Министерство науки и образования Российской Федерации

Пензенский государственный университет

Кафедра «Системы автоматизированного проектирования»

**Отчет**

По лабораторной работе №2

По курсу «Операционные системы реального времени»

На тему «Передача данных с использованием почтовых ящиков и ждущих таймеров, перехват и фильтрация информации с использованием системных перехватчиков»

Выполнил студент гр.20ВВС1

Горбунов Н.А.

Проверил:

Гудков П.А.

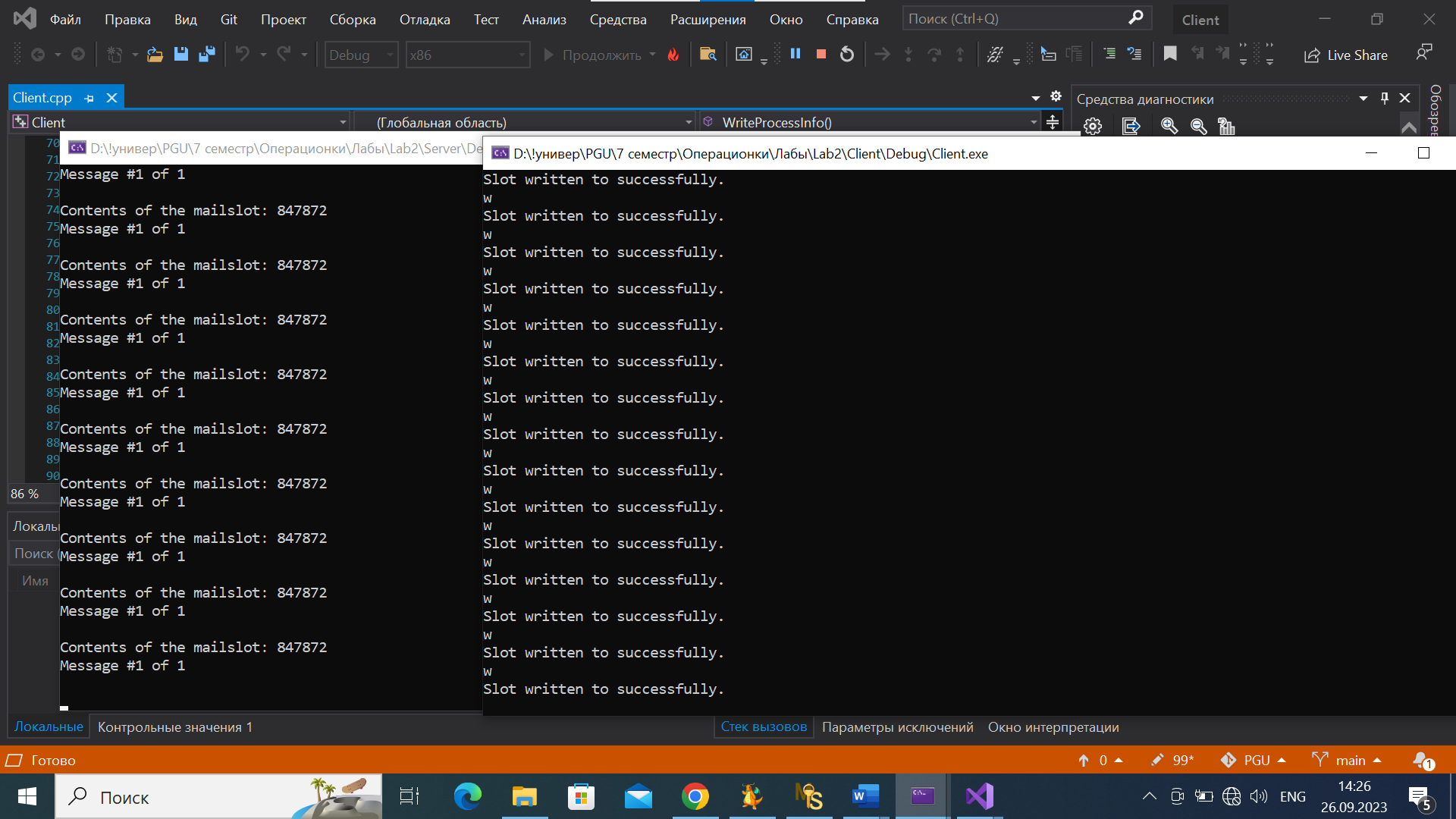
Пенза, 2023

**Цель:** Изучение построения клиент-серверного взаимодействия процессов с использованием почтовых ящиков. Изучение особенностей работы нитей при использовании ждущих таймеров.Изучить методику установки и снятия системных перехватчиков для эффективной реакции программы на действия пользователя.изучение методов и средств порождения процессов и нитей, изучение способов синхронизации процессов и нитей через ожидание окончания их выполнения.

Написали 2 программы -  
1) Сервер  
2) Клиент

Сервер запускается первый и ждёт, пока клиенты не запишут что-нибудь в слот.

Клиент запускается и ждет пока пользователь нажмет на кнопку отправки.



**Листинг Сервера:**

#include <iostream>

#include <Windows.h>

#include <Psapi.h>

#include <WinBase.h>

#include <stdio.h>

#include <tchar.h>

#include <strsafe.h>

HANDLE hSlot;

LPCTSTR SlotName = TEXT("\\\\.\\mailslot\\sample\_mailslot");

BOOL WINAPI MakeSlot(LPCTSTR lpszSlotName)

{

hSlot = CreateMailslot(lpszSlotName,

0, // no maximum message size

MAILSLOT\_WAIT\_FOREVER, // no time-out for operations

(LPSECURITY\_ATTRIBUTES)NULL); // default security

if (hSlot == INVALID\_HANDLE\_VALUE)

{

printf("CreateMailslot failed with %d\n", GetLastError());

return FALSE;

}

else printf("Mailslot created successfully.\n");

return TRUE;

}

BOOL ReadSlot()

{

DWORD cbMessage, cMessage, cbRead;

BOOL fResult;

LPTSTR lpszBuffer;

TCHAR achID[80];

DWORD cAllMessages;

HANDLE hEvent;

OVERLAPPED ov;

cbMessage = cMessage = cbRead = 0;

hEvent = CreateEvent(NULL, FALSE, FALSE, TEXT("ExampleSlot"));

if (NULL == hEvent)

return FALSE;

ov.Offset = 0;

ov.OffsetHigh = 0;

ov.hEvent = hEvent;

fResult = GetMailslotInfo(hSlot, // mailslot handle

(LPDWORD)NULL, // no maximum message size

&cbMessage, // size of next message

&cMessage, // number of messages

(LPDWORD)NULL); // no read time-out

if (!fResult)

{

printf("GetMailslotInfo failed with %d.\n", GetLastError());

return FALSE;

}

if (cbMessage == MAILSLOT\_NO\_MESSAGE)

{

//printf("Waiting for a message...\n");

return TRUE;

}

cAllMessages = cMessage;

while (cMessage != 0) // retrieve all messages

{

// Create a message-number string.

StringCchPrintf((LPTSTR)achID,

80,

TEXT("\nMessage #%d of %d\n"),

cAllMessages - cMessage + 1,

cAllMessages);

// Allocate memory for the message.

lpszBuffer = (LPTSTR)GlobalAlloc(GPTR,

lstrlen((LPTSTR)achID) \* sizeof(TCHAR) + cbMessage);

if (NULL == lpszBuffer)

return FALSE;

lpszBuffer[0] = '\0';

fResult = ReadFile(hSlot,

lpszBuffer,

cbMessage,

&cbRead,

&ov);

if (!fResult)

{

printf("ReadFile failed with %d.\n", GetLastError());

GlobalFree((HGLOBAL)lpszBuffer);

return FALSE;

}

// Concatenate the message and the message-number string.

StringCbCat(lpszBuffer,

lstrlen((LPTSTR)achID) \* sizeof(TCHAR) + cbMessage,

(LPTSTR)achID);

// Display the message.

\_tprintf(TEXT("Contents of the mailslot: %s\n"), lpszBuffer);

GlobalFree((HGLOBAL)lpszBuffer);

fResult = GetMailslotInfo(hSlot, // mailslot handle

(LPDWORD)NULL, // no maximum message size

&cbMessage, // size of next message

&cMessage, // number of messages

(LPDWORD)NULL); // no read time-out

if (!fResult)

{

printf("GetMailslotInfo failed (%d)\n", GetLastError());

return FALSE;

}

}

CloseHandle(hEvent);

return TRUE;

}

int main()

{

MakeSlot(SlotName);

while (TRUE)

{

ReadSlot();

Sleep(30);

}

}

**Листинг Клиента:**

#include <windows.h>

#include <stdio.h>

#include <iostream>

#include <Psapi.h>

#include <WinBase.h>

#include <string>

LPCTSTR SlotName = TEXT("\\\\.\\mailslot\\sample\_mailslot");

HANDLE hFile;

BOOL WriteSlot(HANDLE hSlot, LPCTSTR lpszMessage)

{

BOOL fResult;

DWORD cbWritten;

fResult = WriteFile(hSlot,

lpszMessage,

(DWORD)(lstrlen(lpszMessage) + 1) \* sizeof(TCHAR),

&cbWritten,

(LPOVERLAPPED)NULL);

if (!fResult)

{

printf("WriteFile failed with %d.\n", GetLastError());

return FALSE;

}

printf("Slot written to successfully.\n");

return TRUE;

}

BOOL WriteProcessInfo()

{

PROCESS\_MEMORY\_COUNTERS memCounter;

BOOL result = GetProcessMemoryInfo(GetCurrentProcess(),

&memCounter,

sizeof(memCounter));

//std::wstring testString =

std::wstring string = (std::to\_wstring(memCounter.PagefileUsage));

wchar\_t\* textShit = (wchar\_t\*)string.c\_str();

return WriteSlot(hFile, textShit);

}

int main()

{

hFile = CreateFile(SlotName,

GENERIC\_WRITE,

FILE\_SHARE\_READ,

(LPSECURITY\_ATTRIBUTES)NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,

(HANDLE)NULL);

if (hFile == INVALID\_HANDLE\_VALUE)

{

printf("CreateFile failed with %d.\n", GetLastError());

return FALSE;

}

while (true) {

char key;

std::cin >> key;

if (key == 'w')

{

WriteProcessInfo();

}

if (key == 'e') {

break;

}

}

CloseHandle(hFile);

return TRUE;

}

**Вывод:**

Научился взаимодействовать с почтовым ящиком.