*IPC (InterProcess Communication)* - механизмы, предназначенные для обмена данными между процессами и для синхронизации процессов.

**Именованным каналом** называется объект ядра операционной системы, который обеспечивает обмен данными между процессами, выполняющимися на компьютерах в одной локальной сети.

Процесс, создающий именованный канал, называется **сервером именованного канала**. Процессы, которые связываются с именованным каналом, называются **клиентами именованного канала**. Любой именованный канал идентифицируется своим именем, которое задается при создании канала.

**\\servname\pipe\xxxxx**

* **servname -** имя компьютера – сервера именованного канала;
* **pipe -** фиксированное слово;
* **xxxxx -** имя канала

1. при создании канала всегда используется локальный формат имени;
2. если клиент удаленный (на другом компьютере), то он всегда должен использовать сетевой формат имени; при этом обмен данными между клиентом и сервером осуществляется сообщениями;
3. если клиент локальный и использует сетевой формат имени при подсоединении к каналу (функция CreateFile), то обмен данными осуществляется сообщениями;
4. если клиент локальный и использует локальный формат имени канала, то обмен данными осуществляется потоком.

Сервер

Функция **CreateNamedPipe** создает именованный канал с заданными параметрами:

* PIPE\_ACCESS\_DUPLEX: канал будет доступен для чтения и записи.
* PIPE\_TYPE\_MESSAGE: сообщения будут обрабатываться как сообщения.
* PIPE\_READMODE\_MESSAGE: чтение будет происходить в режиме сообщений.
* PIPE\_WAIT: функция блокирует вызов, пока клиент не подключится.
* Параметры PIPE\_UNLIMITED\_INSTANCES, 0, INFINITE, nullptr задают дополнительные свойства канала. (максимально возможное количество экземпляров именованного канала. задает время ожидания (таймаут) в миллисекундах для операций на канале. указывая, что операция (например, подключение клиента или ожидание данных) может продолжаться бесконечно долго.)

Функция **ExchangeMessages** выполняет обмен сообщениями между клиентом и сервером:

* ReadFile считывает данные из канала в rbuf, и возвращает количество прочитанных байтов в bytesRead.
* Если прочитано сообщение длиной 1 байт, равное нулю ('\0'), то сервер завершает работу.
* Если сообщение успешно прочитано, оно выводится на экран, а затем отправляется обратно клиенту с помощью WriteFile.

Функция ExchangeMessages выполняет отправку и получение сообщений:

* Генерируется сообщение для отправки, состоящее из строки "Hello from Client" и номера сообщения.
* Сначала отправляется сообщение через WriteFile.
* Затем производится чтение ответа от сервера через ReadFile.
* Время выполнения обмена сообщениями вычисляется и выводится.
* Отправляется сообщение нулевой длины для завершения работы сервера.

Функция **CreatePipeConnection** создает подключение к существующему именованному каналу:

* GENERIC\_READ | GENERIC\_WRITE: канал будет доступен как для чтения, так и для записи.
* OPEN\_EXISTING: открывает существующий канал.

 **Сервер** создает именованный канал и ждет подключения от клиента.

 **Клиент** подключается к этому каналу и начинает обмениваться сообщениями.

 Сервер считывает сообщение от клиента и отправляет его обратно.

 Этот процесс продолжается, пока не будет отправлено сообщение нулевой длины, указывающее на завершение работы сервера.

 После завершения общения каналы закрываются, и ресурсы освобождаются.

Клиент нп

**CreatePipeConnection** — функция, которая создает подключение к именованному каналу:

* CreateFileA: открывает существующий канал с правами на чтение и запись (GENERIC\_READ | GENERIC\_WRITE).
* Если hPipe равно INVALID\_HANDLE\_VALUE, значит, подключение не удалось, и вызывается HandleError.

Клиент нпт

Для обмена сообщениями по сети может использоваться функция TansactNamedPipe, которая объединяет операции чтения и записи в одну операцию. Такую объединенную операцию называют ***транзакцией***  именованного канала. После обмена данными осуществляется разрыв связи с именованным каналом.

**ExchangeMessagesWithTransact** — функция для обмена сообщениями:

* Создается буфер rbuf для чтения сообщений от сервера.
* clock\_t start и clock\_t end используются для измерения времени передачи сообщений.
* Цикл for выполняется messageCount раз, где:
  + Формируется строка сообщения, которая включает индекс сообщения.
  + **TransactNamedPipe**: отправляет сообщение на сервер и ожидает ответа в одном вызове.
    - Параметры:
      * hPipe: дескриптор канала.
      * (LPVOID)message.c\_str(): указатель на строку сообщения.
      * message.size() + 1: размер отправляемых данных, включая завершающий нулевой символ.
      * rbuf: буфер для полученного сообщения.
      * sizeof(rbuf) - 1: размер буфера для чтения.
      * &bytesRead: указатель на переменную, в которую будет записано количество прочитанных байтов.
      * nullptr: для синхронного выполнения.
  + Если TransactNamedPipe возвращает FALSE, вызывается HandleError.
  + Прочитанное сообщение завершается нулевым символом и выводится на экран.
* Время передачи всех сообщений вычисляется и выводится.
* Отправляется сообщение нулевой длины для завершения работы сервера с помощью TransactNamedPipe.

 **Клиент**:

* Подключается к серверу через именованный канал.
* Отправляет заданное количество сообщений.
* Получает ответы от сервера и выводит их на экран.
* Завершает соединение, отправив сообщение нулевой длины.

Клиент нпст

WaitForPipe — функция, которая ожидает, пока именованный канал станет доступным для подключения.

* WaitNamedPipeA: блокирует выполнение, пока канал не станет доступен.
* Если ожидание не удается, генерируется исключение с сообщением об ошибке.

SendMessageToPipe — функция для отправки сообщения и получения ответа:

* Определяются буферы buffer (для отправляемого сообщения) и answer (для получаемого ответа).
* Формируется строка сообщения, и она копируется в buffer с помощью strncpy\_s, что предотвращает переполнение буфера.
* clock\_t start: запускается таймер для измерения времени передачи.
* **CallNamedPipeA**: используется для отправки сообщения и ожидания ответа от сервера. Параметры:
  + pipeName: имя канала.
  + buffer: указатель на отправляемое сообщение.
  + strlen(buffer) + 1: длина сообщения с завершающим нулевым символом.
  + answer: указатель на буфер для полученного ответа.
  + BUFFER\_SIZE: размер буфера для ответа.
  + &dwRead: указатель на переменную для записи количества прочитанных байтов.
  + 1000: тайм-аут в миллисекундах на операцию.
* Если вызов не удается, выводится сообщение об ошибке. Если успешно, выводится ответ от сервера.
* После этого фиксируется время окончания передачи, и вычисляется время, потраченное на отправку сообщения.

**CallNamedPipe**

BOOL CallNamedPipe(

LPCTSTR nP, // [in] указатель на имя канала

LPVOID pw, // [in] указатель на буфер для записи

DWORD sw, // [in] размер буфера для записи

LPVOID pr, // [out] указатель на буфер для чтения

DWORD sr, // [in] размер буфера для чтения

LPDWORD pr, // [out] количество прочитанных байт

DWORD to // [in] интервал ожидания

);

**Применение**: Используется для синхронного вызова именованного канала. Вызов блокирует выполнение до тех пор, пока сервер не обработает запрос.

**Возврат**: TRUE в случае успешного выполнения, FALSE в случае ошибки.

**Замечание**: Использует синхронный подход и ожидает, пока сервер обработает сообщение, прежде чем продолжить выполнение.

**2. TransactNamedPipe**

BOOL TransactNamedPipe(

HANDLE hP, // [in] дескриптор канала

LPVOID pw, // [in] указатель на буфер для записи

DWORD sw, // [in] размер буфера для записи

LPVOID pr, // [out] указатель на буфер для чтения

DWORD sr, // [in] размер буфера для чтения

LPDWORD pr, // [out] количество прочитанных байт

LPOVERLAPPED ol // [in,out] для асинхронного доступа

);

**Применение**: Используется для синхронной передачи данных с поддержкой транзакционного обмена, что позволяет отправлять и получать данные в одном вызове.

**Возврат**: TRUE в случае успешного выполнения, FALSE в случае ошибки.

**Замечание**: Позволяет отправлять данные и сразу же получать ответ, что удобно для транзакционного обмена. Может использоваться с асинхронным доступом, если указать OVERLAPPED структуру.

**3. ReadFile**

BOOL ReadFile(

HANDLE hP, // [in] дескриптор канала

LPVOID pb, // [out] указатель на буфер ввода

DWORD sb, // [in] количество читаемых байт

LPDWORD ps, // [out] количество прочитанных байт

LPOVERLAPPED ol // [in,out] для асинхронной обработки

);

**Применение**: Используется для чтения данных из канала, уже установленного с сервером.

**Возврат**: TRUE в случае успешного выполнения, FALSE в случае ошибки.

**Замечание**: Читает данные из уже существующего соединения. Не инициирует соединение, а только считывает данные.

**4. WriteFile**

BOOL WriteFile(

HANDLE hP, // [in] дескриптор канала

LPVOID pb, // [in] указатель на буфер вывода

DWORD sb, // [in] количество записываемых байт

LPDWORD ps, // [out] количество записанных байт

LPOVERLAPPED ol // [in,out] для асинхронной обработки

);

* **Применение**: Используется для записи данных в именованный канал, который был открыт ранее.
* **Возврат**: TRUE в случае успешного выполнения, FALSE в случае ошибки.
* **Замечание**: Записывает данные в уже установленное соединение, не инициируя его.

**CreateNamedPipe** создает именованный канал и подготавливает его к ожиданию соединений. Это базовая функция для сервера именованных каналов. После создания канала можно вызывать **ConnectNamedPipe** для ожидания клиента.

**Основные параметры:**

* lpName: Имя канала (в формате \\.\pipe\<имя>).
* dwOpenMode: Режим доступа (дуплексный, односторонний и т.д.).
* dwPipeMode: Режим передачи данных (буферизация, сообщение и т.д.).
* Другие параметры задают, например, количество экземпляров канала.

**Использование:**

Сервер создает канал через **CreateNamedPipe**, затем ждет подключений клиентов через **ConnectNamedPipe**.

**TransactNamedPipe** Эта функция используется для выполнения атомарной (неделимой) операции чтения и записи через именованный канал.

**Описание:**

* Клиент записывает данные в канал, и сервер отвечает в рамках одного вызова функции.
* Это упрощает взаимодействие, так как позволяет избежать явного чтения и записи (всё происходит одним вызовом).

**Основные параметры:**

* hNamedPipe: Дескриптор открытого канала.
* lpInBuffer: Буфер для записи данных (от клиента к серверу).
* nInBufferSize: Размер входного буфера.
* lpOutBuffer: Буфер для чтения ответа от сервера.
* nOutBufferSize: Размер выходного буфера.
* lpBytesRead: Количество прочитанных байт.
* lpOverlapped: Структура для асинхронных операций (необязательно).

**Особенности:**

* Это атомарная операция: сервер обрабатывает запрос и возвращает ответ без дополнительных вызовов чтения/записи.
* Удобно для простого обмена запросами и ответами.

**CallNamedPipeA**

Эта функция представляет собой высокоуровневую обертку для использования именованных каналов с клиентской стороны.

**Описание:**

* Выполняет атомарную операцию соединения, записи, чтения и разъединения канала.
* Не требует явного подключения или отключения клиента.

**Основные параметры:**

* lpNamedPipeName: Имя канала.
* lpInBuffer: Буфер для записи данных (от клиента).
* nInBufferSize: Размер входного буфера.
* lpOutBuffer: Буфер для чтения данных (ответ от сервера).
* nOutBufferSize: Размер выходного буфера.
* lpBytesRead: Количество прочитанных байт.
* nTimeOut: Таймаут операции (в миллисекундах).

**Особенности:**

* Удобна для быстрого взаимодействия: клиент выполняет всю работу одним вызовом.
* Подразумевает, что сервер уже ожидает подключения.