# C# Professional

## Межпроцессное взаимодействие

otus.ru


Проверить, идет ли запись

# Меня хорошо видно && слышно?

Ставим “+”, если все хорошо
“-”, если есть проблемы

Тема вебинара

# Межпроцессное взаимодействие


**Александр Усманов**

https://t.me/AlexanderUsmanov


# Правила вебинара

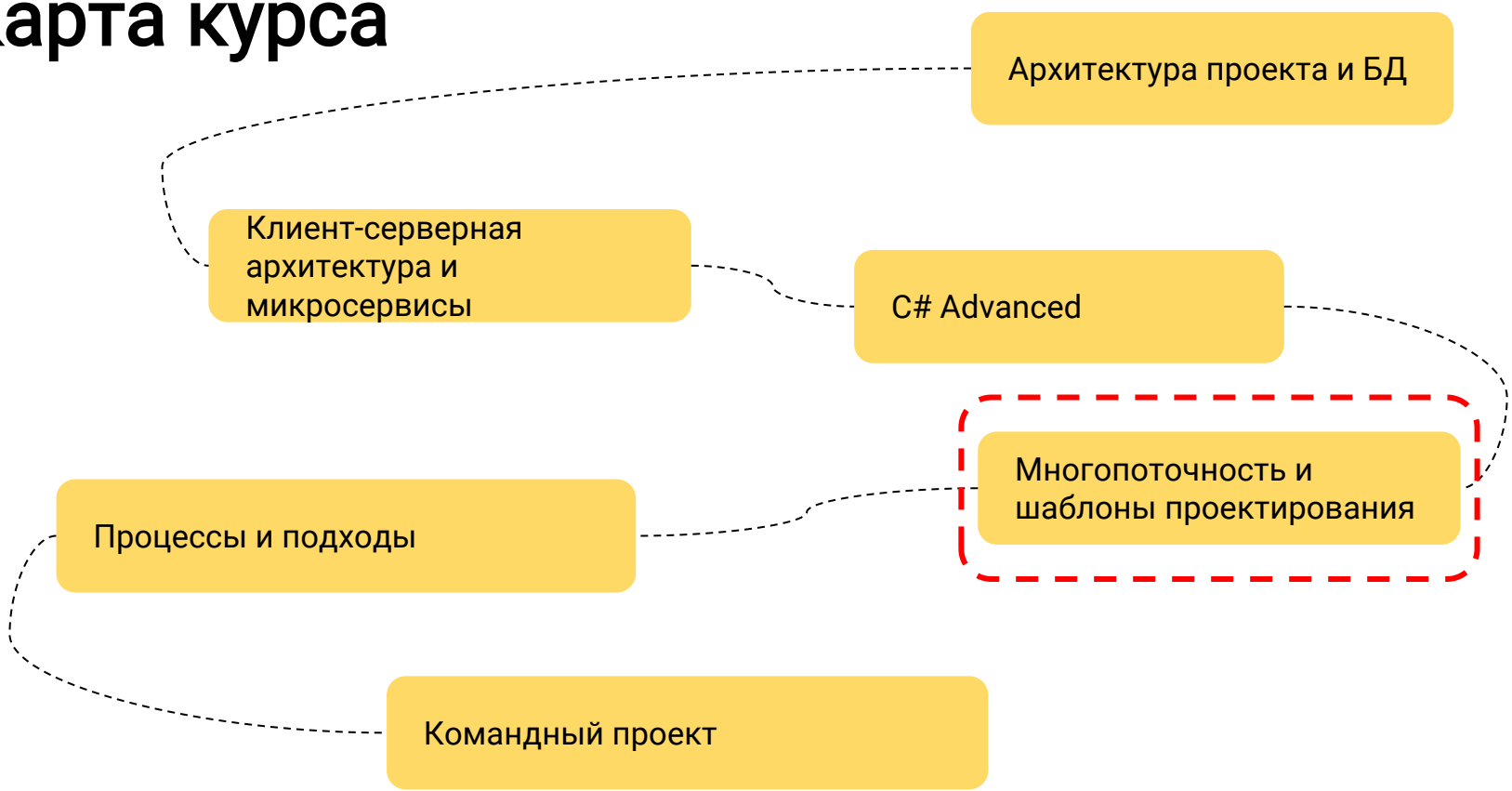
Активно
участвуем

Off-topic обсуждаем
в общем чате учебной группы
в telegram

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

# Карта курса


Архитектура проекта и БД
Клиент-серверная архитектура и микросервисы
C# Advanced
Многопоточность и шаблоны проектирования
Процессы и подходы
Командный проект

# Маршрут вебинара

- Знакомство
- IPC
- Shared Memory, Socket, Pipes
- RPC,WCF, gRPC
- Примеры
- Рефлексия

# Цели вебинара

## К концу занятия вы сможете

1. Знать основные способы организации IPC
2. Понимать как оценивается эффективность организации IPC
3. Применять некоторые способы IPC в своих проектах

# Смысл

## Зачем вам это уметь

1. Обеспечивать межпроцессное взаимодействие в ваших решениях
2. Выбирать наиболее оптимальные решения для обеспечения межпроцессного взаимодействия

# IPC

# IPC - inter-process communication

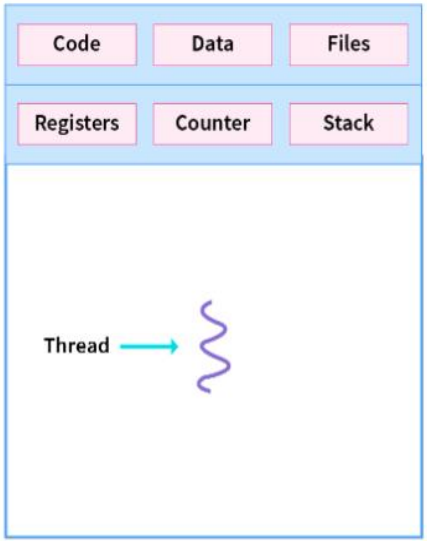
**Межпроцессное взаимодействие** - механизм , позволяющий процессам обмениваться данными и координировать свои действия.

Для современных ОС механизмы IPC являются ключевыми, обеспечивая возможность взаимодействия множества потоков в системе и совместного использования ими ресурсов при необходимости.

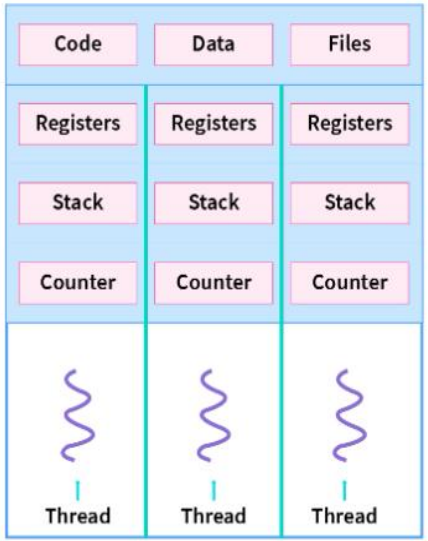
Процессы, вовлеченные в IPC, могут принадлежать как одному приложению так и совершенно разным программам. IPC может использоваться, как в пределах одной локальной машины, так и в распределенной системе.

IPC нацелен на обеспечение согласованности данных, совместное использование ресурсов и координацию при работе в мультизадачных средах

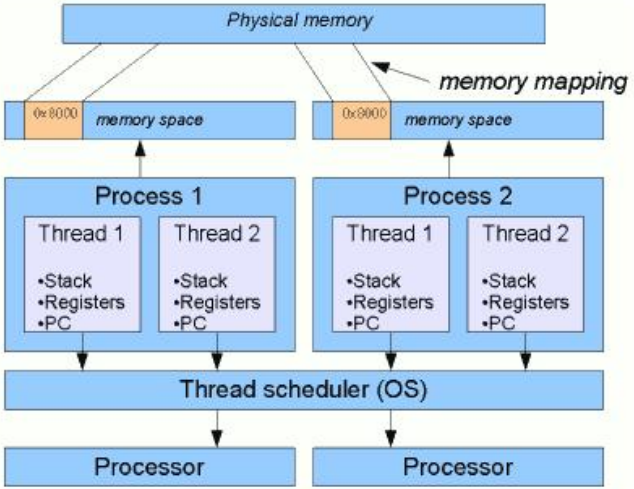
# Процессы



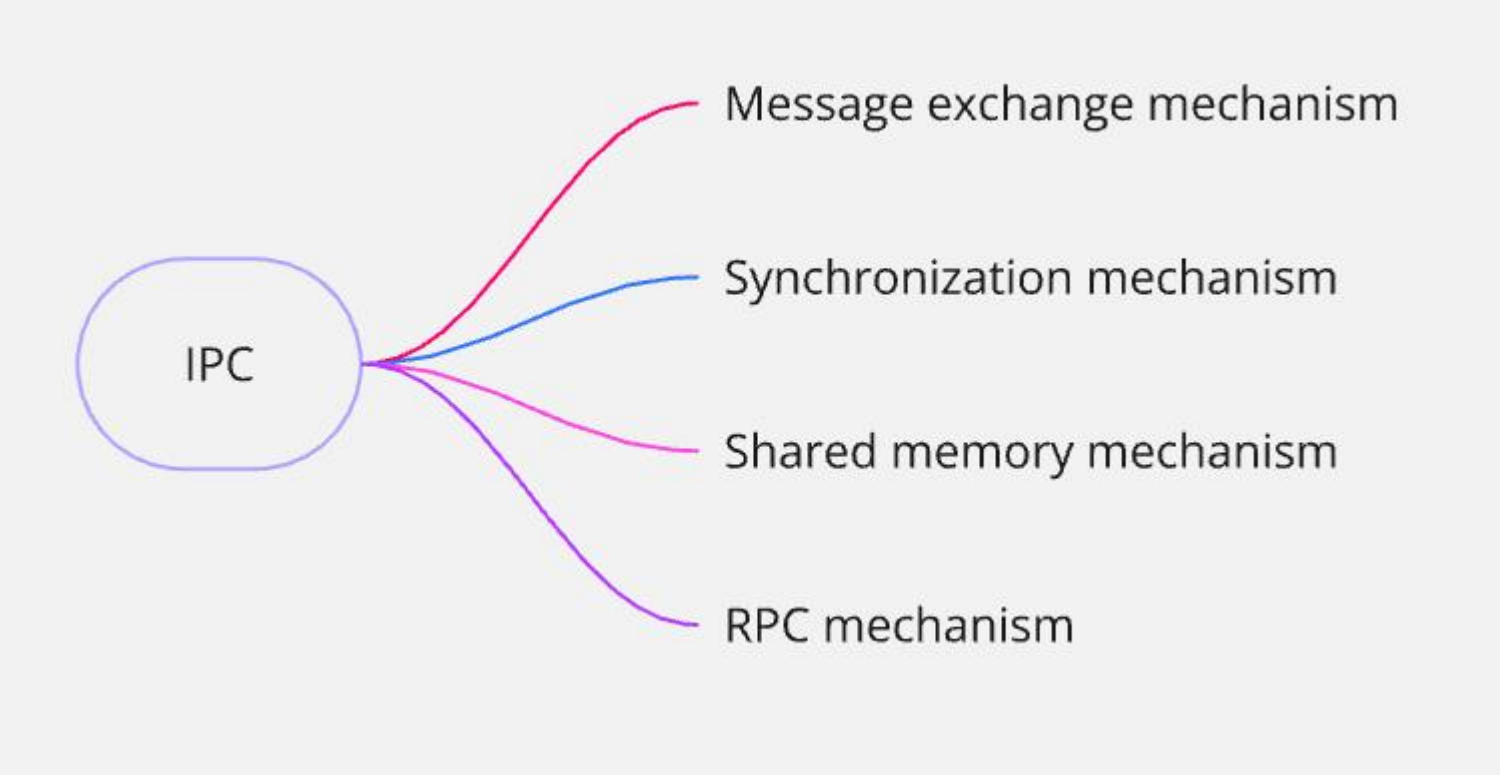

Single-threaded process




Multithreaded process

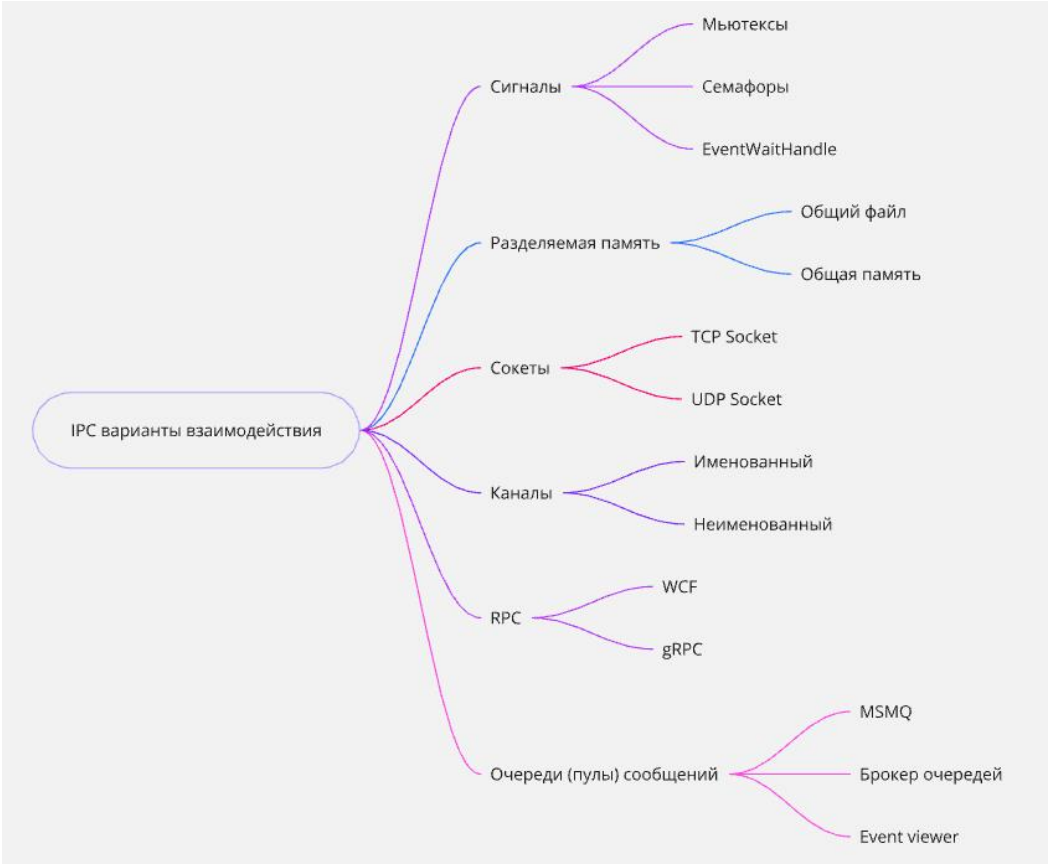


# IPC - механизмы межпроцессного взаимодействия



# IPC - оценка эффективности организации межпроцессного взаимодействия

# IPC - варианты взаимодействия



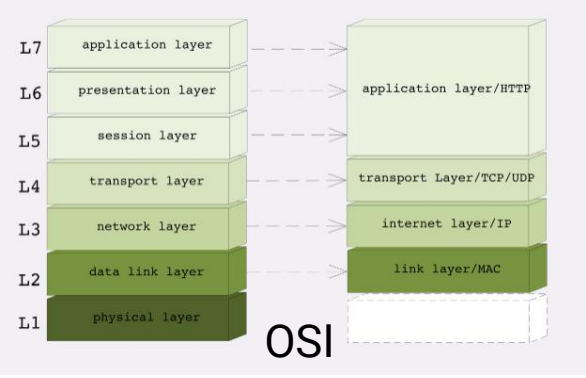
# Socket

# Сокеты, сокеты Беркли

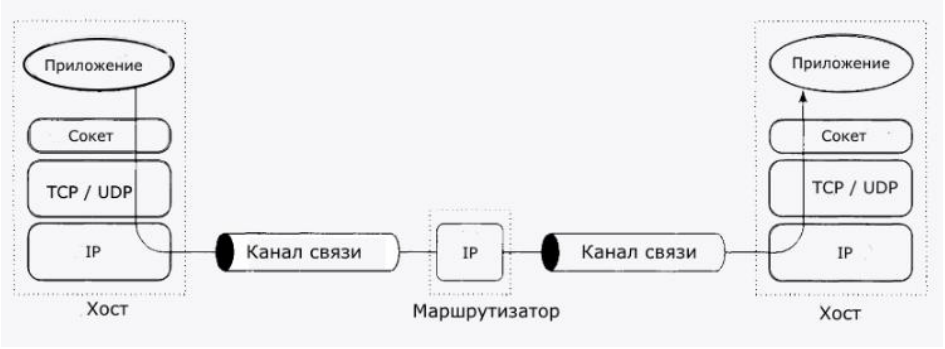
**Сокет** — программный интерфейс для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой только сетью.
Сокет — абстрактный объект, представляющий конечную точку соединения.
Следует различать клиентские и серверные сокеты.

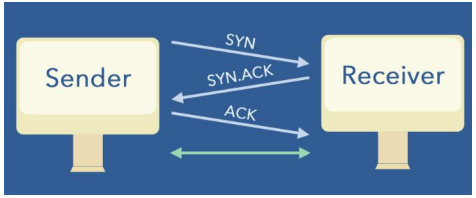
Сокеты были впервые разработаны в Калифорнийском университете Беркли в 80-х годах 20-го века, отсюда получили название сокеты Беркли.
Соответствуют POSIX стандартам.

# TCP и UDP сокеты

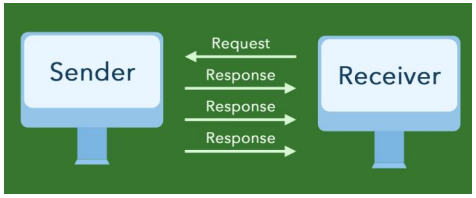


OSI



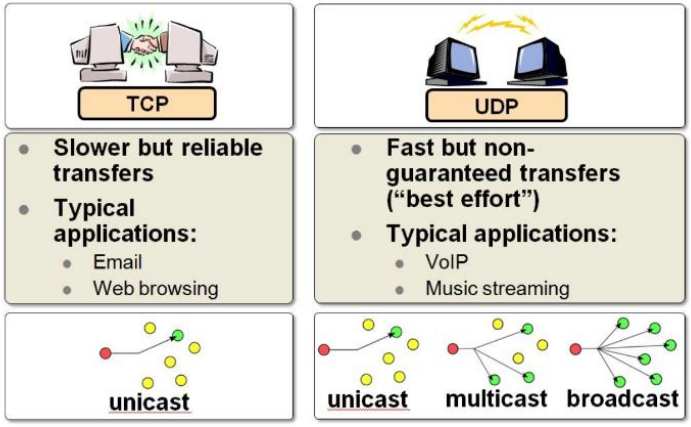



TCP

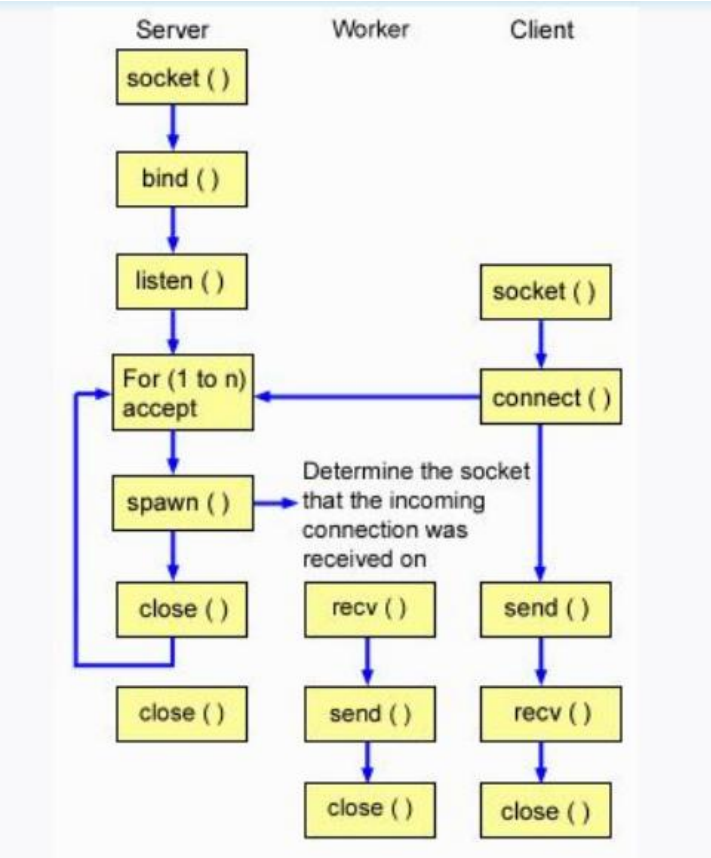


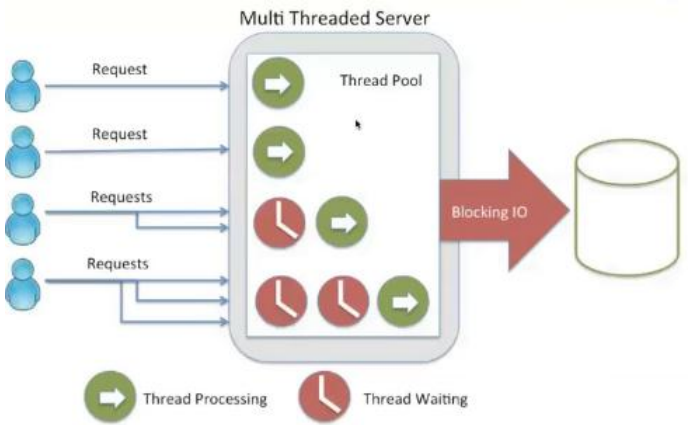
UDP

| TCP | UDP |
|---|---|
| Reliable | Unreliable |
| Connection-oriented | Connectionless |
| Acknowledgement | No acknowledgement |
| Low transmission | Fast transmission |
| Segment sequencing | No sequencing |

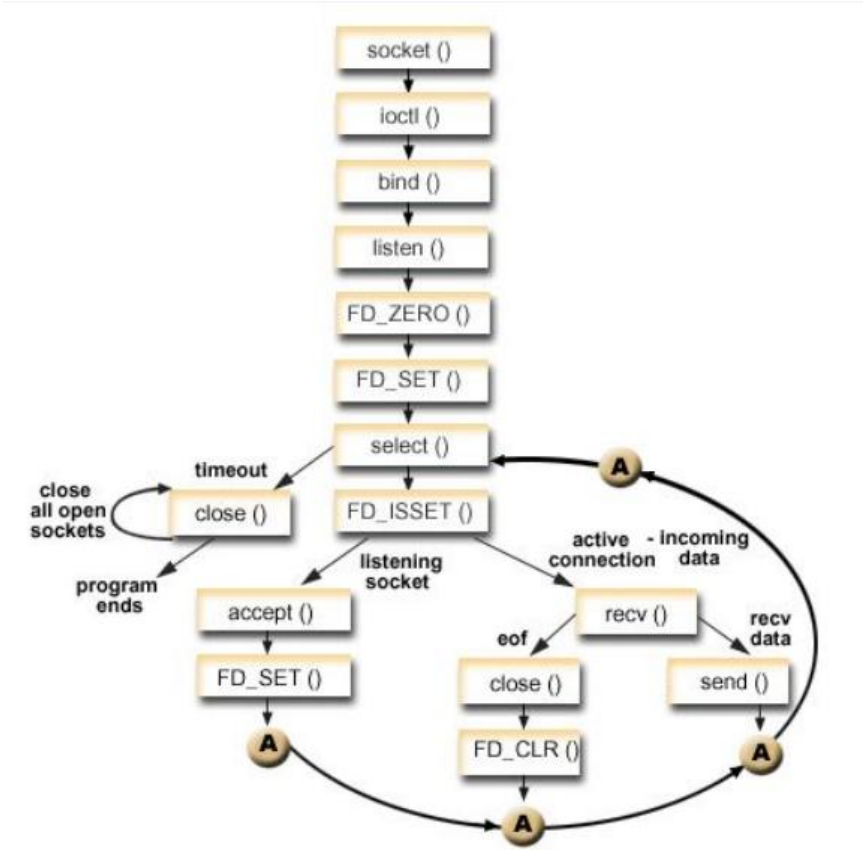


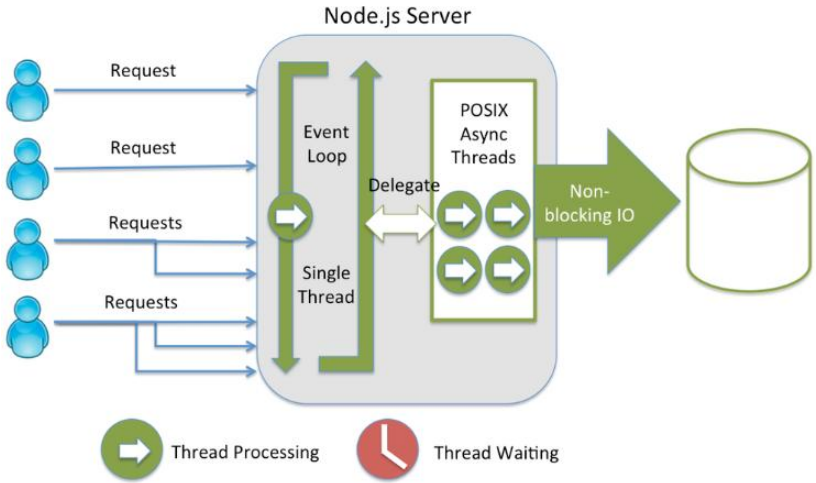
# Многопоточный блокирующий сервер



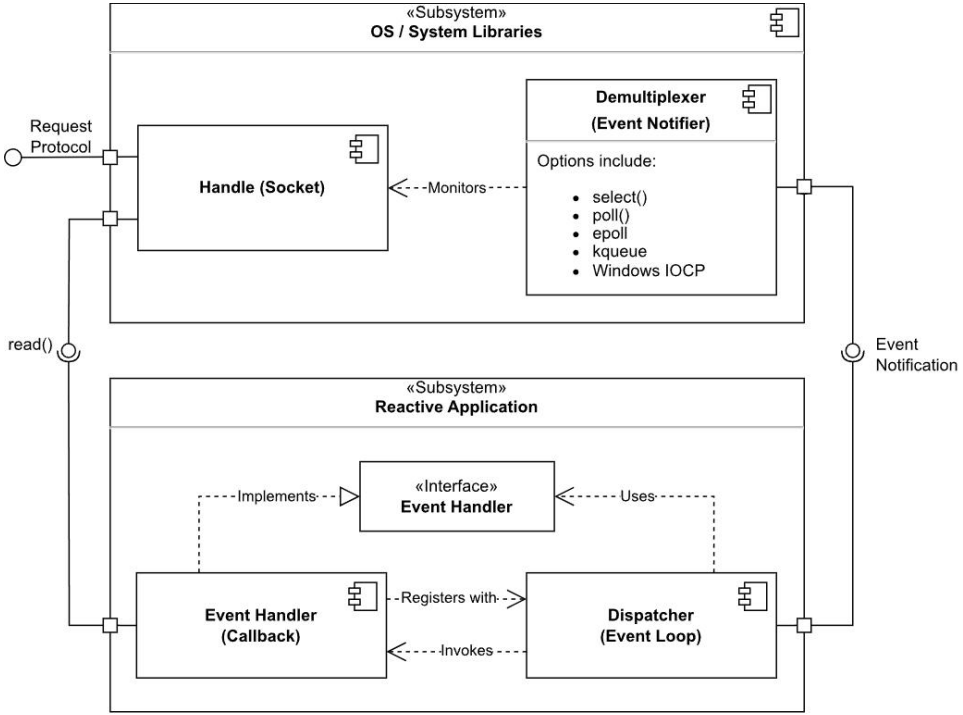



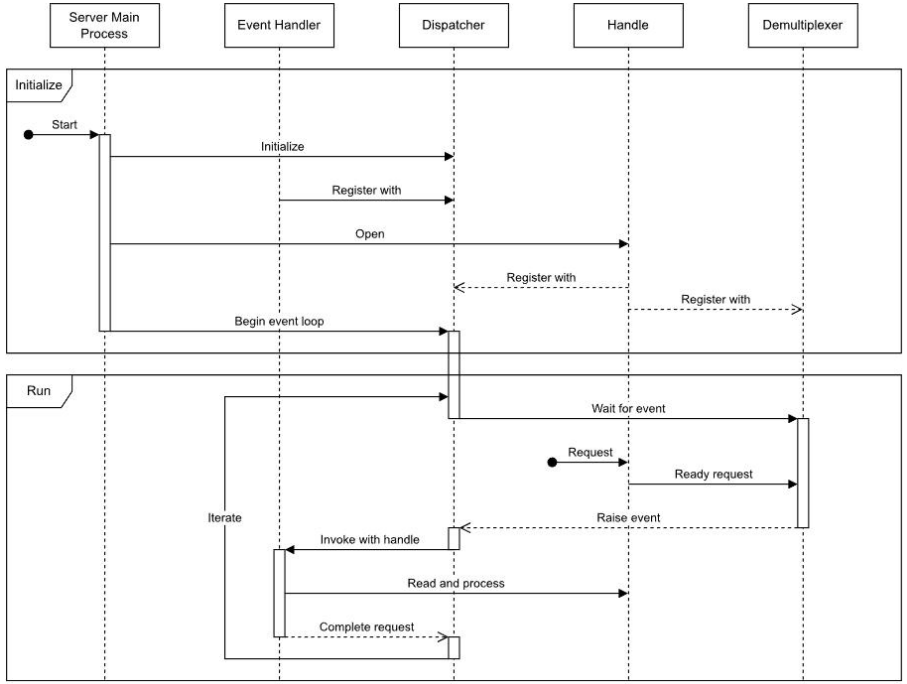
# Однопоточный неблокирующий сервер



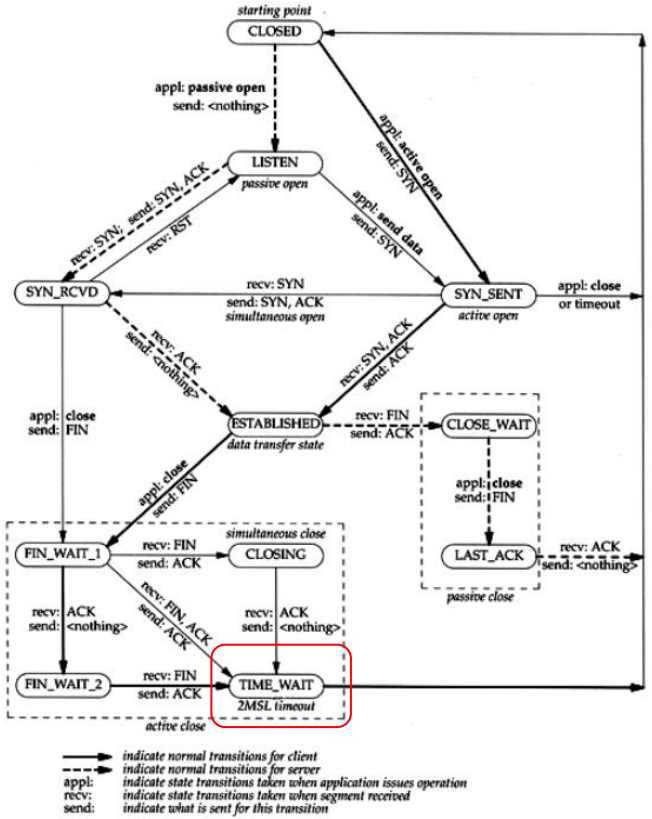



# Паттерн реактор



# Жизненный цикл TCP сокета RFC 793



# Именованные и неименованные каналы

# Именованные и неименованные каналы

**Канал** — это абстракция ОС, канал данных, который используется для односторонней или двусторонней связи между процессами.

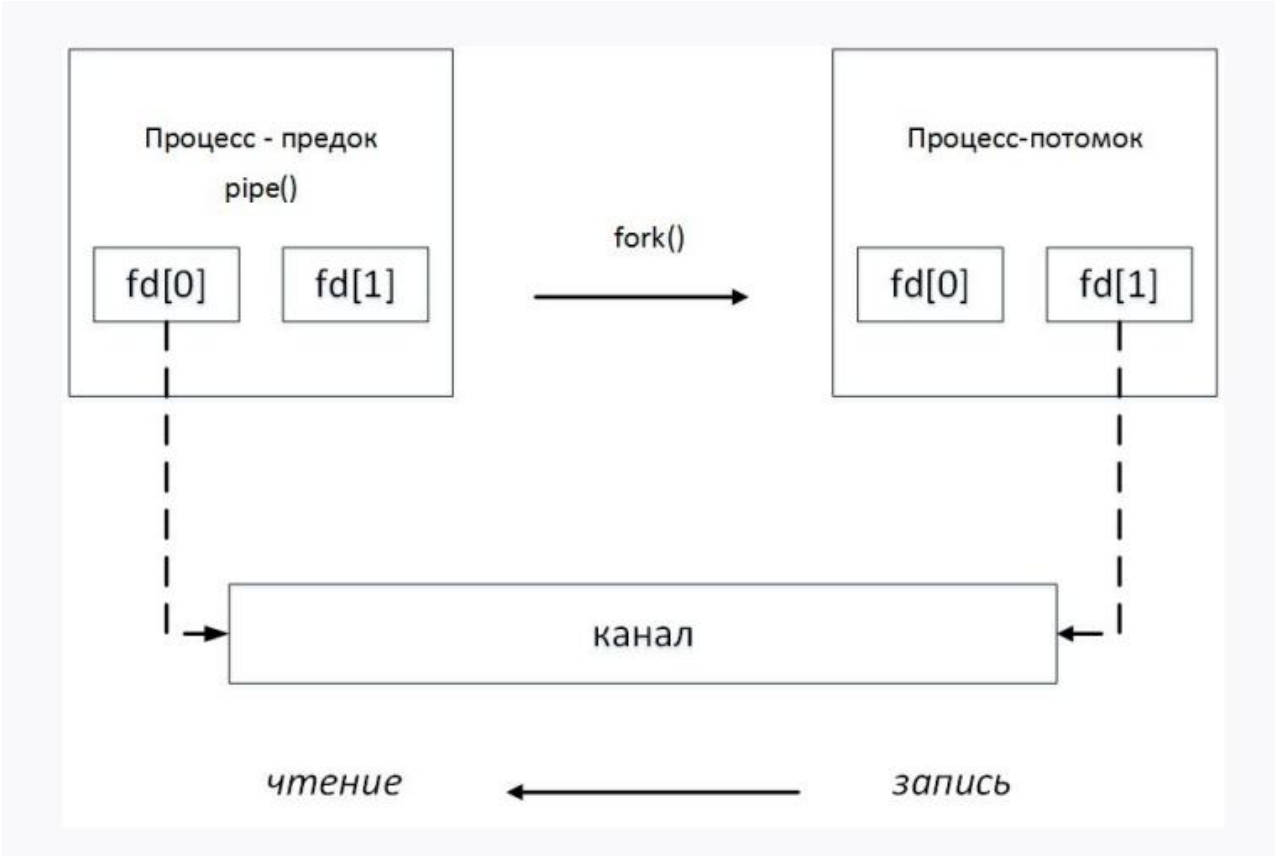
Каналы бывают двух видов:

- Анонимный канал - используются для связи между связанными процессами (т. е. родительским и дочерним).
- Именованный канал - используются для связи между несвязанными процессами, даже на разных машинах.

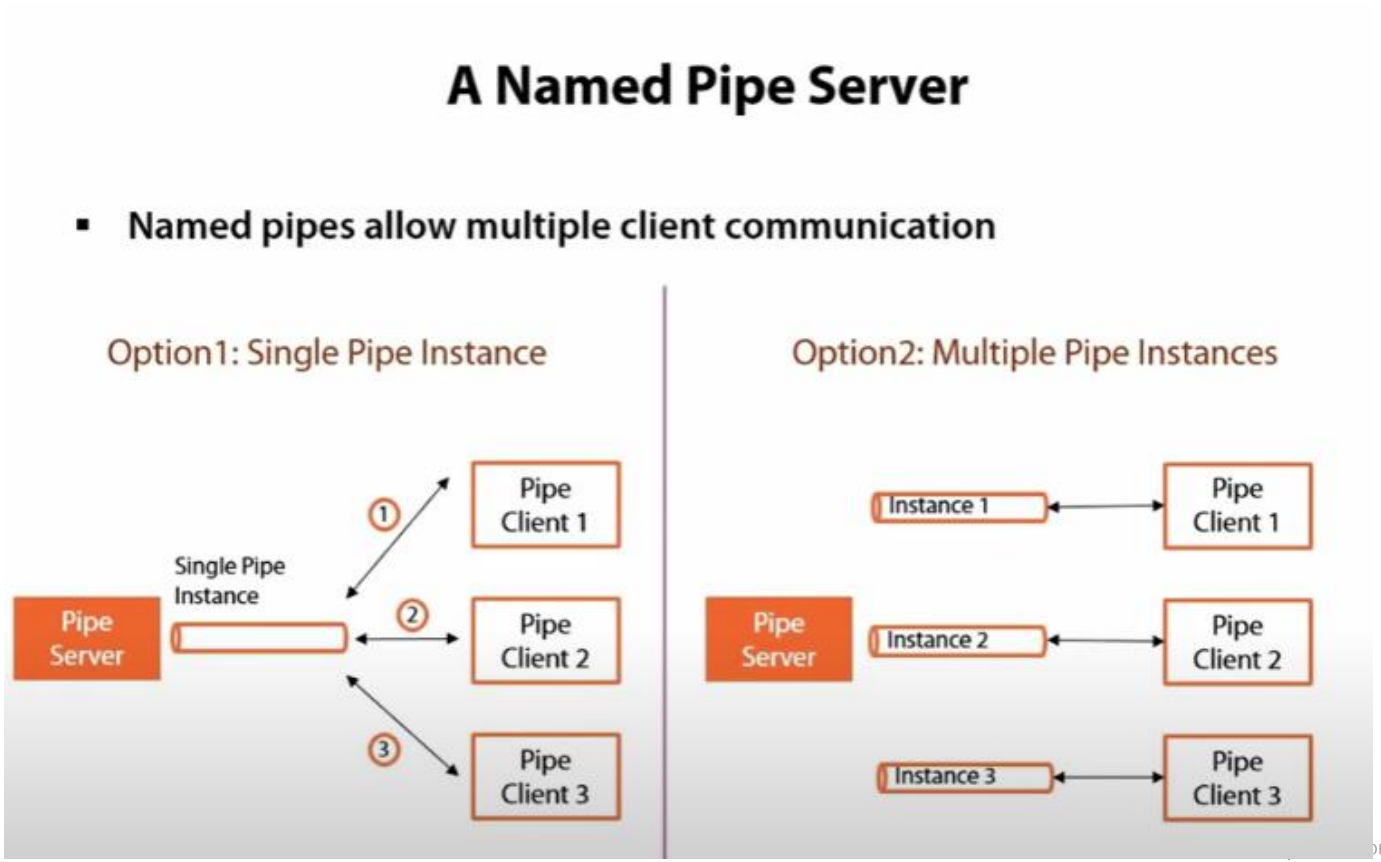
Именованные каналы доступны по имени в файловой системе. Поддерживают архитектуру клиент-сервер, где один процесс действует как сервер (создавая канал), а другие действуют как клиенты (подключаясь к каналу).

Именованные каналы могут работать локально или по сети.

# Неименованный канал



# Именованный канал



# Основные различия между именованными и неименованными каналами

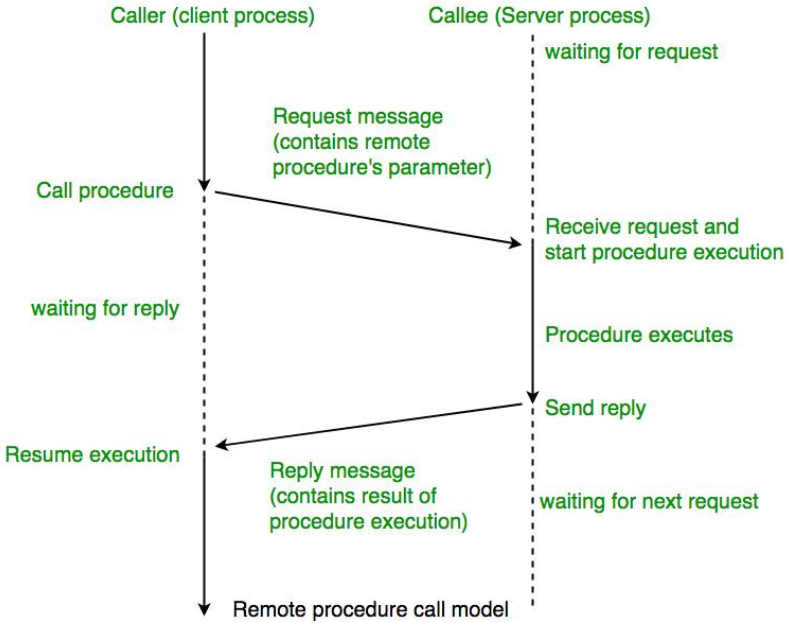
| Характеристика | Неименованные каналы | Именованные каналы |
|---|---|---|
| **Связь** | Между родительским и дочерним процессами | Между любыми процессами |
| **Доступ** | Доступен только для родственных процессов | Доступен для любых процессов (локально и через сеть) |
| **Направленность** | Обычно однонаправленные | Могут быть двунаправленными |
| **Постоянство** | Временные, удаляются по завершению процесса | Постоянные, могут сохраняться в системе |
| **Использование через сеть** | Только локально | Локально и через сеть |

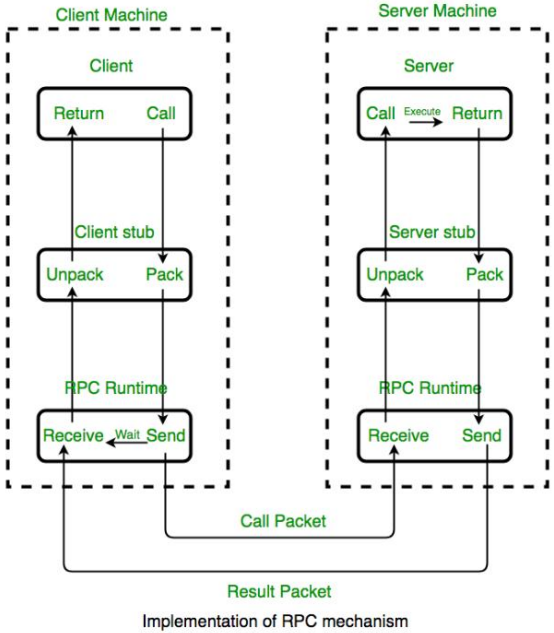
# RPC

# RPC

**Удаленный вызов процедур** (RPC) — один из методов построения распределенных клиент-серверных приложений . Он основан на расширении обычного локального вызова процедур, так что вызываемая процедура не обязательно должна находиться в том же адресном пространстве, что и вызывающая процедура . Два процесса могут находиться в одной системе или в разных системах, соединенных сетью.

# RPC




Implementation of RPC mechanism

# WCF основные возможности

- **Поддержка разных типов связи**: WCF поддерживает множество протоколов, таких как HTTP, TCP, MSMQ, WebSockets и Named Pipes, что позволяет реализовать гибкие решения для связи между приложениями.

- **SOAP и REST**: WCF поддерживает как SOAP (формат для создания веб-сервисов), так и RESTful службы. SOAP более формальный и строго типизированный, в то время как REST предпочтителен для простых веб-сервисов.

- **Безопасность**: WCF предлагает встроенные механизмы для обеспечения безопасности, такие как аутентификация, авторизация, шифрование и управление сертификатами.

- **Обмен сообщениями**: Поддержка различных шаблонов взаимодействия, таких как "один к одному", "один ко многим" и двусторонние обмены сообщениями.

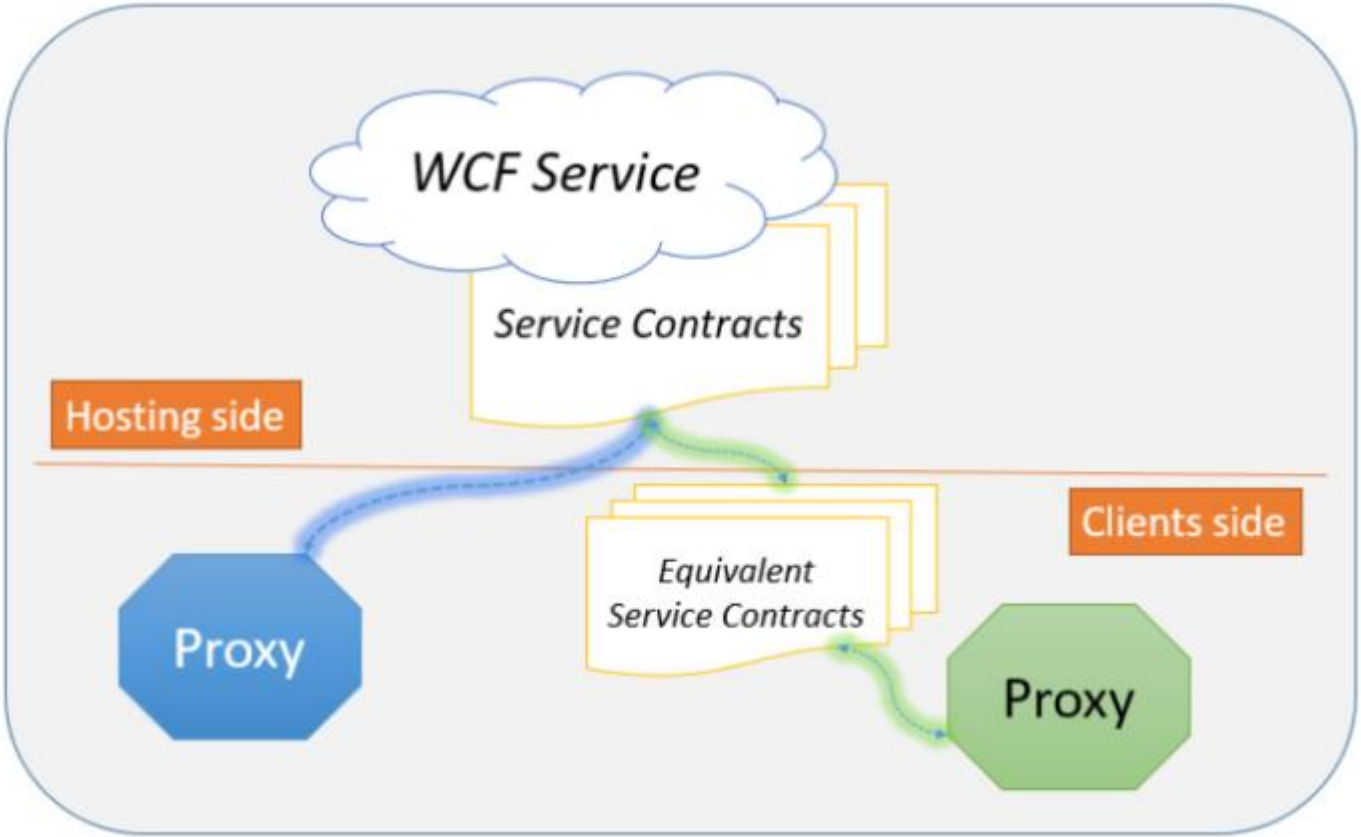
- **Транзакции**: WCF поддерживает распределённые транзакции, что важно для обеспечения целостности данных в сложных системах.

- **Расширяемость**: Фреймворк WCF разработан с учётом возможности кастомизации. Разработчики могут расширять или изменять стандартные поведения WCF, добавляя свои обработчики сообщений, расширяя протоколы безопасности или изменяя способ обработки ошибок.

# WCF основные компоненты

- **Service**: Служба WCF, которая предоставляет определённый набор функций, размещенная по определенному адресу . **Address**
- **Client**: Клиент, который обращается к сервису для выполнения операций.
- **Binding**: Определяет способ, которым клиент и служба взаимодействуют друг с другом (протоколы, формат сообщений).
- **Contract**: Описывает, какие операции поддерживает служба (контракт операции), и формат передаваемых данных (контракт данных).
- **Hosting**: Служба WCF может быть размещена в различных окружениях, таких как IIS, Windows-сервисы или любое консольное приложение.

# WCF


WCF Service
Service Contracts
Hosting side
Clients side
Equivalent
Service Contracts
Proxy
Proxy

# gRPC основные особенности

- **HTTP/2**: Использует HTTP/2 для передачи данных, что обеспечивает такие возможности, как мультиплексирование (одновременные запросы), сжатие заголовков и эффективное управление соединениями.

- **Protocol Buffers (Protobuf)**: Протокол сериализации данных, который позволяет передавать данные в компактном бинарном формате. Это быстрее и эффективнее, чем текстовые форматы, такие как JSON или XML.

- **Двустороннее потоковое взаимодействие**: Поддерживает разные типы вызовов — односторонние и двусторонние потоки, что позволяет клиенту и серверу одновременно обмениваться данными.

- **Межъязыковая поддержка**: gRPC поддерживает множество языков программирования, включая Python, Go, Java, C#, Node.js и многие другие.

- **Идентификация служб**: gRPC использует строгое разделение сервисов и методов через файл .proto, где определяются интерфейсы (методы) и структуры данных.

# gRPC как работает

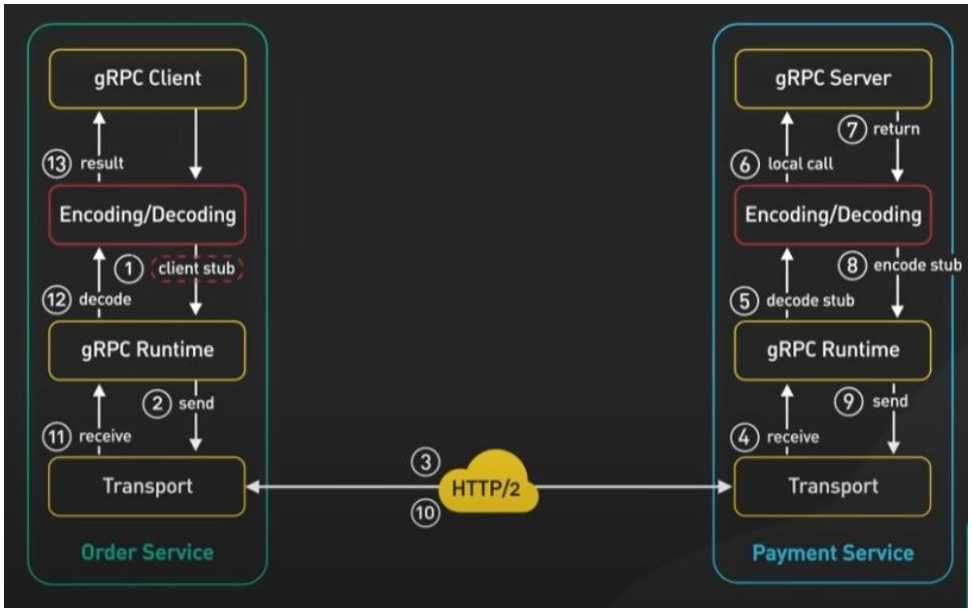
- **Определение контракта (gRPC-сервиса)**: Вначале создается файл .proto, где определяются все методы и типы данных, которые будут использоваться для коммуникации между клиентом и сервером.

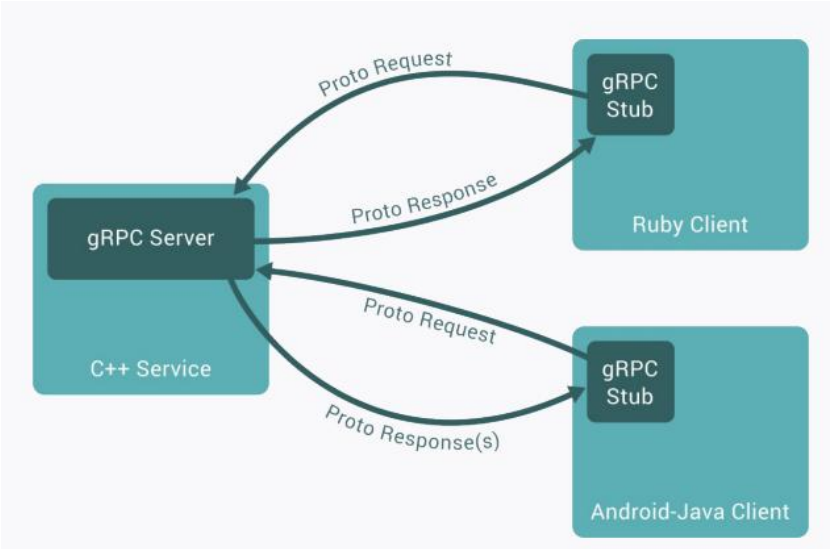
- **Генерация кода**: Из файла .proto автоматически генерируются серверные и клиентские библиотеки для работы с gRPC в выбранных языках программирования.

- **Клиент и сервер**: Сервер реализует методы, описанные в контракте, а клиент вызывает эти методы через gRPC.

- **Сериализация и десериализация**: Все данные, передаваемые между клиентом и сервером, сериализуются в компактный бинарный формат с помощью Protocol Buffers.

# gRPC


gRPC Client
13 result
Encoding/Decoding
1 client stub
12 decode
gRPC Runtime
2 send
11 receive
Transport
Order Service
3
HTTP/2
10
gRPC Server
7 return
6 local call
Encoding/Decoding
8 encode stub
5 decode stub
gRPC Runtime
9 send
4 receive
Transport
Payment Service


Proto Request
gRPC Stub
Proto Response
Ruby Client
gRPC Server
C++ Service
Proto Request
gRPC Stub
Proto Response(s)
Android-Java Client

# LIVE

# Список материалов для изучения

1. Рихтер Дж. "CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C#"
2. https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BA%D0%B5%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) - сокеты
3. https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BA%D0%B5%D1%82%D1%8B_%D0%91%D0%B5%D1%80%D0%BA%D0%BB%D0%B8#bind() - сокеты беркли
4. https://stackoverflow.com/questions/25338862/why-time-wait-state-need-to-be-2msl-long
5. https://server-gu.ru/tcp-syn/
6. https://en.wikipedia.org/wiki/Maximum_segment_lifetime
7. https://ru.wikipedia.org/wiki/POSIX - Posix стандарт
8. https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BA%D1%82%D0%BE%D1%80_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) - шаблон реактор
9. https://inventos.ru/tpost/pyogrkbmf1-kogda-tcp-soketi-otkazivayutsya-umirat задержки по ответам при потеряхпакетов
10. https://xvpn.io/blog/tcp-vs-udp tcp vs udp socket
11. https://www.guru99.com/tcp-3-way-handshake.html
12. https://flylib.com/books/en/3.223.1.188/1/ socket lifecycle
13. https://metanit.com/sharp/net/1.1.php
14. https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%B5%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%B0%D0%BD%D0%B0%D0%BB - именованные каналы
15. https://www.geeksforgeeks.org/remote-procedure-call-rpc-in-operating-system/ RPC
16. https://www.f5.com/company/blog/nginx/building-microservices-inter-process-communication
17. https://ru.wikipedia.org/wiki/SOAP
18. https://chsakell.com/2015/10/15/wcf-proxies-from-beginner-to-expert/ wcf
19. https://www.youtube.com/watch?app=desktop&v=gnchfOojMk4&ab_channel=ByteByteGo Grpc
20. https://gitlab.com/otus-demo/ipc
21. https://gitlab.com/otus-demo/grpc1

# Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

# Рефлексия

# Заполните, пожалуйста, опрос о занятии по ссылке в чате

Спасибо за внимание!

# Приходите на следующие вебинары