Каналы передачи данных Pipe

В среде операционной системы Microsoft Windows вам доступно такое удобное средство передачи данных между параллельно работающими процессами, как каналы типа Pipe. Это средство позволяет организовать передачу данных между локальными процессами, а также между процессами, запущенными на различных рабочих станциях в сети.

Каналы типа Pipe больше всего похожи на файлы, поэтому они достаточно просты в использовании.

Через канал можно передавать данные только между двумя процессами. Один из процессов создает канал, другой открывает его. После этого оба процесса могут передавать данные через канал в одну или обе стороны, используя для этого хорошо знакомые вам функции, предназначенные для работы с файлами, такие как ReadFile и WriteFile. Заметим, что приложения могут выполнять над каналами Pipe синхронные или асинхронные операции, аналогично тому, как это можно делать с файлами. В случае использования асинхронных операций необходимо отдельно побеспокоиться об организации синхронизации.

Именованные и анонимные каналы

Существуют две разновидности каналов Pipe - именованные (Named Pipes) и анонимные (Anonymous Pipes).

Как видно из названия, именованным каналам при создании присваивается имя, которое доступно для других процессов. Зная имя какой-либо рабочей станции в сети, процесс может получить доступ к каналу, созданному на этой рабочей станции.

Анонимные каналы обычно используются для организации передачи данных между родительскими и дочерними процессами, запущенными на одной рабочей станции или на “отдельно стоящем” компьютере.

Имена каналов

Имена каналов в общем случае имеют следующий вид:

```
\\ИмяСервера\pipe\ИмяКанала
```

Если процесс открывает канал, созданный на другой рабочей станции, он должен указать имя сервера. Если же процесс создает канал или открывает канал на своей рабочей станции, вместо имени указывается символ точки:

```
\\.\pipe\ИмяКанала
```

В любом случае процесс может создать канал только на той рабочей станции, где он запущен, поэтому при создании канала имя сервера никогда не указывается.

Реализации каналов

В простейшем случае один серверный процесс создает один канал (точнее говоря, одну реализацию канала) для работы с одним клиентским процессом.

Однако часто требуется организовать взаимодействие одного серверного процесса с несколькими клиентскими. Например, сервер базы данных может принимать от клиентов запросы и рассылать ответы на них.

В случае такой необходимости серверный процесс может создать несколько реализаций канала, по одной реализации для каждого клиентского процесса.

Функции для работы с каналами

В этом разделе мы опишем наиболее важные функции программного интерфейса Microsoft Windows NT, предназначенные для работы с каналами Pipes. Более подробную информацию вы найдете в документации, которая поставляется в составе библиотеки разработчика Microsoft Development Library (MSDN).

Создание канала

Для создания именованных и анонимных каналов Pipes используются функции CreatePipe и CreateNamedPipe.

Функция CreatePipe

Анонимный канал создается функцией CreatePipe, имеющей следующий прототип:

```
BOOL CreatePipe(
  PHANDLE hReadPipe,  // адрес переменной, в которую будет
                      // записан идентификатор канала для
                      // чтения данных
  PHANDLE hWritePipe, // адрес переменной, в которую будет
                      // записан идентификатор канала для
                      // записи данных
  LPSECURITY_ATTRIBUTES lpPipeAttributes, // адрес переменной
                      // для атрибутов защиты
  DWORD nSize);       // количество байт памяти,
```

```
                                // зарезервированной для канала
```

Канал может использоваться как для записи в него данных, так и для чтения. Поэтому при создании канала функция CreatePipe возвращает два идентификатора, записывая их по адресу, заданному в параметрах hReadPipe и hWritePipe.

Идентификатор, записанный по адресу hReadPipe, можно передавать в качестве параметра функции ReadFile или ReadFileEx для выполнения операции чтения. Идентификатор, записанный по адресу hWritePipe, передается функции WriteFile или WriteFileEx для выполнения операции записи.

Через параметр lpPipeAttributes передается адрес переменной, содержащей атрибуты защиты для создаваемого канала. В наших приложениях мы будем указывать этот параметр как NULL. В результате канал будет иметь атрибуты защиты, принятые по умолчанию.

И, наконец, параметр nSize определяет размер буфера для создаваемого канала. Если этот размер указан как нуль, будет создан буфер с размером, принятым по умолчанию. Заметим, что при необходимости система может изменить указанный вами размер буфера.

В случае успеха функция CreatePipe возвращает значение TRUE, при ошибке - FALSE. В последнем случае для уточнения причины возникновения ошибки вы можете воспользоваться функцией GetLastError.

Функция CreateNamedPipe

Для создания именованного канала Pipe вы должны использовать функцию CreateNamedPipe. Вот прототип этой функции:

```
HANDLE CreateNamedPipe(
  LPCTSTR lpName,           // адрес строки имени канала
  DWORD   dwOpenMode,       // режим открытия канала
  DWORD   dwPipeMode,       // режим работы канала
  DWORD   nMaxInstances,    // максимальное количество
                            // реализаций канала
  DWORD   nOutBufferSize,   // размер выходного буфера в байтах
  DWORD   nInBufferSize,    // размер входного буфера в байтах
  DWORD   nDefaultTimeOut, // время ожидания в миллисекундах
  LPSECURITY_ATTRIBUTES lpSecurityAttributes); // адрес
                            // переменной для атрибутов защиты
```

Через параметр lpName передается адрес строки имени канала в форме \\.\pipe\ИмяКанала (напомним, что при создании канала имя сервера не

указывается, так как канал можно создать только на той рабочей станции, где запущен процесс, создающий канал).

Параметр dwOpenMode задает режим, в котором открывается канал. Остановимся на этом параметре подробнее.

Канал Pipe может быть ориентирован либо на передачу потока байт, либо на передачу сообщений. В первом случае данные через канал передаются по байтам, во втором - отдельными блоками заданной длины.

Режим работы канала (ориентированный на передачу байт или сообщений) задается, соответственно, константами PIPE_TYPE_BYTE или PIPE_TYPE_MESSAGE, которые указываются в параметре dwOpenMode. По умолчанию используется режим PIPE_TYPE_BYTE.

Помимо способа передачи данных через канал, с помощью параметра dwOpenMode можно указать, будет ли данный канал использован только для чтения данных, только для записи или одновременно для чтения и записи. Способ использования канала задается указанием одной из следующих констант:

Константа	Использование канала
PIPE_ACCESS_INBOUND	Только для чтения
PIPE_ACCESS_OUTBOUND	Только для записи
PIPE_ACCESS_DUPLEX	Для чтения и записи

Перечисленные выше параметры должны быть одинаковы для всех реализаций канала (о реализациях канала мы говорили выше). Далее мы перечислим параметры, которые могут отличаться для разных реализаций канала:

Константа	Использование канала
PIPE_READMODE_BYTE	Канал открывается на чтение в режиме последовательной передачи отдельных байт
PIPE_READMODE_MESSAGE	Канал открывается на чтение в режиме передачи отдельных сообщений указанной длины
PIPE_WAIT	Канал будет работать в блокирующем режиме, когда процесс переводится в состояние ожидания до завершения операций в канале
PIPE_NOWAIT	Неблокирующий режим работы канала. Если операция

	не может быть выполнена немедленно, в неблокирующем режиме функция завершается с ошибкой
FILE_FLAG_OVERLAPPED	Использование асинхронных операций (ввод и вывод с перекрытием). Данный режим позволяет процессу выполнять полезную работу параллельно с проведением операций в канале
FILE_FLAG_WRITE_THROUGH	В этом режиме функции, работающие с каналом, не возвращают управление до тех пор, пока не будет полностью завершена операция на удаленном компьютере. Используется только с каналом, ориентированном на передачу отдельных байт и только в том случае, когда канал создан между процессами, запущенными на различных станциях сети

Дополнительно к перечисленным выше флагам, через параметр dwOpenMode можно передавать следующие флаги защиты:

Флаг	Описание
WRITE_DAC	Вызывающий процесс должен иметь права доступа на запись к произвольному управляющему списку доступа именованного канала access control list (ACL)
WRITE_OWNER	Вызывающий процесс должен иметь права доступа на запись к процессу, владеющему именованным каналом Pipe
ACCESS_SYSTEM_SECURITY	Вызывающий процесс должен иметь права доступа на запись к управляющему списку доступа именованного канала access control list (ACL)

Подробное описание этих флагов выходит за рамки нашей книги. При необходимости обратитесь к документации, входящей в состав SDK.

Теперь перейдем к параметру dwPipeMode, определяющему режим работы канала. В этом параметре вы можете указать перечисленные выше константы PIPE_TYPE_BYTE, PIPE_TYPE_MESSAGE, PIPE_READMODE_BYTE, PIPE_READMODE_MESSAGE, PIPE_WAIT и PIPE_NOWAIT. Для всех реализаций канала необходимо указывать один и тот же набор констант.

Параметр nMaxInstances определяет максимальное количество реализаций, которые могут быть созданы для канала. Вы можете указывать здесь значения от 1 до PIPE_UNLIMITED_INSTANCES. В последнем случае максимальное количество реализаций ограничивается только наличием свободных системных ресурсов.

Заметим, что если один серверный процесс использует несколько реализаций канала для связи с несколькими клиенсткими, то общее количество реализаций может быть меньше, чем потенциальное максимальное количество клиентов. Это связано с тем, что клиенты могут использовать реализации по очереди, если только они не пожелают связаться с серверным процессом все одновременно.

Параметры nOutBufferSize и nInBufferSize определяют, соответственно, размер буферов, используемых для записи в канал и чтения из канала. При необходимости система может использовать буферы других, по сравнению с указанными, размеров.

Параметр nDefaultTimeOut определяет время ожидания для реализации канала. Для всех реализаций необходимо указывать одинаковое значение этого параметра.

Через параметр lpPipeAttributes передается адрес переменной, содержащей атрибуты защиты для создаваемого канала. В наших приложениях мы будем указывать этот параметр как NULL. В результате канал будет иметь атрибуты защиты, принятые по умолчанию.

В случае успеха функция CreateNamedPipe возвращает идентификатор созданной реализации канала, который можно использовать в операциях чтения и записи, выполняемых с помощью таких функций, как ReadFile и WriteFile.

При ошибке функция CreateNamedPipe возвращает значение INVALID_HANDLE_VALUE. Код ошибки вы можете уточнить, вызвав функцию GetLastError.

Использование фукции CreateFile

Функция CreateFile, предназначенная для работы с файлами, может также быть использована для создания новых каналов и открытия существующих. При этом вместо имени файла вы должны указать этой функции имя канала Pipe.

Пример использования функции CreateNamedPipe

Приведем пример использования функции CreateNamedPipe для создания именованного канала Pipe с именем $MyPipe$, предназначенным для чтения и записи данных, работающем в блокирующем режиме и допускающем создание неограниченного количества реализаций:

```
HANDLE hNamedPipe;
LPSTR  lpszPipeName = "\\\\.\\pipe\\$MyPipe$";
hNamedPipe = CreateNamedPipe(
    lpszPipeName,
    PIPE_ACCESS_DUPLEX,
    PIPE_TYPE_MESSAGE | PIPE_READMODE_MESSAGE | PIPE_WAIT,
    PIPE_UNLIMITED_INSTANCES,
    512, 512, 5000, NULL);
```

Через создаваемый канал передаются сообщения (так как указана константа PIPE_TYPE_MESSAGE). Данная реализация предназначена только для чтения (константа PIPE_READMODE_MESSAGE).

При создании канала мы указали размер буферов ввода и вывода, равный 512 байт. Время ожидания операций выбрано равным 5 секунд. Атрибуты защиты не указаны, поэтому используются значения, принятые по умолчанию.

Установка соединения с каналом со стороны сервера

После того как серверный процесс создал канал, он может перейти в режим соединения с клиентским процессом. Соединение со стороны сервера выполняется с помощью функции ConnectNamedPipe.

Прототип функции ConnectNamedPipe представлен ниже:

```
BOOL ConnectNamedPipe(
  HANDLE hNamedPipe,       // идентификатор именованного канала
  LPOVERLAPPED lpOverlapped); // адрес структуры OVERLAPPED
```

Через первый параметр серверный процесс передает этой функции идентификатор канала, полученный от функции CreateNamedPipe.

Второй параметр используется только для огранизации асинхронного обмена данными через канал. Если вы используете только синхронные операции, в качестве значения для этого параметра можно указать NULL.

В случае успеха функция ConnectNamedPipe возвращает значение TRUE, а при ошибке - FALSE. Код ошибки можно получить с помощью функции GelLastError.

В зависимости от различных условий функция ConnectNamedPipe может вести себя по разному.

Если параметр lpOverlapped указан как NULL, функция выполняется в синхронном режиме. В противном случае используется асинхронный режим.

Для канала, созданного в синхронном блокирующем режиме (с использованием константы PIPE_WAIT), функция ConnectNamedPipe переходит в состояние ожидания соединения с клиентским процессом. Именно этот режим мы будем использовать в наших примерах программ, исходные тексты которых вы найдете ниже.

Если канал создан в синхронном неблокирующем режиме, функция ConnectNamedPipe немедленно возвращает управление с кодом TRUE, если только клиент был отключен от данной реализации канала и возможно подключение этого клиента. В противном случае возвращается значение FALSE. Дальнейший анализ необходимо выполнять с помощью функции GetLastError XE "GetLastError" . Эта функция может вернуть значение ERROR_PIPE_LISTENING (если к серверу еще не подключен ни один клиент), ERROR_PIPE_CONNECTED XE "ERROR_PIPE_CONNECTED" (если клиент уже подключен) или ERROR_NO_DATA (если предыдущий клиент отключился от сервера, но клиент еще не завершил соединение).

Ниже мы привели пример использования функции ConnectNamedPipe:

```
HANDLE hNamedPipe;
LPSTR  lpszPipeName = "\\\\.\\pipe\\$MyPipe$";
hNamedPipe = CreateNamedPipe(
    lpszPipeName,
    PIPE_ACCESS_DUPLEX,
    PIPE_TYPE_MESSAGE | PIPE_READMODE_MESSAGE | PIPE_WAIT,
    PIPE_UNLIMITED_INSTANCES,
    512, 512, 5000, NULL);
fConnected = ConnectNamedPipe(hNamedPipe, NULL);
```

В данном случае функция ConnectNamedPipe перейдет в состояние ожидания, так как канал был создан для работы в синхронном блокирующем режиме.

Установка соединения с каналом со стороны клиента

Для создания канала клиентский процесс может воспользоваться функцией CreateFile. Как вы помните, эта функция предназначена для работы с файлами, однако с ее помощью можно также открыть канал, указав его имя вместо имени файла. Забегая вперед, скажем, что функция CreateFile позволяет открывать не только файлы или каналы Pipe, но и другие системные ресурсы, например, устройства и каналы Mailslot.

Функция CreateFile была нами описана в предыдущем томе “Библиотеки системного программиста”, однако для удобства мы повторим прототип этой функции и ее краткое описание:

Итак, прототип функции CreateFile:

```
HANDLE CreateFile(
  LPCTSTR lpFileName,     // адрес строки имени файла
  DWORD  dwDesiredAccess, // режим доступа
  DWORD  dwShareMode, // режим совместного использования файла
  LPSECURITY_ATTRIBUTES lpSecurityAttributes, // дескриптор
                                              // защиты
  DWORD  dwCreationDistribution, // параметры создания
  DWORD  dwFlagsAndAttributes,   // атрибуты файла
  HANDLE hTemplateFile);  // идентификатор файла с атрибутами
```

Раньше при работе с файлами через параметр lpFileName вы передавали этой функции адрес строки, содержащей имя файла, который вы собираетесь создать или открыть. Строка должна быть закрыта двоичным нулем. Если функция CreateFile работает с каналом Pipe, параметр lpFileName определяет имя канала.

Параметр dwDesiredAccess определяет тип доступа, который должен быть предоставлен к открываемому файлу. В нашем случае этот тип доступа будет относиться к каналу Pipe. Здесь вы можете использовать логическую комбинацию следующих констант:

Константа	Описание
0	Доступ запрещен, однако приложение может определять атрибуты файла, канала или устройства, открываемого при помощи функции CreateFile XE "CreateFile"
GENERIC_READ	Разрешен доступ на чтение из файла или канала Pipe
GENERIC_WRITE	Разрешен доступ на запись в файл или канал Pipe

Тип доступа, указанный при помощи параметра dwDesiredAccess, не должен противоречить типу доступа для канала, заданного при его создании функцией CreateNamedPipe.

С помощью параметра dwShareMode задаются режимы совместного использования открываемого или создаваемого файла. Для этого параметра вы можете указать комбинацию следующих констант:

Константа	Описание
0	Совместное использование файла запрещено
FILE_SHARE_READ	Другие приложения могут открывать файл с помощью функции CreateFile для чтения
FILE_SHARE_WRITE	Аналогично предыдущему, но на запись

Через параметр lpSecurityAttributes необходимо передать указатель на дескриптор защиты или значение NULL, если этот дескриптор не используется. В наших приложениях мы не работаем с дескриптором защиты.

Параметр dwCreationDistribution определяет действия, выполняемые функцией CreateFile, если приложение пытается создать файл, который уже существует. Для этого параметра вы можете указать одну из следующих констант:

Константа	Описание
CREATE_NEW	Если создаваемый файл уже существует, функция CreateFile возвращает код ошибки
CREATE_ALWAYS	Существующий файл перезаписывается, при этом содержимое старого файла теряется
OPEN_EXISTING	Открывается существующий файл. Если файл с указанным именем не существует, функция CreateFile возвращает код ошибки
OPEN_ALWAYS	Если указанный файл существует, он открывается. Если файл не существует, он будет создан
TRUNCATE_EXISTING	Если файл существует, он открывается, после чего длина файла устанавливается равной нулю.

	Содержимое старого файла теряется. Если же файл не существует, функция CreateFile возвращает код ошибки

Параметр dwFlagsAndAttributes задает атрибуты и флаги для файла.

При этом можно использовать любые логические комбинации следующих атрибутов (кроме атрибута FILE_ATTRIBUTE_NORMAL, который можно использовать только отдельно):

Атрибут	Описание
FILE_ATTRIBUTE_ARCHIVE	Файл был архивирован (выгружен)
FILE_ATTRIBUTE_COMPRESSED	Файл, имеющий этот атрибут, динамически сжимается при записи и восстанавливается при чтении. Если этот атрибут имеет каталог, то для всех расположенных в нем файлов и каталогов также выполняется динамическое сжатие данных
FILE_ATTRIBUTE_NORMAL	Остальные перечисленные в этом списка атрибуты не установлены
FILE_ATTRIBUTE_HIDDEN	Скрытый файл
FILE_ATTRIBUTE_READONLY	Файл можно только читать
FILE_ATTRIBUTE_SYSTEM	Файл является частью операционной системы

В дополнение к перечисленным выше атрибутам, через параметр dwFlagsAndAttributes вы можете передать любую логическую комбинацию флагов, перечисленных ниже:

Флаг	Описание
FILE_FLAG_WRITE_THROUGH	Отмена промежуточного кэширования данных для уменьшения вероятности потери данных при аварии
FILE_FLAG_NO_BUFFERING	Отмена промежуточной буферизации или кэширования. При использовании этого флага необходимо выполнять чтение

	и запись порциями, кратными размеру сектора (обычно 512 байт)
FILE_FLAG_OVERLAPPED	Асинхронное выполнение чтения и записи. Во время асинхронного чтения или записи приложение может продолжать обработку данных
FILE_FLAG_RANDOM_ACCESS	Указывает, что к файлу будет выполняться произвольный доступ. Флаг предназначен для оптимизации кэширования
FILE_FLAG_SEQUENTIAL_SCAN	Указывает, что к файлу будет выполняться последовательный доступ от начала файла к его концу. Флаг предназначен для оптимизации кэширования
FILE_FLAG_DELETE_ON_CLOSE	Файл будет удален сразу после того как приложение закроет его идентификтор. Этот флаг удобно использовать для временных файлов
FILE_FLAG_BACKUP_SEMANTICS	Файл будет использован для выполнения операции выгрузки или восстановления. При этом выполняется проверка прав доступа
FILE_FLAG_POSIX_SEMANTICS	Доступ к файлу будет выполняться в соответствии со спецификацией POSIX

И, наконец, последний параметр hTemplateFile предназначен для доступа к файлу шаблона с расширенными атрибутами создаваемого файла.

В случае успешного завершения функция CreateFile возвращает идентификатор созданного или открытого файла (или каталога), а при работе с каналом Pipe - идентификатор реализации канала.

При ошибке возвращается значение INVALID_HANDLE_VALUE (а не NULL, как можно было бы предположить). Код ошибки можно определить при помощи функции GetLastError.

В том случае, если файл уже существует и были указаны константы CREATE_ALWAYS или OPEN_ALWAYS, функция CreateFile не возвращает код ошибки. В то же время в этой ситуации функция GetLastError возвращает значение ERROR_ALREADY_EXISTS.

Приведем фрагмент исходного текста клиентского приложения, открывающего канал с именем $MyPipe$ при помощи функции CreateFile:

```
char   szPipeName[256];
HANDLE hNamedPipe;
strcpy(szPipeName, "\\\\.\\pipe\\$MyPipe$");
hNamedPipe = CreateFile(
    szPipeName, GENERIC_READ | GENERIC_WRITE,
    0, NULL, OPEN_EXISTING, 0, NULL);
```

Здесь канал открывается как для записи, так и для чтения.

Отключение серверного процесса от клиентского процесса

Если сервер работает с несколькими клиентскими процессами, то он может использовать для этого несколько реализаций канала, причем одни и те же реализации могут применяться по очереди.

Установив канал с клиентским процессом при помощи функции ConnectNamedPipe, серверный процесс может затем разорвать канал, вызвав для этого функцию DisconnectNamedPipe. После этого реализация канала может быть вновь использована для соединения с другим клиентским процессом.

Прототип функции DisconnectNamedPipe мы привели ниже:

```
BOOL DisconnectNamedPipe(HANDLE hNamedPipe);
```

Через параметр hNamedPipe функции передается идентификатор реализации канала Pipe, полученный от функции CreateNamedPipe.

В случае успеха функция возвращает значение TRUE, а при ошибке - FALSE. Код ошибки можно получить от функции GetLastError.

Закрывание идентификатора канала

Если канал больше не нужен, после отключения от клиентского процесса серверный и клиентский процессы должны закрыть его идентификатор функцией CloseHandle:

```
CloseHandle(hNamedPipe);
```

Запись данных в канал

Запись данных в открытый канал выполняется с помощью функции WriteFile, аналогично записи в обычный файл:

```
HANDLE hNamedPipe;
DWORD  cbWritten;
char   szBuf[256];
WriteFile(hNamedPipe, szBuf, strlen(szBuf) + 1,
  &cbWritten, NULL);
```

Через первый параметр функции WriteFile передается идентификатор реализации канала. Через второй параметр передается адрес буфера, данные из которого будут записаны в канал. Размер этого буфера указывается при помощи третьего параметра. Предпоследний параметр используется для определения количества байт данных, действительно записанных в канал. И, наконец, последний параметр задан как NULL, поэтому запись будет выполняться в синхронном режиме.

Учтите, что если канал был создан для работы в блокирующем режиме, и функция WriteFile работает синхронно (без использования вывода с перекрытием), то эта функция не вернет управление до тех пор, пока данные не будут записаны в канал.

Чтение данных из канала

Как и следовало ожидать, для чтения данных из канала можно воспользоваться функцией ReadFile, например, так:

```
HANDLE hNamedPipe;
DWORD  cbRead;
char   szBuf[256];
ReadFile(hNamedPipe, szBuf, 512, &cbRead, NULL);
```

Данные, прочитанные из канала hNamedPipe, будут записаны в буфер szBuf, имеющий размер 512 байт. Количество действительно прочитанных байт данных будет сохранено функцией ReadFile в переменной cbRead. Так как последний параметр функции указан как NULL, используется синхронный режим работы без перекрытия.

Другие функции

Среди других функций, предназначенных для работы с каналами Pipe, мы рассмотрим функции CallNamedPipe, TransactNamedPipe, PeekNamedPipe, WaitNamedPipe, SetNamedPipeHandleState, GetNamedPipeInfo, GetNamedPipeHandleState.

Функция CallNamedPipe

Обычно сценарий взаимодействия клиентского процесса с серверным заключается в выполнении следующих операций:

· подключение к каналу с помощью функции CreateFile;

· выполнение операций чтения или записи такими функциями как ReadFile или WriteFile;

· отключение от канала функцией CloseHandle.

Функция CallNamedPipe позволяет выполнить эти операции за один прием, при условии что канал открыт в режиме передачи сообщений и что клиент посылает одно сообщение серверу и в ответ также получает от сервера одно сообщение.

Ниже мы привели прототип функции CallNamedPipe:

```
BOOL CallNamedPipe(
  LPCTSTR lpNamedPipeName, // адрес имени канала
  LPVOID  lpOutBuffer,     // адрес буфера для записи
  DWORD   nOutBufferSize,  // размер буфера для записи
  LPVOID  lpInBuffer,      // адрес буфера для чтения
  DWORD   nInBufferSize,   // размер буфера для чтения
  LPDWORD lpBytesRead,     // адрес переменной для записи
                        // количества прочитанных байт данных
  DWORD   nTimeOut);       // время ожидания в миллисекундах
```

Перед вызовом функции CallNamedPipe вы должны записать в параметр lpNamedPipeName указатель на текстовую строку, содержащую имя канала Pipe. При формировании этой строки вы должны руководствоваться теми же правилами, что и при использовании функции CreateFile.

Кроме имени канала, вы также должны подготовить буфер, содержащий передаваемые через канал данные. Адрес и размер этого буфера следует указать функции CallNamedPipe, соответственно, через параметры lpOutBuffer и nOutBufferSize.

Данные, полученные от сервера в ответ на посланное ему сообщение, будут записаны в буфер, который вы тоже должны подготовить заранее. Адрес

этого буфера необходимо указать через параметр lpInBuffer, а размер буфера - через параметр nInBufferSize.

В переменную, адрес которой указан через параметр lpBytesRead, записывается количество байт, полученных через канал от сервера.

Параметр nTimeOut определяет, в течении какого времени функция CallNamedPipe будет ожидать доступности канала Pipe, прежде чем она вернет код ошибки. Помимо численного значения в миллисекундах, вы можете указать в этом параметре одну из следующих констант:

Константа	Описание
NMPWAIT_NOWAIT	Ожидание канала Pipe не выполняется. Если канал не доступен, функция CallNamedPipe сразу возвращает код ошибки
NMPWAIT_WAIT_FOREVER	Ожидание выполняется бесконечно долго
NMPWAIT_USE_DEFAULT_WAITXE	Ожидание выполняется в течении периода времени, указанного при вызове функции CreateNamedPipe

В случае успешного завершения функция CallNamedPipe возвращает значение TRUE, а при ошибке - FALSE. Код ошибки можно получить, вызвав функцию GetLastError.

Заметим, что может возникнуть ситуация, при которой длина сообщения, полученного от сервера, превосходит размер буфера, предусмотренного процессом. В этом случае функция CallNamedPipe завершится с ошибкой, а функция GetLastError вернет значение ERROR_MORE_DATA. Не существует никакого способа получить через канал оставшуюся часть сообщения, так как перед возвращением управления функция CallNamedPipe закрывает канал с сервером.

Функция TransactNamedPipe

Функция TransactNamedPipe, также как и функция CallNamedPipe, предназначена для выполнения передачи и приема данных от клиентского процесса серверному. Однако эта функция более гибкая в использовании, чем функция CallNamedPipe.

Прежде всего, перед использованием функции TransactNamedPipe клиентский процесс должен открыть канал с сервером, воспользовавшись для этого, например, функцией CreateFile.

Кроме того, клиентский процесс может выполнять обмен данными с сервером, вызывая функцию TransactNamedPipe много раз. При этом не будет происходить многократного открытия и закрытия канала.

Прототип функции TransactNamedPipe представлен ниже:

```
BOOL TransactNamedPipe(
  HANDLE  hNamedPipe,  // идентификатор канала Pipe
  LPVOID  lpvWriteBuf, // адрес буфера для записи
  DWORD   cbWriteBuf,  // размер буфера для записи
  LPVOID  lpvReadBuf,  // адрес буфера для чтения
  DWORD   cbReadBuf,   // размер буфера для чтения
  LPDWORD lpcbRead,    // адрес переменной, в которую будет
       // записано количество действительно прочитанных байт
  LPOVERLAPPED lpov);  // адрес структуры типа OVERLAPPED
```

Через параметр hNamedPipe вы должны передать функции TransactNamedPipe идентификатор предварительно открытого канала. Канал следует открыть в режиме передачи сообщений.

Параметры lpvWriteBuf и cbWriteBuf задают, соответственно, адрес и размер буфера, содержимое которого будет передано через канал.

Ответное сообщение, полученное от сервера, будет записано в буфер с адресом lpvReadBuf. Размер этого буфера следует передать функции TransactNamedPipe через параметр cbReadBuf.

После того как функция TransactNamedPipe вернет управление, в переменную, адрес которой передавался через параметр lpcbRead, будет записан размер принятого от сервера сообщения в байтах.

Если операция передачи данных через канал должна выполняться с перекрытием (асинхронно), вы должны подготовить структуру типа OVERLAPPED и передать ее адрес через параметр lpov. В противном случае параметр lpov должен быть задан как NULL.

В случае успешного завершения функция TransactNamedPipe возвращает значение TRUE, а при ошибке - FALSE. Код ошибки можно получить, вызвав функцию GetLastError.

Если возникает ситуация, при которой длина сообщения, полученного от сервера, превосходит размер буфера, предусмотренного процессом, функция

TransactNamedPipe завершится с ошибкой, а функция GetLastError вернет значение ERROR_MORE_DATA. Оставшуюся часть сообщения можно прочитать с помощью такой функции, как ReadFile.

Функция PeekNamedPipe

Чтение данных из канала функцией ReadFile вызывает удаление прочитанных данных. В противоположность этому, функция PeekNamedPipe позволяет получить данные из именованного или анонимного канала без удаления, так что при последующих вызовах этой функции или функции ReadFile будут получены все те же данные, что и при первом вызове функции PeekNamedPipe.

Еще одно отличие заключается в том, что функция PeekNamedPipe никогда не переходит в состояние ожидания, сразу возвращая управление вне зависимости от того, есть данные в канале или нет.

Прототип функции PeekNamedPipe представлен ниже:

```
BOOL PeekNamedPipe(
  HANDLE  hPipe,     // идентификатор канала Pipe
  LPVOID  lpvBuffer, // адрес буфера для прочитанных данных
  DWORD   cbBuffer,  // размер буфера прочитанных данных
  LPDWORD lpcbRead,  // адрес переменной, в которую будет
                     // записано количество действительно
                     // прочитанных байт данных
  LPDWORD lpcbAvail, // адрес переменной, в которую будет
                     // записано общее количество байт данных,
                     // доступных в канале для чтения
  LPDWORD lpcbMessage); // адрес переменной, в которую будет
                     // записано количество непрочитанных
                     // байт в данном сообщении
```

Через параметр hPipe функции PeekNamedPipe нужно передать идентификатор открытого анонимного или именованного канала Pipe.

Данные, полученные из канала, будут записаны в буфер lpvBuffer, имеющий размер cbBuffer байт. При этом количество действительно прочитанных байт будет сохранено в переменной, адрес которой передается функции PeekNamedPipe через параметр lpcbRead.

В случае успешного завершения функция PeekNamedPipe возвращает значение TRUE, а при ошибке - FALSE. Код ошибки можно получить, вызвав функцию GetLastError.

Функция WaitNamedPipe

С помощью функции WaitNamedPipe процесс может выполнять ожидание момента, когда канал Pipe будет доступен для соединения:

```
BOOL WaitNamedPipe(
  LPCTSTR lpszPipeName, // адрес имени канала Pipe
  DWORD   dwTimeout);   // время ожидания в миллисекундах
```

Через параметр lpszPipeName задается имя канала, для которого выполняется ожидание готовности к соединению. Время ожидания в миллисекундах задается через параметр dwTimeout.

Помимо численного значения в миллисекундах, вы можете указать в этом параметре одну из следующих констант:

Константа	Описание
NMPWAIT_WAIT_FOREVER	Ожидание выполняется бесконечно долго
NMPWAIT_USE_DEFAULT_WAIT	Ожидание выполняется в течении периода времени, указанного при вызове функции CreateNamedPipe

Если канал стал доступен до истечения периода времени, заданного параметром dwTimeout, функция WaitNamedPipe возвращает значение TRUE. В противном случае возвращается значение FALSE и вы можете воспользоваться функцией GetLastError.

Функция SetNamedPipeHandleState

При необходимости вы можете изменить режимы работы для уже созданного канала Pipe. Для этого предназначена функция SetNamedPipeHandleState, прототип которой мы привели ниже:

```
BOOL SetNamedPipeHandleState(
  HANDLE  hNamedPipe, // идентификатор канала Pipe
  LPDWORD lpdwMode,   // адрес переменной, в которой указан
                      // новый режим канала
  LPDWORD lpcbMaxCollect, // адрес переменной, в которой
                      // указывается максимальный размер
                      // пакета, передаваемого в канал
  LPDWORD lpdwCollectDataTimeout); // адрес максимальной
                      // задержки перед передачей данных
```

Параметр hNamedPipe задает идентификатор канала Pipe, режим работы которого будет изменен.

Новый режим работы записывается в переменную, адрес которой задан через параметр lpdwMode. Вы можете указать одну из следующих констант, определяющих режим работы канала:

Константа	Использование канала
PIPE_READMODE_BYTE	Канал открывается на чтение в режиме последовательной передачи отдельных байт
PIPE_READMODE_MESSAGE	Канал открывается на чтение в режиме передачи отдельных сообщений указанной длины
PIPE_WAIT	Канал будет работать в блокирующем режиме, когда процесс переводится в состояние ожидания до завершения операций в канале
PIPE_NOWAIT	Неблокирующий режим работы канала. Если операция не может быть выполнена немедленно, в неблокирующем режиме функция завершается с ошибкой

Константы PIPE_WAIT и PIPE_NOWAIT, задающие блокирующий и неблокирующий режим соответственно, можно комбинировать при помощи логической операции ИЛИ с константами PIPE_READMODE_BYTE и PIPE_READMODE_MESSAGE.

Если текущий режим работы канала изменять не нужно, для параметра lpdwMode следует указать значение NULL.

Теперь рассмотрим назначение параметра lpcbMaxCollect.

Если при открытии канала клиентским процессом функцией CreateFile не была указана константа FILE_FLAG_WRITE_THROUGH, то данные передаются пакетами, которые собираются из отдельных сообщений. Размер такого пакета как раз и определяет параметр lpcbMaxCollect.

В том случае, когда вы не собираетесь изменять размер пакета, укажите для параметра lpcbMaxCollect значение NULL.

Параметр lpdwCollectDataTimeout задает максимальный интервал между передачами данных по сети. Если функция SetNamedPipeHandleState изменяет параметры канала со стороны сервера, или если сервер и клиент работают на одном и том же компьютере, параметр lpdwCollectDataTimeout должен быть задан как NULL.

В случае успешного завершения функция SetNamedPipeHandleState возвращает значение TRUE, а при ошибке - FALSE. Код ошибки можно получить, вызвав функцию GetLastError.

Функция GetNamedPipeHandleState

С помощью функции GetNamedPipeHandleState процесс может определить состояние канала Pipe, зная его идентификатор.

Прототип функции GetNamedPipeHandleState мы привели ниже:

```
BOOL GetNamedPipeHandleState(
  HANDLE  hNamedPipe,     // идентификатор именованного канала
  LPDWORD lpState,        // адрес флагов состояния канала
  LPDWORD lpCurInstances, // адрес количества реализаций
  LPDWORD lpMaxCollectionCount, // адрес размера пакета
                          // передаваемых данных
  LPDWORD lpCollectDataTimeout, // адрес максимального
                          // времени ожидания
  LPTSTR  lpUserName,     // адрес имени пользователя
                          // клиентского процесса
  DWORD   nMaxUserNameSize); // размер буфера для
                    // имени пользователя клиентского процесса
```

Идентификатор канала, для которого нужно определить состояние, передается функции GetNamedPipeHandleState через параметр hNamedPipe.

Через параметр lpState нужно передать указатель на переменную типа DWORD, в которую будет записан один из флагов состояния канала:

Флаги состояния	Описание
PIPE_WAIT	Канал будет работать в блокирующем режиме, когда процесс переводится в состояние ожидания до завершения операций в канале
PIPE_NOWAIT	Неблокирующий режим работы канала. Если операция не может быть выполнена немедленно, в неблокирующем

	режиме функция завершается с ошибкой

Если информация о состоянии канала не требуется, в качестве значения для параметра lpState следует использовать константу NULL.

В переменную, адрес которой передается через параметр lpCurInstances, записывается текущее количество реализаций канала. Если эта информация вам не нужна, передайте через параметр lpCurInstances значение NULL.

Параметры lpMaxCollectionCount и lpCollectDataTimeout позволяют определить, соответственно, размер пакета передаваемых данных и максимальное время ожидания между передачами.

Через параметр lpUserName вы должны передать указатель на буфер, в который функция GetNamedPipeHandleState запишет имя пользователя клиентского процесса. Размер этого буфера задается в параметре nMaxUserNameSize.

При необходимости вы можете задать значения параметров lpMaxCollectionCount, lpCollectDataTimeout и lpUserName как NULL. В этом случае соответствующая информация не будет извлекаться.

В случае успешного завершения функция GetNamedPipeHandleState возвращает значение TRUE, а при ошибке - FALSE. Код ошибки можно получить, вызвав функцию GetLastError.

Функция GetNamedPipeInfo

Еще одна функция, позволяющая получить информацию об именованном канале по его идентификатору, называется GetNamedPipeInfo:

```
BOOL GetNamedPipeInfo(
  HANDLE  hNamedPipe,       // идентификатор канала
  LPDWORD lpFlags,          // адрес флагов типа канала
  LPDWORD lpOutBufferSize, // адрес размера выходного буфера
  LPDWORD lpInBufferSize,  // адрес размера входного буфера
  LPDWORD lpMaxInstances); // адрес максимального количества
                           // реализаций канала
```

Параметр hNamedPipe задает идентфикатор именованного канала Pipe, для которого требуется получить информацию.

Через параметр lpFlags функции GetNamedPipeInfo необходимо передать адрес переменной типа DWORD или NULL, если флаги определять не требуется. Ниже мы привели возможные значения флагов:

Флаг	Описание
PIPE_CLIENT_END	Идентификатор ссылается на клиентскую часть канала
PIPE_SERVER_END	Идентификатор ссылается на серверную часть канала
PIPE_TYPE_MESSAGE	Канал работает в режиме передачи сообщений

В переменные, адреса которых задаются через параметры lpOutBufferSize и lpInBufferSize, функция GetNamedPipeInfo заносит размеры входного и выходного буфера, соответственно. Если эта информация не нужна, передайте через параметры lpOutBufferSize и lpInBufferSize значение NULL.

И, наконец, через параметр lpMaxInstances передается адрес переменной, в которую будет записано максимальное значение реализаций, которое можно создать для данного канала. Если после вызова функции GetNamedPipeInfo в этой переменной записано значение PIPE_UNLIMITED_INSTANCES, количество реализаций ограничивается только свободными системными ресурсами.

В случае успешного завершения функция GetNamedPipeInfo возвращает значение TRUE, а при ошибке - FALSE. Код ошибки можно получить, вызвав функцию GetLastError.

Pipe в Java

PipedReader, PipedWriter

```
try {
  // Connect to the pipe
  RandomAccessFile pipe = new
RandomAccessFile("\\\\.\\pipe\\testpipe", "rw");
  String echoText = "Hello word\n";
  // write to pipe
  pipe.write ( echoText.getBytes() );
  // read response
  String echoResponse = pipe.readLine();
  System.out.println("Response: " + echoResponse );
  pipe.close();
} catch (Exception e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
}
```

Windows Pipe в Java

```
try {
// Connect to the pipe
RandomAccessFile pipe = new
RandomAccessFile("\\\\.\\pipe\\testpipe", "rw");
String echoText = "Hello word\n";
// write to pipe
pipe.write ( echoText.getBytes() );
// read response
String echoResponse = pipe.readLine();
System.out.println("Response: " + echoResponse );
pipe.close();

} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
```