#define STOP "STOP"

#define NAME L[\\\\Across\\pipe\\Tube](file:///\\\\Across\\pipe\\Tube) Задаются макросы для имени канала и для сообщения, которым клиент будет завершать работу канала.

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

HANDLE cH; // дескриптор клиентского конца канала

DWORD mode = PIPE\_READMODE\_MESSAGE; // режим работы канала

DWORD lp; // переменная для получения информации о количестве байтов, переданных через канал

char ibuf[50] = "Hello from Client", // буфер для чтения данных

obuf[50]; // буфер для записи данных

try {

cout << "ClientNPt\n\n"; // выводим информационное сообщение на экран

if ((cH = CreateFile(NAME, GENERIC\_READ | GENERIC\_WRITE, // пытаемся создать файл-канал

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE,

NULL, OPEN\_EXISTING, NULL, NULL)) == INVALID\_HANDLE\_VALUE) {

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

}

if (!SetNamedPipeHandleState(cH, &mode, NULL, NULL)) { // устанавливаем режим работы канала

cout << GetLastError();

throw SetPipeError("SetNamedPipeHandleState: ", GetLastError()); // в случае ошибки выбрасываем исключение

}

}

int countMessage;

cout << "Number of messages: ";

cin >> countMessage; // запрашиваем у пользователя количество сообщений для отправки

for (int i = 1; i <= countMessage; i++) {

string obufstr = "Hello from ClientNPt " + to\_string(i); // формируем сообщение для отправки на сервер

strcpy\_s(obuf, obufstr.c\_str()); // копируем сообщение в буфер для записи данных

if (!TransactNamedPipe(cH, obuf, sizeof(obuf), ibuf, sizeof(ibuf), &lp, NULL)) { // отправляем сообщение на сервер и ожидаем ответа

throw SetPipeError("TransactNamedPipe: ", GetLastError()); // в случае ошибки выбрасываем исключение

}

cout << ibuf << endl; // выводим полученный ответ на экран

}

if (!WriteFile(cH, STOP, sizeof(STOP), &lp, NULL)) { // отправляем серверу сигнал о завершении передачи данных

throw SetPipeError("WriteFile: ", GetLastError()); // в случае ошибки выбрасываем исключение

}

if (!CloseHandle(cH)) { // закрываем дескриптор клиентского конца канала

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

}

system("pause");

}

catch (string ErrorPipeText) {

cout << endl << ErrorPipeText;

}

}