***Именнованые каналы: назначение, область применения, их использование.***

***Создать метод CalculateSalary для расчета заработной платы сотрудника по количеству отработанных дней и месяцев соответственно. Учитывать, что оклад рассчитывается на 23 рабочих дня.***

**Назначение именованных каналов в C#** — передача данных между процессами, выполняющимися на компьютерах в одной локальной сети, или между процессами на одном компьютере.

**Область применения именованных каналов включает, например:**

синхронизацию процессов;

связь клиент-сервер;

передачу данных по сценариям оболочки;

безопасное туннелирование данных по SSH;

ведение журнала в реальном времени;

конвейеры обработки данных.

**Для реализации** именованных каналов используются классы NamedPipeServerStream и NamedPipeClientStream.

**Порядок работы с именованными каналами:**

Создание именованного канала сервером.

Ожидание сервера подключения клиента к экземпляру именованного канала.

Создание именованного канала клиентом.

Соединение клиента с экземпляром именованного канала.

Обмен данными по именованному каналу.

Закрытие именованного канала клиентом и сервером.

**Именованные каналы** (named pipes) представляют собой механизм межпроцессного взаимодействия (IPC), который позволяет устанавливать двунаправленное соединение между процессами, работающими как на одной, так и на разных машинах в сети. В отличие от анонимных каналов, именованные каналы имеют имя, которое позволяет процессам находить друг друга и устанавливать связь.

**Назначение:**

Двунаправленная связь: Именованные каналы обеспечивают двунаправленный обмен данными между процессами, что позволяет процессам как отправлять, так и получать данные.

Взаимодействие между процессами, не имеющими отношения “родитель-потомок”: В отличие от анонимных каналов, именованные каналы могут использоваться для связи между любыми процессами, если они имеют доступ к одному и тому же имени канала.

Коммуникация между процессами на разных машинах (в сети): Именованные каналы могут использоваться для установления связи между процессами, работающими на разных компьютерах в сети, что делает их полезным инструментом для распределенных приложений.

**Область применения:**

Взаимодействие между службами и приложениями: Именованные каналы часто используются для связи между службами Windows и приложениями, работающими на том же или другом компьютере.

Межпроцессное взаимодействие в рамках одной системы: Именованные каналы могут использоваться для организации обмена данными между разными приложениями, работающими на одной машине.

Распределенные системы: Именованные каналы могут быть частью архитектуры распределенных систем, обеспечивая обмен данными между различными компонентами.

Клиент-серверная архитектура: Именованный канал может использоваться в качестве механизма связи между сервером и клиентами.

Реализация плагинной архитектуры: Основное приложение может использовать именованный канал для взаимодействия с плагинами, работающими в отдельных процессах.

**Использование в C#:**

В .NET именованные каналы представлены классами NamedPipeServerStream и NamedPipeClientStream в пространстве имен System.IO.Pipes.

**Пример: Клиент-серверное взаимодействие с использованием именованных каналов**

**Сервер (NamedPipeServer.exe):**

using System;

using System.IO;

using System.IO.Pipes;

using System.Text;

using System.Threading;

public class NamedPipeServer

{

public static void Main(string[] args)

{

string pipeName = "MyNamedPipe";

Console.WriteLine($"[Сервер]: Запускаю сервер с именем канала: {pipeName}");

while (true)

{

try

{

// 1. Создаем именованный канал

using (NamedPipeServerStream pipeServer = new NamedPipeServerStream(pipeName, PipeDirection.InOut, 1, PipeTransmissionMode.Byte, PipeOptions.Asynchronous))

{

// 2. Ожидаем подключения клиента

Console.WriteLine("[Сервер]: Ожидание подключения клиента...");

pipeServer.WaitForConnection();

Console.WriteLine("[Сервер]: Клиент подключился.");

// 3. Обработка запросов клиента (в отдельном потоке)

ThreadPool.QueueUserWorkItem((state) => HandleClient(pipeServer));

}

}

catch (Exception ex)

{

Console.WriteLine($"[Сервер]: Ошибка: {ex.Message}");

}

// Задержка для избежания избыточного потребления ресурсов

Thread.Sleep(1000);

}

}

private static void HandleClient(NamedPipeServerStream pipeServer)

{

try

{

using (StreamReader reader = new StreamReader(pipeServer, Encoding.UTF8))

using (StreamWriter writer = new StreamWriter(pipeServer, Encoding.UTF8))

{

writer.AutoFlush = true;

while (pipeServer.IsConnected)

{

// Читаем запрос от клиента

string messageFromClient = reader.ReadLine();

if (messageFromClient == null)

{

Console.WriteLine("[Сервер]: Клиент разорвал соединение.");

break; // Клиент отключился

}

Console.WriteLine($"[Сервер]: Получено от клиента: {messageFromClient}");

// Отправляем ответ клиенту

string responseMessage = $"Echo from Server: {messageFromClient}";

writer.WriteLine(responseMessage);

Console.WriteLine($"[Сервер]: Отправлен ответ клиенту: {responseMessage}");

// Проверка на команду завершения сеанса

if (messageFromClient.Equals("exit", StringComparison.OrdinalIgnoreCase))

{

Console.WriteLine("[Сервер]: Клиент запросил завершение сеанса.");

break;

}

}

}

}

catch (Exception ex)

{

Console.WriteLine($"[Сервер]: Ошибка при обработке клиента: {ex.Message}");

}

finally

{

Console.WriteLine("[Сервер]: Завершение обработки клиента.");

}

}

}

**Клиент (NamedPipeClient.exe):**

using System;

using System.IO;

using System.IO.Pipes;

using System.Text;

public class NamedPipeClient

{

public static void Main(string[] args)

{

string pipeName = "MyNamedPipe";

Console.WriteLine($"[Клиент]: Подключаюсь к каналу с именем: {pipeName}");

try

{

// 1. Подключаемся к именованному каналу

using (NamedPipeClientStream pipeClient = new NamedPipeClientStream(".", pipeName, PipeDirection.InOut, PipeOptions.Asynchronous))

{

// Соединяемся с сервером

Console.WriteLine("[Клиент]: Ожидание соединения с сервером...");

pipeClient.Connect(5000); // 5 секунд таймаут на соединение

Console.WriteLine("[Клиент]: Соединен с сервером.");

// 2. Обмениваемся данными с сервером

using (StreamWriter writer = new StreamWriter(pipeClient, Encoding.UTF8))

using (StreamReader reader = new StreamReader(pipeClient, Encoding.UTF8))

{

writer.AutoFlush = true;

while(true)

{

Console.Write("[Клиент]: Введите сообщение (или 'exit' для завершения): ");

string messageToSend = Console.ReadLine();

writer.WriteLine(messageToSend);

Console.WriteLine($"[Клиент]: Отправил сообщение: {messageToSend}");

if (messageToSend.Equals("exit", StringComparison.OrdinalIgnoreCase))

{

Console.WriteLine("[Клиент]: Завершаю работу.");

break;

}

string responseFromServer = reader.ReadLine();

Console.WriteLine($"[Клиент]: Получен ответ от сервера: {responseFromServer}");

}

}

}

}

catch (TimeoutException)

{

Console.WriteLine("[Клиент]: Превышено время ожидания подключения.");

}

catch(Exception ex)

{

Console.WriteLine($"[Клиент]: Ошибка: {ex.Message}");

}

finally

{

Console.WriteLine("[Клиент]: Завершил работу.");

}

}

}

**Пояснения:**

**Сервер:**

Создает NamedPipeServerStream с указанным именем канала и направлением InOut (для двунаправленной связи).

Использует цикл while(true) для постоянного прослушивания подключений.

WaitForConnection() блокирует выполнение до подключения клиента.

Создаёт новый поток для обработки запросов каждого клиента с помощью ThreadPool.QueueUserWorkItem.

Внутри обработчика, использует StreamReader и StreamWriter для чтения и записи данных.

**Клиент:**

Создает NamedPipeClientStream с именем сервера (“.”) и именем канала.

Connect() пытается установить соединение с сервером, принимает таймаут в миллисекундах.

Использует StreamWriter для отправки сообщений и StreamReader для приема ответов.

**Ключевые моменты:**

Имя канала: Имя канала должно совпадать на сервере и клиенте, чтобы они могли установить связь.

Направление канала: Используйте PipeDirection.InOut для двунаправленного обмена.

Асинхронная работа: Можно использовать асинхронные методы BeginRead/EndRead и BeginWrite/EndWrite для более эффективной обработки данных, особенно при больших объемах.

Обработка ошибок: Обработайте возможные ошибки, такие как потеря соединения, тайм-ауты и т.д.

Безопасность: Именованные каналы могут быть настроены для ограничения доступа и проверки подлинности, если это необходимо.

**Преимущества именованных каналов:**

Двунаправленная связь: Позволяют обмениваться данными в обоих направлениях.

Гибкость: Могут использоваться между любыми процессами, а не только между родительским и дочерним процессами.

Поддержка сетевых соединений: Могут использоваться для связи между процессами на разных компьютерах.

**Недостатки именованных каналов:**

Сложность: Сложнее в использовании, чем анонимные каналы.

Настройка: Требуется явная настройка имени канала и прав доступа.