Порт – номер приложения, которое идентифицирует себя как получателя. Порт != Идентификационный номер приложения. Приложения занимают свободные порты при каждом запуске компьютера. Если приложению требуется какой-то конкретный порт, а он занят другим приложением, то это приложение не запуститься.

Порт 80 – веб-сервер.

Сокет – объединенная информация об IP и о порте. Сокет записывается следующим образом:

54.239.33.90:80,

где:

54.239.33.90 – IP;

80 – порт.

Клиент – это тот, кто инициирует сеанс связи. Сервер – тот, который его получает, а затем отвечает некоторой информацией.

Сокет — это интерфейс между прикладным и транспортным уровнями внутри хоста.

Интерфейс сокета Беркли — API, позволяющий реализовывать взаимодействие между компьютерами или между процессами на одном компьютере. Данная технология может работать со множеством различных устройств ввода-вывода и драйверов, несмотря на то, что их поддержка зависит от реализации операционной системы. Подобная реализация интерфейса лежит в основе TCP/IP, благодаря чему считается одной из фундаментальных технологий, на которых основывается Интернет.

Сокеты бывают блокирующие и неблокирующие. Суть в том, что в случае блокирующих сокетов при попытке прочитать (и записать) данные функция чтения будет ждать до тех пор, пока не прочитает хотя бы один байт или произойдет разрыв соединения или придет сигнал. В случае неблокирующих сокетов функция чтения проверяет, есть ли данные в буфере, и если есть - сразу возвращает, если нет, то она не ждет и также сразу возвращает, что прочитано 0 байт.

Определение функции:

int socket (int domain, int type, int protocol)

Аргумент domain задает используемый для взаимодействия набор протоколов (вид коммуникационной области), для стека протоколов TCP/IP он должен иметь символьное значение AF\_INET или PF\_INET.

Аргумент type задает режим взаимодействия:

SOCK\_STREAM - с установлением соединения; обеспечивают надежный дуплексный протокол на основе установления логического соединения. Если говорится о семействе протоколов TCP/IP, то это TCP;

SOCK\_DGRAM - без установления соединения; обеспечивают ненадежный сервис доставки датаграмм. В рамках TCP/IP это будет протокол UDP;

SOCK\_RAW –предоставляют доступ к некоторым датаграммам на уровне протокола IP. Они используются в особых случаях, например для просмотра всех ICMP-сообщений.

Аргумент protocol задает конкретный протокол транспортного уровня (из нескольких возможных в стеке протоколов). Если этот аргумент задан равным 0, то будет использован протокол "по умолчанию" (TCP для SOCK\_STREAM и UDP для SOCK\_DGRAM при использовании комплекта протоколов TCP/IP).

При удачном завершении своей работы данная функция возвращает дескриптор сокета - целое неотрицательное число, однозначно его идентифицирующее.

При обнаружении ошибки в ходе своей работы функция возвращает число "-1".

Все процессы (приложения или библиотеки DLL), вызывающие функции Winsock, должны инициализировать использование библиотеки DLL сокетов Windows перед вызовом других функций Winsock. Это также гарантирует, что Winsock поддерживается в системе.

Инициализация Winsock:

- Создайте объект WSADATA с именем wsaData.

- Вызовите WSAStartup и верните его значение в виде целого числа и проверьте наличие ошибок.

Функция WSAStartup вызывается для запуска использования WS2\_32.dll.

Структура WSADATA содержит сведения о реализации сокетов Windows. Параметр MAKEWORD(2,2) WSAStartup запрашивает версию 2.2 Winsock в системе и задает переданную версию в качестве самой высокой версии Windows Sockets поддержки, которую может использовать вызывающий объект.

Чтобы сервер принимал клиентские подключения, он должен быть привязан к сетевому адресу в системе. В следующем коде показано, как привязать сокет, который уже был создан с IP-адресом и портом. Клиентские приложения используют IP-адрес и порт для подключения к сети узла.

int bind (SOCKET s, SOCKADDR\_IN \*addr, int addrlen)

Пример:

err = bind( s, (LPSOCKADDR)&sin, sizeof(sin) );

Аргумент s задает дескриптор связываемого сокета.

Аргумент addr в общем случае должен указывать на структуру данных, содержащую локальный адрес, приписываемый сокету. Для сетей TCP/IP такой структурой является SOCKADDR\_IN.

Аргумент addrlen задает размер (в байтах) структуры данных, указываемой аргументом addr.

В случае успеха bind возвращает 0, в противном случае - "-1".

Системный вызов listen выражает желание выдавшей его программы- сервера ожидать запросы к ней от программ-клиентов и имеет следующий вид:

int listen (SOCKET s, int n);

Пример:

err = listen(s, SOMAXCONN);

Аргумент s задает дескриптор сокета, через который программа будет ожидать запросы к ней от клиентов. Socket должен быть предварительно создан системным вызовом socket и обеспечен адресом с помощью системного вызова bind.

Аргумент n определяет максимальную длину очереди входящих запросов на установление связи. Если какой-либо клиент выдаст запрос на установление связи при полной очереди, то этот запрос будет отвергнут.

Признаком удачного завершения системного вызова listen служит нулевой код возврата.

Для приема запросов от программ-клиентов на установление связи в программах-серверах используется системный вызов accept, имеющий следующий прототип:

int accept (SOCKET s, sockaddr\_in \*addr, int \*p\_addrlen;

Аргумент s задает дескриптор сокета, через который программа-сервер получила запрос на соединение (посредством системного запроса listen ).

Аргумент addr должен указывать на область памяти, размер которой позволял бы разместить в ней структуру данных, содержащую адрес сокета программы-клиента, сделавшей запрос на соединение. Никакой инициализации этой области не требуется.

Аргумент p\_addrlen должен указывать на область памяти в виде целого числа, задающего размер (в байтах) области памяти, указываемой аргументом addr.

Системный вызов accept извлекает из очереди, организованной системным вызовом listen, первый запрос на соединение и возвращает дескриптор нового (автоматически созданного) сокета с теми же свойствами, что и socket, задаваемый аргументом s. Этот новый дескриптор необходимо использовать во всех последующих операциях обмена данными.

Если очередь запросов на момент выполнения accept пуста, то программа переходит в состояние ожидания поступления запросов от клиентов на неопределенное время (хотя такое поведение accept можно и изменить).

Признаком неудачного завершения accept служит отрицательное возвращенное значение (дескриптор сокета отрицательным быть не может). Обычно для облуживания сервером краткосрочного взаимодействия со множеством клиентов можно реализовать «бесконечный» цикл в котором извлекается новый клиент из очереди, идет взаимодействие с ним по приему и передаче данных и последующее отключение, если же взаимодействие требует времени, или какой-то длительности по обмену сообщениями, то обслуживание клиента реализуется в новом потоке

После установления соединения с клиентом можно передавать и получать данные. Для этого в операционной системе Window используются системные вызовы recv для чтения и send для записи.

Системные вызовы recv и send имеют следующие прототипы:

int recv (SOCKET s, void \*buf, size\_t len, int flags);

int send (SOCKET s, const void \*buf, size\_t len, int flags);

Возвращаемое значение:

число принятых или переданных байтов в случае успеха или -1 в случае ошибки Аргумент s задает дескриптор сокета, через который принимаются данные.

Аргумент buf для вызова recv указывает на область памяти, предназначенную для размещения принимаемых данных, а для вызова send - область памяти, содержащая передаваемые данные.

Аргумент len задает размер (в байтах) области buf.

Аргумент flags зависит от системы, но и UNIX, и Windows поддерживают следующие флаги:

MSG\_OOB – следует послать или принять срочные данные.

MSG\_PEEK – используется для просмотра поступивших данных без их удаления из приемного буфера. После возврата из системного вызова данные еще могут быть получены при последующем вызове recv.

MSG\_DONTROUTE – сообщает ядру, что не надо выполнять обычный алгоритм маршрутизации. Как правило, используется программами маршрутизации или для диагностических целей.

Функция select определяет состояние одного или нескольких сокетов, ожидая, если необходимо, выполнения синхронного ввода-вывода.

int WSAAPI select(

[in] int nfds,

[in, out] fd\_set \*readfds,

[in, out] fd\_set \*writefds,

[in, out] fd\_set \*exceptfds,

[in] const timeval \*timeout

);

*[in] nfds*

Игнорируется. Параметр nfds включен только для совместимости с сокетами Berkeley.

*[in, out] readfds*

Необязательный указатель на набор сокетов, которые нужно проверить на удобочитаемость.

*[in, out] writefds*

Необязательный указатель на набор сокетов, которые нужно проверить на возможность записи.

*[in, out] exceptfds*

Необязательный указатель на набор сокетов, которые нужно проверить на наличие ошибок.

*[in] timeout*

Максимальное время ожидания select , представленное в виде структуры TIMEVAL . Установите для параметра тайм -аута значение null для блокировки операций.