МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт КНТ

Факультет ФИСП

Кафедра ПИ им. Л.П. Фельдмана

Лабораторная работа №6

Тема: Организация межпроцессного взаимодействия

Курс: Системное программирование

Выполнил

ст. гр. ПИ-18б

Моргунов А.Г.

Проверил

Филипишин Д.А.

Донецк – 2021

**Задание**

Задание. Разработать две Windows-программы на языке С.:

1) программа-клиент, в которой производится запуск программы-сервера, формирование матрицы, передача её серверу с помощью средства IPC, получение от сервера преобразованной матрицы, вывод на экран обеих матриц, передача через буфер обмена указанного в варианте объекта и отображение его на экране;

2) программа-сервер, в которой производится получение от программы-клиента исходной матрицы, её преобразование согласно заданию по варианту, передача клиенту преобразованной матрицы с помощью заданного средства IPC, получение из буфера обмена указанного объекта и отображение его в окне.

Матрица, с которой выполняется работа, имеет размер 8х8, а её элементы – это случайные числа в диапазоне от -9 до 9.

Запуск сервера выполняется с помощью функции CreateProcess(), а имя исполняемого файла для него выбирается пользователем с помощью блока диалога открытия файла.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № варианта | Средства IPC для прямой и обратной передачи данных | Обработка массива, выполняема сервером | Переда данных через буфер обмена | Вид передачи |
| 5 | Анонимный канал  Именованный канал | В матрице B поменять местами столбцы, содержащие минимальный и максимальный отрицательные элементы | Текст из 4-х строк | Прямая |

**Листинг клиентского приложения**

// SP\_lab\_6.cpp : Определяет точку входа для приложения.

//

#include "framework.h"

#include "SP\_lab\_6.h"

#include <commdlg.h>

#include <string>

#include <string>

#include <tchar.h>

#include <array>

#include <exception>

#include <windowsx.h>

#include <commctrl.h>

#include <stdexcept>

#include <cmath>

#include <commdlg.h>

#define MAX\_LOADSTRING 100

HWND main\_window;

constexpr int dimension = 8;

constexpr int size{ dimension \* dimension };

using String = std::basic\_string<TCHAR>;

template <typename CHAR\_TYPE>

std::basic\_string<CHAR\_TYPE> ToStringHelper(int);

template <>

std::string ToStringHelper<char>(int value) {

return std::to\_string(value);

}

template <>

std::wstring ToStringHelper<wchar\_t>(int value) {

return std::to\_wstring(value);

}

String ToString(int value) {

return ToStringHelper<TCHAR>(value);

}

struct WindowSize {

int width;

int height;

};

WindowSize GetWindowSize(HWND window\_handle) {

RECT window\_rect;

GetClientRect(window\_handle, &window\_rect);

WindowSize result;

result.width = window\_rect.right - window\_rect.left;

result.height = window\_rect.bottom - window\_rect.top;

return result;

}

int RandomInt(int range\_min, int range\_max) {

return rand() % (range\_max - range\_min + 1) + range\_min;

}

template <size\_t kSize>

void DrawArray(const std::array<int, kSize>& arr, RECT rect) {

HDC hdc = GetDC(main\_window);

String numbers;

for (size\_t i = 0; i < dimension; ++i) {

numbers.clear();

for (size\_t j = 0; j < dimension; ++j) {

numbers += std::to\_wstring(arr[i \* dimension + j]) + L", ";

}

int height = DrawText(hdc, numbers.data(), (int)numbers.size(), &rect, 0);

OffsetRect(&rect, 0, height);

}

ReleaseDC(main\_window, hdc);

}

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

WCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

WCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: Разместите код здесь.

// Инициализация глобальных строк

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_SPLAB6, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_SPLAB6));

MSG msg;

// Цикл основного сообщения:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

//

// ФУНКЦИЯ: MyRegisterClass()

//

// ЦЕЛЬ: Регистрирует класс окна.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_SPLAB6));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_SPLAB6);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// ЦЕЛЬ: Сохраняет маркер экземпляра и создает главное окно

//

// КОММЕНТАРИИ:

//

// В этой функции маркер экземпляра сохраняется в глобальной переменной, а также

// создается и выводится главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить маркер экземпляра в глобальной переменной

main\_window = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!main\_window)

{

return FALSE;

}

ShowWindow(main\_window, nCmdShow);

UpdateWindow(main\_window);

return TRUE;

}

//

// ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// ЦЕЛЬ: Обрабатывает сообщения в главном окне.

//

// WM\_COMMAND - обработать меню приложения

// WM\_PAINT - Отрисовка главного окна

// WM\_DESTROY - отправить сообщение о выходе и вернуться

//

//

HANDLE hPipe;

DWORD cbRead;

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case IDM\_COPY\_DATA: {

OpenClipboard(0);

EmptyClipboard();

constexpr auto message = "Line1\r\Line2\r\Line3\r\nLine4\r\n";

const auto size = strlen(message) + 1;

HGLOBAL h = GlobalAlloc(GMEM\_MOVEABLE, size);

memcpy(GlobalLock(h), message, size);

GlobalUnlock(h);

SetClipboardData(CF\_TEXT, h);

CloseClipboard();

break;

}

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_START\_SERVER: {

hPipe = CreateNamedPipe(L"\\\\.\\pipe\\MyPipe", PIPE\_ACCESS\_DUPLEX, PIPE\_TYPE\_BYTE | PIPE\_WAIT, 1, 1024, 1024, 0, NULL);

if (hPipe == INVALID\_HANDLE\_VALUE)

MessageBox(NULL, L"INVALID VALUE", L" ", MB\_OK);

wchar\_t szFile[100];

wchar\_t szFileTitle[100];

OPENFILENAME ofn;

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = NULL;

ofn.lpstrFile = szFile;

ofn.lpstrFile[0] = '\0';

ofn.nMaxFile = sizeof(szFile);

ofn.lpstrFilter = L"All\0\*.\*\0";

ofn.nFilterIndex = 1;

ofn.lpstrFileTitle = szFileTitle;

ofn.nMaxFileTitle = sizeof(szFileTitle);

ofn.lpstrInitialDir = NULL;

ofn.Flags = OFN\_PATHMUSTEXIST | OFN\_FILEMUSTEXIST;

if (!GetOpenFileName(&ofn)) {

break;

}

//MessageBox(NULL, ofn.lpstrFile, L"File name", MB\_OK);

STARTUPINFO si;

PROCESS\_INFORMATION pi;

ZeroMemory(&si, sizeof(si));

si.cb = sizeof(si);

ZeroMemory(&pi, sizeof(pi));

String cmdline(ofn.lpstrFile);

String str(300, 0);

SECURITY\_ATTRIBUTES sa;

sa.nLength = sizeof(sa);

sa.bInheritHandle = TRUE;

sa.lpSecurityDescriptor = nullptr;

HANDLE r\_pipe, w\_pipe;

if (!CreatePipe(&r\_pipe, &w\_pipe, &sa, 0)) {

std::abort();

}

int len = swprintf\_s(str.data(), str.size(), L"%s %x %x", ofn.lpstrFileTitle, hWnd, r\_pipe);

str.resize(len);

SetWindowText(main\_window, str.data());

if (!CreateProcess(cmdline.data(), str.data(),

NULL, // Process handle not inheritable

NULL, // Thread handle not inheritable

TRUE, // Set handle inheritance to TRUE

0, // No creation flags

NULL, // Use parent's environment block

NULL, // Use parent's starting directory

&si, // Pointer to STARTUPINFO structure

&pi)) {

std::abort();

}

std::array<int, size> arr;

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

arr[i] = RandomInt(-9, 9);

}

WriteFile(w\_pipe, arr.data(), size \* sizeof(int), nullptr, nullptr);

CloseHandle(w\_pipe);

std::array<int, size> server\_arr{0};

Sleep(100);

auto read\_success = ReadFile(hPipe, server\_arr.data(), size \* sizeof(int), &cbRead, NULL);

DrawArray(arr, { 0, 0, 200, 200 });

DrawArray(server\_arr, { 200, 0, 400, 200 });

break;

}

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

// TODO: Добавьте сюда любой код прорисовки, использующий HDC...

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

// Обработчик сообщений для окна "О программе".

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

**Листинг серверного приложения**

// SP\_lab\_6\_server.cpp : Определяет точку входа для приложения.

//

#include "framework.h"

#include "SP\_lab\_6\_server.h"

#include <string>

#include <tchar.h>

#include <exception>

#include <windowsx.h>

#include <commctrl.h>

#include <stdexcept>

#include <array>

#include <algorithm>

#include <cmath>

#include <sstream>

#define MAX\_LOADSTRING 100

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

WCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

WCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

HWND main\_window;

HANDLE r\_pipe;

HWND client\_window;

std::array<int, 64> client\_array, server\_solution;

using String = std::basic\_string<TCHAR>;

template <typename CHAR\_TYPE>

std::basic\_string<CHAR\_TYPE> ToStringHelper(int);

template <>

std::string ToStringHelper<char>(int value) {

return std::to\_string(value);

}

template <>

std::wstring ToStringHelper<wchar\_t>(int value) {

return std::to\_wstring(value);

}

String ToString(int value) {

return ToStringHelper<TCHAR>(value);

}

template <size\_t kSize>

void DrawArray(const std::array<int, kSize>& arr, RECT rect) {

HDC hdc = GetDC(main\_window);

String numbers;

constexpr auto dimension = 8;

for (size\_t i = 0; i < dimension; ++i) {

numbers.clear();

for (size\_t j = 0; j < dimension; ++j) {

numbers += std::to\_wstring(arr[i \* dimension + j]) + L", ";

}

int height = DrawText(hdc, numbers.data(), (int)numbers.size(), &rect, 0);

OffsetRect(&rect, 0, height);

}

ReleaseDC(main\_window, hdc);

}

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

using Array = std::array<int, 64>;

void SwapColumn(Array& arr, size\_t a, size\_t b) {

for (size\_t j = 0; j < 8; ++j) {

std::swap(arr[a + j \* 8], arr[b + j \* 8]);

}

}

Array FindColumnAndSwap(Array& arr) {

int min\_neg = 0;

int max\_neg = -10;

int min\_neg\_col = 0;

int max\_neg\_col = 0;

for (size\_t i = 0; i < 8; ++i) {

for (size\_t j = 0; j < 8; ++j) {

if (arr[i + j \* 8] < min\_neg) {

min\_neg = arr[i + j \* 8];

min\_neg\_col = i;

}

if (arr[i + j \* 8] > max\_neg && arr[i + j \* 8] < 0 && i != min\_neg\_col) {

max\_neg = arr[i + j \* 8];

max\_neg\_col = i;

}

}

}

SwapColumn(arr, min\_neg\_col, max\_neg\_col);

return arr;

}

int RandomInt(int range\_min, int range\_max) {

return rand() % (range\_max - range\_min + 1) + range\_min;

}

//

// ФУНКЦИЯ: MyRegisterClass()

//

// ЦЕЛЬ: Регистрирует класс окна.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_SPLAB6SERVER));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_SPLAB6SERVER);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// ЦЕЛЬ: Сохраняет маркер экземпляра и создает главное окно

//

// КОММЕНТАРИИ:

//

// В этой функции маркер экземпляра сохраняется в глобальной переменной, а также

// создается и выводится главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить маркер экземпляра в глобальной переменной

main\_window = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!main\_window)

{

return FALSE;

}

ShowWindow(main\_window, nCmdShow);

UpdateWindow(main\_window);

return TRUE;

}

//

// ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// ЦЕЛЬ: Обрабатывает сообщения в главном окне.

//

// WM\_COMMAND - обработать меню приложения

// WM\_PAINT - Отрисовка главного окна

// WM\_DESTROY - отправить сообщение о выходе и вернуться

//

//

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_CREATE:

//DrawArray(client\_array, { 0, 0, 200, 200 });

//DrawArray(server\_solution, { 200, 0, 200, 200 });

//break;

case WM\_RBUTTONUP:

DrawArray(client\_array, { 0, 0, 200, 200 });

DrawArray(server\_solution, { 200, 0, 400, 200 });

break;

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case IDM\_PASTE\_DATA: {

OpenClipboard(hWnd);

HANDLE hglb = GetClipboardData(CF\_TEXT);

void\* lptstr = GlobalLock(hglb);

int len = strlen((char\*)lptstr);

SetWindowTextA(hWnd, (char\*)lptstr);

HDC hdc = GetDC(hWnd);

RECT rect{ 0, 200, 200, 400 };

DrawTextA(hdc, (char\*)lptstr, len, &rect, 0);

ReleaseDC(hWnd, hdc);

GlobalUnlock(hglb);

CloseClipboard();

break;

}

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

// TODO: Добавьте сюда любой код прорисовки, использующий HDC...

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

HANDLE hPipe;

// Обработчик сообщений для окна "О программе".

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

DWORD cbWritten;

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

swscanf\_s(lpCmdLine, L"%x %x", &client\_window, &r\_pipe);

//MessageBox(NULL, lpCmdLine, L"Info", MB\_OK);

constexpr auto dimension = 8;

constexpr auto size = dimension \* dimension;

Array arr;

if (!ReadFile(r\_pipe, arr.data(), size \* sizeof(int), nullptr, nullptr)) {

std::wstringstream ss;

ss << GetLastError();

String str = ss.str();

MessageBox(NULL, str.data(), L"Error", MB\_OK);

}

CloseHandle(r\_pipe);

client\_array = arr;

FindColumnAndSwap(arr);

server\_solution = arr;

hPipe = CreateFile(L"\\\\.\\pipe\\MyPipe", GENERIC\_ALL, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hPipe == INVALID\_HANDLE\_VALUE)

MessageBox(NULL, L"INVALID VALUE", L" ", MB\_OK);

auto written = WriteFile(hPipe, server\_solution.data(), size \* sizeof(int), &cbWritten, NULL);

// Инициализация глобальных строк

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_SPLAB6SERVER, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_SPLAB6SERVER));

MSG msg;

// Цикл основного сообщения:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int)msg.wParam;

}

**Демонстрация работы приложения приведена в видео SP\_lab\_6.mp4**