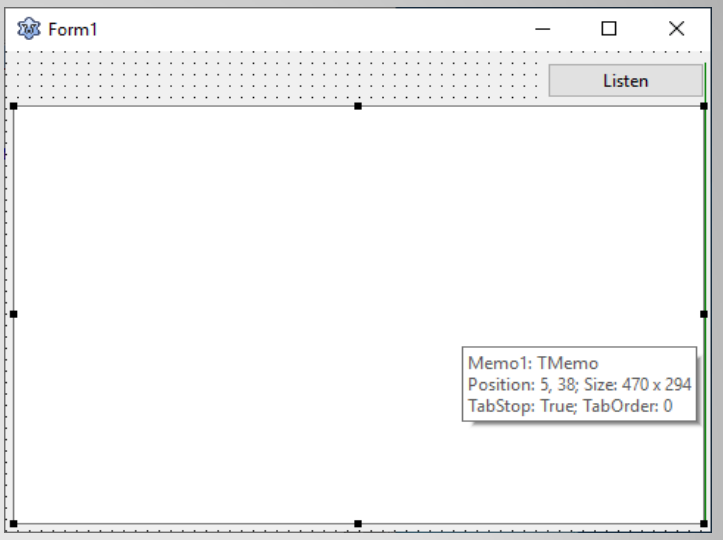
Именованные каналы (Named pipe)

Цели:

- Получить навыки создания именованного канала
- Получить навыки подключения к именованному каналу
- Получить навыки обмена данными между приложениями по именованному каналу

Создание канала (Сервер)

Создадим новый проект, на главной форме разместим компонент TMemo (Memo1) и TButton (Button1).



Добавим библиотеку Windows в раздел uses.
Данное приложение будет компонентом сервера, которое создает именованный канал (named pipe) и прослушивает его в ожидании подключения клиентов.
После подключения клиента сервер отображает полученное от него сообщение в Memo1 и отправляет клиенту подтверждение получения сообщения.

API именованных каналов

Для создания и работы с именованными каналами используются функции windows API:
CreateNamedPipe(имя, доступ, тип, количество, размер буфера1, размер буфера2,таймаут,дескриптор безопасности) – создание именованного канала.
Рассмотрим наиболее важные параметры подробнее:
имя – имя создаваемого канала в формате \\сервер\pipe\имя_канала. Здесь сервер - это имя компьютера, на котором открыт канал. Вслучае локальной машины (localhost) обозначается точкой.
доступ - режим доступа к каналу: дуплексный, входной или выходной.
тип – режим работы канала: блочный (message) или побайтовый (byte).
дескриптор безопасности – содержит указания по передаче прав на именованный канал дочерним процессам и потокам.

API именованных каналов

Ожидание клиента:
ConnectNamedPipe(handle,режим) – ожидание соединения клиента
handle – дескриптор канала полученный от CreateNamedPipe.
режим – указатель на структуру данных в асинхронном режиме работы.

На стороне клиента подключение к каналу осуществляется как к файлу с помощью функции CreateFile, которой в качестве параметров вместо имени файла передается имя канала, а вместо типа доступа (доступ) режим доступа к файлу (чтение/запись).

Обмен данными на обоих сторонах (клиент и сервер) осуществляется функциями чтения записи для файлов ReadFile и WriteFile. В качестве идентификатора файла (Handle) выступает идентификатор канала.

API именованных каналов

Со стороны клиента используются следующие функции:
WaitNamedPipe(имя,таймаут) – ожидание создания канала сервером. Как только канал создан, функция возвращает истину.
имя – имя канала, совпадает с именем в CreateNamedPipe сервера.
таймаут – время ожидания в миллисекундах.

SetNamedPipeHandleState(pipe,dwMode,nil,nil) – установка режима работы с каналом(dwMode) со стороны клиента. Для правильной синхронной работы, режимы работы канала клиента и сервера должны совпадать.

DisconnectNamedPipe(handle) – отключение от канала (отключение клиента)

CloseHandle(handle) – закрытие объекта (канала)

Прослушивание запросов (сервер)

Напишем код сервера для создания канала и его прослушивания. Этот код поместим в событие нажатия на единственную кнопку приложения сервера:

```
procedure TForm1.Button1Click(Sender: TObject);
  var
    SD: SECURITY_DESCRIPTOR;
    SA: SECURITY_ATTRIBUTES;
    pipe : THandle;
    s : String;
    nbyte : DWORD;
const
    PIPE_UNLIMITED_INSTANCES = 255;
```

Прослушивание запросов (сервер)

```
begin
  InitializeSecurityDescriptor(@SD,
  SECURITY_DESCRIPTOR_REVISION);
  SetSecurityDescriptorDacl(@SD, true, nil{ACL}, false);
  SA.lpSecurityDescriptor := @SD;
  SA.nLength := SizeOf(SA);
  SA.bInheritHandle := true;
                    // Создание канала
  pipe:=CreateNamedPipe(PChar('\\.\pipe\testpipe'),
                        PIPE_ACCESS_DUPLEX,
                        PIPE_TYPE_BYTE or PIPE_WAIT,
                        PIPE_UNLIMITED_INSTANCES,
                        1024, 1024, 50, @SA);
```

Прослушивание запросов (сервер)

```
// Проверка на корректное создание канала

if pipe=INVALID_HANDLE_VALUE then
    begin
      Memo1.Lines.Add('Error '+IntToStr(GetLastError()));
      exit;
    end else
    Memo1.Lines.Add('Create ok');
```

Прослушивание запросов (сервер)

```
s:=' ';                          // Главный цикл прослушивания
while true do
  begin                    // Ожидание соединения
    Memo1.Lines.Add('Awaiting for connection .. ');
    if not ConnectNamedPipe(pipe,nil) then
      begin
        Memo1.Lines.Add('Connect error '
                        +IntToStr(GetLastError()));
        break;
      end else
      Memo1.Lines.Add('Connect ok');
```

Прослушивание запросов (сервер)

```
// Чтение данных сервера
      if not ReadFile(pipe,s[1],255,nbyte,nil) then
        Memo1.Lines.Add('Read error '
                                +IntToStr(GetLastError())) else
        begin
          setlength(s,nbyte); // Длина по длине принятого
          Memo1.Lines.Add('Got: '''+s+''''');
        end;
```

Прослушивание запросов (сервер)

```
// Ответ серверу
  s:='Server got '+s;
     if  not  WriteFile(Pipe,  s[1],  Length(s),  {out}nbyte,  nil)
  then
       Memo1.Lines.Add('Write error'
                       +IntToStr(GetLastError())) else
      Memo1.Lines.Add('Write ok');
     FlushFileBuffers(pipe);
     DisconnectNamedPipe(pipe); // Конец сеанса
```

Прослушивание запросов (сервер)

```
  // Если команда DONE, завершаем прослушивание
     if s='Server got DONE' then
       begin
         Memo1.Lines.Add('Closing ..');
         break;
       end;
     Application.ProcessMessages;
   end;
  CloseHandle(pipe);    // Закрываем канал
end;
```

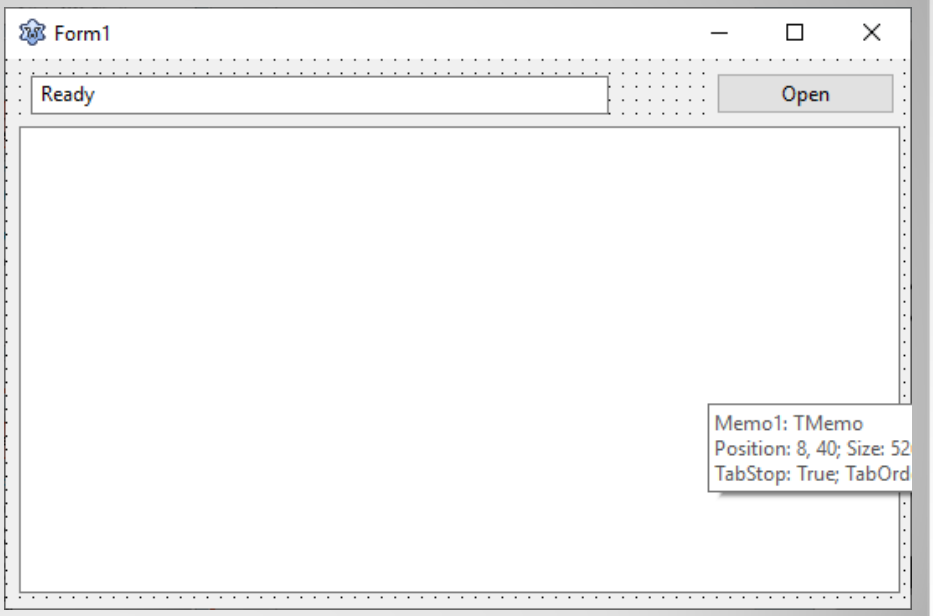
Создание канала (Клиент)

Создадим новый проект, на главной форме разместим компонент TMemo (Memo1), Tedit (Edit1) и TButton (Button1).

Добавим библиотеку Windows в раздел uses.

Данное приложение будет клиентом, который подключается к именованному каналу (named pipe) созданному сервером.

После подключения, клиент отсылает ему сообщение из Edit1 и записывает полученное подтверждение в Memo1.



Сеанс связи (клиент)

Напишем код клиента для подключение к каналу. Этот код поместим в событие нажатия на единственную кнопку приложения клиента:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  SD: SECURITY_DESCRIPTOR;
  SA: SECURITY_ATTRIBUTES;
  dwMode: DWORD;
  cb    : ULONG;
  pipe  : THandle;
  s : String;
  nbyte : DWORD;
```

Сеанс связи (клиент)

```
// Инициализация Security Descriptor
begin
  cb:=GetSidLengthRequired(1);

  InitializeSecurityDescriptor(@SD,
   SECURITY_DESCRIPTOR_REVISION);
  SetSecurityDescriptorDacl(@SD, true, nil{ACL}, false);
  SA.lpSecurityDescriptor := @SD;
  SA.nLength := SizeOf(SA);
  SA.bInheritHandle := true;
```

Сеанс связи (клиент)

```
// Ожидаем создания канала на стороне сервера
Memo1.Lines.Add('Waiting pipe .. ');
  if   not   WaitNamedPipe(PChar('\\.\pipe\testpipe'),5000)
   then
    begin
     Memo1.Lines.Add('Wait error '
                              +IntToStr(GetLastError()));
     exit;
    end else
    Memo1.Lines.Add('Wait ok');
```

Сеанс связи (клиент)

```
// Если дождались – подключаемся к каналу
  pipe:=CreateFile(PChar('\\.\pipe\testpipe'),
                   GENERIC_READ or GENERIC_WRITE,
                   0,@SA,OPEN_EXISTING,0,0);
// Проверяем, удалось ли подключиться
   if pipe=INVALID_HANDLE_VALUE then
     begin
       Memo1.Lines.Add('Error '+IntToStr(GetLastError()));
       exit;
     end else
     Memo1.Lines.Add('Open ok');
```

Сеанс связи (клиент)

```
// Настраиваем режим работы канала
  dwMode:=PIPE_TYPE_BYTE;
  if    not    SetNamedPipeHandleState(pipe,dwMode,nil,nil)
  then
  Memo1.Lines.Add('Setmode error '
                      +IntToStr(GetLastError())) else
  Memo1.Lines.Add('Setmode ok');
```

Сеанс связи (клиент)

```
// Отправляем данные серверу (содержимое Edit1)
  s:=Edit1.Text;
  if not WriteFile(Pipe, s[1], Length(s), {out}nbyte, nil) then
    Memo1.Lines.Add('Write error '+
                        IntToStr(GetLastError())) else
    Memo1.Lines.Add('Write ok');
//   FlushFileBuffers(pipe); - попробовать добавить
//  и посмотреть что измениться
```

Сеанс связи (клиент)

```
// Принимаем ответ сервера
   if not ReadFile(pipe,s[1],255,nbyte,nil) then
     Memo1.Lines.Add('Read error '+
                     IntToStr(GetLastError())) else
     begin
       setlength(s,nbyte);
       Memo1.Lines.Add('Respond: '+s);
     end;
   CloseHandle(pipe); // Закрываем канал
end;
```