#include <iostream>

#include <clocale>

#include <ctime>

#include "ErrorMsgText.h"

#include "Windows.h"

#define NAME L"\\\\.\\pipe\\Tube"

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus"); установка русской локали

HANDLE sH; - объявление переменной типа HANDLE, которая будет хранить дескриптор именованного канала.

DWORD lp; - объявление переменной типа DWORD, которая будет хранить количество байт, прочитанных/записанных при использовании функций ReadFile/WriteFile.

char buf[50]; - объявление буфера для передачи данных по именованному каналу.

блок кода, который может выбросить исключение в случае возникновения ошибки при работе с именованным каналом. Если исключение будет выброшено, то оно будет обработано в блоке catch, где переменная ErrorPipeText будет содержать сообщение об ошибке

try {

cout << "ServerNP\n\n"; - вывод на экран сообщения "ServerNP".

if ((sH = CreateNamedPipe(NAME,

PIPE\_ACCESS\_DUPLEX, PIPE\_TYPE\_MESSAGE | PIPE\_WAIT,

1, NULL, NULL, INFINITE, NULL)) == INVALID\_HANDLE\_VALUE) {

throw SetPipeError("CreateNamedPipe: ", GetLastError());

} - создание именованного канала с именем NAME при помощи функции CreateNamedPipe(). Если функция CreateNamedPipe() вернула значение INVALID\_HANDLE\_VALUE, то будет выброшено исключение при помощи функции SetPipeError(), которая принимает строку с текстом ошибки и код ошибки, полученный при вызове функции GetLastError().

Параметры функции

- NAME - имя именованного канала;

- PIPE\_ACCESS\_DUPLEX - режим доступа к каналу (для чтения и записи);

- PIPE\_TYPE\_MESSAGE | PIPE\_WAIT - тип канала (сообщения) и режим ожидания (блокирующий);

- 1 - максимальное количество экземпляров канала;

- NULL, NULL - размеры входного и выходного буферов (NULL означает использование по умолчанию);

- INFINITE - таймаут ожидания соединения (бесконечный);

NULL - безопасность (используется по умолчанию).

cout << "Waiting for connect...\n";

while (true) {

if (!ConnectNamedPipe(sH, NULL)) {

throw SetPipeError("ConnectNamedPipe: ", GetLastError());

} - происходит ожидание подключения клиента при помощи функции ConnectNamedPipe, которая блокирует выполнение программы до тех пор, пока клиент не подключится к именованному каналу.

while (true) {

if (ReadFile(sH, buf, sizeof(buf), &lp, NULL)) { - Далее внутри бесконечного цикла считываются данные из канала при помощи функции ReadFile, которая блокируется до тех пор, пока в канале не появятся данные. Полученные данные сохраняются в буфер buf.

if (strcmp(buf, "STOP") == 0) {

cout << endl;

break;

} - Если в канал была передана команда "STOP", то цикл чтения данных из канала прерывается, и программа переходит к следующему циклу ожидания клиента.

cout << buf << endl;

if (WriteFile(sH, buf, sizeof(buf), &lp, NULL)) {

if (strstr(buf, "ClientNPct")) {

break;

}

} - В противном случае данные выводятся на экран при помощи cout, и если в полученных данных встречается подстрока "ClientNPct", то программа выходит из цикла чтения данных из канала. Затем данные повторно записываются в канал при помощи функции WriteFile. Если запись прошла успешно, то программа проверяет, необходимо ли прервать цикл записи, и в случае необходимости, программа выходит из цикла записи.

else {

throw SetPipeError("WriteFile: ", GetLastError());

}

}

else {

throw SetPipeError("ReadFile: ", GetLastError());

}

}

if (!DisconnectNamedPipe(sH)) { - закрываем соединение с клиентом с помощью функции DisconnectNamedPipe

throw SetPipeError("DisconnectNamedPipe: ", GetLastError());

}

}

if (!CloseHandle(sH)) { - Закрываем созданный именованный канал с помощью функции CloseHandle.

throw SetPipeError("CloseHandle: ", GetLastError());

}

system("pause");

}

catch (string ErrorPipeText) {

cout << endl << ErrorPipeText;

}

}

Эта программа создает именованный канал и ожидает соединения от клиента. После установления соединения она считывает данные от клиента и отправляет их обратно. Внутренний цикл нужен для поддержания соединения с клиентом, пока он не передаст строку, содержащую "ClientNPct". После получения этой строки соединение разрывается.