#include <iostream>

#include "windows.h"

#include <string>

using namespace std;

string GetErrorMail(int code)

{

string msgText = "";

switch (code)

{

case WSAEINTR: msgText = "WSAEINTR"; break;

case WSAEACCES: msgText = "WSAEACCES"; break;

case WSAEFAULT: msgText = "WSAEFAULT"; break;

default: msgText = "Error";

};

return msgText;

} Эта функция возвращает текстовое описание ошибки, код которой передан в качестве параметра.

string SetErrorMail(string msgText, int code)

{

return msgText + GetErrorMail(code);

} Эта функция используется для формирования текста ошибки.

int main()

{

setlocale(LC\_ALL, "rus"); Функция setlocale устанавливает локаль приложения.

try {

HANDLE serverMailSlot;

int i = 0;

double t1, t2;

Здесь объявляются переменные: HANDLE serverMailSlot - дескриптор создаваемого сервером почтового ящика; int i - счетчик полученных сообщений; double t1, t2 - переменные для замера времени передачи сообщений.

SECURITY\_DESCRIPTOR pSD;

SECURITY\_ATTRIBUTES SA;

InitializeSecurityDescriptor(&pSD, SECURITY\_DESCRIPTOR\_REVISION);

SetSecurityDescriptorDacl(&pSD, true, NULL, false);

SA.nLength = sizeof(SA);

SA.lpSecurityDescriptor = &pSD;

SA.bInheritHandle = true;

Этот код создает объект безопасности и структуру атрибутов безопасности, которые будут использоваться при создании почтового ящика с помощью функции CreateMailslot.

SECURITY\_DESCRIPTOR pSD; - создается объект безопасности, который представляет собой структуру SECURITY\_DESCRIPTOR, содержащую информацию о безопасности.

SECURITY\_ATTRIBUTES SA; - создается структура, которая содержит атрибуты безопасности.

InitializeSecurityDescriptor(&pSD, SECURITY\_DESCRIPTOR\_REVISION); - инициализирует объект безопасности, передавая указатель на объект безопасности pSD и версию безопасности SECURITY\_DESCRIPTOR\_REVISION.

SetSecurityDescriptorDacl(&pSD, true, NULL, false); - устанавливает дескриптор управления доступом (DACL) в объекте безопасности. Функция устанавливает атрибут безопасности, указывающий, что объект защищен дескриптором управления доступом.

SA.nLength = sizeof(SA); - устанавливает длину структуры атрибутов безопасности в байтах.

SA.lpSecurityDescriptor = &pSD; - устанавливает указатель на объект безопасности в структуре атрибутов безопасности.

SA.bInheritHandle = true; - устанавливает атрибут безопасности, указывающий, что процесс может наследовать дескриптор ящика при запуске других процессов.

//возвращает дескриптор почтового ящика

if ((serverMailSlot = CreateMailslot(L"\\\\.\\mailslot\\BOX", // символическое имя ящика

/\*300 ,\*/500, //максимальную длину сообщения равную 300 байт

MAILSLOT\_WAIT\_FOREVER, /\*180000,\*/ //бесконечный интервал ожидания для функции ReadFile

&SA)) == INVALID\_HANDLE\_VALUE) //атрибуты безопасности

throw SetErrorMail("CreateMailslot", GetLastError());

Эти строки используются для создания структуры SECURITY\_ATTRIBUTES с заданными атрибутами безопасности. В данном случае, устанавливается разрешение на чтение и запись почтового ящика для всех пользователей.

cout << "Hello i am server" << endl;

char readBuf[50];

DWORD readMsg;

Затем в бесконечном цикле выполняется операция чтения из почтового ящика с помощью функции ReadFile(). Дескриптор почтового ящика передается в качестве первого аргумента. Если операция чтения прошла успешно, то данные записываются в буфер readBuf, а количество прочитанных байтов - в переменную readMsg.

do {

i++;

if (!ReadFile(serverMailSlot, readBuf, sizeof(readBuf), &readMsg, NULL))

throw SetErrorMail("ReadFile", GetLastError());

SetMailslotInfo(serverMailSlot, 5000);

Функция SetMailslotInfo() устанавливает новое время ожидания для функции ReadFile() в пять секунд.

if (i == 1)

t1 = clock();

cout << readBuf << " " << i << endl;

//Sleep(10);

} while (strcmp(readBuf, " ") != 0);

t2 = clock();

cout << "Время передачи: " << (t2 - t1) / 1000 << " сек." << endl;

cout << "Количество сообщений: " << i - 1 << endl << endl;

if (!CloseHandle(serverMailSlot))

throw "Error: CloseHandle";

Дескриптор почтового ящика закрывается с помощью функции CloseHandle().

system("pause");

}

catch (string e) {

if (e == "ReadFileError") {

cout << e << " Timeout";

return 0;

}

cout << e << endl;

}

return 0;

}