**Версии, структура и состав интерфейса Windows Sockets**

Существует две основные версии интерфейса: Windows Sockets 1.1 и Windows Sockets 2. В состав каждой версии входит динамическая библиотека, библиотека экспорта и заголовочный файл, необходимый для работы с библиотеками.

Для использования интерфейса Winsock2 в исходный текст программы следует включить следующую последовательность директив компилятора C++.

**#include "Winsock2.h" //** заголовок WS2\_32.dll

**#pragma comment(lib, "WS2\_32.lib") //** экспорт WS2\_32.dll – содержит все функции Winsock2

Функция WSAGetLastError вызывается, непосредственно сразу после функции Winsock2, завершившейся с ошибкой.

Для обработки ошибок используется функция SetErrorMsgText, которая в качестве параметра получает префикс формируемого сообщения об ошибке.

Для инициализации библиотеки WS2\_32.DLL предназначена функция **WSAStartup**.[ версия Windows Sockets, указатель на WSADATA-шаблон струтктуры содержится в Winsock2.h].

Для завершения работы с библиотекой WS2\_32.DLL используется функция **WSACleanup**.

Для создания сокета используется функция **socket**.[формат адреса, тип сокета, протокол]. **SOCK\_STREEM** – сокет ориентированный на поток;

**SOCK\_DGRAM** – сокет, ориентированный на сообщения(UDP).

После завершения работы с сокетом, обычно, его закрывают (освобождают ресурс). Для закрытия сокета применяется функция **closesocket**.[дескриптор сокета].

Для установки параметров существующего сокета используется функция **bind.** [сокет, указатель на структуру **SOCKADDR\_IN,** длина **SOCKADDR\_IN** в байтах].

После создания сокета и выполнения функции bind сокет остается недоступным для подсоединения клиента. Чтобы сделать доступным уже связанный сокет, необходимо его переключить, в так называемый, прослушивающий режим. Переключение осуществляется с помощью функции **listen.** [дескриптор связанного сокета, максимальная длина очереди]. при успешном завершении функция возвращает нуль, иначе возвращается значение **SOCKET\_ERROR.**

Канал связи (или соединение) создается между двумя сокетами, ориентированными на поток. На стороне сервера это должен быть связанный (функция **bind**) и переключенный в режим прослушивания (функция **listen**) сокет. На стороне клиента должен быть создан дескриптор ориентированного на поток сокета (функция **socket**).

Канал связи создается в результате взаимодействия функций **accept** (на стороне сервера) и **connect** (на стороне клиента). Алгоритм взаимодействия этих функций зависит от установленного режима ввода-вывода для участвующих в создании канала сокетов. Accept[дескриптор связанного сокета, указатель на **SOCKADDR\_IN,** указатель на длину **SOCKADDR\_IN**].

Функция **accept** приостанавливает выполнение программы сервера до момента срабатывания в программе клиента функции. В результате работы функции **accept** создается новый сокет, предназначенный для обмена данными с клиентом.

На стороне клиента создание канала осуществляется с помощью функции **connect.** Для того, чтобы выполнить функцию **connect**, достаточно просто предварительно создать сокет (функция **socket**), ориентированный на поток. Функция **connect** указывает модулю TCP сокет клиента, который будет использоваться для соединения с сокетом сервера (его параметры указываются через параметры **connect**).

Обмен данными по каналу связи осуществляется между двумя сокетами и возможен сразу после того, как этот канал создан (выполнена функция **accept** на стороне сервера и функция **connect** на стороне клиента). Для пересылки данных по каналу Winsock2 предоставляет функции **send** и **recv** (рисунки 3.11.1 и 3.11.2). Функция **send** пересылает по каналу, указанного сокета, определенное количество байт данных. Функция **recv** принимает по каналу, указанного сокета, определенное количество байт данных. Для работы обеих функций в программе необходимо выделить память (буфер) для приема или оправления данных. Размер буфера для приема данных и для отправления данных указывается в параметрах функций.

Работа функций **send** и **recv** является синхронной, т.е. до тех пор, пока не будет выполнена пересылка или прием данных выполнение программы приостанавливается. Поэтому, если одной из сторон распределенного будет выдана функция **recv** для которой нет данных для приема и при этом соединение не разорвано, то это приведет к зависанию программы на некоторое время