Глава 5. Интерфейс Named Pipe

5.1. Предисловие к главе

В этой главе рассматривается еще один IPC – механизм, поддерживаемый операционной системой Windows и имеющий название ***Mailslots (почтовый ящик)***. Также как и Named Pipe механизм Mailslots может быть использован для обмена данными между распределенными в локальной сети процессами.

5.2. Назначение и состав интерфейса Mailslot

Почтовым ящиком (Mailslot) называется объект ядра операционной системы, который обеспечивает передачу данных от процессов-клиентов к процессам-серверам, выполняющимся на компьютерах в одной локальной сети. Процесс, создающий почтовый ящик называется ***сервером почтового ящика***. Процессы, которые связываются с почтовым ящиком, называются ***клиентами почтового*** ящика.

Каждый почтовый ящик имеет имя, которое определяется сервером при создании и используется клиентами для доступа. Передача может осуществляться только сообщениями и в одном направлении – от клиента к серверу. Обмен данными может происходить в синхронном и асинхронном режимах. Допускается создание нескольких серверов с одинаковым именем почтового ящика – в этом случае все отправляемые клиентом сообщения будут поступать во все почтовые ящики, имеющие имя, указанное клиентом. Однако, следует сказать, что такая рассылка сообщений возможна только в том случае, когда длина оправляемых сообщений не превышает 425 байт.

В том случае, если клиент отправляет сообщение размером меньше, чем 425 байт, то пересылка осуществляется без гарантии доставки. Пересылка сообщения размером более 425 байт возможна только от одного клиента к одному серверу.

Перечень функций интерфейса Mailslot API приводится в таблице 5.2.1. Функции CreateFile, ReadFile, WriteFile являются универсальными и используются также для работы с именованными каналами, файловой системой, сокетами и т.д.

Таблица 5.2.1

Наименование функции	Назначение
`CreateFile`	Открыть почтовый ящик
`CreateMailslot`	Создать почтовый ящик
`GetMailslotInfo`	Получить информацию о почтовом ящике
`ReadFile`	Читать данные из почтового ящика
`SetMailslotInfo`	Изменить время ожидания сообщения
`WriteFile`	Писать данные в почтовый ящик

Как и в случае с именованными каналами, для использования функций Mailslot API в программе на языке C++ достаточно включить в ее текст заголовочный файл Widows.h.

На рисунке 5.2.1 изображена схема взаимодействия процесса-сервера и процесса-клиента в простейшем случае. Каждая программа разбита на три блока. Сплошной направленной линий обозначается движение данных от одного процесса к другому.

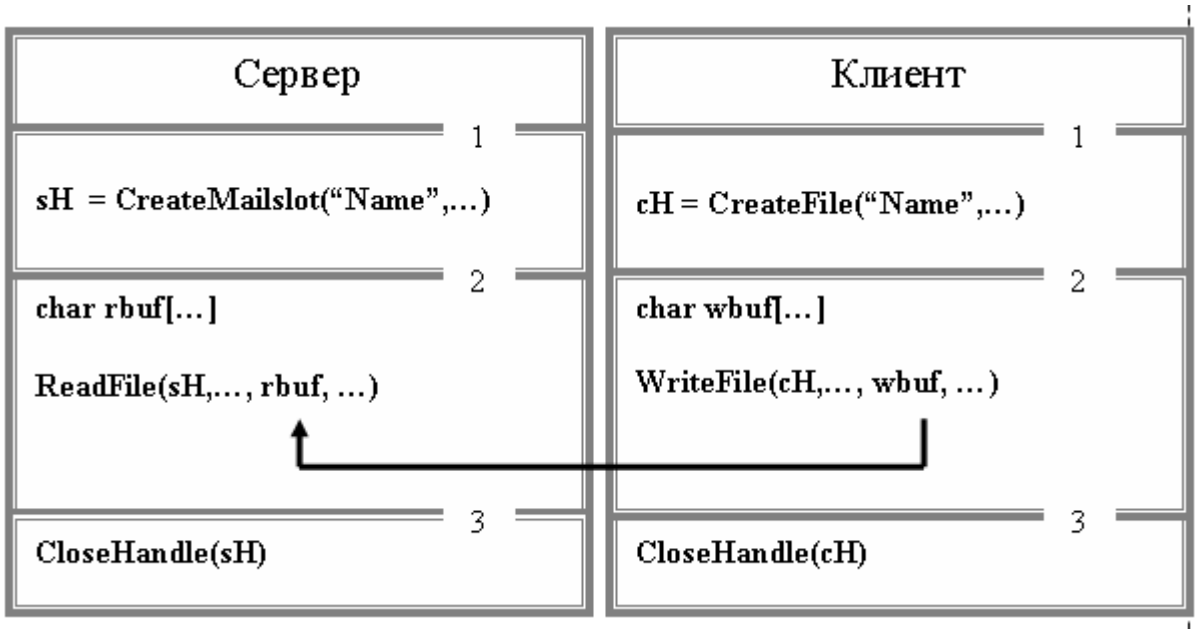

Рисунок 5.2.1. Схема взаимодействия процессов использующих Mailslot API

В первом блоке программы сервера выполняется функция CreateMailslot, создающая почтовый ящик. В случае успешного завершения функция возвращает дескриптор почтового ящика, который будет использоваться дальше. Кроме того, один из параметров функции CreateMailslot определяет время ожидания функцией ReadFile, очередного сообщения от клиента. В простейшем случае можно установить бесконечное время ожидания. Во втором блоке сервера осуществляется считывание данных из почтового ящика. В последнем третьем блоке сервера закрывается дескриптор почтового ящика, что приводит к его уничтожению.

В первом блоке программы клиента осуществляется подсоединение клиента к почтовому ящику с помощью функции CreateFile (открыть почтовый ящик). В случае успешного выполнения, функции формирует дескриптор почтового ящика, который потом используется функцией WriteFile (второй блок клиента) для записи данных в почтовый сервер. При завершении программы, следует закрыть дескриптор почтового ящика с помощью функции CloseHandle.

В принципе, между процессами обмен данными можно организовать в обе стороны. Для этого необходимо в рамках каждого процесса создать свой почтовый ящик, который бы использовался для приема сообщений.

5.3. Создание почтового ящика

Для создания почтового ящика используется функция CreateMailslot (рисунок 5.3.1).

```
// -- создать почтовый ящик
// Назначение: функция  предназначена для создания почтового
//             ящика

 HANDLE CreateMailslot
           (
     LPCTSTR        pname,    // [in] символическое имя ящика
     DWORD          maxms,    // [in] максимальная длина сообщения
     DWORD          timeo,    // [in] интервал ожидания
     LPSECURITY_ATTRIBUTES sattr // [in] атрибуты безопасности
            );

// Код возврата: в случае успешного завершения функция
//       возвращает дескриптор почтового ящика, иначе
//       значение  INVALID_HANDLE_VALUE
// Примечание: pname - указывает на строку именем канала в
//       локальном формате;
//       timeo – параметр устанавливает время ожидания
//       собщения  функцией ReadFile; для задания бесконечного
//       ожидания, следует установить значение
//       MAILSLOT_WAIT_FOREVER;
//       sattr – для установки атрибутов безопасности
//       по умолчанию  следует установить значение NULL
```

Рисунок 5.3.1. Функция CreateMailslot

Параметр функции CreateMailslot, задающий имя почтового ящика, подразумевает, что это имя задано в локальном формате. На рисунках 5.3.2 – 5.3.4 указаны три возможных формата имени почтового ящика.

```
\\.\mailslot\xxxxx
где: точка (.) - обозначает  локальный компьютер;
     mailslot  - фиксированное слово;
     xxxxx     - имя почтового ящика
```

Рисунок 5.3.2. Локальный формат имени почтового ящика

Локальный формат имени почтового ящика используется при создании почтового ящика (ящик всегда создается на локальном для сервера компьютере), а также программой клиентом при открытии ящика, если предполагается использовать для записи все ящики с заданным именем на одном локальном компьютере.

Сетевой формат имени почтового ящика используется программой клиента, для записи сообщений в группу одноименных почтовых ящиков, которые находятся на компьютере, указанном в имени.

```
\\servname\mailslot\xxxxx
где: servname  - имя  компьютера-сервера почтового ящика;
     mailslot  - фиксированное слово;
     xxxxx     - имя почтового ящика
```

Рисунок 5.3.3. Сетевой формат имени почтового ящика

```
\\domain\mailslot\xxxxx
где: domain    - имя  домена компьютеров или *;
     mailslot  - фиксированное слово;
     xxxxx     - имя почтового ящика
```

Рисунок 5.3.4. Доменный формат имени почтового ящика

Доменный формат имени почтового ящика используется программой клиента для записи сообщений в группу одноименных почтовых ящиков, которые находятся на всех компьютерах указанного домена. Если необходимо записать в сообщение в группу почтовых ящиков, которые находятся на компьютерах первичного домена, то вместо имени домена можно указать символ *.

5.4. Соединение клиентов с почтовым ящиком

Для установки связи с почтовым ящиком программа клиента использует функцию CreateFile (рисунок 5.4.1).

Как уже отмечалось, функция CreateFile является универсальной и значения ее параметров практически ничем не отличаются от значений, применяющихся для связи клиента именованного канала с сервером именованного канала. В описании функции и приведенных примерах не рассматриваются никакие параметры, определяющие атрибуты безопасности. Использование атрибутов безопасности установленных по умолчанию, приводит к тому, что связь может быть установлена только между процессами, которые запущены от одного имени и с одним общим паролем. Для знакомства с возможностями интерфейса Mailslots, связанными с системой безопасности операционной системы Windows рекомендуется обратиться к источникам [4] или http://msdn2.miscrosoft.com

Следует обратить внимание на формат имени открываемого почтового ящика. Этот формат определяет пространство поиска почтовых ящиков, с которыми будет установлена связь.

```
// -- открыть почтовый ящик
// Назначение: функция  предназначена для подключения клиента
//             к  почтовому ящику

 HANDLE CreateFile
                  (
     LPCTSTR    mname, // [in] символическое имя почтового ящика
     DWORD      accss, // [in] чтение или запись
     DWORD      share, // [in] режим совместного использования
     LPSECURITY_ATTRIBUTES sattr // [in] атрибуты безопасности
     DWORD      oflag, // [in] флаг открытия почтового ящика
     DWORD      aflag, // [in] флаги и атрибуты
     HANDLE     exten, // [in] дополнительные атрибуты
                  );

// Код возврата: в случае успешного завершения функция
//       возвращает дескриптор именованного канала, иначе
//       INVALID_HANDLE_VALUE -  неудачное завершение
// Примечание:- параметр mname указывается в локальном,
//       сетевом или доменном  формате:  в зависимости от
//       способа применения;
//     - параметр accss  должен принимать значение
//        GENERIC_WRITE
//     - параметр share может принимать значения
//        FILE_SHARE_READ (совместное чтение),
//        FILE_SHARE_WRITE (совместная запись),
//       FILE_SHARE_READ| FILE_SHARE_WRITE (чтение и запись);
//     - параметр sattr для установки атрибутов безопасности
//       по умолчанию, следует установить значение NULL;
//     - значение параметра oflag всегда устанавливается  в
//       OPEN_EXISTING(открытие существующего ящика);
//     - значение параметра aflag можно установить в NULL,
//       что определяет значения флагов и атрибутов  по
//       умолчанию или установить FILE_ATTRIBUTE_NORMAL;
//     - значение параметра exten следует становить в  NULL
```

Рисунок 5.4.1. Функция CreateFile

5.5. Обмен данными через почтовый ящик

Для записи данных в почтовый ящик используется функция WriteFile, а для чтения данных из почтового ящика функция ReadFile. Значения параметров, используемые в этих универсальных функциях при работе с почтовыми ящиками, практически ничем не отличаются от значений, применяемых при работе с именованными каналами. Разница заключается лишь в том, что в одном случае функции используют дескрипторы каналов, а другом – дескрипторы почтовых ящиков. Кроме того, следует помнить, что

функцию ReadFile может выполнять только программа сервера, а WriteFle могут выполнять и сервер (сервер может записывать в свой собственный ящик) и клиент.

```
//.............................................................
HANDLE hM; // дескриптор почтового ящика
DWORD rb;                              // длина почитанного сообщения
char rbuf[100];                        // буфер ввода
try
{
   if ((hM = CreateMailslot("\\\\.\\mailslot\\myslot",
                NULL,
                MAILSLOT_WAIT_FOREVER,       // ждать вечно
                NULL)) == INVALID_HANDLE_VALUE)
                   throw "CreateMailslotError";
  if(!ReadFile(hM,
              rbuf,                          // буфер
              sizeof(rbuf),                  // размер буфера
              &rb,                           // прочитано
              NULL))
                    throw "ReadFileError";
//.............................................................
}
//.............................................................
```

Рисунок 5.5.1. Создание почтового ящика

```
//.............................................................
HANDLE hM; // дескриптор почтового ящика
DWORD wb;                          // длина записанного сообщения
char wbuf[] = "Hello Mailslot";  // буфер вывода
try
{
   if ((hM = CreateFile("\\\\isit301\\mailslot\\myslot",
                 GENERIC_WRITE,    // будем писать в ящик
                 FILE_SHARE_READ,  // разрешаем одновременно читать
                 NULL,
                 OPEN_EXISTING,    //  ящик уже есть
                 NULL, NULL)) == INVALID_HANDLE_VALUE)
                 throw "CreateFileError";
   if(!WriteFile(hM,
                 wbuf,               // буфер
                 sizeof(wbuf),       // размер буфера
                 &wb,                // записано
                 NULL))
                  throw "ReadFileError";

//.............................................................
}
//.............................................................
```

Рисунок 5.5.2. Соединение клиента с почтовым ящиком

На рисунках 5.5.1 и 5.5.2 представлены фрагменты программ сервера и клиента. В программе сервера создается почтовый ящик и читается сообщение из него. В программе клиента осуществляется подключение к почтовому ящику и записывается в него сообщение. Следует обратить внимание, что программы клиента и сервера находятся на разных компьютерах, т.к. символическое имя почтового ящика в функции CreateFile указано в сетевом формате.

5.6. Получение информации о почтовом ящике

Получить информацию о характеристиках почтового ящика можно с помощью функции GetMailslotInfo (рисунок 5.6.1).

```
// -- получить информацию о почтовом ящике
// Назначение: функция  предназначена для получения
//             характеристик созданного почтового ящика

 BOOL GetMailslotInfo
                  (
     HANDLE     hM,    // [in] дескриптор почтового ящика
     LPDWORD    ml,    // [out] максимальная длина сообщения
     LPDWORD    nl,    // [out] длина следующего сообщения
     LPDWORD    nm     // [out] количество сообщений
     LPDWORD    to     // [out] интервал ожидания сообщения
                 );

// Код возврата: в случае успешного завершения функция
//       возвращает TRUE, иначе FALSE
```

Рисунок 5.6.1. Функция GetMailslotInfo

Функция GetMailslotInfo может быть использована только на стороне сервера почтового ящика и параметр hM должен быть получен в результате выполнения функции CreateMailslot. Чаще всего функция применяется для выяснения количества непрочитанных сообщений накопившихся в почтовом ящике.

5.7. Изменение интервала ожидания сообщения

Время ожидания функцией ReadFile поступления сообщения в почтовый ящик первоначально устанавливается при создании почтового ящика с помощью функции CreateMailslot. В процессе работы, может оказаться необходимым изменить значение этого интервала или вообще сделать его нулевым. Для этого применяется функция SetMailslotInfo (рисунок 5.6.1). Функция может быть выполнена только в программе сервера и использует в качестве аргумента дескриптор почтового сервера, который был получен при выполнении функции CreateMailslot.

```
// -- изменить время ожидания сообщения
// Назначение: функция  предназначена для изменения
     //             одной характеристики почтового ящика –
//              интервала времени ожидания

 BOOL SetMailslotInfo
                  (
     HANDLE      hM, // [in] дескриптор почтового ящика
     PDWORD      to  // [in] новое значение интервала
                  );

// Код возврата: в случае успешного завершения функция
//       возвращает TRUE, иначе FALSE
```

Рисунок 5.7.1. Функция SetMailslotInfo

5.8. Итоги главы

1. Механизм Mailslots (почтовый ящик) является одним из IPC-механизмов операционной системы Windows, позволяющий создавать распределенные приложения архитектуры клиент-сервер в локальной сети TCP/IP.
2. Почтовый ящик представляет собой объект операционной системы, предоставляющий возможность пересылать данные в одном направлении: от клиента к серверу.
3. Почтовый ящик идентифицируется своим именем. Сервером называется процесс создающий почтовый ящик. Клиентом – процесс, который подключается к почтовому ящику и записывает в него данные.
4. Обмен данными осуществляется сообщениями и может происходить в синхронном и асинхронном режимах. Если клиент и сервер находятся на разных компьютерах, доставка сообщений не гарантируется.
5. Допускается создание нескольких ящиков с одним и тем же именем. Если пересылаемые сообщения не превышают 425 байт, то возможна передача данных одновременно нескольким почтовым ящикам.
6. В состав Mailslots API входят функции создания почтового ящика, подсоединения клиента к почтовому ящику, функции записи и чтения сообщений, а также функции для получения и установки характеристик почтового ящика.