UNIVERSIDAD DE OVIEDO
Departamento de Ingeniería Eléctrica,
Electrónica, de Computadores y Sistemas

Programación de sockets Windows lenguaje C bajo Qt

Ignacio Alvarez García – Diciembre 2016

INDICE

1. Preparación del programa

Añadir enlace a la librería ws2_32.lib en archivo de proyecto .pro

```
LIBS += -lws2_32
```

Añadir #include <winsock2.h> en el archivo de código fuente.

```
#include <winsock2.h>
```

Añadir inicialización de Windows Sockets al principio de main()

```
WSADATA WsaData;
WORD wVersionRequerida = MAKEWORD (2, 2);
WSAStartup (wVersionRequerida, &WsaData);
```

2. Creación de socket

El tipo de datos para un identificador socket en Windows es SOCKET (a diferencia del estándar que utiliza int).

1) Crear una variable de este tipo.
2) Inicializar la variable con llamada a función socket( ), familia AF_INET, tipo SOCK_STREAM para sockets TCP (con conexión) o SOCK_DGRAM para sockets UDP (sin conexión), protocolo 0.
3) Preparar variable tipo struct sockaddr_in con una dirección local y puerto válidos.
   La dirección local puede ser 127.0.0.1 (localhost) para comunicaciones dentro del mismo equipo.
   El puerto es un valor de 16 bits, que puede ser 0 para el cliente, debe ser un valor preferiblemente entre 49152 y 65535 para el servidor – utilizar función htons( ) para convertir el nº deseado al formato de red).
4) Enlazar con un puerto local con la función bind( )

3. Conexión de socket TCP

La conexión se realiza de forma diferente en el socket cliente y el servidor.

3.1. Socket cliente

5.c) Preparar variable tipo struct sockaddr_in con una dirección remota y puerto válidos de un servidor conocido.
6.c) Conectar con el socket remoto mediante la función connect( ).
7.c) Si todo es correcto, ya se pueden utilizar las funciones send( ) y recv( ) sobre el socket cliente.

3.2. Socket servidor

5.s) Llamar a función listen( ) para anunciar que se está dispuesto a recibir conexiones.
6.s) Preparar variable tipo struct sockaddr_in donde se rellenará la dirección del cliente que se conecta.

7.s) Esperar conexión mediante la función accept( ). Esta función devolverá un nuevo identificador de socket que se almacenará en una 2ª variable, la cual se utilizará en la comunicación con las funciones send( ) y recv( ).

4. Conexión de socket UDP

No es necesaria una conexión para sockets UDP. Una vez creados, se pueden enviar datos con las funciones sendto( ) y recvfrom( ).

5. Envío y recepción

Sockets con conexión: enviar con función send( ), recibir con función recv( ).
Sockets sin conexión: preparar variable tipo struct sockaddr_in para enviar a la dirección deseada con sendto( ), o para recibir con recvfrom( ).

6. Desconexión

Cerrar socket con función closesocket( ).

7. Ejemplos y documentación

Documentación completa de sockets "estilo Windows" en:
https://msdn.microsoft.com/es-es/library/windows/desktop/ms741416(v=vs.85).aspx

Tutorial para realización de cliente y servidor (para Windows) en:
http://www.binarytides.com/winsock-socket-programming-tutorial/

Documentación completa de sockets POSIX (para Linux y casi todas las plataformas no Windows):
http://pubs.opengroup.org/onlinepubs/9699919799/functions/V2_chap02.html#tag_15_10

Tutoriales para realización de cliente y servidor con sockets POSIX en:
http://www.linuxhowtos.org/C_C++/socket.htm
http://www.binarytides.com/socket-programming-c-linux-tutorial/