Анонимные и именованные каналы

Каналы

Канал — это разновидность псевдофайла, которым, как показано на рисунке, можно воспользоваться для соединения двух процессов.

Два процесса, соединенные каналом

Обмен данными через канал

Если процессам A и B необходимо обменяться данными с помощью канала, то они должны установить его заранее. Когда процессу A нужно отправить данные процессу B, он осуществляет запись в канал, как будто он имеет дело с выходным файлом. Процесс B может прочитать данные, осуществляя операцию чтения из канала, как будто он имеет дело с входным файлом.

Отличительные свойства канала

- Невозможен доступ по имени
- Канал не существует вне процесса
- Последовательный доступ к данным FIFO

Анонимные каналы

Создается автоматически операционной системой при создании нового дочернего процесса. Он представляет собой однонаправленный поток данных, который связывает родительский процесс с дочерним процессом.

Анонимные каналы

- Однонаправленность
- Автоматическое создание
- Ограниченность
- Простота использования

Именованные каналы

Позволяет обмниваться данными между любыми процессами, даже теми, которые не связаны друг с другом через fork().

Именованные каналы

Универсальность

Создание вручную

Двустороннее взаимодействие

Отображаются в файловой системе

C#

PipeStream

NamedPipeClientStream

NamedPipeServerStream

PipeStream

Используется для создания каналов связи между двумя процессами. Это позволяет передавать данные от одного потока к другому через именованный канал.

PipeStream

PipeStream состоит из двух частей:

- Серверная часть – принимает входящие соединения и обрабатывает их

- Клиентская часть – инициирует соединение с серверной частью и отправляет/принимает данные

PipeServerStream

Предоставляет возможность создавать и управлять серверными именованными каналами. Предназначен для реализации серверной стороны коммуникации между процессами через именованные каналы.

Конструктор

PipeServerStream(string pipeName, PipeDirection direction, int maxNumberOfServerInstances, PipeTransmisionMode transmissionMode, PipeOption options);

Параметры конструктора

pipeName: имя канала

direction: направление передачи данных (In, Out< InOut

maxNumberOfServerInstances: максимальное кол-во экземпляров сервера, которое может обслуживать данный канал одновременно

Параметры конструктора

transmissionMode: Режим передачи данных (Byte, Message)

options: дополнительные опции канала

Методы и свойства

WaitForConnection() ожидает подключение клиента к серверу. Блокируется до тех пор, пока клиент не установит соединение

Read() чтение канала

Write() запись канала

isConnected подключен ли клиент к серверу

Методы и свойства

Close() закрывает экземпляр канала и освобождает ресурсы

PipeClientStream

представляет собой клиентскую сторону именованного канала (pipe), который используется для межпроцессного взаимодействия. Этот класс позволяет клиентам подключаться к существующему серверному каналу и отправлять или получать данные.

Конструктор

public PipeClientStream(string serverName, string pipeName, PipeDirection direction, PipeOptions options);

Параметры конструкторов

serverName: Имя сервера, к которому осуществляется подключение.
pipeName: Имя канала, к которому подключается клиент.
direction: Направление передачи данных(In, Out, InOut)
options: Опции канала, такие как Asynchronous, WriteThrough и другие.

Методы и свойства

Connect() устанавливает соединение с серверным каналом. Если сервер недоступен, метод может заблокироваться до тех пор, пока сервер не станет доступен

Read() и Write() методы для чтения и записи данных в канал.

IsConnected подключен ли клиент к серверу

Close()Закрывает текущий экземпляр канала

Методы и свойства

Connect() устанавливает соединение с серверным каналом. Если сервер недоступен, метод может заблокироваться до тех пор, пока сервер не станет доступен

Read() и Write() методы для чтения и записи данных в канал.

IsConnected подключен ли клиент к серверу

Close()Закрывает текущий экземпляр канала

1. Создание сервера

```
//создает именованный канал
NamedPipeServerStream server = new NamedPipeServerStream("myPipe");
//ожидаем подключение клиента
server.WaitForConnection();
```

2. Чтение данных от клиента

```
server.WaitForConnection();

byte[] buffer = new byte[1024];

int bytesRead = server.Read(buffer, 0, buffer.Length);

string message = Encoding.UTF8.GetString(buffer, 0, bytesRead);

Console.WriteLine("Message: " + message);
```

3. Отправка данных клиенту

```
string response = "Hello from server!";
byte[] responseBytes = Encoding.UTF8.GetBytes(response);
server.Write(responseBytes, 0, responseBytes.Length);
```

4. Закрытие соединения

```
server.Close();
```

Клиентская сторона

На стороне клиента процесс аналогичен, но вместо ожидания подключения он инициирует его.

1. Подкючение к серверу

```
//Подключаемся к серверному каналу

NamedPipeClientStream client = new
NamedPipeClientStream(".", "myPipe", PipeDirection.InOut);

//Устанавливаем соединение

client.Connect();
```

2. Отправка данных на сервер

```
string message = "Hello from client!";

byte[] messageBytes = Encoding.UTF8.GetBytes(message);

client.Write(messageBytes, 0, messageBytes.Length);
```

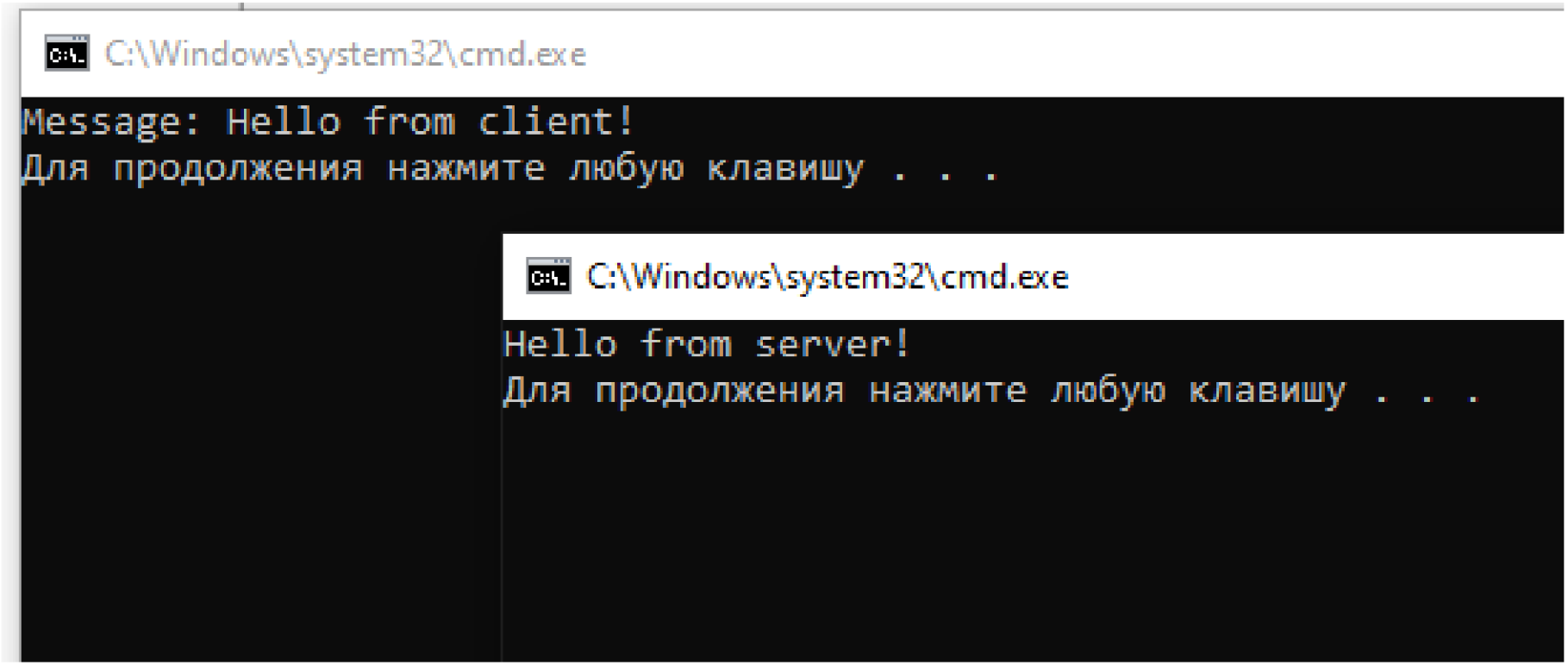
3. Получение ответа от сервера

```
byte[] buffer = new byte[1024];

int bytesRead = client.Read(buffer, 0, buffer.Length);

string response = Encoding.UTF8.GetString(buffer, 0,
bytesRead);

Console.WriteLine(response);
```

4. Закрытие соединения

```
client.Close();
```

Результат


C:\Windows\system32\cmd.exe
Message: Hello from client!
Для продолжения нажмите любую клавишу . . .
C:\Windows\system32\cmd.exe
Hello from server!
Для продолжения нажмите любую клавишу . . .

Задание

Изменить клиентское приложение так, чтоб у пользователя была возможность указать свое имя и ввести сообщение. На сервере должно отобразиться имя пользователя и сообщение.