По возможности повторим…

**Коды программы:**

**Form1.cs**

namespace qip

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

if (\_enterConnInf.ShowDialog() != DialogResult.OK && \_enterConnInf.StrMailBox != "Введите имя нового ящика")

{

toolStripStatusLabel2.Text = \_s + \_enterConnInf.StrMailBox;

\_preMailBox = \_enterConnInf.StrMailBox;

PrintDelegateFunc = new PrintInFCTB(PrintFunc);

tt=new Thread(new ParameterizedThreadStart(Serv))

{

IsBackground = true,

Priority = ThreadPriority.Lowest

};

tt.Start(true);

this.Show();

fctb.Invoke(PrintDelegateFunc, new object[] { dsdsd });

}else toolStripStatusLabel2.Text = \_s + "ПУСТО";

style = new GifImageStyle(fctb);

style.ImagesByText.Add(@":bb", Properties.Resources.bye);

style.ImagesByText.Add(@":D", Properties.Resources.lol);

style.ImagesByText.Add(@"8)", Properties.Resources.rolleyes);

style.ImagesByText.Add(@":@", Properties.Resources.unsure);

style.ImagesByText.Add(@":)", Properties.Resources.smile\_16x16);

style.ImagesByText.Add(@":(", Properties.Resources.sad\_16x16);

style.ImagesByText.Add(@"``", Properties.Resources.greenMsg);

style.ImagesByText.Add(@"~~", Properties.Resources.redMsg);

style.ImagesByText.Add(@"^^", Properties.Resources.cli\_qutim\_ico\_0);

style.StartAnimation();

fctb.OnTextChanged();

}

private string dsdsd = "^^" + "OnLine", \_preMailBox = "";

private delegate void PrintInFCTB(string str);

private PrintInFCTB PrintDelegateFunc;

private void PrintFunc(string str)

{

fctb.AppendText(str);

}

private Thread tt;

private const string \_s = "Созданный ящик: ";

private readonly ConnectInfo \_enterConnInf = new ConnectInfo();

private readonly ClientToServer \_clientToServB = new ClientToServer();

/// <summary>

/// Сервер\_А

/// </summary>

private void Serv(object state)

{

string mailName = null;

EventWaitHandle hEvent = null;

IntPtr hMailslot = IntPtr.Zero;

BinaryReader br = null;

int[] mailInfo = new int[4];

hMailslot = clAPI.CreateMailslot(

@"\\.\mailslot\" + (mailName = \_enterConnInf.StrMailBox),

0, -1, IntPtr.Zero

);

if (hMailslot == clAPI.INVALID\_HANDLE\_VALUE)

throw new Win32Exception(Marshal.GetLastWin32Error());

hEvent = new EventWaitHandle(

false,

EventResetMode.ManualReset,

@"Global\Mailslot\_" + mailName

);

br = new BinaryReader(//вместо ReadFile

new FileStream(hMailslot, FileAccess.ReadWrite),Encoding.Default);

while ((bool)state)

{

hEvent.WaitOne();

do

{

if (!clAPI.GetMailslotInfo(

hMailslot, out mailInfo[0], out mailInfo[1],

out mailInfo[2], out mailInfo[3])

)

{

MessageBox.Show("WARNING!!! GetMailslotInfo error: "+Marshal.GetLastWin32Error());

Debugger.Break();

break;

}

// No messages

if (mailInfo[1] == -1)

break;

/\*Ник контакта\*/ fctb.Invoke(PrintDelegateFunc, new object[] { (Environment.NewLine +"``"+\_clientToServB.StrClientToServB + Environment.NewLine) });

//dsdsd+="Message:"+Environment.NewLine;

fctb.Invoke(PrintDelegateFunc, new object[] { (Encoding.Default.GetString(br.ReadBytes(mailInfo[1]))) });

//dsdsd+=(Encoding.Default.GetString(br.ReadBytes(mailInfo[1])));

} while (mailInfo[2] != 0);

hEvent.Reset();

// Console.WriteLine("Continue? Y/N");

// if (Console.ReadLine().Equals("N", StringComparison.InvariantCultureIgnoreCase))

// break;

// Console.Clear();

// Console.WriteLine("mail: " + mailName);

//break;

}

br.Close();

hEvent.Close();

}

private void toolStripStatusLabel1\_Click(object sender, EventArgs e)

{

if (\_enterConnInf.ShowDialog() != DialogResult.OK && \_enterConnInf.StrMailBox != "Введите имя нового ящика" && \_enterConnInf.StrMailBox != \_preMailBox)

{

toolStripStatusLabel2.Text = \_s + \_enterConnInf.StrMailBox;

PrintDelegateFunc = new PrintInFCTB(PrintFunc);

tt = new Thread(new ParameterizedThreadStart(Serv))

{

IsBackground = true,

Priority = ThreadPriority.Lowest

};

tt.Start(true);

this.Show();

//fctb.Invoke(PrintDelegateFunc, new object[] { dsdsd });

}else toolStripStatusLabel2.Text = \_s + \_preMailBox;

}

#region fctb\_TxtChanged

GifImageStyle style;

static string RegexSpecSymbolsPattern = @"[\^\$\[\]\(\)\.\\\\*\+\|\?\{\}]";

private void fctb\_TextChanged(object sender, TextChangedEventArgs e)

{

if (style == null) return;

e.ChangedRange.ClearStyle(StyleIndex.All);

foreach (var key in style.ImagesByText.Keys)

{

string pattern = Regex.Replace(key, RegexSpecSymbolsPattern, "\\$0");

e.ChangedRange.SetStyle(style, pattern);

}

txtBoxClient.Focus();

}

#endregion

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

{

string Id = "~~" + \_enterConnInf.StrMailBox + Environment.NewLine;

string messaga = txtBoxClient.Text;

StrMessage = messaga;

Client(true);

txtBoxClient.Text = "";

fctb.Text += Environment.NewLine+Id+messaga;

//scroll to end

fctb.Select();

SendKeys.SendWait("^{END}");

txtBoxClient.Focus();

}

private void checkBoxEnterLock\_CheckedChanged(object sender, EventArgs e)

{

txtBoxClient.Focus();

}

private void txtBoxClient\_KeyUp(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter && checkBoxEnterLock.Checked)

{

txtBoxClient.Text = txtBoxClient.Text.TrimEnd('\r', '\n');

txtBoxClient.Select(0, 0);

btnSend\_Click(sender, e);

}

}

private bool \_state;

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

{//нужно подключение к серверу\_Б как клиента

//снять имя сервера\_Б со свойства \_clientToServB.StrClientToServB

if (\_clientToServB.ShowDialog() != DialogResult.OK && \_clientToServB.StrClientToServB != "Имя сервера\_Б" && \_clientToServB.StrClientToServB!=null)

{

Client(\_state);//MessageBox.Show(\_clientToServB.StrClientToServB);

}

txtBoxClient.Focus();

}

private void Client(object state)

{

string mailName = null;

EventWaitHandle hEvent = null;

IntPtr hMailslot = IntPtr.Zero;

int temp = 0;

//Console.WriteLine("Enter mail:");

hMailslot = clClientAPI.CreateFile(

@"\\.\mailslot\" + (mailName = \_clientToServB.StrClientToServB),

0x40000000U /\* GENERIC\_WRITE \*/,

0x1U /\* FILE\_SHARE\_READ \*/,

IntPtr.Zero,

0x3U /\* OPEN\_EXISTING \*/,

0x80U /\* FILE\_ATTRIBUTE\_NORMAL \*/,

IntPtr.Zero

);

if (hMailslot == clAPI.INVALID\_HANDLE\_VALUE)

throw new Win32Exception(Marshal.GetLastWin32Error());

hEvent = EventWaitHandle.OpenExisting(@"Global\Mailslot\_" + mailName);

while ((bool)state)

{

byte[] msg = null;

//Console.WriteLine("Write a message:");

//с textBoxClient.Text надо считать message

msg = Encoding.Default.GetBytes(StrMessage ?? "");

if (!clClientAPI.WriteFile(hMailslot, msg, msg.Length, out temp, IntPtr.Zero))

MessageBox.Show("WARNING!!! WriteFile error: "+Marshal.GetLastWin32Error());

else

hEvent.Set();

//Console.WriteLine("Continue? Y/N");

// if (Console.ReadLine().Equals("N", StringComparison.InvariantCultureIgnoreCase))

break;

}

clAPI.CloseHandle(hMailslot);

hEvent.Close();

}

/// <summary>

/// Тут храним тексты с textBoxClient.Text

/// </summary>

private string StrMessage { get; set; }

}

}

**Коды clAPI.cs:**

namespace qip

{

class clAPI

{

#region Server

public static readonly IntPtr INVALID\_HANDLE\_VALUE = (IntPtr)(-1);

[DllImport("kernel32.dll", SetLastError = true, CharSet = CharSet.Unicode)]

public static extern IntPtr **CreateMailslot**(//Создание канала

string lpName, // адрес имени канала Mailslot

int nMaxMessageSize, // максимальный размер сообщения

int lReadTimeout, // время ожидания для чтения

IntPtr lpSecurityAttributes // адрес структуры защиты

);

[DllImport("kernel32.dll", SetLastError = true)]

[return: MarshalAs(UnmanagedType.Bool)]

public static extern bool **GetMailslotInfo**(//Определение состояния канала Mailslot

IntPtr hMailslot, // идентификатор канала Mailslot

out int lpMaxMessageSize, // адрес максимального размера сообщения

out int lpNextSize, // адрес размера следующего сообщения

out int lpMessageCount, // адрес количества сообщений

out int lpReadTimeout // адрес времени ожидания

);

[DllImport("kernel32.dll", SetLastError = true)]

[return: MarshalAs(UnmanagedType.Bool)]

public static extern bool **CloseHandle**(

IntPtr hObject

);

#endregion

}

internal class clClientAPI

{

#region Client

[DllImport("kernel32.dll", SetLastError = true, CharSet = CharSet.Unicode)]

public static extern IntPtr **CreateFile**(//Открытие канала Mailslot

string fileName,//передается имя канала Mailslot

uint fileAccess,//над открываемым каналом будет выполняться операция r/w/rw

uint fileShare,

IntPtr securityAttributes,

uint creationDisposition,//открывает/создает существующий/новый канал

uint flags,

IntPtr template

);

[DllImport("kernel32.dll", SetLastError = true)]

[return: MarshalAs(UnmanagedType.Bool)]

public static extern bool **WriteFile**(//Запись сообщений в канал Mailslot

IntPtr hFile,//идентификатор канала Mailslot

byte[] lpBuffer,//определяет адрес буфера с сообщением

int nNumberOfBytesToWrite,//размер сообщения

out int lpNumberOfBytesWritten,

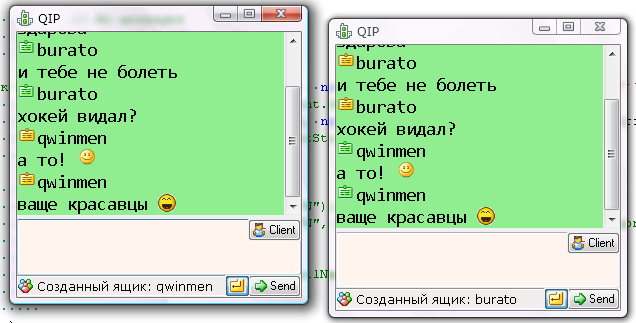
IntPtr lpOverlapped

);

#endregion

}

}

И вот как это работает: