Использование сокетов в программе на С++.

Для работы с сокетами в С++ предназначен интерфейс Winsock, который упрощает разработку сетевых приложений для ОС Windows. Winsock представляет собой интерфейс между самим процессом и протоколом, выполняющим передачу информации (TCP/UDP). Конечно же есть готовые библиотеки для работы с сетью, но не всегда они удовлетворяют необходимым требованиям.

Для начала работы с сокетами в С++ необходимо подключить соответствующие библиотеки.

#include "WinSock2.h"

#pragma comment(lib, "Ws2\_32.lib")

Перед началом использованием функций для работы с сокетами, необходимо вызвать функцию WSAStartup, которая позволяет приложению зарегистрировать себя в библиотеке WSOCK32.DLL, чтобы иметь доступ к интерфейсу Windows Sockets:

int WSAStartup(WORD wVersionRequired, LPWSADATA lpWSAData);

где в качестве первого аргумента необходимо передать слово, у которого младший байт определяет версию, а старший байт — подверсию Winsock. Сделать это можно, используя макрос MAKEWORD.

Если мы используем версию Winsock 2, то первый аргумент функции будет иметь следующий вид: MAKEWORD(2, 2).

Аргумент lpWSAData — это указатель на структуру WSADATA, которая в результате работы функции получит информацию о реализации текущей версии Windows Socket. Пример использования функции:

WSAStartup(MAKEWORD(2, 2), &wsaData);

Если всё прошло успешно, функция WSAStartup возвращает 0, а иначе — код ошибки. Весь список кодов ошибок можно посмотреть на MSDN.

Если смотреть со стороны WinsockAPI, то сокет — это некая абстракция, которая позволяет произвести обмен информацией. Доступ к этой абстракции обеспечивает дескриптор сокета. Всё, что нам необходимо — создать сокет с подходящей конфигурацией и использовать его для обмена информацией. При создании сокета нам необходимо указать тип протокола, который будет использоваться для передачи данных, должен ли сокет поддерживать связь с другим адресатом или нет.

Для использования сокета его сначала нужно объявить:

SOCKET \_socket;

где тип SOCKET описывает дескриптор конкретного сокета, идентифицирует его.

Рассмотрим типичную последовательность вызова функций для сервера:

1. Для начала необходимо вызвать функцию WSAStartup для инициализации библиотеки для работы с сокетами.

2. Создаем сокет. Для этого используем функцию socket, указав в качестве аргументов спецификацию семейства адресов, тип сокета (потоковый или дейтаграммный) и протокол передачи данных.

3. После этого следует ассоциировать сокет с адресом и портом. Для этого необходимо вызвать функцию bind, передав ей в качестве аргументов дескриптор сокета, который необходимо связать с адресом и портом, указатель на структуру sockaddr, которая хранит информацию о семействе адреса, IP и порт, а также размер структуры sockaddr.

4. Установить сокет в режим прослушивания входящих подключений. Для этого предназначена функция listen, которая в качестве аргументов принимает дескриптор сокета, который необходимо установить в режим прослушивания, а также максимальный размер очереди входящих подключений.

5. Принять подключение клиента с помощью функции accept, которая при новом подключении возвращает новый сокет, идентифицирующий соответствующее подключение.

6. После этого можно выполнять передачу и прием данных с помощью функций send и recv соответственно. Например, для отправки данных можно вызвать функцию send следующим образом:

send(\_socket, buffer, strlen(buffer), 0);

для приема данных:

int i = recv(\_socket, buffer, size, 0);

7. Для закрытия соединения использовать функцию closesocket, передав ей в качестве аргумента дескриптор сокета, который необходимо закрыть.

8. Вызвать функцию WSACleanup для завершения работы с библиотекой Winsock 2.



Теперь рассмотрим типичную последовательность вызова функций для клиента:

1. Вызвать функцию WSAStartup для инициализации библиотеки для работы с сокетами.

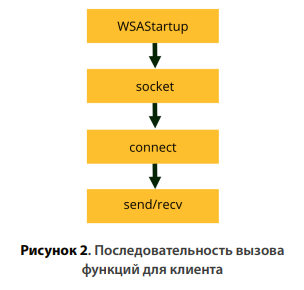
2. Создать сокет.

3. Вызвать метод connect, указав в качестве аргументов дескриптор сокета-источника, указатель на структуру типа sockaddr, которая хранит информацию про адрес, с которым необходимо установить подключение, а также размер вышеупомянутой структуры в байтах.

4. Выполнять передачу и приём данных с помощью функций send и recv.

5. После завершения работы закрыть соединение с помощью функции closesocket.

6. Вызвать функцию WSACleanup для завершения работы с библиотекой Winsock 2.



Функции для работы с сокетами

Функция socket

Функция socket возвращает дескриптор нового сокета в случае успешной работы, иначе — возвращает ошибку INVALID\_SOCKET. Более подробную информацию об ошибке можно получить с помощью функции WSAGetLastError.

Синтаксис функции: SOCKET socket(int af,int type,int protocol);

Аргумент af задает спецификацию семейства адресов. Возможные значения для семейства адресов определены в заголовочном файле Winsock2.h. В настоящее время поддерживаются значения AF\_INET или AF\_INET6, которые являются форматами семейства интернет-адресов для IPv4 и IPv6.

Аргумент type задает тип создаваемого сокета. Обычно используется SOCK\_STREAM (потоковый сокет) или SOCK\_DGRAM (дейтаграмный сокет).

Аргумент protocol задает используемый протокол передачи данных. Если указано значение 0, клиент не желает указывать протокол, а используемый протокол будет выбран поставщиком услуг. Например, для потоковых сокетов будет использоваться протокол TCP, а для дейтаграммных — UDP.

Список доступных протоколов можно посмотреть на MSDN.

Приведем пример создания сокета с помощью функции socket:

SOCKET \_socket = socket (AF\_INET, SOCK\_STREAM, 0);

Функция setsockopt

Функция setsockopt позволяет выполнить «тонкую» настройку сокета.

Синтаксис функции: int setsockopt(SOCKET s,int level, int optname, const char \*optval, int optlen);

■ s — дескриптор сокета;

■ level — уровень протокола, на котором задается параметр сокета;

■ optname — параметр сокета, для которого задается значение, например SO\_DEBUG;

■ optval — указатель на буфер, в котором указано значение, необходимое для установки;

■ optlen — размер буфера в байтах, на который указывает optval.

В случае успешной работы функция возвращает 0. С более подробной информацией о работе функции вы можете ознакомиться в соответствующем разделе MSDN.

Функция bind

Функция bind предназначена для ассоциации сокета с адресом и портом.

Синтаксис функции: int bind(SOCKET s, const sockaddr \*addr, int namelen);

Аргумент s — дескриптор сокета, который нужно связать с адресом:

■ addr — указатель на структуру типа sockaddr, которая хранит информацию о семействе адреса, IP-адрес, а также порт;

■ namelen — размер структуры addr.

Пример заполнения структуры sockaddr:

sockaddr\_in addr; addr.sin\_family = AF\_INET;

inet\_pton(AF\_INET,"0.0.0.0", &addr.sin\_addr);

addr.sin\_port = htons(port);

Функция inet\_pton строку символов в структуру сетевого адреса указанного сетевого семейства адресов, результат сохраняет в dst.

int inet\_pton(int af, const char \*src, void \*dst);

Важен тот факт, что обычно такой «ручной» биндинг (т.е. привязка сокета к конкретному адресу и порту) необходим на стороне сервера, но на клиенте это тоже возможно сделать. Но в стандартной реализации функция connect автоматически биндит сокет к адресу хоста, при этом выбирая случайный порт.

Функция listen

Функция listen предназначена для установки сокета в режим прослушивания входящих подключений. Её необходимо вызывать после вызова функции bind, когда сокет привязан к определенному адресу и порту. Исключение составляют дейтаграммные сокеты, которые не требуют установки соединения. В таком случае можно сразу вызывать функцию для приема данных.

Синтаксис функции: int listen(SOCKET s, int backlog);

■ аргумент s— дескриптор сокета, который необходимо установить в режим прослушивания,

■ backlog — максимальный размер очереди входящих подключений.

Если всё прошло успешно, функция возвращает 0.

Пример использования функции: listen(\_socket, 1);

открыть для прослушивания входящих подключений сокет с размером очереди подключений равным 1.

Если в качестве параметра backlog установлено значение SOMAXCONN, базовый поставщик услуг, ответственный за сокет \_socket, установит размер очереди на максимально допустимое значение.

Функция accept

Функция accept используется для приёма сокетом входящих подключений.

Синтаксис функции: SOCKET accept(SOCKET s, sockaddr \*addr, int \*addrlen);

■ s — дескриптор сокета, который будет принимать входящее подключение,

■ addr — указатель на структуру типа sockaddr, в которую будет записана информация об адресе сокета, который подключается;

■ addrlen — длина адреса.

Последние 2 аргумента не обязательны, если информация об адресе клиента не важна, тогда в качестве значений можно передать NULL.

Если функция отработала без ошибок, то она возвращает дескриптор нового созданного сокета, который будет использоваться для связи с присоединившимся сокетом.

Пример использования функции

SOCKET acceptSocket;

sockaddr\_in acceptSockAddr;

int len = sizeof (acceptSockAddr);

acceptSocket = accept (\_socket, (sockaddr\* )&acceptSockAddr, &len);

Также следует учесть, что данная функция является блокирующей, т.е. если в очереди нет клиентов на под ключение, функция не возвращает управление в вызывающий код.

Если есть необходимость выполнять обработку нескольких клиентов, то необходимо задействовать дополнительные потоки, чтобы при подключении очередного клиента освобождать «слушающий» сокет.

Функция connect

Функция connect позволяет установить соединение с сокетом.

Синтаксис функции: int connect(SOCKET s, const sockaddr \*name, int namelen);

■ s — дескриптор сокета-источника;

■ name — указатель на структуру типа sockaddr, которая хранит информацию об адресе, с которым необходимо установить соединение;

■ namelen — размер в байтах структуры, на которую указывает указатель name.

Если подключение установлено успешно, функция возвращает 0, иначе — SOCKET\_ERROR. Код ошибки можно извлечь с помощью функции WSAGetLastError.

Пример использования функции

sockaddr\_in addr;

s\_addr.sin\_family = AF\_INET; // тип адреса (TCP/IP)

// адрес сервера. TCP/IP представляет адреса

// в числовом виде, поэтому для представления адреса

// используем функцию inet\_pton;

inet\_pton(AF\_INET, "10.5.54.16", &addr.sin\_addr);

s\_addr.sin\_port = htons (1234);

// Функция htons

// позволяет задать номер порта в TCP/IP представлении

connect (s, (sockaddr \*) &addr, sizeof (addr));

В случае использования дейтаграммных сокетов вызов функции connect не обязателен.

После установки соединения сокеты могут обмениваться информацией с помощью вызова функций send и recv. В случае дейтаграммных сокетов можно использовать функции sendto и recvfrom.

После использования сокета его необходимо закрыть с помощью функции closesocket.

int closesocket(SOCKET s);