ОДЕССКИЙ НАЦИОНАЛЬНЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИНСТИТУТ КОМПЬЮТЕРНЫХ СИСТЕМ

КАФЕДРА «КОМПЬЮТЕРНЫЕ ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ И СЕТИ»

Лабораторная работа №8

по дисциплине

«Системное программирование»

тема

«Межпроцессорное взаимодействие»

Выполнили: Човганский Б.Ю.

Жмыхов Д.С.

Группа: АЕ-144

Бригада: 5

Одесса 2017

**Цель работы:**

Разработать программу с использованием среды разработки Visual Studio или DevCpp. Получить практические навыки работы со схемой клиент-сервер.

**Задание на лабораторную работу:**

Вариант 5:

Реализовать операции со множествами целых чисел: объединение, пересечение, разность.

Задание:

Создать клиент-серверное приложение, где клиентское приложение выполняет интерактивное взаимодействие с пользователем: ввод данных с клавиатуры и вывод результатов на экран. А серверное приложение получает от клиента набор данных для обработки и посылает результат обратно клиенту.

Передача данных между клиентом и сервером должна осуществляться с использованием общей памяти или именованных каналов по выбору студента. Для каждого подключения клиента сервер должен создавать отдельный поток, в котором будет выполняться обработка запроса пользователя.

Передаваемые данные, должны быть размещены в буфере приема/отправки в формате, оптимальном для обработки процессором, т.е. если в задании речь идет об обработке:

* целых чисел - тип данных должен быть целочисленных (int или long);
* вещественных чисел - тип данных float или double;
* строковые значения – тип ASCIIZ-строка или массив символов со счетчиком байт.

**Код программы:**

1. **Клиент**

// Lab8Client.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include <windows.h>

#include <stdio.h>

#include <conio.h>

#define BUFSIZE 512

int main(void)

{

LPCWSTR pipename = \_T("\\\\.\\pipe\\mynamedpipe");

HANDLE hPipe;

char inputBuffer[BUFSIZE];

char chBuf[BUFSIZE];

BOOL fSuccess;

DWORD cbRead, cbWritten, dwMode;

char ch;

while (TRUE) {

if (INVALID\_HANDLE\_VALUE != (hPipe = CreateFile(pipename,

GENERIC\_READ | GENERIC\_WRITE, 0, NULL, OPEN\_EXISTING, 0, NULL)))

break;

// ERROR\_PIPE\_BUSY - это не ошибка

if (ERROR\_PIPE\_BUSY != GetLastError()) {

printf("Could not open pipe. Error 0x%08X\n", GetLastError());

return 0;

}

// Все экземпяры канала заняты, ждем

if (!WaitNamedPipe(pipename, NMPWAIT\_WAIT\_FOREVER)) {

printf("Could not open pipe.");

return 0;

}

}

printf("Connected to server\n");

dwMode = PIPE\_READMODE\_MESSAGE;

if (!(fSuccess = SetNamedPipeHandleState(hPipe, &dwMode, NULL, NULL))) {

printf("SetNamedPipeHandleState failed with error 0x%08X\n", GetLastError());

return 0;

}

do {

// Получение первого массива чисел

char oneNumber[BUFSIZE] = { '\0' };

int a[BUFSIZE / 4];

int counter1 = 0;

do {

try {

printf("Enter first array of integers (separate them by comas without spaces):\n");

scanf("%s", inputBuffer);

counter1 = 0;

oneNumber[0] = '\0';

for (int i = 0; i < strlen(inputBuffer)+1; i++) {

if (inputBuffer[i] != ',' && inputBuffer[i] !='\0') {

char x = inputBuffer[i];

strncat(oneNumber, &x,1);

}

else {

if (atoi(oneNumber) == NULL) throw(1);

a[counter1] = atoi(oneNumber);

counter1++;

oneNumber[0] = '\0';

}

}

}

catch (int error) {

printf("You have made mistake, try again\n");

continue;

}

break;

} while (true);

int b[BUFSIZE / 4];

int counter2 = 0;

// Получение второго массива чисел

do {

try {

printf("Enter second array of integers (as first):\n");

scanf("%s", inputBuffer);

counter2 = 0;

oneNumber[0] = '\0';

for (int i = 0; i < strlen(inputBuffer) + 1; i++) {

if (inputBuffer[i] != ',' && inputBuffer[i] != '\0') {

char x = inputBuffer[i];

strncat(oneNumber, &x, 1);

}

else {

if (atoi(oneNumber) == NULL) throw(1);

b[counter2] = atoi(oneNumber);

counter2++;

oneNumber[0] = '\0';

}

}

}

catch (int error) {

printf("You have made mistake, try again\n");

continue;

}

break;

} while (true);

//Получение операции

do {

printf("Select operation (U-union, I-interselection, R-relative component):");

ch = \_getch();

printf("%c\n\n", ch);

if(toupper(ch) == 'U' || toupper(ch) == 'I' || toupper(ch) == 'R') break;

} while (true);

//Склеивание данных

inputBuffer[0] = '\0';

itoa(counter1,inputBuffer,10);

strncat(inputBuffer, ",",1);

for (int i = 0; i < counter1; i++) {

itoa(a[i], oneNumber, 10);

strcat(inputBuffer, oneNumber);

strncat(inputBuffer, ",", 1);

}

itoa(counter2, oneNumber, 10);

strcat(inputBuffer, oneNumber);

strncat(inputBuffer, ",", 1);

for (int i = 0; i < counter2; i++) {

itoa(b[i], oneNumber, 10);

strcat(inputBuffer, oneNumber);

strncat(inputBuffer, ",", 1);

}

strncat(inputBuffer, &ch, 1);

// Шлем сообщение

if (!(fSuccess = WriteFile(

hPipe,

inputBuffer,

strlen(inputBuffer) + 1,

&cbWritten,

NULL)))

{

printf("WriteFile failed with error 0x%08X\n", GetLastError());

return 0;

}

if (!(fSuccess = ReadFile(

hPipe,

chBuf,

BUFSIZE,

&cbRead,

NULL)))

{

if (GetLastError() != ERROR\_MORE\_DATA) {

printf("ReadFile failed with error 0x%08X\n", GetLastError());

break;

}

else {

abort();

}

}

printf("Reply from server (A %c B): %s\n",toupper(ch), chBuf);

printf("\n\nTry again (Y/N)? ");

ch = \_getch();

printf("%c\n", ch);

} while ('N' != toupper(ch));

CloseHandle(hPipe);

return 0;

}

**2. Сервер**

// Lab8Server.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <windows.h>

#include <process.h>

#include <stdio.h>

#define BUFSIZE 4096

DWORD WINAPI thread\_proc(LPVOID);

VOID GetAnswerToRequest(LPTSTR, LPTSTR, LPDWORD);

int main(VOID)

{

BOOL fConnected;

HANDLE hPipe, hThread;

LPCWSTR pipename = \_T("\\\\.\\pipe\\mynamedpipe");

printf("Server is ready\n");

while (TRUE) {

hPipe = CreateNamedPipe(

pipename,

PIPE\_ACCESS\_DUPLEX,

PIPE\_TYPE\_MESSAGE |

PIPE\_READMODE\_MESSAGE |

PIPE\_WAIT,

PIPE\_UNLIMITED\_INSTANCES,

BUFSIZE,

BUFSIZE,

0,

NULL);

if (hPipe == INVALID\_HANDLE\_VALUE) {

printf("CreatePipe failed with error 0x%08X\n", GetLastError());

return 0;

}

fConnected = ConnectNamedPipe(hPipe, NULL) ? TRUE

: (GetLastError() == ERROR\_PIPE\_CONNECTED);

if (fConnected) {

printf("Got a new connection.\n");

if (NULL == (hThread = (HANDLE)\_beginthreadex(NULL, 0, (\_beginthreadex\_proc\_type)thread\_proc, (void\*)hPipe, 0, NULL))) {

printf("\_beginthreadex failed with error %d\n", errno);

return 0;

}

else

CloseHandle(hThread);

}

else

CloseHandle(hPipe);

}

return 0;

}

DWORD WINAPI thread\_proc(LPVOID lpvParam)

{

char chRequest[BUFSIZE];

char chReply[BUFSIZE] = {'\0'};

char buff[BUFSIZE] = { '\0' };

DWORD cbBytesRead=0, cbReplyBytes=0, cbWritten;

BOOL fSuccess;

HANDLE hPipe;

hPipe = (HANDLE)lpvParam;

while (TRUE)

{

// Читаем запрос от клиента

fSuccess = ReadFile(

hPipe,

chRequest,

sizeof(chRequest),

&cbBytesRead,

NULL);

if (!fSuccess || cbBytesRead == 0)

break;

// Считываем данные

char oneNumber[BUFSIZE] = { '\0' };

char operation;

int a[BUFSIZE / 4],b[BUFSIZE / 4];

int asize, bsize;

int counter = 0;

for (int i = 0; i < strlen(chRequest) + 1; i++) {

if (chRequest[i] != ',' && chRequest[i] != '\0') {

char x = chRequest[i];

strncat(oneNumber, &x, 1);

}

else {

if (counter == 0) asize = atoi(oneNumber);

else if (counter == asize+1) bsize = atoi(oneNumber);

else if (counter < asize+1) a[counter-1]= atoi(oneNumber);

else if (counter < bsize+asize+2) b[counter - asize - 2] = atoi(oneNumber);

else operation = chRequest[i-1];

counter++;

oneNumber[0] = '\0';

}

}

// Расчитываем ответ

chReply[0] = '\0';

bool flag;

switch (toupper(operation))

{

case 'U':

flag = true;

for (int i = 0; i < asize; i++) {

itoa(a[i], buff, 10);

cbReplyBytes += strlen(buff) + 1;

strcat(chReply, buff);

strncat(chReply, ",",1);

}

for (int i = 0; i < bsize; i++) {

for (int j = 0; j < asize; j++) if (b[i] == a[j]) flag = false;

if (flag) {

itoa(b[i], buff, 10);

cbReplyBytes += strlen(buff) + 1;

strcat(chReply, buff);

strncat(chReply, ",", 1);

}

flag = true;

}

break;

case 'I':

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

for (int j = 0; j < bsize; j++) {

if (a[i] == b[j]) {

itoa(a[i], buff, 10);

cbReplyBytes += strlen(buff) + 1;

strcat(chReply, buff);

strncat(chReply, ",", 1);

}

}

break;

case 'R':

flag = true;

for (int i = 0; i < asize; i++) {

for (int j = 0; j < bsize; j++) if (a[i] == b[j]) flag = false;

if (flag) {

itoa(a[i], buff, 10);

cbReplyBytes += strlen(buff) + 1;

strcat(chReply, buff);

strncat(chReply, ",", 1);

}

flag = true;

}

break;

default:

break;

}

chReply[strlen(chReply) - 1] = '\0';

// Пишем ответ в канал

fSuccess = WriteFile(

hPipe,

chReply,

cbReplyBytes,

&cbWritten,

NULL);

if (!fSuccess || cbReplyBytes != cbWritten)

break;

}

// Выполним принудительный сброс буферов на сторону клиента

FlushFileBuffers(hPipe);

DisconnectNamedPipe(hPipe);

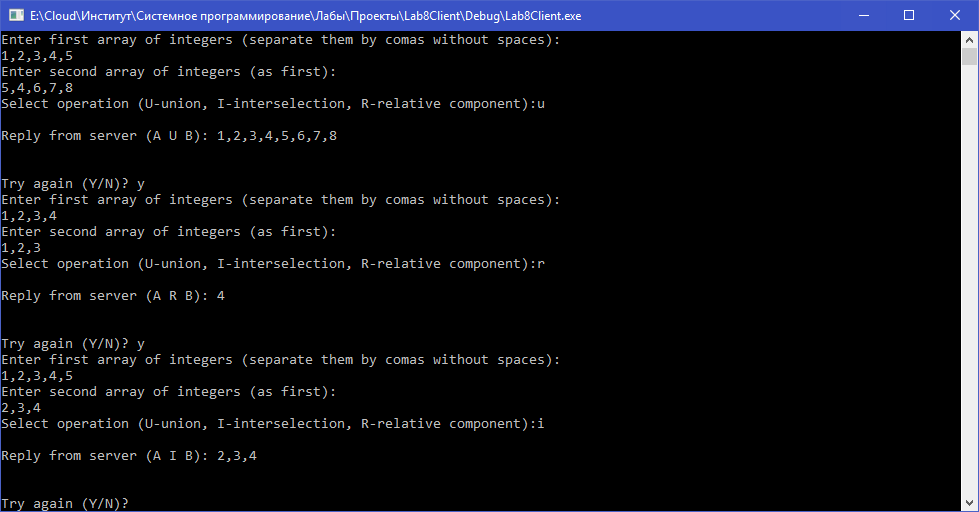
CloseHandle(hPipe);

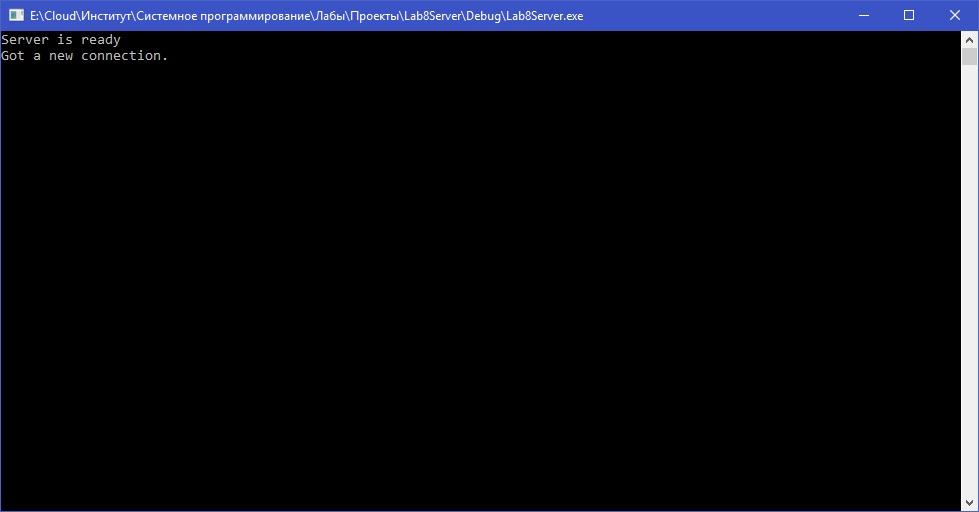
printf("Client terminated\n");

return 1;

}

**Работа программы:**





**Вывод:** практические навыки клиент-серверной схемы помогут при создании бэкендов сайтов или приложений, так что работа выполнена успешно.