Вариант 6

**1**

Создадим два приложения, один – клиент, другой - сервер

Код клиента:

#include <windows.h>

#include <stdio.h>

#include <conio.h>

#include <iostream>

DWORD main(int argc, char\* argv[])

{

HANDLE handleMailSlot;

char text;

LPCTSTR sizeMailslotName = TEXT("\\\\.\\mailslot\\sample\_mailslot");;

char SizeBuf[512];

DWORD cbWritten;

// Создаем канал с процессом MSLOTS

handleMailSlot = CreateFile(

sizeMailslotName, GENERIC\_WRITE,

FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, 0, NULL);

// В случае ошибки выводим код и завершаем работу

if (handleMailSlot == INVALID\_HANDLE\_VALUE)

{

fprintf(stdout, "CreateFile: Error %ld\n",

GetLastError());

return 0;

}

// Пишем, что канал создан

fprintf(stdout, "\nConnected. Type 'exit' to terminate\n");

// Посылаем команды через канал

while (1)

{

// Сообщение-приглашение ввести команду

printf("cmd>");

// Вводим строку

std::cin >> SizeBuf;

// Передаем серверному процессу

// в качестве команды

if (!WriteFile(handleMailSlot, SizeBuf, strlen(SizeBuf) + 1,

&cbWritten, NULL))

break;

// В ответ на команду "exit" завершаем цикл

// обмена данными с серверным процессом

if (!strcmp(SizeBuf, "exit"))

break;

}

// Закрываем идентификатор канала

CloseHandle(handleMailSlot);

return 0;

}

Код сервера:

#include <windows.h>

#include <stdio.h>

#include <conio.h>

int main()

{

// Код возврата из функций

BOOL fReturn;

// Размер сообщения в байтах

DWORD ValueOfMessagesBytes;

// Количество сообщений в канале Mailslot

DWORD MsgNumber;

// Идентификатор канала Mailslot

HANDLE handleMailSlot;

// Имя создаваемого канала Mailslot

LPCTSTR lpsizeMailslotName = TEXT("\\\\.\\mailslot\\sample\_mailslot");

// Буфер для передачи данных через канал

char SizeBuf[512];

// Количество байт данных, принятых через канал

DWORD cbRead;

// Создаем канал Mailslot

handleMailSlot = CreateMailslot(

lpsizeMailslotName, 0,

MAILSLOT\_WAIT\_FOREVER, NULL);

// В случае ошибки выводим код и завершаем работу

if (handleMailSlot == INVALID\_HANDLE\_VALUE)

{

fprintf(stdout, "CreateMailslot: Error %ld\n",

GetLastError());

return 0;

}

// Пишем, что канал создан

fprintf(stdout, "Mailslot created!!!\n");

// Посылаем команды через канал

while (1)

{

// Определяем состояние канала Mailslot

fReturn = GetMailslotInfo(

handleMailSlot, NULL, &ValueOfMessagesBytes,

&MsgNumber, NULL);

if (!fReturn)

{

fprintf(stdout, "GetMailslotInfo: Error %ld\n",

GetLastError());

break;

}

// Если в канале есть Mailslot сообщения,

// читаем первое из них и выводим

if (MsgNumber != 0)

{

if (ReadFile(handleMailSlot, SizeBuf, 512, &cbRead, NULL))

{

// Выводим принятую строку на консоль

printf("Received: <%s>\n", SizeBuf);

// Если пришла команда "exit",

// завершаем работу

if (!strcmp(SizeBuf, "exit"))

break;

}

else

{

fprintf(stdout, "ReadFile: Error %ld\n",

GetLastError());

break;

}

}

// Выполняем задержку на 200 миллисекунд

Sleep(200);

}

// Перед завершением приложения закрываем

// идентификатор канала Mailslot

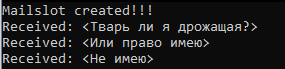
CloseHandle(handleMailSlot);

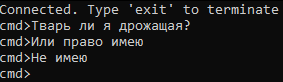
return 0;

}

**2**

В результате получили, что при запуске двух процессов сервер принимает сообщения от клиента и выводит его.





Успех!