Федеральное государственное бюджетное образовательное учреждение высшего образования «Национальный исследовательский университет «МЭИ»

Институт информационных и вычислительных технологий

Кафедра Управления и интеллектуальных технологий

**Отчёт по лабораторной работе № 2**

# «Межпроцессное взаимодействие»

**По курсу «Системное программное обеспечение»**

# Выполнил студент группы А-01-19:

# Китайский А.С.

# Проверил:

# Козлюк Д.А.

# Москва 2022

# Вариант 4. Запуск дочернего процесса с перенаправлением потоков ввода-вывода

# Написать программу для обучения пользователя командной строки вежливости с использованием перенаправления потоков ввода и вывода через анонимные каналы.

# 1. Создать два анонимных канала функцией CreatePipe().

# 2. Запустить процесс cmd.exe, соединив один из созданных каналов с потоком ввода, а другой канал — с потоками вывода и ошибок дочернего процесса.

# 3. Считать из канала, связанного с потоком вывода дочернего процесса, все данные функцией ReadFile() и вывести их на экран.

# 4. Запросить у пользователя полную строку-команду.

# 5. Если введенная строка не начинается со слова «please» (до первого пробела), уведомить об этом пользователя и перейти к пункту 4.

# 6. Если введена строка «thanks», остановить дочерний процесс функцией TerminateProcess(), закрыть анонимные каналы функцией CloseHandle() и завершить работу программы.

# 7. Записать в канал, связанный с потоком ввода дочернего процесса, оставшуюся часть команды и символ перевода строки '\n'.

# 8. Перейти к пункту 3 (цикл).

#include <iostream>

#include <windows.h>

using namespace std;

HANDLE input\_pipe\_read\_end;

HANDLE input\_pipe\_write\_end;

HANDLE output\_pipe\_read\_end;

HANDLE output\_pipe\_write\_end;

DWORD bytes\_read;

void CreateChildProcess();

void ReadInvite();

int main()

{

SECURITY\_ATTRIBUTES attributes;

// настройка параметров структуры SECURITY\_ATTRIBUTES

attributes.nLength = sizeof(SECURITY\_ATTRIBUTES); // размер структуры в байтах

attributes.bInheritHandle = TRUE; // при создании нового процесса дескриптор канала наследуется

attributes.lpSecurityDescriptor = NULL; // присвоение объекту дескриптора по умолчанию

// создание анонимного канала ввода

CreatePipe(&input\_pipe\_read\_end, &input\_pipe\_write\_end, &attributes, 0);

// создание анонимного канала вывода

CreatePipe(&output\_pipe\_read\_end, &output\_pipe\_write\_end, &attributes, 0);

// разрешение наследования дескрипторов-концов, которые остаются в текущем процессе

SetHandleInformation(input\_pipe\_write\_end, HANDLE\_FLAG\_INHERIT, 0);

SetHandleInformation(output\_pipe\_read\_end, HANDLE\_FLAG\_INHERIT, 0);

CreateChildProcess();

ReadInvite();

while (true)

{

const char PLEASE[] = "please";

char\* input = NULL;

char buffer[256];

while (!input)

{

fgets(buffer, sizeof(buffer), stdin); // получает строку из потока

if (!strncmp(buffer, "thanks", 6)) // сравнивает символы buffer с "thanks"

{

break; //завершает цикл в случае равенства строк

}

else if (strncmp(buffer, "please", 6)) // сравнивает символы buffer с "please"

{

fprintf(stderr, "Please ask politely!\n>"); // печатает строку в поток

}

else

{

input = buffer + sizeof(PLEASE); //input указывает на начало непосредственно команды

}

}

if (input != NULL)

{

WriteFile(input\_pipe\_write\_end, input, strlen(input), NULL, NULL); // записывает данные в канал дочернего процесса

ReadInvite();

}

else

{

break;

}

}

return 0;

}

void CreateChildProcess()

// создание дочернего процесса, который использует ранее созданные каналы

{

wchar\_t cmd[] = L"cmd.exe";

PROCESS\_INFORMATION piProcInfo;

STARTUPINFO siStartInfo;

// настройка параметров структуры STARTUPINFO

ZeroMemory(&siStartInfo, sizeof(siStartInfo)); // заполняет блок памяти нулями, заданный указателем на начало и длиной

siStartInfo.cb = sizeof(siStartInfo); // размер структуры в байтах

siStartInfo.hStdInput = input\_pipe\_read\_end; // дескриптор ввода для процесса

siStartInfo.hStdOutput = output\_pipe\_write\_end; // дескриптор вывода для процесса

siStartInfo.hStdError = output\_pipe\_write\_end; // дескриптор ошибки для процесса

siStartInfo.dwFlags |= STARTF\_USESTDHANDLES; // дескрипторы наследуются

// запуск дочернего процесса

if (!CreateProcess(

NULL, // нет имени модуля

cmd, // командная строка

NULL, // дескриптор процесса не наследуется

NULL, // дескриптор потока не наследуется

TRUE, // дескриптор наследуется

0, // нет флагов создания

NULL, // используется родительский блок конфигурации

NULL, // используется начальный родительский каталог

&siStartInfo, // указатель на структуру STARTUPINFO

&piProcInfo)) // указатель на структуру PROCESS\_INFORMATION

{

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

}

// закрытие дескрипторов дочернего процесса и его потока

CloseHandle(piProcInfo.hProcess);

CloseHandle(piProcInfo.hThread);

// закрытие дескрипторов каналов

CloseHandle(input\_pipe\_read\_end);

CloseHandle(output\_pipe\_write\_end);

}

void ReadInvite()

// отображения приглашений командной строки, чтения команд и выдачи результатов

{

char chBuf[64];

do

{

ReadFile(output\_pipe\_read\_end, chBuf, sizeof(chBuf), &bytes\_read, NULL); //считывает данные из канала дочернего процесса

fwrite(chBuf, bytes\_read, 1, stdout); // записывает данные в поток

} while ((bytes\_read != 0) && (chBuf[bytes\_read - 1] != '>'));

}

Пример работы программы: