Tutorial sobre winsock en C por MazarD
http://www.mazard.info

1.-Introducción

Al final me he decidido a escribir el tutorial sobre winsock, espero que os guste.
En éste tutorial vamos a introducirnos en el mundo de los sockets en c para windows, para hacerlo fácil voy a explicar sólo las conexiones tcp/ip.
Supongo que ya lo sabrás pero por si acaso decir que en una conexión a través de internet (o en lan, da igual) conectarán dos pcs, para ello necesitan dos parametros, el primero es la ip que sería cómo la dirección de tu casa y el otro el puerto que sería con quien quiere hablar dentro de la casa: con la mula, con el messenger, con el internet explorer...

Para ello uno de los dos ordenadores tiene que estar escuchando y esperando a que alguien conecte, a este ordenador se le llama el servidor, y al que conecta se le llama el cliente. Cuando cliente y servidor han conectado ya pueden recibir o enviar datos sin preocuparse de nada mas hasta que se cierre la conexión.

Bueno dejemos las explicaciones by barrio sésamo y vamos a lo que interesa.

2.-Funciones de winsock

Aquí intentaré explicar todas las funciones que vamos a utilizar en el programa. Si vas a programar un servidor, solo necesitas las funciones 2.01,2.02,2.04,2.06,2.07,2.08,2.09 y 2.09 y si vas a programar un cliente necesitas entender las 2.01,2.02,2.03,2.04,2.05,2.09 y 2.10.

2.01.-WSAStartup

Esta función lo que hace es inicializar el winsock en nuestro programa y definir la versión que vamos a utilizar, lo que cambia entre versiones es que unas tienen mas funciones a usar y otras menos, nosotros cómo vamos a lo básico sólo nos interesa compatibilidad y por tanto usaremos la version 2.0 que viene con todos los windows.
Para ello haremos lo siguiente:

Definimos una variable para la inicializacion
```
WSADATA wsa;
```

Inicializamos determinando la versión 2.0 para ello usamos MAKEWORD que pasará ese 2.0 a una variable de tipo WORD (16bits) que es lo que pide la función. También pasamos wsa por referéncia para que la modifique.

```
WSAStartup(MAKEWORD(2,0),&wsa);
```

2.02.-socket

Con la funcion socket lo que hacemos es crear un socket (una conexión para que se entienda) definiendo el protocolo de la conexión y el tipo de información que se trasmitirá a través de él.

definimos un socket
```
SOCKET sock;
```

Creamos el socket de tipo AF_INET (protocolo ipv4), SOCK_STREAM (tipo de paquetes para tcp), IPPROTOTCP (protocolo tcp).

```
sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
```

2.03.-gethostbyname

Con gethostbyname lo que hacemos es resolver un dominio cómo podría ser www.google.com, irc.irc-hispano.org, mazard.no-ip.org... a la ip que le corresponde, así luego podremos conectar.

Definimos una estructura del tipo hostent, esta estructura guardará toda la información a bajo nivel sobre el host al que vayamos a conectar. Si a gethostbyname le pasamos una ip directamente no la resolverá y la asignará a direc correctamente.

struct hostent *direc;

Ahora resolvemos el nombre del host y se lo asignamos a la direccion.
direc=gethostbyname("www.google.es");

Nota: Si no quisieramos resolver el nombre de dominio podriamos usar la función inet_addr para transformar la ip de cadena de caracteres a una variable de tipo in_addr. Pero es mejor utilizar esta función siempre, tiene la misma utilidad que in_addr y además te resuelve nombres de dominio.

2.04.-htons

Con esta funcion guardaremos el puerto al que vamos a conectar mas adelante.
En el protocolo de red los puertos no se expresan igual que los expresamos nosotros, no entraremos en detalles pero si tienes curiosidad busca big endian, little endian o tcp/ip network byte order en google.

un unsigned short se ajusta exactamente al rango en el que pueden ir los puertos, desde 1 hasta 65535.

u_short puerto;

Así transformamos el puerto 9999 "al lenguaje que entenderá" el socket.
puerto=htons(9999);

2.05.-connect

Con esta función conseguiremos conectar con la máquina remota, esta es la parte mas delicada, intentaré explicarla lo mas detalladamente posible. Para ello haremos uso de las funciones explicadas anteriormente.

Definimos una variable del tipo sockaddr_in que contendrá la información final de la dirección a la que vamos a conectar.

struct sockaddr_in remoto;

definimos que la direccion remota usa el protocolo ipv4
remoto.sin_family = AF_INET;
definimos el puerto al que vamos a conectar usando la función htons explicada anteriormente

remoto.sin_port = htons(9999);

Ahora viene lo mas delicado, tenemos que la estructura hostent definida anteriormente tiene un elemento que es la dirección ip (h_addr) usamos -> en lugar de '.' para el elemento puesto que direc es un puntero. Pero el h_addr podemos tratarlo cómo un in_addr (son estructuras diferentes pero pueden usarse igual) pero el compilador para sin_addr sólo admite in_addr así que le decimos que lo trate cómo tal con (struct in_addr *) a esto que hemos hecho se le llama casting, ahora tenemos que direc->h_addr es un puntero a un in_addr. Pero con esto no nos basta, ya que sin_addr tiene que ser un valor no un puntero así que a todo le añadimos * delante para que guarde el contenido del puntero, no el puntero en sí.
Se que si no has tratado mucho con punteros y castings puede resultar un poco rebuscado pero lo he explicado lo mejor que he podido. Para dudas en el foro.

```
remoto.sin_addr = *((struct in_addr *)direc->h_addr);
```

Tal y cómo se define en la msdn inicializamos el elemento sin_zero con zeros.

```
memset(remoto.sin_zero,0,8);
```

Ya podemos usar el connect con sock creado con la función socket, y la estructura remoto que hemos inicializado, pero connect necesita de un puntero a sockaddr no de un sockaddr_in así que volvemos a hacer el casting cogiendo la dirección de memoria de remoto con & y pasandole un puntero a esta direccion tratada como sockaddr.
Además tenemos que pasarle a la función el tamaño de la estructura.

```
connect(sock, (sockaddr *)&remoto, sizeof(sockaddr));
```

Una vez hecho esto si no se ha producido ningún error(cómo que la dirección a conectar no existe...)
ya podemos empezar a enviar y recibir datos.

2.06.-bind

La función bind lo que hace es asociar un socket a una dirección local (por si tubieramos varias) en nuestro caso podemos considerar que estamos preparando el socket para ponerlo a la escucha.
Para ello utilizaremos algunas de las funciones explicadas anteriormente.

Definimos una variable del tipo sockaddr_in que contendrá la información de nuestro pc para que puedan conectar.

```
struct sockaddr_in local;
```

definimos que estamos usando el protocolo ipv4

```
local.sin_family = AF_INET;
```

Definimos la ip local "por defecto"

```
local.sin_addr.s_addr = INADDR_ANY;
```

Definimos el puerto que vamos a poner a la escucha.

```
local.sin_port = htons(puerto);
```

Llamamos a la funcion bind con el socket que habiamos creado la dirección local definida pasada cómo puntero y tratada como SOCKADDR, podríamos decir para entendernos que SOCKADDR y sockaddr_in son comptabiles y por tanto podemos tratarlo así.
Además pasamos el tamaño de la estructura (cosas internas para la función)

```
bind(sock, (SOCKADDR*) &local, sizeof(local))
```

2.07.-listen

Con esta función pondremos definitivamente el socket a la escucha, no se saldrá de ésta función hasta que alguien haya conectado o se produzca un error.

Ponemos a la escucha el socket creado y le decimos que sólo espere a una conexión.
listen(sock,1);

2.08.-accept

Una vez alguien a conectado a nuestro puerto que estaba a la escucha, tenemos que aceptar dicha conexión, esto lo hacemos con la función accept.

int len
len=sizeof(struct sockaddr);

Al aceptar la conexión local contendrá la información de la conexión que se ha establecido una vez mas tiene que tratarse como sockaddr.
sock=accept(sock,(sockaddr*)&local,&len);

Ahora ya hemos establecido la conexión y estamos preparados para enviar y recibir datos.

2.09.-recv

Con ésta función recibiremos datos de la máquina remota cuando los envie.

int i;
char Buffer[1025];

Le decimos el socket del que queremos recibir datos, la cadena que recibirá la información, el número de carácteres (bytes) que cómo máximo queremos recibir, el último parámetro es un flag que marca cómo va actuar la función, a lo simple vamos a poner 0 que es además lo mas común.
La función devolverá el número de carácteres recibidos, -1 si se ha producido un error o 0 si se ha cerrado el socket.

i=recv(sock, buffer, 1024, 0);

Lógicamente la función recv no nos guarda el final de cadena en el buffer, lo haremos nosotros a partir de los carácteres recibidos. (cuidado con ésto, hay que controlar que i no valga -1 antes de seguir con todo el tratamiento)
buffer*=0;*

2.10.-send

Usando esta función enviaremos datos a la máquina remota.

char Buffer[]="Fucking world";

Le decimos que envie al socket conectado sock la cadena "Fucking world" tenemos que decirle cuantos carácteres hay que enviar de la cadena. El último parámetro definiria el comportamiento de los paquetes al ser enviados, le daremos 0 que es lo mas normal.
La función devolverá el número de carácteres que se han enviado o -1 en caso de que ya no estemos conectados.

int enviado=0;
enviado=send(sock,Buffer,strlen(Buffer), 0);

3.-Código de ejemplo

Ahora que ya sabemos cómo funciona cada una de las funciones que vamos a utilizar vamos a programar el típico ejemplo en temas de estos de sockets, es decir un chat. Por si no lo sabes para hacer pruebas puedes usar el cliente y el servidor en el mismo ordenador poniendo cómo ip "127.0.0.1"
Si te funciona así puedes estar seguro de que funcionará en internet, sin tener en cuenta el tema de abrir puertos en los routers, firewalls, blablabla

En el chat el servidor solo recibira datos y el cliente sólo los enviará, pero está claro que puedes usar recv y send tanto en servidor como en cliente. Está así para que no quede cutre pero tampoco liar a meter hilos en el ejemplo.

3.1.-Código de un chat cómo cliente

```
int main()
{
   WSADATA wsa;
   SOCKET sock;
   struct hostent *host;
   struct sockaddr_in direc;
   int conex;
   char Buffer[1024];
   int len;

   //Inicializamos
   WSAStartup(MAKEWORD(2,2),&wsa);

   //resolvemos el nombre de dominio localhost, esto se resolverá a 127.0.0.1
   host=gethostbyname("localhost");

   //creamos el socket
   sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
   if (sock==-1)
   {
      printf("Error al crear el socket");
      return -1;
   }
   //Definimos la dirección a conectar que hemos recibido desde el gethostbyname
   //y decimos que el puerto al que deberá conectar es el 9999 con el protocolo ipv4
   direc.sin_family=AF_INET;
   direc.sin_port=htons(9999);
   direc.sin_addr = *((struct in_addr *)host->h_addr);
   memset(direc.sin_zero,0,8);

   //Intentamos establecer la conexión
   conex=connect(sock,(sockaddr *)&direc, sizeof(sockaddr));
   if (conex==-1)  //si no se ha podido conectar porque no se ha encontrado el host o no
                   //está el puerto abierto
   {
      printf("No se ha podido conectar\n");
      return -1;
   }

   printf("[MiniXat para tutorial de sockets MazarDZone Foro]\n");
   printf("[escribe el texto a enviar o 'salir' para salir  ]\n");
```

```
   while (len!=-1 && strcmp(Buffer,"salir")!=0) //mientras el socket no se haya
desconectado
                                        //y no se escriba salir
   {
      printf("Texto a enviar:");
      fgets(Buffer,1023,stdin);  //pedir texto a enviar por pantalla
      len=send(sock,Buffer,strlen(Buffer),0); //enviar el texto que se ha introducido
   }
   return 0;
}
```

3.2.-Código de un chat cómo servidor

```
#include <winsock2.h> //la cabezera para usar las funciones de winsock
#include <stdio.h>

/*linkamos a la libreria del winsock, también puedes hacerlo desde
  project->settings->link si usas ms visual c++ */
#pragma comment(lib,"ws2_32.lib")

int main()
{
   WSADATA wsa;
   SOCKET sock;
   struct sockaddr_in local;
   int len=0;
   char Buffer[1024];

   //Inicializamos
   WSAStartup(MAKEWORD(2,0),&wsa);

   //Creamos el socket
   sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);

   //defnimos dirección por defecto, ipv4 y el puerto 9999
   local.sin_family = AF_INET;
   local.sin_addr.s_addr = INADDR_ANY;
   local.sin_port = htons(9999);

   //asociamos el socket al puerto
   if (bind(sock, (SOCKADDR*) &local, sizeof(local))==-1)
   {
      printf("error en el bind\n");
      return -1;
   }

   //ponemos el socket a la escucha
   if (listen(sock,1)==-1)
   {
      printf("error en el listen\n");
      return -1;
   }

   len=sizeof(struct sockaddr);
```

```
   //hay una conexión entrante y la aceptamos
   sock=accept(sock,(sockaddr*)&local,&len);

   printf("[SERVIDOR MiniXat para tutorial de sockets MazarDZone Foro]\n");
   printf("[Cuando se vaya recibiendo texto aparecera en pantalla    ]\n");

   while (len!=0) //mientras estemos conectados con el otro pc
   {
      len=recv(sock,Buffer,1023,0); //recibimos los datos que envie

      if (len>0)  //si seguimos conectados
      {
         Buffer[len]=0; //le ponemos el final de cadena
         printf("Texto recibido:%s",Buffer); //imprimimos la cadena recibida
      }
   }

   return 0;
}
```