

WIN API

17/02/2017

1.1

|  |  |
| --- | --- |
| **Nombre.** | **Matricula** |
| **Jesús Alberto Goiz Barrales.** | **24400085** |
| **Juan Alberto Gutiérrez.** | **24400063** |
| **Brenda Robles Antonio.** | **24400073** |
| **Rene Moratilla Montes.** | **24400075** |
| **Guillermo Vivaldo Vazquez** | **24400093** |

**Control De Versiones**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre Del Archivo | Versión | Fecha de modificación | Autor | Comentarios |
| tsp WIN API.docx | 1.0 | 17/02/2017 | BRA |  |
| Csoft-WIN\_API-170217.docx | 1.1 | 22/02/2017 | JAGC | Se cambio el documento a el formato estándar, se modificó el nombre a formato estándar. |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**WIN API (Creación De Interfaces en C)**

Para el presente curso supondré que estás familiarizado con la programación en C y C++ y también con las aplicaciones y el entorno Windows, al menos al nivel de usuario. Sin embargo, no se requerirán muchos más conocimientos.

El curso pretende ser una explicación de la forma en que se realizan los programas en Windows usando el API. Las explicaciones de las funciones y los mensajes del API son meras traducciones del fichero de ayuda de WIN32 de Microsoft, y sólo se incluyen como complemento.

Para empezar, vamos a ponernos en antecedentes. Veamos primero algunas características especiales de la programación en Windows.

[^](http://winapi.conclase.net/curso/index.php?cap=000#inicio)

Los programas Windows son independientes de la máquina en la que se ejecutan (o al menos deberían serlo), el acceso a los dispositivos físicos se hace a través de interfaces, y nunca se accede directamente a ellos. Esta es una de las principales ventajas para el programador, ya que no hay que preocuparse por el modelo de tarjeta gráfica o de impresora, la aplicación funcionará con todas, y será el sistema operativo el que se encargue de que así sea.

Recursos: Un concepto importante es el de recurso. Desde el punto de vista de Windows, un recurso es todo aquello que puede ser usado por una o varias aplicaciones. Existen recursos físicos, que son los dispositivos que componen el ordenador, como la memoria, la impresora, el teclado o el ratón y recursos virtuales o lógicos, como los gráficos, los iconos o las cadenas de caracteres.

Por ejemplo, si nuestra aplicación requiere el uso de un puerto serie, primero debe averiguar si está disponible, es decir, si existe y si no lo está usando otra aplicación; y después lo reservará para su uso. Esto es necesario porque este tipo de recurso no puede ser compartido.

Lo mismo pasa con la memoria o con la tarjeta de sonido, aunque son casos diferentes. Por ejemplo, la memoria puede ser compartida, pero de una forma general, cada porción de memoria no puede compartirse, (al menos en los casos normales, veremos que es posible hacer aplicaciones con memoria compartida), y se trata de un recurso finito. Las tarjetas de sonido, dependiendo del modelo, podrán o no compartirse por varias aplicaciones. Otros recursos como el ratón y el teclado también se comparten, pero se asigna su uso automáticamente a la aplicación activa, a la que normalmente nos referiremos como la que tiene el "foco", es decir, la que mantiene contacto con el usuario.

Desde nuestro punto de vista, como programadores, también consideramos recursos varios componentes como los menús, los iconos, los cuadros de diálogo, las cadenas de caracteres, los mapas de bits, los cursores, etc. En sus programas, el Windows almacena separados el código y los recursos, dentro del mismo fichero, y estos últimos pueden ser editados por separado, permitiendo por ejemplo, hacer versiones de los programas en distintos idiomas sin tener acceso a los ficheros fuente de la aplicación.

Los programas en Windows están orientados a eventos, esto significa que normalmente los programas están esperando a que se produzca un acontecimiento que les incumba, y mientras tanto permanecen aletargados o dormidos.

Un evento puede ser por ejemplo, el movimiento del ratón, la activación de un menú, la llegada de información desde el puerto serie, una pulsación de una tecla...

Esto es así porque Windows es un sistema operativo multitarea, y el tiempo del microprocesador ha de repartirse entre todos los programas que se estén ejecutando. Si los programas fueran secuenciales puros, esto no sería posible, ya que hasta que una aplicación finalice, el sistema no podría atender al resto.

Debido a la complejidad de los programas Windows, normalmente los dividiremos en varios ficheros fuente, que compilaremos por separado y enlazaremos juntos.

Cada compilador puede tener diferencias, más o menos grandes, a la hora de trabajar con proyectos. Sin embargo creo que no deberías tener grandes dificultades para adaptarse a cada uno de ellos.

En el presente curso trabajaremos con el compilador de "Bloodshed", que es público y gratuito, y puede descargarse de Internet en la siguiente URL: <http://www.bloodshed.net/>.

Para crear un proyecto Windows usando este compilador elegiremos el menú "File/New Project". Se abrirá un cuadro de diálogo donde podremos elegir el tipo de proyecto. Elegiremos "Windows Application" y "C++ Project". A continuación pulsamos "Aceptar".

El compilador crea un proyecto con un fichero C++, con el esqueleto de una aplicación para una ventana, a partir de ahí empieza nuestro trabajo.

Convenciones: En parte para que no te resulte muy difícil adaptarte a la terminología de Windows, y a la documentación existente, y en parte para seguir mi propia costumbre, en la mayoría de los casos me referiré a componentes y propiedades de Windows con sus nombres en inglés. Por ejemplo, hablaremos de "button", "check box", "radio button", "list box", "combo box" o "property sheet", aunque algunas veces traduzca sus nombres a español, por ejemplo, "dialog box" se nombrará a menudo como "cuadro de diálogo".

Además hablaremos a menudo de ventanas "overlapped" o superponibles, que son las ventanas corrientes. Para el término "pop-up" he desistido de buscar una traducción.

También se usarán a menudo, con relación a "check boxes", términos ingleses como checked, unchecked o grayed, en lugar de marcado, no marcado o gris.

Owner-draw, es un estilo que indica que una ventana o control no es la encargada de actualizarse en pantalla, esa responsabilidad es transferida a la ventana dueña del control o ventana.

Para "bitmap" se usará a menudo la expresión "mapa de bits".

Los controles son la forma en que las aplicaciones Windows intercambian datos con el usuario. Normalmente se usan dentro de los cuadros de diálogo, pero en realidad pueden usarse en cualquier ventana.

Existen bastantes, y los iremos viendo poco a poco, al mismo tiempo que aprendemos a manejarlos.

Los más importantes y conocidos son:

* control estatic: son etiquetas, marcos, iconos o dibujos.
* control edit: permiten que el usuario introduzca datos alfanuméricos en la aplicación.
* control list box: el usuario puede escoger entre varias opciones de una lista.
* control combo box: es una combinación entre un edit y un list box.
* control scroll bar: barras de desplazamiento, para la introducción de valores entre márgenes definidos.
* control button: realizan acciones o comandos, de button de derivan otros dos controles muy comunes:
* control check box: permite leer variables de dos estados "checked" o "unchecked"
* control radio button: se usa en grupos, dentro de cada grupo sólo uno puede ser activado.

Referencia: <http://winapi.conclase.net/curso/index.php?cap=000>