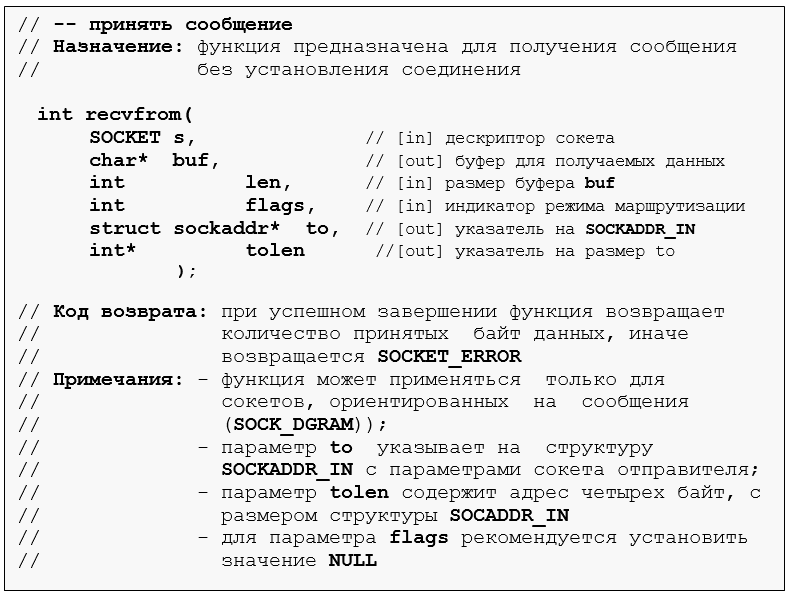
**Вопрос 1. Что такое широковещательный адрес и для чего он используется?** Широковещательный адрес (broadcast address) — это специальный тип IP-адреса, который используется для отправки данных всем узлам в сети одновременно.

Когда устройство отправляет данные на широковещательный адрес, пакет будет доставлен каждому узлу, подключенному к сети.

Широковещательный адрес — это последний IP-адрес в диапазоне адресов сети. Этот адрес формируется путем замены всех битов в хостовой части IP-адреса на единицы.

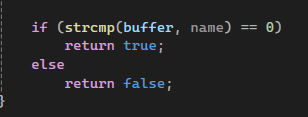
Подсетевое широковещание. Этот тип широковещания используется для отправки сообщения **всем устройствам в твоей локальной сети**, **независимо от того, в какой они подсети**. В этом случае используется специальный адрес — **255.255.255.255**.

**Вопрос 2. Как работает функция recvfrom и в каких случаях она может завершиться аварийно.** Переводит программу в состояние ожидания, до поступления сообщения (функция sendto). Функция recvfrom использует структуру SOCKADDR\_IN – в нее автоматически помещаются параметры сокета клиента/сервера, после приема от него сообщения. Данные поступают в буфер, который обеспечивает принимающая сторона.



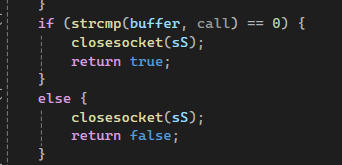
1. **Отключение сокета**: Локальный сокет закрыт (вызов closesocket) или удалённый узел завершил соединение.
2. **Сетевые проблемы**: Потеря связи, тайм-ауты, проблемы маршрутизации.
3. **ICMP ошибки**: Получен ICMP-сигнал о недоступности удалённого хоста.
4. **Неверные параметры**: Некорректный сокет, слишком маленький буфер, неправильные флаги.
5. **Ошибки Winsock**:
   * **WSAECONNRESET**: Соединение сброшено удалённым узлом.
   * **WSAEFAULT**: Неверный адрес буфера.
   * **WSAENETDOWN**: Сетевой интерфейс недоступен.
6. **Недостаток ресурсов**: Нехватка системных ресурсов для обработки запроса.

**Вопрос 3. В чём отличие правильного и неправильного позывного для сервера? Как сервер на это реаргирует?** В конце функции GetRequestFromClient

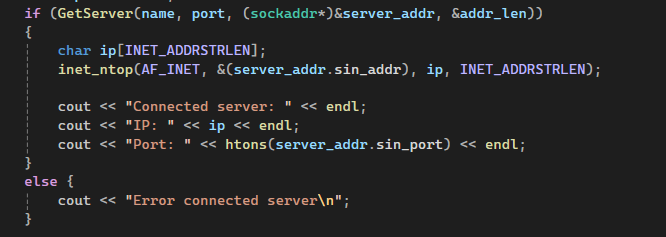


Если он возвращает true, то вызывается PutAnswerToClient и отправляется ответ клиенту.

В клиенте в GetServer:



Если GetServer возвращает true, то клиент понимает, что подключение произошло и выводит инфу:



**Вопрос 4. Какие сетевые вызовы используются для отправки данных серверу и клиенту в широковещательном режиме?**

Использование широковещательных адресов возможно только в протоколе UDP. Поэтому при создании дескрипторов сокетов (в программах клиентов и серверов) при вызове функции socket значение параметраtype должно быть SOCK\_DGRAM**,** а для обмена данными этом случае используются функции sendto и recvfrom**.**

**Вопрос 5. Алгоритм взаимодействия сервера и клиента.**

На сервере:

Прописывается позывной, создаётся структура для хранения данных клиента, инициализируется библиотека WSAStartup(). Дальше идёт функция GetRequestFromClient – получение сообщение от клиента. В качестве параметров передаём наш позывной, порт, sockaddr\_in, в котором мы будем хранить данные клиента.

Внутри GetRequestFromClient:

Создаём сокет, который будет использоваться для получения UDP сообщений от клиентов.

Прописываем sockaddr\_in для указание сетевого адреса сервера. Туда прописываем порт, полученный из параметров функции и INADDR\_ANY (сервер будет слушать все сетевые интерфейсы).

Биндим sockaddr\_in к сокету.

recvfrom сообщения. На этот моменте сервер приостанавливает работу и ждёт сообщения от клиента. Передаём sockaddr\_in клиента чтобы структура заполнилась.

На клиенте:

Создаётся структура для хранения данных сервера. Инициализируется библиотека WSAStartup(). Вызывается GetServer. В него передаётся позывной, порт, структура для хранения данных сервера.

В GetServer:

Создаём сокет UDP.

Создаём sockaddr\_in для клиента. В адресе sin\_addr прописываем **INADDR\_BROADCAST** — это константа, которая представляет широковещательный IP-адрес для IPv4. Обычно этот адрес равен 255.255.255.255.

Setsockopt прописываем параметры для сокета. Меняем параметр SO\_BROADCAST на 1, то есть разрешаем широковещательную рассылку.

Вызываем sendto и в качестве двух последних параметров указываем наш sockaddr\_in.

Прописываем recvfrom. Клиент может принимать сообщения от любого источника, который отправляет данные на тот же порт, на котором он слушает.

Дальше на сервере:

Сервер сравнивает полученное сообщение со своим позывным, если всё хорошо, то вызывает PutAnswerToClient.

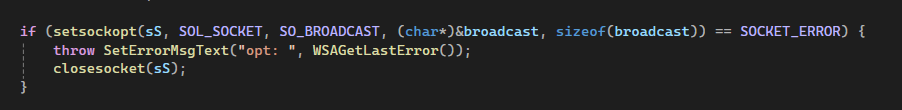
Важно: В GetRequestFromClient мы создаём и настраиваем структуру SOCKADDR\_IN для приемника (сервера), чтобы он мог принимать входящие соединения. В PutAnswerToClient мы используем уже существующий адрес клиента для отправки ответа, поэтому необходимость в создании нового адреса отпадает.

Отправляем sendto свой позывной клиенту, используя его sockaddr\_in.

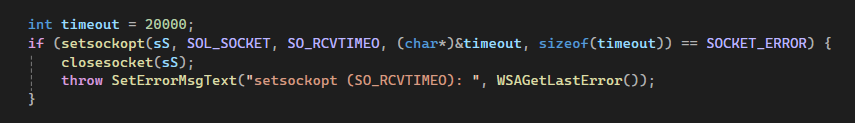
Далее продолжается работа GetServer на клиенте:

Срабатывает recvfrom. В структуру sockaddr\_in записывается данные сервера. Клиент сравнивает полученное и отправленное сообщение.

**Вопрос 6. Как с помощью setsockopt настроить сокет на работу с широковещательным адресом.** Setsockopt прописываем параметры для сокета. Меняем параметр SO\_BROADCAST на 1, то есть разрешаем широковещательную рассылку.

****

**Вопрос 7. Какие сценарии могут привести к завершению работы с кодом WSATIMEDOUT?**

****

Ошибка WSATIMEDOUT

**Вопрос 8. Как реализовать многократное взаимодействие сервера с разными клиентами в одном приложении?**

Цикл в котором постоянно будет проигрываться функция GetRequestFromClient и SendResponseToClient

**Вопрос 9. Наличие других серверов с таким же позывным?**

На сервере также вызвать GetServer. Отличие от GetServer клиента: Сервер будет принимать широковещательные сообщения в цикле (чтобы отловить все сервера, а не один). Цикл не бесконечный, а на какое-то время. При приёме сообщения от другого сервера, вносит его в массив уникальных значений.

**Вопрос 10.**

Разработанные функции  **GetRequestFromClient** и **GetServer**  имеют существенный недостаток. После вызова функции **recvfrom**  они переводят поток в режим ожидания. Выход из этого состояния возможен лишь в том случае, если в адрес сокета поступило сообщение или будет исчерпан допустимый интервал ожидания. Такой алгоритм работы делает эти функции мало применимыми.

1. Что такое символическое имя?

Символьное имя — это человекочитаемое название компьютера или сервера в сети. Оно используется вместо IP-адреса для упрощения идентификации устройств. Например, вместо использования IP-адреса 192.168.0.1, можно использовать символическое имя my-computer.

Символьное имя обычно соответствует доменному имени в сети, которое может быть разрешено с помощью DNS или NetBIOS.

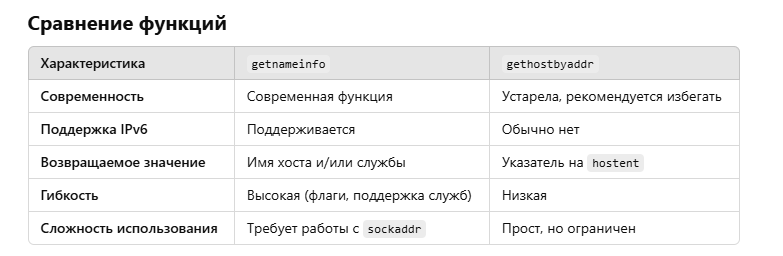
2. Назначение функций

gethostname

Функция gethostname используется для получения символического имени текущего компьютера.

Она возвращает имя хоста, настроенное в системе.

Это имя может быть полезно для сетевых операций или идентификации устройства в локальной сети.



getnameinfo

Функция getnameinfo предназначена для преобразования IP-адреса в символическое имя хоста и (опционально) имени службы. Она современная, поддерживает как IPv4, так и IPv6, и предоставляет гибкий интерфейс через использование структуры sockaddr. Дополнительные флаги позволяют настроить возвращаемую информацию, например, получить только имя хоста или только имя службы. Благодаря универсальности и совместимости с новыми протоколами, getnameinfo является предпочтительным выбором для современных приложений.

gethostbyaddr

Функция gethostbyaddr предназначена для получения информации о хосте по его IP-адресу. Она возвращает указатель на структуру hostent, содержащую символическое имя хоста, список альтернативных имен и связанных с ними адресов. Однако, эта функция считается устаревшей (deprecated), поскольку не поддерживает IPv6 и уступает getnameinfo по гибкости. Она подходит только для приложений, работающих с IPv4 и требующих минимальных изменений в старом коде.

3. DNS-служба в локальной сети

DNS (Domain Name System) — это служба, которая преобразует символические имена хостов в их IP-адреса. В локальной сети может быть настроен локальный DNS-сервер, выполняющий следующие функции:

* Разрешение имен устройств внутри сети.
* Ускорение сетевого взаимодействия путем кэширования запросов.
* Упрощение управления сетью, так как пользователям не нужно запоминать IP-адреса.

Если DNS-служба отсутствует, возможно использование NetBIOS для разрешения имен в пределах локальной сети.

4. Как обратиться к собственному компьютеру?

Есть несколько способов обратиться к своему компьютеру:

1. Через символическое имя

Используйте имя, настроенное для вашего компьютера, которое можно узнать с помощью gethostname.

2. Через IP-адрес

Локальный IP-адрес: 127.0.0.1 (или ::1 для IPv6).

Ваш реальный IP-адрес в локальной сети, например, 192.168.x.x

3. Через зарезервированное имя

Символическое имя localhost всегда указывает на ваш компьютер.