UNIVERSIDAD LASALLISTA BENAVENTE

ESCUELA DE INGENERIA EN COMPUTACION

CON ESTUDIOS INCORPORADOS ALA

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
CLAVE: 8793-16

"MANEJO DE LAS APIS ESTANDARES DE WINDOWS CON VISUAL BASIC"

T E S I S :

QUE PARA OBTENER EL TITULO DE:
INGENIERO EN COMPUTACION

P R E S E N T A:
ADOLPH HEERVER RODRIGUEZ SOSALES

ASESOR:
ING. CLUDIA MAYELA ALCATRAZ AVEDAÑO

CELAYA GTO **2003**

Dedico esta tesis a mi mamita, la cual siempre me ha demostrado su amor, a la que quiero, respeto mucho y le debo lo que soy.

ÍNDICE

TEMA **PÁGINA**

Agradecimientos personales

A Diosito por dejarme vivir de manera tranquila y satisfactoria estos 5 años de mi vida y darme la oportunidad de haber terminado y seguir adelante.

A mi mamita Margarita que desde el comienzo de la carrera casi siempre que llegaba me preguntaba que como me había ido, que siempre estuvo a un lado mío y se esforzó por ayudarme en lo que estuviera a su alcance para verme terminar la carrera.

A mi papito Raúl que de algún modo u otro aunque no fuera directamente conmigo se preocupaba de cómo me iba de de alguna manera me dio la libertad de llevar la carrera como yo quisiera, sin imponerme nada ni reclamarme nada.

A mi hermano low que ahora atraviesa la preparatoria, un lugar bonito y difícil en el que se encuentra, por ser mi hermano y que aunque le canso y le caigo gordo a veces, es mi amigo de toda la vida y que nos veremos para siempre, para tener alguien con quien pelearme y jugar.

A mi angelito de la guarda que me cuida y esta al pendiente de mi siempre, que a veces lo olvido, pero que seguro estuvo a mi lado cuando me desvelaba haciendo la tesis.

A mi mismo por haber terminado la carrera, que me costo cierto trabajo pero que con ayuda de todos a mi alrededor y yo pude sacar, no exitoso como espere terminarla al final pero digna y orgullosamente con mis calificaciones de acuerdo a lo que estudie y aprendí, con ciertas ayudas como todo para levantar el animo.

A mis amigos, de toda la vida, que conforme los conocía y después nos apartábamos por motivos de estudio nos seguimos frecuentando, luis, pollo, niño, ulises, gustavo, emilio, mauricio, balta, pepe, manuel, afortunadamente tengo varios que los estimo, y desde luego a los que conocí en la carrera, arturo, héctor, melgarito, beto, ivan y mis compañeros y compañeras de clase, que conviví 5 años de mi vida con ellos casi diario.

Agradecimientos Académicos

Universidad Lasallista Benavente ULSAB. Por haberme dejado ser uno de sus estudiantes.

Ingeniero Miguel Ángel Jamaica Arreguín. Por tenernos paciencia y dedicar tiempo de su clase durante toda la carrera para platicar con nosotros de diferentes temas interesantes en su momento y polémicos.

Ingeniero Claudia Mayela Alcaraz Avendaño. Por haber sido mi asesor de tesis y dar sus clases muy apegadas a las reglas y estructura académica.

Todos mis profesores que a veces me hacían pasar desvelos por sus trabajos y exámenes, a los disgustos y satisfacciones, a los siempre fueron dedicados y los que no tanto en su trabajo, a Araceli por presionarnos y alentarnos a sacar adelante la Tesis, en fin les agradezco sinceramente.

Introducción

API (Aplication Program Interface), aplicación de interfase de programa, estas son funciones que ejecuta continuamente el sistema operativo Windows para su funcionamiento correcto en determinadas tareas en todos sus aspectos, desde crear el marco de una ventana, manejo de Internet y navegadores, redes, aplicaciones para el manejo de gráficos en 2D y 3D, audio, dispositivos como es el teclado, ratón, monitor, scanner, aplicaciones con las que trabaja Windows como los diálogos comunes que son estándares para todos los programas bajo esta plataforma, herramientas de uso común como información del sistema y manejo del registro de Windows, conectividad en base de datos, entre muchas otras cosas.

Estas están organizadas y estructuradas por familias, ordenadas de tal manera para utilizarlas desde el programa que las requiera cuantas veces sea necesario, las APIs son utilizadas libremente desde el lenguaje de programación que se utilice en su diseño desde Visual Basic, C++, por mencionar algunos, de los mas poderosos y utilizados actualmente.

Existen familias de APIs que son específicas para alguna función en concreto y que son desarrolladas por empresas externas a Microsoft y el sistema operativo Windows pero que funcionan correctamente bajo esta plataforma como son las de Sun y las de conectividad a sus bases de datos desde su lenguaje Java y otras como lo es MySQL desde Internet por mencionar algunas y otras en librerías externas a las que son básicas para el sistema operativo como las de ciertos dispositivos como lo puede ser una tarjeta gráfica por mencionar la Voodoo 3dfx que contiene su propio controlador en su librería especial para gráficos que Windows no contiene en su sistema llamado Glide, como lo es el estándar actual DirectX y que ahora todas las tarjetas trabajan bajo estas APIs de DirectX para interactuar con el sistema operativo Windows, al igual que las tarjetas de audio y ciertos dispositivos externos como palancas de juegos.

Las APIs en Windows surgieron para hacer más sencillo la manera de programar para el sistema operativo y obtener el máximo provecho de él, además de crear un estándar en librerías y formatos, ser equitativo para los diferentes lenguajes de programación, hacerlo mas robusto y fácil de corregir, utilizando la programación con herencia, funciones, estructuras y boques entendibles, fáciles de corregir y actualizar.

Cabe mencionar que la mayoría de las APIs anteriores desde un Windows 98 y actuales Windows XP son compatibles, varían pocas cosas y son sistemas operativos con un núcleo muy diferente ya que Windows XP esta basado en la familia de servidores de Microsoft que son Windows NT y 2000, que próximamente serán Windows .Net, mientras que Windows 98 es para el hogar, pero sin dejar de lado que la mayoría de programas pueden ser ejecutados en ambas plataformas, muchos no funcionaran correctamente ya que necesariamente requerirán de ciertos cambios, pero si internamente utilizan APIs generales, como cajas de dialogo, el registro entre otros funcionaran bien con tan solo instalarlos en ambas plataformas.

Se requieren de familias de APIs especiales y concretas como las que actualmente utiliza el programa de comunicación en línea popular Windows Messenger de la página de correo gratuito Hotmail será necesario adecuar el programa para la versión de Windows a utilizar. Concluyendo se quiere dar a entender que las APIs van en constante evolución y cambio, como las hay genéricas y especificas que en donde su utilización requiere un cobro por su completa documentación, pero que las utilizadas en esta tesis son libres.

En esta tesis se mostraran las capacidades de lo que se puede hacer son las APIs, con el fin de demostrar que sin tomar mucho esfuerzo ni derrochar recursos en comprar controles especiales (que internamente utilizan las APIs) para su utilización de nuestro programa se puede hacer un software profesional y competitivo, en diferentes tipos de lenguajes de programación y bajo los diferentes sistemas operativos de Windows tomando en cuenta esta herramienta y sacando el máximo provecho de la misma.

También es necesario conocer la historia de Windows, el por que la necesidad de las APIs, el desconocimiento en el nivel educativo ni la apertura de información masiva y gratuita, ya sea desde Internet o bibliografía variada y especializada al alcance del estudiante promedio, como el marco teórico, las características de las APIs, estructura, modo de utilización y herramientas trabajadas.

Las APIs son tan necesarias para programar en sistemas operativos Windows como el mismo lenguaje de programación lo es, sin ellas seria extremadamente difícil obtener el máximo provecho de la aplicación a desarrollar sin importar a que este dirigida, desde sistemas con base de datos, video juegos o conectividad en red e Internet.

A lo largo de los cuatro capítulos se mostrarán las características y diferentes familias estándares y principales para el sistema operativo Windows, comenzando con el marco teórico, introducción y características en su modo de empleo al igual que herramientas usadas para los primeros dos capítulos. En el tercer capítulo se comienzan a dar ejemplos prácticos y reales de la utilización de las APIs, no solo las características de estas si no un ejemplo para plasmar su uso y modo de empleo, el cuarto capítulo al igual que el tercero esta dividido por familias y estructuras en las APIs de Windows que están organizadas de tal modo que sea fácil su entendimiento y manejo, tratando de dar ejemplo de cada una de ellas, es necesario aclarar que estas no son todas las APIs existentes, solo son algunas de ellas, las mas utilizadas, faltan muchísimas mas, pero estas ilustran de modo a enseñanza que familias hay, que se hace con ellas, que se puede hacer y que no se puede hacer con ellas. Se utilizan en el lenguaje de programación de Visual Basic, para un fácil entendimiento ya que el lenguaje se presta para su uso de manera entendible y estructurada, sin complicar otras características como el diseño de las ventanas a la hora de mostrar las APIs, pero concluyendo que en el lenguaje que mas se utilizan es C++, y que se mejoran para su fácil uso en el .Net de Microsoft con C#, C++ .Net y Visual Basic .Net, ya que estas están saliendo al mercado latinoamericano y su documentación al mismo tiempo de la elaboración de esta tesis.

GENERALIDADES

CAPITULO I

1.1 ¿Qué es un sistema operativo?

Un sistema operativo es el software (es la parte de programas con las que trabaja una computadora) básico que controla una computadora en todos sus aspectos, éste se encarga de la manipulación y coordinación del hardware (es la parte física de la computadora, las piezas internas y electrónicas), como las memorias, teclado, mouse, etc.; organiza los archivos en diferentes tipos de almacenamiento como discos flexibles, discos duros, cintas magnéticas, cd's, y trabaja con los errores que pueda haber con el hardware y la pérdida de datos.

Los sistemas operativos controlan diferentes procesos en la computadora. Uno de los procesos más importante es la interpretación de los comandos que permiten al usuario comunicarse con el ordenador. Algunos intérpretes de instrucciones están basados en texto y exigen que las instrucciones sean tecleadas (uso del teclado). Otros están basados en gráficos, y permiten al usuario comunicarse señalando y haciendo clic en un icono (uso del mouse). Por lo general, los intérpretes basados en gráficos son más sencillos de utilizar.

Los sistemas operativos pueden ser de tarea única o multitarea (varios procesos al mismo tiempo). Los sistemas operativos de tarea única, más primitivos, sólo pueden manejar un proceso en cada momento. Por ejemplo, cuando la computadora está imprimiendo un documento, no puede iniciar otro proceso ni responder a nuevas instrucciones hasta que se termine la impresión.

Todos los sistemas operativos modernos son multitarea y pueden ejecutar varios procesos simultáneamente; esto no quiere decir que al mismo tiempo, las computadoras tienen una Unidad Central de Procesamiento llamada CPU, éste no es más que el procesador de la computadora, el cerebro principal de la misma.

El mecanismo que se emplea más a menudo para lograr esta ilusión es la multitarea por segmentación de tiempos, en la que cada proceso se ejecuta individualmente durante un periodo de tiempo determinado. Si el proceso no finaliza en el tiempo asignado, se suspende y se ejecuta otro proceso. El sistema operativo se encarga de controlar el estado de los procesos suspendidos. También cuenta con un mecanismo que determina el siguiente proceso que debe ejecutarse. Los procesos parecen efectuarse simultáneamente por la alta velocidad del cambio de contexto.

1.1.1 Sistemas operativos actuales

Los sistemas operativos empleados normalmente son UNIX, Macintosh OS, MS-DOS, OS/2 y Windows. El UNIX y sus clones permiten múltiples tareas y múltiples usuarios. Su sistema de archivos proporciona un método sencillo de organizar archivos y permite la protección de archivos.

Sin embargo, las instrucciones del UNIX no son intuitivas; éstas son a través de comandos y tecleadas directamente por el usuario. Para obtener un buen provecho de este sistema operativo se necesita tener buenos conocimientos técnicos del mismo; uno de sus hijos sería el famoso LINUX, que tiene la base de UNIX pero tratado para ser más sencillo de utilizar y ligero, con simuladores de ambientes gráficos.

Otros sistemas operativos multiusuario y multitarea son: OS/2, desarrollado inicialmente por Microsoft Corporation e International Business Machines (IBM), y Windows, desarrollado por Microsoft. El sistema operativo multitarea de las computadoras Apple se denomina Macintosh OS. El DOS y su sucesor, el MS-DOS, son sistemas operativos populares entre los usuarios de computadoras personales. Estos sólo permiten un usuario y una tarea.

1.1.2 Tecnologías futuras

Los sistemas operativos siguen evolucionando constantemente y de manera rapidísima y cada vez se hacen más estrechas, en algunos casos, y en otros más amplias sus diferencias. ¿Qué quiero decir con esto? pues fácil, los sistemas operativos específicos para servidores son internamente diferentes a los de usuarios caseros, pues desde el manejo del hardware, como pueden ser estaciones de trabajos no con uno sino hasta cuatro procesadores o dos monitores y seguridad de archivos muy alta, en cambio un usuario casero no necesita esto, por lo tanto su sistema operativo es notablemente diferente, más sencillo y ligero (en esto aclaro que son muy diferentes); pero, por otro lado, son muy parecidos, ya que actualmente hay diferentes marcas a escoger, y éstas ofrecen sus sistemas operativos con muchas similitudes y mejoras para estar siempre a la cabeza y competir entre ellas.

Los sistemas operativos distribuidos están diseñados para su uso en un grupo de ordenadores conectados pero independientes que comparten recursos. En un sistema operativo distribuido, un proceso puede ejecutarse en cualquier ordenador de la red (normalmente, un ordenador inactivo en ese momento) para aumentar el rendimiento de ese proceso. En los sistemas distribuidos, todas las funciones básicas de un sistema operativo, como mantener los sistemas de archivos, garantizar un comportamiento razonable y recuperar datos en caso de fallos parciales, resultan más complejas.

Actualmente los dos sistemas operativos más fuertes en el mercado para uso comercial son Windows Xp por Microsoft y Mac Os Jaguar por Macintosh, pero hay competidores nuevos como Linux y todos sus entornos gráficos como el Red Hat, el Mandraque o Gnome; sin embargo, por el lado industrial, éstos son muy diferentes, Microsoft, el desarrollador de sistemas operativos más fuerte tiene a su Windows NT o Windows 2000, y UNIX por el otro.

Entonces, para concluir: existen diferentes sistemas operativos, los cuales son específicos para tareas únicas y otros para ser comerciales; pero hay dos ramas principales: los de tipo servidor y los de estación de trabajo, y diferentes marcas y empresas que los desarrollan, aunque actualmente y desde a principio de los 90 Microsoft y su famoso Windows es quien ha reinado y proliferado de manera asombrosa a lo largo y ancho de todo el mundo, sin barreras de idiomas, y lo ha hecho por su fácil manejo, apariencia y muchas otras cosas más que se tomarán en cuenta a lo largo de toda esta tesis, el cómo comenzó, fue evolucionado y qué se espera para su futuro, y todo lo relacionado con éste en cuanto al fabuloso mundo de las APIS (Interfase de aplicación de Programa).

1.2 Historia de windows

1.2.1 Windows 1.0

Microsoft comenzó el desarrollo del "ADMINISTRADOR DE INTERFAZ", que posteriormente derivó en Microsoft Windows en Septiembre de 1981. La interfaz inicial tenía menús ubicados en la parte inferior de la ventana y la interfaz sufrió un cambio en 1982 cuando se diseñaron los ahora comunes menús desplegables.

Esto ocurrió después de Apple Lisa, un experimento de Apple por llevar una interfaz gráfica al usuario. Sin embargo, ocurrió antes de Macintosh. Windows prometía una interfaz gráfica fácil de usar y la utilización de gráfica independiente del dispositivo, así como el soporte de multitarea.

La Primera versión de Microsoft Windows, fue lanzada el 20 de noviembre de 1985. Tomó un total de 55 programadores para desarrollarlo y no permitía ventanas en cascada, solamente en mosaico.

Microsoft publicó la primera versión de Windows, una interfaz gráfica de usuario (GUI) para su propio sistema operativo (MS-DOS) que había sido incluido en el IBM PC y ordenadores compatibles desde 1981. La interfaz gráfica fue creada después del MacOS de Apple.

Las siguientes fueron las principales características de Windows 1.0:

- Interfaz gráfica con menús desplegables, no había ventanas en cascada y soporte para mouse.
- Gráficos de pantalla e impresora independientes del dispositivo.
- Multitarea cooperativa entre las aplicaciones Windows.

1.2.2 Windows 2.0

Segunda versión de Microsoft Windows, lanzada en 1987. Windows 2.0 que tenía más características que Windows 1.0, tales como iconos y ventanas traslapadas. Nacen aplicaciones como Excel, Word for Windows, Corel Draw y PageMaker.

Las siguientes fueron las principales características de Windows 2.0:

- Ventanas traslapadas.
- Archivos PIF para aplicaciones DOS.

1.2.3 Windows 3.0

Una completa reconstrucción de Windows con muchas nuevas facilidades, tales como la habilidad de direccionar más allá de 640 KB en la memoria. Fue lanzado en 1990, y vendió más de 10 millones de copias.

Las siguientes fueron las principales características de Windows 3.0:

- Modo estándar (286), con soporte de memoria grande.
- Modo Mejorado 386, con memoria grande y soporte de múltiples sesiones para el DOS.
- Se agregó el Administrador de Programas y Administrador de Archivos.
- Soporte para Red.
- Soporte para más de 16 colores.
- Soporte para cajas de selección, menús jerárquicos y los archivos. INI privados para cada aplicación, estos empezaron a cobrar más valor.

1.2.4 OS/2 1

Durante la segunda mitad de los 80, Microsoft e IBM habían estado desarrollando conjuntamente OS/2 como sucesor del DOS, para sacar el máximo provecho a las capacidades del procesador Intel 80286. OS/2 utilizaba el direccionamiento hardware de memoria disponible en el Intel 80286 para poder utilizar hasta 16 MB de memoria.

La mayoría de los programas de DOS estaban, por el contrario, limitados a 640 K de memoria. OS/2 1 también soportaba memoria virtual y multitarea.

Más adelante IBM añadió, en una de las primeras versiones de OS/2 un sistema gráfico llamado Presentation Manager (PM). Aunque en muchos aspectos era superior a Windows, su API (Aplicación de Programa de Interfase), unas de las primeras era incompatible con la que usaban los programas Windows. Entre otras cosas, Presentation Manager localizaba el eje de coordenadas X,Y en la parte inferior izquierda de la pantalla como las coordenadas cartesianas, mientras que Windows situaba el punto 0,0 en la esquina superior izquierda de la pantalla como otros sistemas informáticos basados en ventanas.

A principio de los 90, crecieron las tensiones en la relación entre IBM y Microsoft. Cooperaban entre sí en el desarrollo de sus sistemas operativos para PC y cada uno tenía acceso al código del otro. Microsoft quería desarrollar Windows aún más, mientras IBM deseaba que el futuro trabajo estuviera basado en OS/2. En un intento de resolver estas diferencias, IBM y Microsoft acordaron que IBM desarrollaría OS/2 2.0 para reemplazar a OS/2 1 y Windows 3.0, mientras Microsoft desarrollaría un nuevo sistema operativo, OS/2 3.0.

Este acuerdo pronto fue dejado de lado y la relación entre IBM y Microsoft terminó. IBM continuó desarrollando IBM OS/2 2.0 mientras que Microsoft cambió el nombre de su (todavía no publicado) OS/2 3.0 a Windows NT.

(Microsoft promocionó Windows NT con tanto éxito que la mayoría de la gente no se dio cuenta de que se trataba de un OS/2 mejorado.) Ambos retuvieron los derechos para usar la tecnología de OS/2 y Windows desarrollada hasta la fecha de terminación del acuerdo.

1.2.5 OS/2 2.0

IBM publicó OS/2 versión 2.0 en los noventas. Esta versión suponía un gran avance frente a OS/2 1.3. Incorporaba un nuevo sistema de ventanas orientado a objetos llamado Workplace Shell como sustituto del Presentation Manager, un nuevo sistema de ficheros, HPFS, para reemplazar al sistema de ficheros FAT de DOS usado también en Windows y aprovechaba todas las ventajas de las capacidades de 32 bit del procesador Intel 80386. También podía ejecutar programas DOS y Windows, ya que IBM había retenido los derechos para usar el código de DOS y Windows como resultado de la ruptura.

1.2.6 Windows 3.1 y Windows 3.11

En respuesta a la inminente aparición de OS/2 2.0, Microsoft desarrolló Windows 3.1. Más tarde Microsoft publicó también Windows 3.11 (denominado Windows para trabajo en grupo), que incluía controladores y protocolos mejorados para las comunicaciones en red y soporte para redes punto a punto.

Las siguientes fueron las principales características de Windows 3.1:

- Fuentes TrueType.
- Capacidad para que una aplicación reinicie la máquina.
- Soporte de API de multimedia y red.

1.2.7 Windows NT

Mientras tanto, Microsoft continuó desarrollando Windows NT.

Siendo un sistema operativo completamente nuevo, Windows NT sufrió problemas de compatibilidad con el hardware y el software existentes. También necesitaba gran cantidad de recursos y éstos estaban solamente disponibles en equipos grandes y caros. Debido a esto muchos usuarios no pudieron pasarse a Windows NT.

La interfaz gráfica de NT todavía estaba basada en la de Windows 3.1 que era inferior a la Workplace Shell de OS/2.

1.2.8 Windows 95

En respuesta a ello Microsoft comenzó a desarrollar un sucesor para Windows 3.1 cuyo nombre clave era Chicago.

Chicago iba encaminado a incorporar una nueva interfaz gráfica que compitiera con la de OS/2. También se pretendía introducir arquitectura de 32 bits y dar soporte a multitarea preventiva, como OS/2 o el mismo Windows NT. Sin embargo, sólo una parte de Chicago comenzó a utilizar arquitectura de 32 bits, la mayor parte siguió usando una arquitectura de 16 bits, Microsoft argumentaba que una conversión completa retrasaría demasiado la publicación de Chicago y sería demasiado costosa.

Microsoft desarrolló una nueva API para remplazar la API de Windows de 16 bits. Esta API fue denominada Win32, desde entonces Microsoft denominó a la antigua API de 16 bits como Win16. Esta API fue desarrollada en tres versiones: una para Windows NT, una para Chicago y otra llamada Win32s, que era un subconjunto de Win32 que podía ser utilizado en sistemas con Windows 3.1. De este modo Microsoft intentó asegurar algún grado de compatibilidad entre Chicago y Windows NT, aunque los dos sistemas tenían arquitecturas radicalmente diferentes.

Lanzado el 24 de agosto de 1995. En contraste con las anteriores versiones de Windows, Win95 es un sistema operativo más que una interfaz gráfica de usuario que corre sobre DOS.

Provee soporte para aplicaciones de 32 bits, multitarea con desalojo, soporte de red incorporado (TCP/IP,IPX, SLIP, PPP, y Windows Sockets). Incluye MS-DOS 7.0 como una aplicación. La interfaz gráfica, aunque similar a las previas versiones, fue significativamente mejorada.

1.2.9 OS/2 3.0 y 4.0

IBM continuó vendiendo OS/2, produciendo versiones posteriores como OS/2 3.0 y 4.0 (también llamado Warp). Pero con la llegada de Windows 95 OS/2 comenzó a perder cuota de mercado.

Aunque OS/2 seguía corriendo aplicaciones de Windows 3.0, carecía de soporte para las nuevas aplicaciones que requerían Windows 95. Al contrario que con Windows 3.0, IBM no tenía acceso al código fuente de Windows 95 y tampoco tenía el tiempo ni los recursos necesarios para emular el trabajo de los programadores de Microsoft con Windows 95, de esta manera perdió terreno, mercado y fuerza.

1.2.10 Windows NT 4.0

Después de la aparición de Windows 95, Windows NT continuaba usando la interfaz de Windows 3.1. Entonces Microsoft publicó Windows NT 4.0 que tenía la nueva interfaz de Windows 95 pero sobre Windows NT.

1.2.11 Windows 98

El 25 de junio de 1998 llegó Windows 98, que era una revisión menor de Windows 95. Incluía nuevos controladores de hardware y el sistema de ficheros FAT 32 que soportaba particiones mayores a los 2 GB permitidos por Windows 95.

Windows 98, el siguiente escalón en la familia de sistemas operativos Windows de escritorio. De cierta forma es la continuación que se podía esperar de Windows 95.

Como era obvio predecir, esta nueva versión continúa soportando 32 bits en su total dimensión aunque todavía se debe esperar para que se incorpore toda la funcionalidad de seguridad presente en los 32 bits y que hoy es una característica de la familia NT.

Desde el punto de vista del usuario común, Windows 98 no trae nada nuevo. Microsoft no ha hecho cambios relativamente importantes en la interfaz, por lo que, si un usuario sabe usar Windows 95, también sabe usar Windows 98. Se puede decir que la interfaz de Windows 98 es la interfaz que deja Internet Explorer 4.0 cuando se le instala en Windows 95 con la opción "Actualización de Escritorio", que es una versión mejorada de la interfaz nativa de Windows 95.

Así como para un usuario común, Windows 98 será familiar, para un programador Windows también, hasta que abra el velo que cubre a Windows 98 y descubra lo que hay en esta nueva versión de Windows:

- Modelo de Driver Win32 (Win32 Driver Model).
- Soporte para Múltiples Monitores.
- Tecnología de administración de poder OnNow.
- Soporte para USB.

1.2.12 Windows 98 Second Edition (Segunda Edición)

En 1999 Microsoft sacó al mercado Windows 98 Second Edition, cuya característica más notable era la capacidad de compartir entre varios equipos una conexión a Internet a través de una sola línea telefónica.

1.2.13 Windows Millennium Edition (Edición del milenio)

En el 2000 Microsoft introdujo Windows ME que era una copia de Windows 98 con más aplicaciones añadidas. Windows ME fue un proyecto rápido de un año para rellenar el hueco entre Windows 98 y el nuevo Windows XP.

1.2.14 Windows 2000 (NT 5.0)

En este mismo año vio la luz Windows 2000, una nueva versión de Windows NT muy útil para los administradores de redes y con una gran cantidad de servicios de red.

Hasta la versión 4.0 Windows NT se comercializaba en tres versiones: Workstation, Server, y Advanced Server. Desde Windows 2000, también se pierde la nomenclatura Workstation y Server, siendo la siguiente:

- Windows 2000 Professional anteriormente NT Workstation.
- Windows 2000 Server anteriormente NT Server.
- Windows 2000 Advanced Server anteriormente NT Advanced Server.
- Windows 2000 Datacenter Server. Producto nuevo y que es el nuevo y más poderoso sistema operativo de Microsoft con posibilidad de hasta 16 procesadores simétricos y 64 GB de memoria física.

1.2.15 Windows CE

Microsoft Windows CE es una plataforma de sistema operativo para un amplio rango de dispositivos computacionales móviles.

La plataforma Windows CE hará posible que nuevas categorías de dispositivos que no sean PC's puedan comunicarse unos con otros, compartir información almacenada en PC's basados en Windows, y conectarse a Internet.

Windows CE es un sistema operativo nuevo, compacto y portable, construido desde las bases para posibilitar el desarrollo de un gran número de dispositivos comerciales y hogareños, incluyendo PC's de Bolsillo (Handheld PC), "wallet PC", dispositivos inalámbricos tales como teléfonos celulares inteligentes, y la próxima generación de consolas de video juego incluyendo reproductores de DVD.

El sistema operativo Windows CE es un sistema de 32 bits, multitarea que tiene una arquitectura abierta, otorgando un soporte a una variedad de dispositivos.

Windows CE hace posible que se generen nuevas categorías de productos que pueden "hablar" unos con otros, compartir e intercambiar información con PC's basados en Windows, y comunicarse con una amplia variedad de sistemas empresariales o con Internet para el acceso al correo electrónico y a la World Wide Web o tripe WWW.

1.2.16 Windows XP y XP Profesional

La unión de Windows NT/2000 y Windows 3.1/95/98/SE se alcanzó con Windows XP liberado en 2001 en su versión Home y Professional. Windows XP usa el kernel o núcleo de Windows NT.

Este ha cambiado su forma visual aunque sigue siendo el clásico Windows desde su versión del 95, es más accesible al usuario y "bonito", más fácil de manejar e intuitivo, con más características de compatibilidad entre los dispositivos nuevos de hardware y con antiguas aplicaciones para Windows, mejor manejo de la memoria y procesamiento, pues ofrece las características de potencia y confiabilidad del Windows 2000 y la familia NT y la facilidad de uso del Windows 9.X

En un futuro no muy lejano, a mediados del 2003 estará ya en el mercado el nuevo Windows .NET, que es para servidores, según Microsoft éste tendrá extraordinarias mejoras, en cuanto a su facilidad de uso, confiabilidad, escalabilidad, entre muchas otras, con toda la experiencia de sus antecesores, Windows NT y 2000, tendrá también sus diferentes versiones, Standar, Server, Advanced, por mencionar algunas, este trabajara con 64 bits, no con los 32 bits que se trabaja en la actualidad con sistemas operativos de Microsoft como Windows; de esta manera, para cuando esta tesis este completa, ya estará disponible el nuevo Windows .NET con muchas mejoras y nuevas APIS.

1.3 Algunos aspectos de Windows

Windows depende de los drivers, algunos de DOS y de la BIOS para soportar discos flexibles, discos duros y CD-ROM.

La función primaria de Windows es proveer la interfaz WIMPS (Windows, Icons, Mouse, Pointer, Scroll-Bars). Para llevar a cabo esto, la mayoría del código de Windows consiste en rutinas de manejo de botones, cajas, menús, listas desplegables, y otros componentes de las cajas de diálogo.

Para el programa de aplicación, Windows es una gran colección de subrutinas para dibujar cajas y texto en la pantalla o en una página de una impresora. Existen también colecciones más pequeñas que cargan programas, asignan almacenamiento, administran cronómetros (temporizadores o timers) y el puerto COM.

Las colecciones de rutinas útiles que realizan funciones comunes pueden ser reunidas en Bibliotecas de Enlace Dinámico o Dynamic Link Libraries (DLL).

Una DLL es simplemente una forma de módulo EXE sin un punto de entrada principal. Contiene partes de programas, pero no es por sí misma una aplicación. Las rutinas que están en una DLL son "exportadas" por sus nombres. Estas rutinas son agregadas a una tabla que permite a los otros programas llamar a la rutina y requerir su servicio.

Las herramientas de programación construyen dos tipos de módulos. Un programa o utilidad normal es un archivo EXE. Una biblioteca es un archivo DLL.

- EXE: Windows llama a sus tres módulos DLL más importantes (USER, GDI, y KRNL386) con la extensión EXE, que es normalmente utilizada por los programas. Ellas no son programas, son sólo DLL con diferente nombre.
- DRV: Durante la instalación de Windows, el usuario selecciona un tipo particular de teclado, mouse, monitor, e impresora. Las DLL que contienen el soporte para un tipo particular de dispositivo general usualmente tienen la extensión DRV. A este tipo de DLL se le conoce como driver o controlador.
- VBX: Visual Basic Extension. Son controles personalizados de Visual Basic. Hoy en desuso.
- OCX: Son el reemplazo de VBX, aunque su función es la misma que la anterior. Hoy en día se les llama Controles ActiveX.

La última categoría de módulos retiene el tipo de archivo DLL. Ellos residen usualmente en el directorio \WINDOWS\SYSTEM, aunque pueden estar en cualquier lugar de la dirección. Proveen soporte para interfaces adicionales de programas. NETAPI provee rutinas asociadas con los servidores de red. OLE provee las convenciones de Microsoft para que los programas de aplicación cooperen editando documentos compuestos. ODBC es un soporte genérico para acceder a varias bases de datos. WINSOCK provee el acceso a Internet.

Una aplicación llama a una rutina de una DLL para solicitar un servicio de Windows. Todas las aplicaciones Windows pueden también exportar una de sus propias subrutinas para administrar eventos. Cualquier programa Windows es manejado por eventos. El evento puede ser el clic que el usuario hace con el mouse, o el presionar una tecla en el teclado, o el fin de algún período de tiempo.

Cada programa Windows provee una rutina administradora de eventos para recibir y manejar tales eventos. Windows llama a la rutina de Administración de Evento de la misma forma en que una aplicación llama a una DLL de Windows para solicitar un servicio.

Aunque Windows puede ejecutar varias aplicaciones a la vez, cada programa tiene una opción de ejecutarse cuando su rutina de manejo de eventos es llamada, y se detiene cuando retorna al sistema de Windows después de manejar el evento. Si esto toma mucho tiempo, el usuario ve un puntero con un reloj de arena como una indicación que la aplicación está ocupada. Un programa no puede realizar una operación larga y complicada sin apoderarse de la CPU y de la interfaz de usuario.

Estas son características de Windows, con las que los programas sobre este sistema operativo se desenvuelven y trabajan; por lo tanto, es necesario conocer estos términos, diferenciar los archivos, saber para qué sirven y de este modo aplicarlos a nuestros programas para obtener el máximo desempeño de nuestra aplicación sobre el sistema operativo tan famoso hasta el momento.

1.4 ¿Qué es un API?

"API" por sus siglas en ingles Aplication Program Interface. En español quiere decir Interfase de Aplicación del Programa.

Una API es un método específico prescrito por un sistema operativo de computadora o por otro programa de aplicación por medio del cual un programador que escribe un programa de aplicación puede hacer solicitudes del sistema operativo o de otra aplicación.

Haciendo un poco de historia para conocer el nacimiento de las APIS en Windows por Microsoft, se puede recordar que desde mediados de 1994 hasta finales de 1995 (cuando Microsoft saca el Windows 95) el OS/2 de IBM, pese a lo que se pudiera suponer, no experimenta un gran crecimiento en su nuevo sistema operativo. Sin embargo, Microsoft iba a aprovechar ese tiempo muy bien. No fue un retraso debido a mejoras del sistema operativo, ya que cuando Windows 95 salió seguía adoleciendo de la inestabilidad de su antecesor y era más lento que OS/2.

Ese tiempo lo dedicó al esfuerzo más grande que ha existido por desmarcarse del resto de la competencia. Ese año se dedicó a crear las APIS más difíciles del mercado (en pocas palabras, son las herramientas que utilizan los desarrolladores de software para un sistema operativo), y Microsoft daría esa documentación a los desarrolladores siempre y cuando no desarrollasen software en otro sistema operativo, he aquí la astucia y eficacia de Microsoft.

He aquí la jugada ganadora de Microsoft. El por qué de su éxito, el por qué la gente lo compró de forma masiva, ya que la mayoría de las compañías que creaban software comercial se encapsularon y desarrollaron para sistemas operativos de Microsoft.

FUNDAMENTOS

CAPITULO II

2.1 ¿A quién está dirigido?

Muy bien, antes de continuar me gustaría comentar para qué lectores está dirigida, principalmente, esta tesis. Como ya he mencionado antes, es principalmente para los desarrolladores de software en sistemas operativos Windows, no importa qué tipo de programa desarrollen, desde sistemas con bases de datos, programas gráficos o muchos otros.

Deben de tener conocimientos de programación estructurada (C) y orientada a objetos (Visual Basic 6), además de lenguajes de alto nivel nuevos, como lo es Visual Basic .NET, principalmente durante el transcurso de toda la tesis. No me inclinaré sobre algún lenguaje en específico, sino que se estarán dando ejemplos prácticos y variados, se dejarán de lado un poco las APIs ya viejas como para Windows 3.11 y se tomarán las de 32 bits o de Windows 98 en adelante y más recientes como para sistemas el nuevo sistema operativo Windows XP, ya que se provee que éste será un gran sistema para los próximos meses y años.

Actualmente, ahora que me encuentro desarrollando la tesis, leí un artículo relacionado con la PC de ensueño o más poderosa de la actualidad (diciembre del 2002), y éste hacia una comparación de las mejores piezas internas (Hardware) de la máquina y su precio, como lo son las tarjetas aceleradoras de gráficos, memorias, procesadores, audio, etc., y al final del artículo mencionaban que una vez ya teniendo tu PC ideal, ¿qué sistema operativo elegir y que éste fuera el más conveniente para cargarle? Pues bien, la respuesta es fácil, Windows XP Profesional, por su facilidad de uso, estabilidad, y muchas otras cosas más. Ni siquiera tomaron en cuenta al "fabuloso Linux" que es gratuito. Pues bien, el artículo lo leí de una revista de editorial Intelligent S.A. de C.V. llamada *PC Intelligent* (número 5), la cual está tomando mucha fuerza, pues es la primera en México en ofrecer al lector un DVD como obsequio al comprar la revista y por sus acertadas columnas y buenos artículos en cuanto a computación se refiere y tecnología. Bien, regresando pues ¿a qué voy con todo esto? Pues fácil, es un indicador para todos aquellos programadores que creen que Windows pierda fuerza, pues no lo creo, pronto saldrá el nuevo Windows Net, que vendrá a sustituir a su serie de servidores, actualmente el 2000 Server y familia que fueron los sucesores del NT; en fin, por si creen que Microsoft dejara perder terreno en sistemas operativos comerciales, para el hogar mejor dicho, pues no parece ser de esta manera. Entonces, ¿qué hacer? Pues estudiar bastante y aprender las buenas APIs, ya que éstas son una herramienta primordial si se quiere obtener una buena y robusta aplicación bajo sistemas operativos Windows.
Esto no quiere decir que todas las APIs sean iguales, quede claro que una misma puede variar de un sistema operativo Windows a otro, ya sea para mejorar o morir y dar paso a una con mayores características; pero no hay de qué preocuparse, pues aquí obtendrás el conocimiento básico y conocerás a las principales y mejores en la actualidad para realizar diferentes tareas. Entonces, resumiendo, pues yo soy un estudiante (aún) de Ingeniería en Computación, me gusta mucho la programación, y es tan chistoso esto que yo desde que comencé a estudiar programación en el año de 1997 en la preparatoria, me encanto, comencé con Turbo Basic y nunca conocí las APIs de Windows, hasta hace casi un año (principios del 2002), fuera de la escuela por medio de un pequeño desarrollo de una aplicación que necesité del sistema operativo y que las descubrí, y me encantaron por la facilidad y gran apoyo que le dan al programador, y he decidido hacer esta tesis, entonces... resumiendo, esta tesis yo la encuentro perfecta para aquellos estudiantes principiantes y mediados de programación bajo Windows, pues un experto programador, supongo que ya las maneja y tal vez utiliza varias específicas para un trabajo en concreto.

De esta manera, después de hablar del por qué de esta tesis y de mí un poco tratando de dejar más en claro este trabajo, espero sea útil para aquellos que las necesiten y para aquellos que no las conozcan, y hasta por qué no, servir de una guía de referencia para alguna en específico. Bueno por ahora escribo muy metafóricamente, pero conforme se avance ya lo estarán descubriendo, pues entrando en materia y sin tantos rodeos, el primer capítulo lo he hecho porque lo vi necesario para adentrar y especificar el tema. Se necesita que ya al menos el lector esté programando en un lenguaje para Windows, por ejemplo Visual Basic, y que maneje al menos los entornos gráficos y comprenda lo que son las funciones clases, módulos, eventos, etc. Pues bien, pasemos con las herramientas a utilizar y a dar lo mejor para su programa en desarrollo.

2.2 Herramientas a Utilizar

Se necesita obviamente un sistema operativo Windows, recomiendo entre Windows 98 en adelante y Windows 2000 en adelante, los anteriores ya son prácticamente obsoletos y pocos lugares los siguen utilizando, si es que no quedaron ya como programas de recuerdo, comparándolos casi como los juegos de maquinitas arcadia de Atari, con juegos de Pacman, Galagan, etc. Una PC que tenga memoria suficiente para el sistema operativo, procesador y disco duro, esto más que nada para que no tarden en estar compilando tanto tiempo y no caigan en la desesperación (lo escribo por experiencia propia); en fin, lenguajes de programación, claro, yo recomiendo compiladores de mismo Microsoft como su Visual Studio, que ya tiene Visual Basic, C++, aplicaciones como el Visor de APIs, entre otros, si optan por compiladores como C de Borland, no lo recomendaría mucho tan solo por la compatibilidad al 100% solamente, si ya trabajaran con Visual Studio .Net, entonces necesitarán Windows 2000 o Xp, bien, son sugerencias, ustedes deciden, pero aquí se tomará un rango de abanico de todo, principalmente básico y de más utilización, se tomará mucho en cuenta el registro de Windows, y aplicaciones extras o de ayuda complementaria, cuando sean necesarias se verán, se darán las direcciones de Internet, bibliografías y referencias para una mayor ahondamiento del lector sobre algún tema en específico.

2.3 Visor de texto API

Este visor en un pequeño programita que viene con el Visual Studio 6 Edición Empresarial de Microsoft, se llama Visor de Texto API, este se puede agregar al Visual Basic, de la siguiente manera:

1. Abrir un proyecto estándar
2. Menú complementos
3. Opción Administrador de complementos
4. Seleccionar Visor de texto API

Es una utilidad para ver constantes, declaraciones y tipos de la API Win32, de una manera sencilla, te permite buscarlos por índice u orden alfabético, es rápida y efectiva.

Menú desde Visual Basic ya agregado con los pasos anteriores

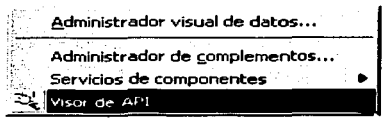

Figura 1.1 Ventana del menú en Visual Basic 6 para visualizar el visor de API[1]

El Visor de API le permite explorar las declaraciones, constantes y tipos incluidos en un archivo de API de texto. Es posible copiar elementos al Portapapeles y, por tanto, copiarlos al código de Visual Basic.

El Cargador de API incluye el archivo de texto WIN32API.TXT para las interfaces Win32.

La lista desplegable que aparece en la parte superior del formulario le permite ver constantes, declaraciones o tipos.

En escribir las palabras a buscar es de gran ayuda, ya que conforme se va escribiendo éste va buscando de manera automática la sentencia más parecida.

Si selecciona una entrada y hace clic en el botón Agregar, podrá agregar un elemento a la lista de elementos seleccionados en la parte inferior del formulario.

Para copiar al Portapapeles elementos de la lista de elementos seleccionados, elija Copiar. Se copiarán todos los elementos de la lista. Después puede incluirlos en módulos si elige Pegar en el menú Edición del módulo. Para quitar una entrada de la lista de elementos seleccionados, haga clic en el botón Quitar.

En la figura 1.2 se ve claramente un ejemplo de lo descrito, con la función ExitWindowsEx, que se verá más adelante. Desde aquí esta función se pegará en cualquier módulo del programa y se mandará llamar como pública para el uso de ella en cualquier parte del programa y de este modo comenzar a utilizar las APIs, (la seleccionada en particular cierra Windows).

[1] Imagen diseñada por el autor de esta obra, las siguientes imágenes a lo largo de toda la tesis son tomadas y creadas por el autor de la obra.

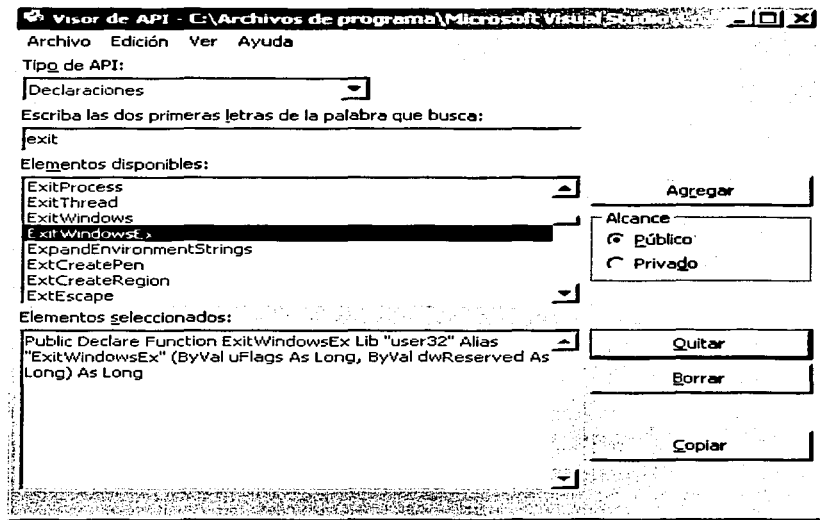

Figura 1.2 Ventana del programa de Visor de API

2.4 Últimos conceptos

Argumento
Un valor sobre el que actúa una función o un procedimiento. Por ejemplo, en la instrucción de Space(10), el número 10 es el argumento.

Expresión
Es cualquier combinación de variables, operadores, constantes, funciones y nombres de campos, controles y propiedades que se evalúa como un único valor. Puede usar expresiones como configuración para numerosas propiedades y para establecer o definir campos calculados en consultas.

Variable
Un lugar de almacenamiento con nombre que puede contener datos que se pueden modificar durante la ejecución del programa. Cada variable tiene un nombre único que la identifica dentro de su nivel de alcance.

Los nombres de las variables para la mayor parte de los lenguajes:

- Deben comenzar con un carácter alfabético.
- No pueden contener un punto o carácter de declaración de tipo incrustados.
- Deben ser únicos dentro del mismo alcance.
- No deben tener una longitud de más de 255 caracteres.

Null
Un valor que indica que una variable no contiene datos válidos o se podría decir que esta vacía. Null es el resultado de:

- Una asignación explícita de Null a una variable.
- Cualquier operación entre expresiones que contienen Null.

Nothing
Un valor especial que indica que una variable de objeto ya no está asociada a ningún objeto real.

Caracteres ASCII
Juego de caracteres de 7 bits denominado ASCII (Código Americano Estándar para Intercambio de Información), ampliamente utilizado para representar letras y símbolos de en un teclado estándar de EE.UU. El juego de caracteres ASCII es igual que los primeros 128 caracteres (0 – 127) del juego de caracteres ANSI.

Parámetro
Es un valor que se transfiere a una función o a un procedimiento. Este valor puede ser transmitido a la función o procedimiento también por valor o por referencia.

Por valor
Modo de pasar un argumento a un procedimiento pasando su valor en vez de su dirección. Esto hace posible el acceso del procedimiento a una copia de la variable. Como resultado, el valor real de la variable no puede ser cambiado por el procedimiento al cual se pasa.

Por referencia
Modo de pasar un argumento a un procedimiento pasando su dirección en vez de su valor. Esto hace posible el acceso del procedimiento a la variable real. Como resultado, el valor real de la variable puede ser cambiado por el procedimiento al cual se pasa.

Función definida por el usuario
Código creado por un usuario que devuelve un valor.

Objeto
Una instancia de una clase que combina datos y procedimientos. Por ejemplo, un control de un formulario en ejecución es un objeto.

Clase
Es la definición formal de un objeto. La clase actúa como una plantilla desde la que se crean las instancias de un objeto en tiempo de ejecución. La clase define las propiedades de un objeto y los métodos utilizados para controlar el comportamiento del objeto.

Propiedad
Un atributo con nombre de un objeto. Las propiedades definen características de un objeto como tamaño, color y ubicación en la pantalla, o el estado de un objeto como habilitado o deshabilitado.

Estos conceptos son básicos para el uso de las APIs, que trabajaremos mas adelante, estos conceptos aplican también a la mayor parte de lenguajes de alto nivel actualmente, manejo de objetos y entornos visuales.

Todas las funciones de API se contienen en dlls del sistema operativo. En los lenguajes de programación como Visual Basic, se puede acceder a estas funciones a través de las llamadas al dll en el que ellos residen. Ya vimos los conceptos que necesitamos, los más importantes y útiles para nosotros, como dll, programas ejecutables *.exe, etc. De aquí en adelante serán muy mencionados aplicados, por eso se necesita tener claramente los conceptos de los que hablaremos para evitar confusiones futuras.

Una función de API es casi exactamente igual que una función Básica Visual. Sin embargo, en lugar de tener el código entero de la función en el programa, solo se necesita incluir una referencia a la función y el dll que los contiene. Todas las funciones deben estar declaradas en un módulo para ser accedido en cualquier parte en el programa.

Bien, una vez ya teniendo todas las herramientas comencemos haciendo un pequeño ejemplo de una llamada a una API para explicar sus partes paso a paso, la manera en que se inserta en el programa, éste es utilizado y sus requisitos, esta función es sencilla, no necesita de muchos parámetros, sin embargo tiene los suficientes para demostrar bien el ejemplo, este es en Visual Basic, (la mayoría de los ejemplos son para este lenguaje, debido a su eficacia, compatibilidad y uso en el mercado).

Esta es la función:

```
Declare Function Beep Lib "kernel32" Alias "Beep" (ByVal dwFreq As Long,
ByVal dwDuration As Long) As Long
```

Toda la función es una sola línea, todas las llamadas de la función contienen varias partes, que a continuación se aclaran.

La primera parte "Declare Function" simplemente le dice a Visual Basic que se está declarando una función que será usada en alguna otra parte en el programa, es conveniente declararla dentro de un modulo. Esto puede prologarse por "Public" si se quiere a la declaración que esté disponible en cualquier parte del programa, o "Private" si lo que se quiere es que esté simplemente sea parte del módulo.

La segunda parte y las restantes se declaran juntas. "Beep" es el nombre de la función que se llamara en el programa. "Alias "Beep"" es el nombre de la función que se encuentra en el dll. Hay que tener cuidado aquí pues es conveniente que no se renombren las funciones para que dentro del proyecto y del dll sean llamadas igual.

La tercera parte de la función, "Lib "kernel32"", simplemente le dice a Visual Basic dónde encontrar la función. En este caso, está en kernel32.dll. Si el dll es uno del dlls del sistema (kernel32, gdi32), entonces todo lo que se necesita es el nombre del dll, sin su extensión (ya que se pueden mandar dlls creados externamente y no pertenecientes a Windows). Si este es el caso entonces habrá que colocar al dll en la carpeta raiz del Sistema como (c:\windows o c:\windows\system) y así poder mandarlo llamar de lo contrario se tendrá que dar la dirección comporta de la localización del archivo dll.

La próxima parte y ultima de la función es la lista de los parámetros. Ésta simplemente es una lista de valores que la función espera para hacer su trabajo. Como con la propia función, la lista de los parámetros contiene varias partes, y puede confundirse un poco si no se trata con cuidado.

La primera parte de la lista de los parámetros es el keyword (llave) "ByVal". Esto le dice al sistema cómo pasar el parámetro. Hay dos posibilidades para este keyword, ByVal y ByRef. ByVal dice el sistema que se está pasando el valor real de la variable. ByRef dice el sistema que se está pasando una referencia a esa variable.

En concreto, si se pasa ByRef, se está pasando la dirección en memoria de ese valor.

Hay que tener mucho cuidado en esta parte, ya que si se manda valores equívocos en la variable a pasar dentro de la función, marcará un error de desbordamiento o mostrará resultados inesperados, comportándose de una manera no prevista.

La segunda parte de la lista de los parámetros es el propio parámetro. En este caso, se tienen dos: dwFreq y dwDuration, con su tipo de variable correspondiente esperada en la dll, en este caso (Long, entero largo). Estos nombres son realmente sólo variables, y son para identificar los valores a trabajar con la función, en este ejemplo son la frecuencia y la duración del sonido beep o pitido.

La parte final de la lista de los parámetros es el tipo del parámetro en sí. La última parte de la función es el tipo de valor de retorno. Esto dice qué tipo de valor la función devolverá. En la mayoría de los casos será un entero largo, y es usado para indicar qué errores habría, si se encontraran en la ejecución de la función.

2.5 Registro

El registro es un programa que ha sido instalado desde el Windows 95 principalmente y en sus sucesores, familia NT y para el hogar Windows 9x hasta la fecha (Windows XP y 2000).
En éste se colocan todas las entradas de los programas y el sistema operativo Windows, aquí se registran los programas comprados, las claves de los usuarios y software con licencia; la mayor parte del registro es para llevar un historial fiable, aquí es donde se encuentran las cadenas de las direcciones de los programas, sus claves, el inicio y carga de Windows, registros, etc. Muchas cosas, éste tiene toda una gama de Apis para la abertura, lectura, grabado y eliminado de las claves del registro, sus archivos de tipo binario o de cadena; en fin, se manejarán muchos ejemplos con el registro, como por qué es que los programas tienen memoria o se acuerdan de cómo fue la ultima configuración que se dejó, al iniciar Windows qué programas se cargan, cuáles están comprados y cuáles no, etc. Muchas cosas.

Trabajar con el registro, puede ser una manera buena de guardar información para el programa, en vez de escribir archivos de tipo .ini o de respaldo de texto, fáciles a los usuarios para entrar y modificar las características del mismo. Ahora bien, trabajar con el registro, si no tiene mucho cuidado sobre lo que se esté haciendo, puede desequilibrar su ambiente de Windows completamente e incluso puede llevar a una reinstalación. Bueno, no es bueno asustarse, mejor mucha atención y "manos a la obra" con el registro.

El programa del registro se llama "regedit.exe" Editor de Registro, éste se encuentra en la carpeta principal de Windows; no importa en qué unidad de disco duro éste se encuentre instalado, para abrirlo:

1. Inicio
2. Ejecutar
3. Escribir regedit

Figura de los pasos descritos de cómo acceder a programa Editor de Registro

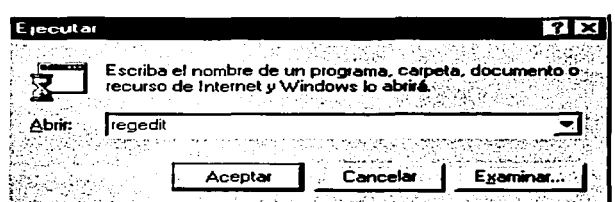

Figura 1.3 Ventana para ejecutar directamente un archivo

Una vez ya abierto es necesario comprender cómo está compuesto; es un tipo Explorador de Windows, del lado izquierdo se encuentra un árbol con la raíz en comenzando en Mi PC y las claves principales, éstas son seis, y del lado derecho la especificación de cada una, su nombre y características.

Llaves principales:

1. HKEY_CLASSES_ROOT
2. HKEY_CURRENT_USER
3. HKEY_LOCAL_MACHINE
4. HKEY_USERS
5. HKEY_CURRENT_CONFIG

En cada una de estas llaves se encuentra alojada la información, ya sea de los programas instalados, las características del sistema operativo, computadora, configuración regional, idioma, hora, etc.

Tipos de valor:
1. Cadena
2. Binario
3. Palabras

Estos valores son los principales, pero hay varios mas, también constantes, y Apis específicas, para leer, borrar, modificar, etc. hacer cambios directamente en el registro del sistema, todos éstos los colocaré en temas concretos posteriores para ahondar y especificar mejor, pero es importante que se tenga el conocimiento previo del registro del sistema.

Figura del programa Editor de Registro de Windows para un mejor entendimiento de lo realizado

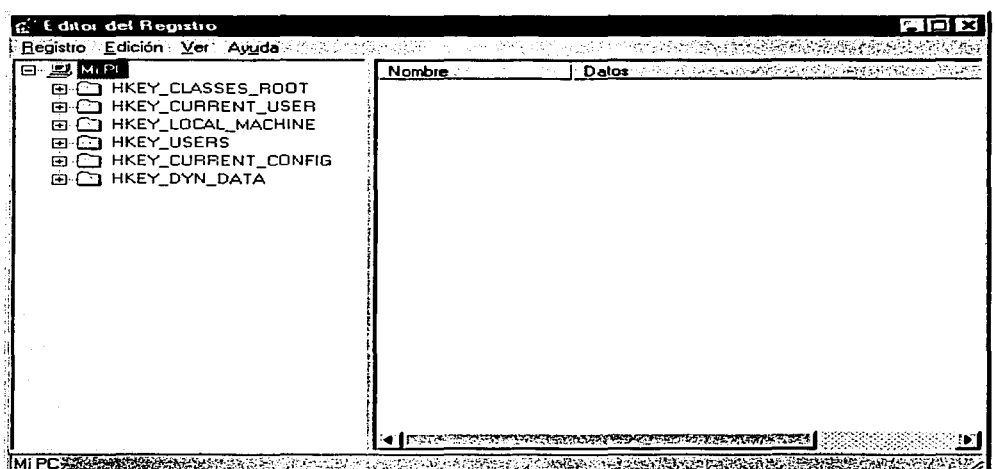

Figura 1.4 Ventana del Editor de registro

FUNCIONES BÁSICAS DE WINDOWS APIS

CAPITULO III

3.1 Información de Windows

La información mostrada por el Sistema Operativo Windows es importante por obvias razones, la de mostrar las características en la cual está instalado Windows, las siguientes APIS son una recopilación de las más importantes, para un programador, ya que con estas enriquece su programa final de una manera profesional.

No muestro en esta sección todas las APIS existentes, ya que no terminaría de mostrarlas todas, ya que estas se dividen en tipos, clases y constantemente salen nuevas con versiones más actuales del Sistema Operativo y mejoras del mismo, al igual que los lenguajes de programación, pues ya que se trabajan de modo diferente para cada uno, es decir la manera en que las mando llamar desde un Java no es lo mismo que Visual Basic.

En esta tesis, muestro las más importantes desde el punto de vista como estudiante de Ingeniería en computación, para algunos programadores, les será insuficiente, pero de gran ayuda, ya que trato de explicar de una manera sencilla y de acomodarlas de acuerdo a sus características y cualidades, mostrando ejemplos prácticos y útiles, cosa que yo no encontré hasta ahora del todo bien, esta información es relativamente cerrada y desde mi punto de vista un poco celada ya que yo nunca las conocí hasta no necesitar de ellas, no es lo mismo que aprender programación, pero van ambas de la mano, una complementa a la otra, bueno sin más rollo, entremos de lleno a esta parte tan útil en la vida de un programador para Windows.

La manera en que se muestran las APIS, será muy estructurada y parecida, primeramente el nombre de la API, su declaración, su información correspondiente, el para qué sirve, el Sistema Operativo compatible, sus parámetros y requisitos, algunas de ellas necesitaran ciertas librerías y versiones mínimas de ciertos programas como lo es el Internet Explorer, y finalmente el código, los pasos a seguir, el código y explicación detallada del mismo, listo para ser utilizado y modificado con las APIS que trabajaremos.

3.1.1 GetUserName

```
Declare Function GetUserName Lib "advapi32.dll" Alias "GetUserNameA" (ByVal lpBuffer As String, nSize As Long) As Long
```

Información
Esta función regresa el nombre de usuario actual con el que se ha abierto sesión en Windows
Sistema Operativo
Windows NT 3.1, Windows 9x

Parámetros
lpBuffer. Apunta al buffer que recibe el nombre de usuario, si es nulo este falla o marca error
nSize. Especifica el tamaño máximo de caracteres que regresa lpBuffer

Código
Este código esta en Visual Basic Net

- *Crear un proyecto nuevo*
- *Seleccionar Visual Basic*
- *Aplicación de Windows*
- *Crear un botón simple en la forma y agregar el código correspondiente*

```
Declare Function GetUserName Lib "advapi32.dll" Alias _
      "GetUserNameA"  (ByVal  lpBuffer  As  String,  ByRef  nSize  As  Integer)  As
      Integer

Function GetUser()
        Dim RetVal As Integer
        Dim UserName As String
        Dim Buffer As String
        Buffer = New String(CChar(" "), 25)
        RetVal = GetUserName(Buffer, 25)
        UserName = Strings.Left(Buffer, InStr(Buffer, Chr(0)) - 1)
        MsgBox("Usuario: " & UserName)
End Function

Private  Sub  Button1_Click(ByVal  sender  As  System.Object,  ByVal  e  As
System.EventArgs) Handles Button1.Click
        GetUser()
End Sub
```

En este código se utiliza un procedimiento y un botón, este es código en Visual Basic .Net, declaro la función GetUserName, después GetUser, dentro de ésta creo 3 variables, un entero y 2 cadenas, a buffer le asigno una longitud de caracteres creándola nueva, a RetVal le asigno el valor regresado por GetUserName, nSize de tamaño 25, y la variable UserName el valor de la cadena, recortada y escribo el valor en un MsgBox.

3.1.2 GetUserNameEx

Declare Function GetUserNameEx Lib "secur32.dll" Alias "GetUserNameExA" (ByVal NameFormat As EXTENDED_NAME_FORMAT, ByVal lpNameBuffer As String, ByRef nSize As Long) As Long

Información
Esta función regresa el nombre de usuario actual con el que se ha abierto sesión en Windows

Sistema Operativo
Windows 2000, Windows XP

```
Private Enum EXTENDED_NAME_FORMAT
    NameUnknown = 0
    NameFullyQualifiedDN = 1
    NameSamCompatible = 2
    NameDisplay = 3
    NameUniqueId = 6
    NameCanonical = 7
    NameUserPrincipal = 8
```

```
    NameCanonicalEx = 9
    NameServicePrincipal = 10
End Enum
```

Es un tipo requerido para especificar el formato deseado, éste no puede ser desconocido, arriba están los diferentes tipos.

lpBuffer. Apunta al buffer que recibe el nombre de usuario, si es nulo éste falla
nSize. Especifica el tamaño máximo de caracteres que regresa lpBuffer

Código
Este código está en Visual Basic Net

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma y agregar el código correspondiente
- Agregar un módulo simple

Hay otra manera de reemplazo en Net, es el siguiente.

```
System.Security.Principal.WindowsIdentity.GetCurrent.Name
Imports System
Imports System.Security
Public Module modmain
   Sub Main()
      MsgBox("Usuario: " & Principal.WindowsIdentity.GetCurrent.Name)
   End Sub
End Module
```

El código es sencillo, colocado en un módulo, y se agrega un msgbox, se manda llamar la cadena de reemplazo y es todo.

3.1.3 GetSystemDirectory

```
Declare Function GetSystemDirectory Lib "kernel32" Alias "GetSystemDirectoryA" (ByVal lpBuffer As String, ByVal nSize As Long) As Long
```

Información
Esta función regresa la cadena de la dirección en donde Windows tiene su sistema actual

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows XP

Parámetros
lpBuffer. Apunta al buffer que recibe el nombre de usuario, si es nulo éste falla o marca error
nSize. Especifica el tamaño máximo de caracteres que regresa lpBuffer

Código
Este código está en Visual Basic Net

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma y agregar el código correspondiente
- Agregar un módulo simple

Hay otra manera de reemplazo en Net, es el siguiente.
System.Environment.SystemDirectory

```
Imports System
Imports System.Environment

Public Module modmain
    Sub Main()
        MsgBox("El directorio del sistema es: " + SystemDirectory)
    End Sub
End Module
```

Una vez más con el Net es más fácil de programar, es todo el código necesario para obtener e directorio del sistema, sin necesidad de declarar la función correspondiente, necesaria en Visual Basic 6 y anteriores.

3.1.4 GetComputerName

Declare Function GetComputerName Lib "kernel32" Alias "GetComputerNameA" (ByVal lpBuffer As String, nSize As Long) As Long

Información
Esta función regresa el nombre de la computadora

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
lpBuffer. Apunta al buffer que recibe el nombre de usuario, si es nulo este falla o marca error
nSize. Especifica el tamaño máximo de caracteres que regresa lpBuffer

Código
Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma y agregar el código correspondiente

```
' en la forma
Private Declare Function GetComputerName Lib "kernel32" Alias "GetComputerNameA"
(ByVal lpBuffer As String, nSize As Long) As Long

Private Sub Command1_Click()
Dim reg As Integer
Dim nSize As Long
Dim tmpVal As String

tmpVal = String$(1024, 0)
nSize = 1024

reg = GetComputerName(tmpVal, nSize)
MsgBox tmpVal
End Sub
```

Igual que en los casos anteriores, se declara el tamaño de nSize y tmpVal, para asignárselo de la función GetComputerName y a una caja de mensaje para mostrar el resultado.

3.1.5 SetComputerName

Declare Function SetComputerName Lib "kernel32" Alias "SetComputerNameA" (ByVal lpComputerName As String) As Long

Información
Esta función coloca el nombre a la computadora, y se verán los cambios hasta después de reiniciar el sistema.

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
lpComputerName. Nombre de la computadora, se deben colocar los caracteres estándares incluidos los siguientes: ! @ # $ % ^ & ') (. - _ { } ~ .

Código
Este código está en Visual Basic 6

- Crear un proyecto nuevo y Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma y agregar el código correspondiente

```
Private Declare Function SetComputerName Lib "kernel32" Alias "SetComputerNameA"
(ByVal lpComputerName As String) As Long

Private Sub Command1_Click()
Dim reg As Integer
Dim nombre As String

nombre = "rafi"
reg = SetComputerName(nombre)
End Sub
```

Este código es sencillo, basta con dar el valor de la variable tipo cadena que se requiere como nombre de la computadora, llamado nombre.

3.1.6 GetWindowsDirectory

Declare Function GetWindowsDirectory Lib "kernel32" Alias "GetWindowsDirectoryA" (ByVal lpBuffer As String, ByVal nSize As Long) As Long

Información

Esta función obtiene el directorio de Windows

Sistema Operativo

Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros

lpBuffer. Apunta al buffer que recibe el nombre de usuario, si es nulo éste falla o marca error

nSize. Especifica el tamaño máximo de caracteres que regresa lpBuffer

Código

Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma y agregar el código correspondiente

```
Private Declare Function GetWindowsDirectory Lib "kernel32" Alias
"GetWindowsDirectoryA" (ByVal lpBuffer As String, ByVal nSize As Long) As Long

Private Sub Command1_Click()
Dim reg As Integer
Dim nSize As Long
Dim tmpVal As String

tmpVal = String$(1024, 0)
nSize = 1024

reg = GetWindowsDirectory(tmpVal, nSize)
MsgBox tmpVal
End Sub
```

El código es sencillo para este caso, las variables declaradas también, no debe de haber mayor problema, es muy parecido al GetUserName y GetComputerName.

La primera figura de la próxima hoja muestra el programa en Visual Basic Net, las tres APIS mostradas anteriormente, la imagen de abajo el resultado del primer botón, Directorio del sistema

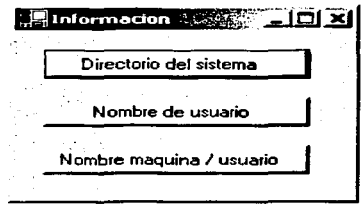

Figura 3.1 Ventana mostrando los botones del código del programa realizado anteriormente

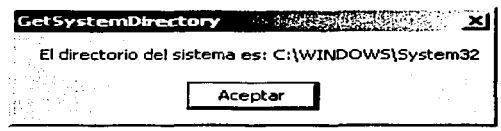

Figura 3.2 Ventana mostrando el resultado del evento del botón "Directorio del sistema" de la figura 3.1

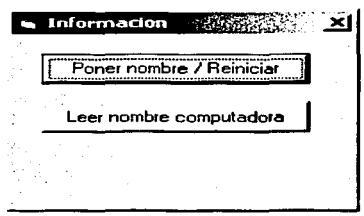

Figura 3.3 Ventana con los dos botones para obtener la información de la computadora

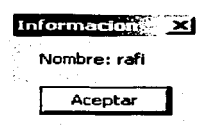

Figura 3.4 Ventana mostrando el resultado de la figura 3.3

3.1.7 ShellAbout

Declare Function ShellAbout Lib "shell32.dll" Alias "ShellAboutA" (ByVal hwnd As Long, ByVal szApp As String, ByVal szOtherStuff As String, ByVal hIcon As Long) As Long

Información

Esta función muestra la caja de diálogo acerca de... de Windows

Sistema Operativo

Windows NT 3.1, Windows 9x, windows XP

Parámetros

hWnd. Identifica la ventana padre, puede ser Null
szApp. Coloca el título de la ventana, tipo cadena
szOtherStuff. Apunta al texto que será mostrado
hIcon. Coloca el icono deseado, si es Null o nulo, se pondrá el de Windows

Código

Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "acerca_de" y agregar el código correspondiente

```
' en la forma
Private Declare Function ShellAbout Lib "shell32.dll" Alias "ShellAboutA" (ByVal
hWnd As Long, ByVal szApp As String, ByVal szOtherStuff As String, ByVal hIcon
As Long) As Long

Private Sub acerca_de_Click()
' ventana acerca de...
 ShellAbout ByVal 0&, " ... ", "Programa creado de muestra ", ByVal 0&
End Sub
```

Esta API puede ser acomodada pasando directamente sus características, como lo es el nombre del creador, ventana e icono, el código mostrado es sencillo, sólo se pasan los parámetros directamente y se declara la función en la forma.

3.1.8 GetCurrentDirectory

Declare Function GetCurrentDirectory Lib "kernel32" Alias "GetCurrentDirectory" (ByVal nBufferLength As Long, ByVal lpBuffer As String) As Long

Información

Esta función muestra la dirección actual de donde se encuentra el archivo programa ejecutable actual.

Sistema Operativo

Windows NT 3.1, Windows 9x, windows XP

Parámetros
nBufferLength. Especifica el tamaño en caracteres del buffer para el tamaño de la cadena que regresará el directorio
lpBuffer. Apunta al buffer de la cadena del directorio

Código
Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "dir_actual" y agregar el código correspondiente

```
' en la forma
Private Declare Function GetCurrentDirectory Lib "kernel32" Alias
"GetCurrentDirectoryA" (ByVal nBufferLength As Long, ByVal lpBuffer As String)
As Long

Private Sub dir_actual_Click()
Dim sSave As String

    ' crear el buffer de la variable
    sSave = String(255, 0)
    ' obtener el directorio actual de donde se encuentra
    ' el programa ejecutable
    GetCurrentDirectory 255, sSave
    MsgBox sSave
End Sub
```

Este código muestra la manera de obtener el directorio actual, se tiene que crear una variable buffer sSave antes de mandar llamar la API, los parámetros son el tamaño, y la variable a recibir la cadena sSave, hecho esto, se coloca dentro de un mensaje par mostrarlo a la vista, pero se puede utilizar internamente bastante, desde saber cómo localizar ciertos programas, instalaciones, entre otras.

3.1.9 SHGetSpecialFolderLocation

Private Declare Function SHGetSpecialFolderLocation Lib "shell32.dll" (ByVal hwndOwner As Long, ByVal nFolder As Long, pidl As ITEMIDLIST) As Long

Información
Esta función obtiene información de las carpetas especiales de Windows

Sistema Operativo
Windows NT 3.1, Windows 9x, windows XP

Parámetros
hwndOwner. Pasa la propia ventana de la aplicación, necesaria
nFolder. Valor que especifica el fólder del cual se quiere su dirección
Ppidl. Dirección que recibe el apuntador a un elemento, identificado por la dirección del fólder

3.1.10 SHGetPathFromIDList

Declare Function SHGetPathFromIDList Lib "shell32" (ByVal pidList As Long, ByVal lpBuffer As String) As Long

Información

Esta función convierte el elemento identificador dado por la API anterior SHGetSpecialFolderLocation en una dirección de las carpetas especiales de Windows.

Sistema Operativo

Windows NT 3.1, Windows 9x, windows XP

Parámetros

pidl. Apuntador al elemento que especifica la dirección de la carpeta

pszPath. Apuntador al buffer que recibe la dirección

Código

Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "carpetas_de_sistema", una caja de texto "Text1" multilínea, barras verticales y agregar el código correspondiente

```
' en la forma [2]
' declaracion de las constantes para las caracteristicas
' especiales de las carpetas de windows
Const CSIDL_DESKTOP = &H0
Const CSIDL_PROGRAMS = &H2
Const CSIDL_CONTROLS = &H3
Const CSIDL_PRINTERS = &H4
Const CSIDL_PERSONAL = &H5
Const CSIDL_FAVORITES = &H6
Const CSIDL_STARTUP = &H7
Const CSIDL_RECENT = &H8
Const CSIDL_SENDTO = &H9
Const CSIDL_BITBUCKET = &HA
Const CSIDL_STARTMENU = &HB
Const CSIDL_DESKTOPDIRECTORY = &H10
Const CSIDL_DRIVES = &H11
Const CSIDL_NETWORK = &H12
Const CSIDL_NETHOOD = &H13
Const CSIDL_FONTS = &H14
Const CSIDL_TEMPLATES = &H15
Const MAX_PATH = 260

' estructuras
Private Type SHITEMID
    cb As Long
    abID As Byte
End Type
```

[2] http://www.allapi.net/

```
Private Type ITEMIDLIST
    mkid As SHITEMID
End Type

' declaracion de las apis
Private Declare Function SHGetSpecialFolderLocation Lib "shell32.dll" (ByVal
hwndOwner As Long, ByVal nFolder As Long, pidl As ITEMIDLIST) As Long

Private Declare Function SHGetPathFromIDList Lib "shell32.dll" Alias
"SHGetPathFromIDListA" (ByVal pidl As Long, ByVal pszPath As String) As Long

Private Sub carpetas_de_sistema_Click()
  ' imprimir las cartetas en Text1
    Text1.Text = "Carpeta de menu de inicio: " +
GetSpecialfolder(CSIDL_STARTMENU) & vbCrLf & _
    "Carpeta de favoritos: " +
GetSpecialfolder(CSIDL_FAVORITES) & vbCrLf & _
    "Carpeta de programas: " +
GetSpecialfolder(CSIDL_PROGRAMS) & vbCrLf & _
    "Carpeta de mandar a: " +
GetSpecialfolder(CSIDL_SENDTO) & vbCrLf & _
    "Carpeta de entorno de red: " +
GetSpecialfolder(CSIDL_NETHOOD) & vbCrLf & _
    "Carpeta de las fuentes: " +
GetSpecialfolder(CSIDL_FONTS) & vbCrLf & _
    "Carpeta de los archivos recientes: " + GetSpecialfolder(CSIDL_RECENT) &
vbCrLf & _
    "Carpeta de las plantillas: " +
GetSpecialfolder(CSIDL_TEMPLATES) & vbCrLf & _
    "Carpeta del escritorio: " +
GetSpecialfolder(CSIDL_DESKTOP)
End Sub

Private Function GetSpecialfolder(CSIDL As Long) As String
    Dim r As Long
    Dim IDL As ITEMIDLIST
    ' obtener la carpeta especial
    r = SHGetSpecialFolderLocation(100, CSIDL, IDL)
    If r = NOERROR Then
        ' se crea el buffer
        Path$ = Space$(512)
        ' obtener la direccion de IDList
        r = SHGetPathFromIDList(ByVal IDL.mkid.cb, ByVal Path$)
        ' quitar los espacios no necesarios
        GetSpecialfolder = Left$(Path, InStr(Path, Chr$(0)) - 1)
        Exit Function
    End If
    GetSpecialfolder = ""
End Function
```

Para explicar este código es necesario aclarar que se están utilizando dos APIS, ésta y la anterior, por ello aquélla no tuvo código de ejemplo, porque aquí se combinan las dos. Este ejemplo puede parecer un poco confuso, pero no es así, si se revisa con atención se entenderá mejor. Comenzamos por declarar las constantes, que éstas no son más que los valores que se pasarán para obtener la carpeta deseada, las estructuras y, por último, las dos APIS.

Al ejecutar el evento clic del botón carpetas_de_sistema, se ejecuta directamente la función GetSpecialfolder() que parece enredoso pero no lo es; primero se pasan los parámetros deseados, la constante y se concatenan todas, con la cadena para indicar el nombre de la carpeta, vbCrLf para brincar a la siguiente línea y todo esto asignado a Text1.

Una vez en la función GetSpecialfolder() tipo cadena, se declaran las variables a utilizar, llama a la API SHGetSpecialFolderLocation y sus parámetros, CSIDL, IDL, valor de la constante y la estructura respectivamente, y lo asigno a r, si r es NOERROR entra al if, asigno una variable cadena y su tamaño, después utilizo SHGetPathFromIDList con las variables IDL.mkid.cb por valor y Path también, ahora GetSpecialfolder tiene la cadena del resultado, le quito los espacios de más, y termino el if, en caso de haber error, GetSpecialfolder es igual a cadena vacía "", regreso a la función, concateno e imprimo en la caja de texto.

A continuación se muestran las imágenes del resultado del programa hecho anteriormente

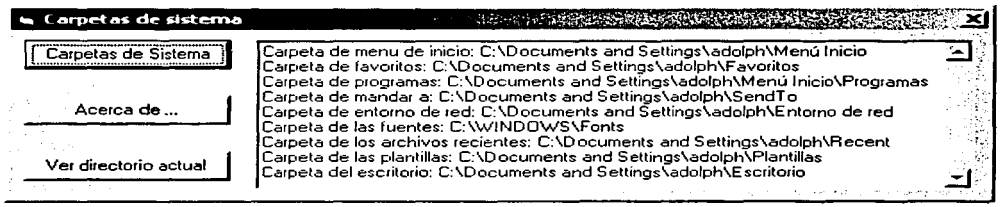

Figura 3.4 Ventana mostrando la localización de las carpetas del sistema

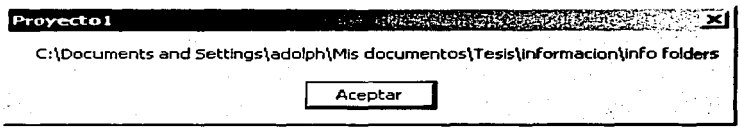

Figura 3.5 Ventana de resultado de la figura 3.4 del botón de "Ver directorio actual"

3.2 Internet y Redes

Este apartado es muy importante, ya que actualmente todas las computadoras se están conectando entre ellas mismas; la computadora que se encuentra fuera de conexión con otras, digamos que no tenga Internet, está limitada a información y comunicación en un tiempo mucho más rápido que si se tuviera que estar transportando los disco de instalación de un programo; por decir un ejemplo de ciudad en ciudad, o hasta es tedioso hacer eso dentro de un mismo centro de cómputo, no hay como la comodidad y rapidez de jalar toda la información de una máquina que prevea de tal programa para instalar. Por ello y mucho más, estas APIS son muy necesarias actualmente y lo serán más en un futuro no muy lejano, tal vez cambien de características, pero habrá quien las reemplace y mejore.

Estas APIs son de las más importantes, utilizan tecnologías de Sockets de Windows, protocolos IP, que son los más usados en programas como el Internet Explorer.

3.2.1 IsNetworkAlive

Declare Function IsNetworkAlive Lib "SENSAPI.DLL" (ByRef lpdwFlags As Long) As Long

Información
Determina si el sistema local PC está conectado a una red y el tipo de red ya sea LAN o WAN

Sistema Operativo
Windows 2000, Windows 9x, Windows 2000/XP, con Internet Explorer 5.0

Parámetros
lpdwFlags. Regresa la información del tipo de red que este disponible

Código
Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "Command2" y agregar el código correspondiente

```
' agregarlo en una forma
' constantes para los diferentes tipos de redes

Const NETWORK_ALIVE_AOL = &H4 [3]
Const NETWORK_ALIVE_LAN = &H1
Const NETWORK_ALIVE_WAN = &H2

Private Declare Function IsNetworkAlive Lib "SENSAPI.DLL" (ByRef lpdwFlags As
Long) As Long

Private Sub Command2_Click()
    Dim Ret As Long
    If IsNetworkAlive(Ret) = 0 Then
        MsgBox "El sistema no esta conectado a la Red"
    Else
        MsgBox "El sistema esta conectado con: " + IIf(Ret = NETWORK_ALIVE_AOL,
"AOL", IIf(Ret = NETWORK_ALIVE_LAN, "LAN", "WAN")) + " RED"
    End If
End Sub
```

Una vez creado el proyecto en Visual Basic 6 recomiendo mantenerlo para agregar más APIS a éste relacionadas con redes, trabajos con IPs e Internet, para la API IsNetworkAlive, es sencilla de utilizar, esta nos indica simplemente que tipo de red esta conectado al equipo.

[3] http://www.allapi.net/

En el código primero verificamos que haya conexión red, de lo contrario mandamos un mensaje indicando que no hay tal, de lo contrario mostramos un mensaje diciendo el tipo de red que hay en existencia en tres simples condiciones en if.

3.2.2 IsDestinationReachable

Declare Function IsDestinationReachable Lib "SENSAPI.DLL" Alias "IsDestinationReachableA" (ByVal lpszDestination As String, ByRef lpQOCInfo As QOCINFO) As Long

Información
Determina si la dirección especificada esta o no disponible para una conexión o enlace con la misma, puede ser una PC en red o dirección en la Internet.

Sistema Operativo
Windows 2000, Windows 9x, Windows 2000/XP, con Internet Explorer 5.0

Parámetros
lpszDestination. Apunta a la cadena que lleva la dirección
lpQOCInfo. Apunta a la estructura que recibe la información de la dirección

Código
Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "Command1" y agregar el código correspondiente

```
' Dentro de una forma
Private Type QOCINFO [4]
    dwSize As Long ' tamano
    dwFlags As Long ' bandera
    dwInSpeed As Long 'bytes/segundo
    dwOutSpeed As Long 'bytes/segundo
End Type

Private Declare Function IsDestinationReachable Lib "SENSAPI.DLL" Alias
"IsDestinationReachableA" (ByVal lpszDestination As String, ByRef lpQOCInfo As
QOCINFO) As Long

Private Sub Command1_Click()
    Dim Ret As QOCINFO
    Ret.dwSize = Len(Ret)
    ' nombre del equipo en la red
    If IsDestinationReachable("rawls", Ret) = 0 Then
        MsgBox "El destino no puede ser contactado"
    Else
        MsgBox "El destino es contactado " + vbCrLf + _
            "La velocidad de la informacion entrante es: " +
Format$(Ret.dwInSpeed / 1024, "#.0") + " Kb/s," + vbCrLf + _
```

[4] http://www.allapi.net/

```
            "La velocidad mandada es:  " + Format$(Ret.dwOutSpeed / 1024, "#.0")
+ " Kb/s."
    End If
End Sub
```

Esta API es un poco más compleja que la anterior por que necesita de una estructura QOCINFO, una vez declarada esta y la API en la forma, creado el botón y su evento clic, se declara a Ret como QOCINFO, después se da el tamaño de Ret y se manda llamar IsDestinationReachable con los valores de la dirección y Ret, si resulta cero el destino es fallido, sino se muestra un mensaje con los datos de la velocidad de transmisión.

3.2.3 InternetGetConnectedStateEx

Declare Function InternetGetConnectedStateEx Lib "wininet.dll" (ByRef lpdwFlags As Long, ByVal lpszConnectionName As String, ByVal dwNameLen As Integer, ByVal dwReserved As Long) As Long

Información
Regresa el estado de la conexión de la dirección especificada en Internet
Sistema Operativo
Windows 2000, Windows 9x, Windows 2000/XP, con Internet Explorer 5.0

Parámetros
lpdwFlags. Regresa el tipo de conexión a una variable long
lpszConnectionName. Apunta la cadena que recibe el nombre de la conexión
dwNameLen. Este contiene la longitud de la cadena
dwReserved. Reservada, debe ser cero

Código
Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "Command3" y agregar el código correspondiente

```
' dentro de la forma la declaracion de la api y constantes
Private Declare Function InternetGetConnectedStateEx Lib "wininet.dll" (ByRef
lpdwFlags As Long, ByVal lpszConnectionName As String, ByVal dwNameLen As
Integer, ByVal dwReserved As Long) As Long
Dim sConnType As String * 255

Private Sub Command3_Click()
  Dim Ret As Long
    Ret = InternetGetConnectedStateEx(Ret, sConnType, 254, 0)
    If Ret = 1 Then
        MsgBox "Coneccion a internet via: " & sConnType, vbInformation
    Else
        MsgBox "No estas conectado a internet", vbInformation
    End If
End Sub
```

Aquí se obtienen los parámetros de Ret primeramente en la mandada a llamar de InternetGetConnectedStateEx con la constante sConnType que recibirá el nombre de la conexión a Internet, el tamaño permitido seguido y la reservada cero, si regresa 1 la conexión es exitosa, de lo contrario se despliega el mensaje de que no se está actualmente conectado a la Internet.

3.2.4 InternetCheckConnection

Declare Function InternetCheckConnection Lib "wininet.dll" Alias "InternetCheckConnectionA" (ByVal lpszUrl As String, ByVal dwFlags As Long, ByVal dwReserved As Long) As Long

Información
Manda llamar una dirección especificada en Internet, para ver si es o no accesible, es revisar la disponibilidad de esta solamente, pero se utiliza por lo regular para mandar llamar la conexión a Internet y conectar a un programa en específico.

Sistema Operativo
Windows 2000, Windows 9x, Windows 2000/XP, con Internet Explorer 3.0

Parámetros
lpszUrl. Apunta a la cadena que contiene la dirección
dwFlags. Contiene el valor de la bandera, donde esta es la única bandera disponible; FLAG_ICC_FORCE_CONNECTION
dwReserved. Reservada, debe ser cero

Código
Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "Command4" y agregar el código correspondiente

```
' dentro de la forma
Private Const FLAG_ICC_FORCE_CONNECTION = &H1
Private Declare Function InternetCheckConnection Lib "wininet.dll" Alias
"InternetCheckConnectionA" (ByVal lpszUrl As String, ByVal dwFlags As Long,
ByVal dwReserved As Long) As Long

Private Sub Command4_Click()
If InternetCheckConnection("http://www.yahoo.com/", FLAG_ICC_FORCE_CONNECTION,
0&) = 0 Then
        MsgBox "Conexion a  http://www.yahoo.com Fallida", vbInformation
    Else
        MsgBox "Conexion a http://www.yahoo.com Exitosa",
vbInformation
    End If
End Sub
```

Este es la mas sencilla de estas APIS de red e Internet pero una de las más útiles pues como se explica en su información se utiliza mucho por los programas actuales que trabajan con Internet, el código es censillo, tenemos la constante FLAG_ICC_FORCE_CONNECTION, y un simple if, donde si el resultado de pedir la conexión es diferente de cero entonces la conexión a la dirección será exitosa.
De lo contrario, será fallida por cuestiones de que no se encuentre la conexión a Internet, se cancele o existan errores como que no haya marcado telefónico a redes o este desinstalado.

3.2.5 InternetAttemptConnect

Declare Function InternetAttemptConnect Lib "wininet" (ByVal dwReserved As Long) As Long

Información
Esta API es utilizada para prácticamente la misma función que la anterior, el detalle se encuentra en la compatibilidad de los Sistemas Operativos de Windows, el programa que la trabaje puede entrar en modo fuera de línea, en caso de no conectar.

Sistema Operativo
Windows NT 4.0, Windows 2000/XP
Parámetros
dwReserved. Reservada, debe ser cero

Código
Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "Command5" y agregar el código correspondiente

```
' en una forma
Private Declare Function InternetAttemptConnect Lib "wininet" (ByVal dwReserved
As Long) As Long

Private Sub Command5_Click()
    If InternetAttemptConnect(ByVal 0&) = 0 Then
        MsgBox "Se puede conectar a Internet", vbInformation
    Else
        MsgBox "No se puede conectar a Internet", vbInformation
    End If
End Sub
```

Este código es sencillo ya que la función sólo requiere de un parámetro, y este es cero por de falta, casi como un tipo de variable booleano, dentro de un if si InternetAttemptConnect regresa cero entra a conexión, de resultar otro valor como dar en el botón de cancelar la conexión mandara el mensaje de que no se conecta a Internet, de este modo un programa seria capaz de saber como trabajar, en o fuera de línea, como lo hace el Internet Explorer de Microsoft.

Las siguientes figuras son las imágenes del programa realizado en Visual Basic 6 con las APIS antes vistas en este capítulo, la reacción de las mismas, la llamada a conexión a Internet, el programa en sí y sus botones.

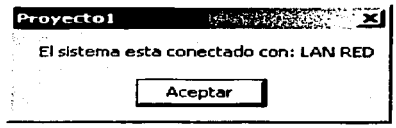

Figura 3.6 Ventana mostrando que existe una conexión a red

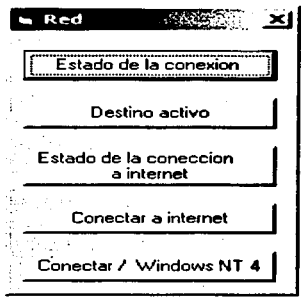

Figura 3.7 Ventana mostrando todos los botones del programa red

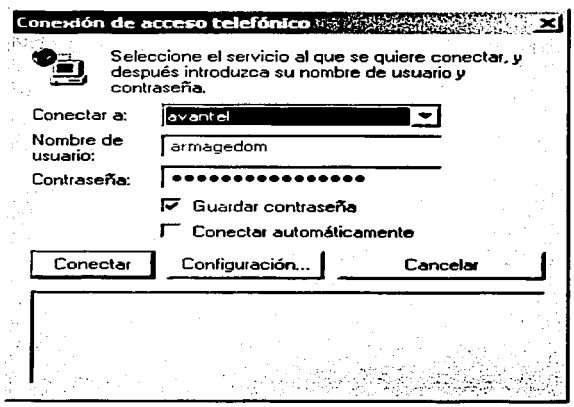

Figura 3.8 Ventana ilustrando el servicio de conexión a Internet

3.2.6 WSAStartup

Declare Function WSAStartup Lib "WSOCK32" (ByVal wVersionRequired As Long, lpWSADATA As WSAData) As Long

Información
Esta función inicializa el uso de los sockets de Windows

Sistema Operativo
Windows NT 3.1, Windows 9x, windows XP, requiere windows Sockets 2.0

Parámetros
wVersionRequested. Verifica la versión actual de Windows sockets
lpWSAData. Apunta a WSADATA, que recibe la información de los sockets

3.2.7 WSACleanup

Declare Function WSACleanup Lib "WSOCK32" () As Long

Información
Esta función finaliza el uso de los sockets de Windows, y su librería dll

Sistema Operativo
Windows NT 3.1, Windows 9x, windows XP, requiere windows Sockets 1.1 o posteriores

Parámetros
No hay parámetros

3.2.8 inet_addr

Declare Function inet_addr Lib "wsock32.dll" (ByVal cp As String) As Long

Información
Esta función convierte el contenido de una cadena de un protocolo de Internet "dirección con punto" en una dirección propia para la estructura de la misma función

Sistema Operativo
Windows NT 3.1, Windows 9x, windows XP, requiere windows Sockets 1.1

Parámetros
cp. Cadena de tipo carácter representando un número expresado en la notación estándar

3.2.9 IcmpSendEcho

Private Declare Function IcmpSendEcho Lib "ICMP" (ByVal IcmpHandle As Long, ByVal DestAddress As Long, ByVal RequestData As String, ByVal RequestSize As Integer, RequestOptns As IP_OPTION_INFORMATION, ReplyBuffer As IP_ECHO_REPLY, ByVal ReplySize As Long, ByVal TimeOut As Long) As Boolean

Información
Esta función manda un ICMP eco y regresa una o más respuestas

Sistema Operativo
Windows NT 3.1, Windows 9x, windows XP, requiere windows Sockets 2.0

Parámetros
IcmpHandle. Llamada de ICMP abierta por IcmpCreateFile
DestinationAddress. Especifica el destino del eco
RequestData. Buffer que contiene la información mandada
RequestSize. Numero de bytes en el buffer de RequestData
RequestOptions. Puntero a las opciones de IP, puede ser Null
ReplyBuffer. Buffer que mantiene cualquier respuesta
ReplySize. Tamaño en bytes del buffer de respuesta
Timeout. Tiempo en milisegundos de la espera de la respuesta

3.2.10 IcmpCreateFile

Declare Function IcmpCreateFile Lib "icmp.dll" () As Long

Información
Esta función crea una llamada en la cual las peticiones al Protocolo de Mensajes del Control de Internet (ICMP) puedan ser publicadas

Sistema Operativo
Windows NT 3.1, Windows 9x, windows XP, requiere windows Sockets 2.0

Parámetros
No hay parámetros

3.2.11 IcmpCloseHandle

Declare Function IcmpCloseHandle Lib "icmp.dll" (ByVal HANDLE As Long) As Boolean

Información
Esta función cierra una llamada de ICMP abierta por IcmpCreateFile

Sistema Operativo
Windows NT 3.1, Windows 9x, windows XP, requiere windows Sockets 2.0

Parámetros
IcmpHandle. Es el ICMP abierto por IcmpCreateFile

Estas APIS son unas de las mas utilizadas en cuestiones de programación para red en LAN o WAN, ya que combinan el poder del protocolo IP y los sockets de Windows, de este es necesario la versión mínima 1.1, pero recomiendo usar la 2 o posteriores, estas primeras estan en la librería de "Wsock32.dll" y la de Protocolo de Mensajes del Control de Internet (ICMP) en "icmp".

En las APIS anteriores no se mostró ejemplo único porque se van a utilizar en el siguiente ejemplo todas, para poder ilustrar de una mejor manera su funcionamiento.

El siguiente programa hará un ping, a una dirección IP ya sea en Internet o en una red, mostrará su estado como respuesta, si es exitoso o existió algún error de hardware, conexión, entre otros, este comando de DOS, "ping", es muy utilizado para probar computadoras conectadas en red principalmente, así que el código es de mucha utilidad, pues bien, sin más preámbulos, vayamos directamente al código.

Código
Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "btn_ping", Una caja de texto "Text1", otra mas "Text2", un módulo y agregar el código correspondiente

Este código se irá desglosando por partes y no puesto todo, se comenzará a explicar; en esta primera parte se colocan las declaraciones de las estructuras, constantes y las APIS que utilizaremos, cabe recalcar que se pondrán todas las constantes para abarcar todos los casos de respuesta del ping que haremos.

```
' ponerlo todo en un modulo
Option Explicit

' declaracion de las constantes
Private Const IP_STATUS_BASE As Long = 11000
Private Const IP_SUCCESS As Long = 0
Private Const IP_BUF_TOO_SMALL As Long = (11000 + 1)
Private Const IP_DEST_NET_UNREACHABLE As Long = (11000 + 2)
Private Const IP_DEST_HOST_UNREACHABLE As Long = (11000 + 3)
Private Const IP_DEST_PROT_UNREACHABLE As Long = (11000 + 4)
Private Const IP_DEST_PORT_UNREACHABLE As Long = (11000 + 5)
Private Const IP_NO_RESOURCES As Long = (11000 + 6)
Private Const IP_BAD_OPTION As Long = (11000 + 7)
Private Const IP_HW_ERROR As Long = (11000 + 8)
Private Const IP_PACKET_TOO_BIG As Long = (11000 + 9)
Private Const IP_REQ_TIMED_OUT As Long = (11000 + 10)
Private Const IP_BAD_REQ As Long = (11000 + 11)
Private Const IP_BAD_ROUTE As Long = (11000 + 12)
Private Const IP_TTL_EXPIRED_TRANSIT As Long = (11000 + 13)
Private Const IP_TTL_EXPIRED_REASSEM As Long = (11000 + 14)
Private Const IP_PARAM_PROBLEM As Long = (11000 + 15)
Private Const IP_SOURCE_QUENCH As Long = (11000 + 16)
```

```
Private Const IP_OPTION_TOO_BIG As Long = (11000 + 17)
Private Const IP_BAD_DESTINATION As Long = (11000 + 18)
Private Const IP_ADDR_DELETED As Long = (11000 + 19)
Private Const IP_SPEC_MTU_CHANGE As Long = (11000 + 20)
Private Const IP_MTU_CHANGE As Long = (11000 + 21)
Private Const IP_UNLOAD As Long = (11000 + 22)
Private Const IP_ADDR_ADDED As Long = (11000 + 23)
Private Const IP_GENERAL_FAILURE As Long = (11000 + 50)
Private Const MAX_IP_STATUS As Long = (11000 + 50)
Private Const IP_PENDING As Long = (11000 + 255)
Private Const PING_TIMEOUT As Long = 500
Private Const WS_VERSION_REQD As Long = &H101
Private Const MIN_SOCKETS_REQD As Long = 1
Private Const SOCKET_ERROR As Long = -1
Private Const INADDR_NONE As Long = &HFFFFFFFF
Private Const MAX_WSADescription As Long = 256
Private Const MAX_WSASYSStatus As Long = 128

' declaracion de las estructuras
Private Type WSADATA [5]
   wVersion As Integer
   wHighVersion As Integer
   szDescription(0 To MAX_WSADescription) As Byte
   szSystemStatus(0 To MAX_WSASYSStatus) As Byte
   wMaxSockets As Long
   wMaxUDPDG As Long
   dwVendorInfo As Long
End Type

Private Type ICMP_OPTIONS
   Ttl               As Byte
   Tos               As Byte
   Flags             As Byte
   OptionsSize       As Byte
   OptionsData       As Long
End Type

Public Type ICMP_ECHO_REPLY
   Address           As Long
   status            As Long
   RoundTripTime     As Long
   DataSize          As Long
   DataPointer       As Long
   Options           As ICMP_OPTIONS
   Data              As String * 250
End Type

' declarar las apis
Private Declare Function WSAStartup Lib "wsock32" (ByVal wVersionRequired As
Long, lpWSADATA As WSADATA) As Long
Private Declare Function WSACleanup Lib "wsock32" () As Long
Private Declare Function IcmpCreateFile Lib "icmp.dll" () As Long
Private Declare Function IcmpCloseHandle Lib "icmp.dll" (ByVal IcmpHandle As
Long) As Long
```

[5] http://www.vbworld.com/ (estructuras y constantes)

```
Private Declare Function IcmpSendEcho Lib "icmp.dll" _
     (ByVal IcmpHandle As Long, _
     ByVal DestinationAddress As Long, _
     ByVal RequestData As String, _
     ByVal RequestSize As Long, _
     ByVal RequestOptions As Long, _
     ReplyBuffer As ICMP_ECHO_REPLY, _
     ByVal ReplySize As Long, _
     ByVal Timeout As Long) As Long
Private Declare Function inet_addr Lib "wsock32" (ByVal s As String) As Long
```

Una vez ya declaradas las constantes y APIS en el módulo, vayamos a la forma, dentro de ella colocar el código del botón "btn_ping".

```
' en la forma
Option Explicit

Private Sub btn_ping_Click()
 Dim ECHO As ICMP_ECHO_REPLY
   Dim pos As Long
   Dim success As Long
   Dim sIPAddress As String

   If SocketsInitialize() Then

      ' ping el IP pasado por la caja Text1
      ' para utilizar la escructura echo
      sIPAddress = Text1.Text
      success = ping(sIPAddress, ("hola"), ECHO)

      ' mostrar los resultados a la caja Text2
      Text2.Text = GetStatusCode(success)

      SocketsCleanup

   Else

        MsgBox "Windows Sockets para 32 bit Windows " & _
               "no esta respondiendo."

   End If
End Sub
```

Se comienza declarando las variables de estructura "eco" y las de ayuda, long y cadena, dentro del evento clic de botón una vez hecho esto se manda llamar la función SocketsInitialize() para inicializar los sockets, en caso de error, simplemente se termina con la opción else, se manda un mensaje y se acaba el programa.

```
Public Function SocketsInitialize() As Boolean
   Dim WSAD As WSADATA
   ' inicializar los sockets
   SocketsInitialize = WSAStartup(WS_VERSION_REQD, WSAD) = IP_SUCCESS
End Function
```

Aquí, en esta función se esta inicializando la librería de los sockets de Windows para trabajarla sin necesidad, de agregar el control como componentes, ni darlo de alta para Visual Basic como control, si no que se manda llamar directa mente y se le utiliza.

Una vez inicializados los sockets en el evento del botón se pasa el valor de la caja Text1 a la variable tipo cadena sIPAddress y se manda como parámetro a la función ping (sIPAddress, ("hola"), ECHO), donde "hola" son los bytes a enviar como información, y ECHO la estructura.

```
Public Function ping(sAddress As String, sDataToSend As String, ECHO As
ICMP_ECHO_REPLY) As Long
   Dim hPort As Long
   Dim dwAddress As Long

   'Convertir la direccion en formato long
   dwAddress = inet_addr(sAddress)

  'si dwAddress es valido
   If dwAddress <> INADDR_NONE Then
     ' abrir un puerto
      hPort = IcmpCreateFile()

     ' si es exitoso
      If hPort Then
        ' hacer el ping
         Call IcmpSendEcho(hPort, dwAddress, sDataToSend, Len(sDataToSend), 0,
ECHO, Len(ECHO), PING_TIMEOUT)
        ' regreso el estado del ping
         ping = ECHO.status
        ' cierro el puerto
         Call IcmpCloseHandle(hPort)
      End If
   Else:
        ' el formato de la direccion es invalido
         ping = INADDR_NONE
   End If
End Function
```

Este código es relativamente sencillo, basta saber los pasos para realizar la operación, aunque no es tan sencillo como utilizar el comando en Dos pero con la ventaja de que éste está programado en nuestro programa y lo utilizaremos tantas veces como queramos y de manera automática cuando lo necesitemos.

La función ping es la más complicada entre comillas, primeramente se declaran las variables y ya obtenidos los parámetros por pase, se crea una nueva dirección con inet_addr, después verifico si ésta es válida o no, una vez aclarada que es válida, abro un puerto con IcmpCreateFile() una vez más si éste es exitoso, ahora sí mando llamar la función cmpSendEcho con sus parámetros correspondientes, dirección y ecos, después ping = ECHO.status la función ping obtendrá el parámetro de estado de echo y su propiedad status, hecho esto cierro el puerto usado con cmpCloseHandle y finalmente regreso al evento clic del botón.

```
Public Function GetStatusCode(status As Long) As String
   Dim msg As String
   ' posibles estados
```

```
    Select Case status [6]
       Case IP_SUCCESS:                msg = "ip success"
       Case INADDR_NONE:               msg = "inet_addr: bad IP format"
       Case IP_BUF_TOO_SMALL:          msg = "ip buf too_small"
       Case IP_DEST_NET_UNREACHABLE:   msg = "ip dest net unreachable"
       Case IP_DEST_HOST_UNREACHABLE:  msg = "ip dest host unreachable"
       Case IP_DEST_PROT_UNREACHABLE:  msg = "ip dest prot unreachable"
       Case IP_DEST_PORT_UNREACHABLE:  msg = "ip dest port unreachable"
       Case IP_NO_RESOURCES:           msg = "ip no resources"
       Case IP_BAD_OPTION:             msg = "ip bad option"
       Case IP_HW_ERROR:               msg = "ip hw_error"
       Case IP_PACKET_TOO_BIG:         msg = "ip packet too_big"
       Case IP_REQ_TIMED_OUT:          msg = "ip req timed out"
       Case IP_BAD_REQ:                msg = "ip bad req"
       Case IP_BAD_ROUTE:              msg = "ip bad route"
       Case IP_TTL_EXPIRED_TRANSIT:    msg = "ip ttl expired transit"
       Case IP_TTL_EXPIRED_REASSEM:    msg = "ip ttl expired reassem"
       Case IP_PARAM_PROBLEM:          msg = "ip param_problem"
       Case IP_SOURCE_QUENCH:          msg = "ip source quench"
       Case IP_OPTION_TOO_BIG:         msg = "ip option too_big"
       Case IP_BAD_DESTINATION:        msg = "ip bad destination"
       Case IP_ADDR_DELETED:           msg = "ip addr deleted"
       Case IP_SPEC_MTU_CHANGE:        msg = "ip spec mtu change"
       Case IP_MTU_CHANGE:             msg = "ip mtu_change"
       Case IP_UNLOAD:                 msg = "ip unload"
       Case IP_ADDR_ADDED:             msg = "ip addr added"
       Case IP_GENERAL_FAILURE:        msg = "ip general failure"
       Case IP_PENDING:                msg = "ip pending"
       Case PING_TIMEOUT:              msg = "ping timeout"
       Case Else:                      msg = "unknown  msg returned"
    End Select

    ' concatenar las cadenas del estado y mensaje
    GetStatusCode = CStr(status) & "   [ " & msg & " ]"
End Function
```

Simplemente para mandar llamar la función *GetStatusCode* y asignarle su valor regresado tipo cadena a la segunda caja Text2, la función, es sencilla, es solo un case y al final una concatenación del valor obtenido y el caso obtenido, de acuerdo al valor de la constante del mensaje del ping.

Terminando el evento del botón se manda llamar *SocketsCleanup()* para cerrar adecuadamente los sockets, dentro de esta función también se mandará un mensaje de error en caso de que se presentara.

```
Public Sub SocketsCleanup()
    ' error de sockets

   If WSACleanup() <> 0 Then
       MsgBox "Error de sockets en Windows", vbExclamation
   End If
End Sub
```

[6] http://www.vbworld.com/

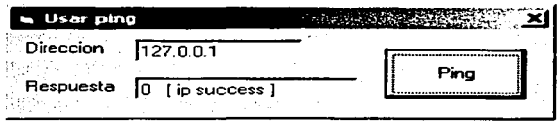

Figura 3.9 Ventana mostrando el programa hecho con el código anterior, que da un ping a una dirección IP asignada en la primera caja de texto.

3.3 El Registro

El registro, bueno como ya se ha visto en el capítulo 2 el registro es digámoslo así la base de datos con la que trabaja Windows, así que el modificar información sin respaldar y sin saber que se hace en específico puede causar un daño parcial al sistema operativo y llegar hasta en casos extremos de ser necesario a volver instalar Windows, así que manejemos con cuidado el registro, del bloque siguiente de APIS serán utilizadas las mas necesarias, útiles y comunes, se construirá un pequeño programa con el cual se podrá crear, leer, borrar y modificar las llaves del registro, pero se darán todos los datos específicos de cada API en caso de ser necesario utilizarlas con otros lenguajes de programación.

3.3.1 RegCreateKey

```
Declare Function RegCreateKey Lib "advapi32.dll" Alias "RegCreateKeyA" (ByVal hKey As Long, ByVal lpSubKey As String, phkResult As Long) As Long
```

Información
Primeramente esta API crea una key o llave en el registro, donde nosotros queramos colocarla, de acuerdo a sus parámetros correspondientes, si la clave ya existe, la abre. En Visual Basic Net ha sido reemplazado por:
Microsoft.Win32.RegistryKey.CreateSubKey

Sistema Operativo
Windows NT 3.1, Windows 9x

Parámetros
hKey. Identifica que tipo de llave es o a cual corresponde
lpSubKey. Apunta a la sub llave o en caso de ser la misma es Null
phkResult. Apunta a la variable que recibe la apertura o creación de la llave

Código
Este código está en Visual Basic Net

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows

- *Crear un botón simple en la forma y agregar el código correspondiente*

```
Imports System
Imports Microsoft.Win32

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        'crear la llave nueva
        Dim RegKey As RegistryKey = Registry.CurrentUser.CreateSubKey("Tesis")
        RegKey.Close()
    End Sub
```

Bien este código es sencillo, una vez creado el proyecto en Visual Basic .Net, se colocan las 2 primeras líneas en las declaraciones de la forma, y después dentro de la declaración del botón, se crea la variable tipo RegistryKey y se le asigna su propiedad CreateSubKey y dentro del parámetro, aquí "Tesis", después se cierra, para verificar abrir el editor de registro, la llave Current user y wuala, ahí se encontrará la carpeta o llave "Tesis".

Bien esto es en caso de utilizar las nuevas características del Visual Basic Net, pero aun así es recomendable conocer las constantes y sus valores, a continuación lo colocó para mayor referencia de todas las existentes, mostrare algunas de las más importantes y necesarias en caso de trabajar con el registro de Windows.

```
' Constantes de las llaves principales [7]
Public Const HKEY_CURRENT_CONFIG = &H80000005
Public Const HKEY_CURRENT_USER = &H80000001
Public Const HKEY_DYN_DATA = &H80000006
Public Const HKEY_LOCAL_MACHINE = &H80000002
Public Const HKEY_PERFORMANCE_DATA = &H80000004
Public Const HKEY_USERS = &H80000003

' Valor de las constantes
Public Const REG_DWORD = 4
Public Const REG_DWORD_BIG_ENDIAN = 5
Public Const REG_DWORD_LITTLE_ENDIAN = 4
Public Const REG_EXPAND_SZ = 2
Public Const REG_FULL_RESOURCE_DESCRIPTOR = 9
Public Const REG_LINK = 6
Public Const REG_MULTI_SZ = 7
Public Const REG_NONE = 0
Public Const REG_RESOURCE_LIST = 8
Public Const REG_RESOURCE_REQUIREMENTS_LIST = 10
Public Const REG_SZ = 1

' Constantes para las subclaves, entre otras
Public Const KEY_CREATE_LINK = &H20
Public Const KEY_CREATE_SUB_KEY = &H4
Public Const KEY_ENUMERATE_SUB_KEYS = &H8
Public Const KEY_EVENT = &H1
Public Const KEY_NOTIFY = &H10
Public Const KEY_QUERY_VALUE = &H1
Public Const KEY_SET_VALUE = &H2
Public Const STANDARD_RIGHTS_ALL = &H1F0000
```

[7] http://guille.costasol.net/vb_api.htm

```
Public Const SYNCHRONIZE = &H100000
Public Const READ_CONTROL = &H20000
Public Const KEY_EXECUTE = (KEY_READ)
Public Const STANDARD_RIGHTS_READ = (READ_CONTROL)
Public Const KEY_WRITE = ((STANDARD_RIGHTS_WRITE Or KEY_SET_VALUE Or
KEY_CREATE_SUB_KEY) And (Not SYNCHRONIZE))
Public Const KEY_ALL_ACCESS = ((STANDARD_RIGHTS_ALL Or KEY_QUERY_VALUE Or
KEY_SET_VALUE Or KEY_CREATE_SUB_KEY Or KEY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY Or
KEY_CREATE_LINK) And (Not SYNCHRONIZE))
Public Const KEY_READ = ((STANDARD_RIGHTS_READ Or KEY_QUERY_VALUE Or
KEY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY) And (Not SYNCHRONIZE))
```

3.3.2 RegOpenKeyEx

Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyExA" (ByVal hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal samDesired As Long, phkResult As Long) As Long

Información
Esta API abre la llave requerida

Sistema Operativo
Windows NT 3.1, Windows 9x

Parámetros
hKey. Identifica que tipo de llave es o a cual corresponde
lpSubKey. Apunta a la sub llave o en caso de ser la misma es Null
ulOptions. Es reservada, debe ponerse un cero
samDesired. Especifica un acceso que describe el tipo de seguridad para la llave que se esta trabajando
phkResult. Apunta a una variable que recibe la llamada de la llave

Código
Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma, llamarlo "abrir llave" y un modulo también, agregar el código correspondiente

```
' dentro del modulo
Public Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyExA"
(ByVal hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal
samDesired As Long, phkResult As Long) As Long
Public Const HKEY_CURRENT_USER = &H80000001
Public Const KEY_WRITE = &H20006

' en la forma
Private Sub abrir_llave_Click()
Dim hregkey As Long ' variable que toma el valor de la llave
Dim subkey As String  ' variable que toma la direccion
Dim retval As Variant ' variable que toma el valor de regreso "0"
```

```
' direcion de la llave
subkey = "Tesis"
' aqui abro la llave
retval = RegOpenKeyEx(HKEY_CURRENT_USER, subkey, 0, KEY_WRITE, hregkey)

' aqui verifico que no haya ningun error, si lo hay manda mensaje
If retval <> 0 Then
    MsgBox "No se puede abrir la llave o no se encuentra"
    Exit Sub
End If

MsgBox "Si se encontro la llave Tesis"
End Sub
```

Ahora bien, dentro del modulo va la declaración de las constantes y la función, en la forma, el código del botón, primeramente se declaran las variables que usaremos, después a subKey se le asigna el valor "Tesis", que es la subcadena a buscar, después a retval se le asignara un valor resultante de la función, donde busco la llave principal, después la subllave, cero por de falta, la tarea a realizar y el andel o valor a regresar al puntero hregkey, después un simple if, si la función regresa algo diferente a cero manda un mensaje alertando de que no se ha encontrado la llave, si es cero si la encontró.

3.3.3 RegSetValueEx

Declare Function RegSetValueEx Lib "advapi32.dll" Alias "RegSetValueExA" (ByVal hKey As Long, ByVal lpValueName As String, ByVal Reserved As Long, ByVal dwType As Long, lpData As Any, ByVal cbData As Long) As Long

Información
Esta API coloca información dentro de una variable creada en la misma función, o sea, que se crea la variable y se le asigna su valor, dentro de una carpeta o llave.

Sistema Operativo
Windows NT 3.1, Windows 9x
Parámetros
hKey. Identifica que tipo de llave es o a cual corresponde
lpValueName. Es una cadena que contiene el nombre de la variable
Reserved. Reservada, poner un cero
dwType. Aquí poner el tipo de variable que será, binaria, cadena, etc.
lpData. Apunta al buffer donde estará el valor de la variable
cbData. Especifica el tamaño en bytes del tamaño de la variable

Código
Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma, llamarlo "poner valor" y un modulo también, agregar el código correspondiente

```
' dentro del modulo
Public Const HKEY_CURRENT_USER = &H80000001
Public Const KEY_WRITE = &H20006
Public Const REG_DWORD = 4

Public Declare Function RegSetValueEx Lib "advapi32.dll" Alias "RegSetValueExA"
(ByVal hKey As Long, ByVal lpValueName As String, ByVal Reserved As Long, ByVal
dwType As Long, lpData As Any, ByVal cbData As Long) As Long
Public Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyExA"
(ByVal hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal
samDesired As Long, phkResult As Long) As Long

' en la forma
Private Sub poner_valor_Click()
Dim hregkey As Long ' variable que toma el valor de la llave
Dim subkey As String  ' variable que toma la direccion
Dim retval As Variant ' variable que toma el valor de regreso
Dim tipo As Long ' tiene que ser long para el valor a colocar
tipo = 1
subkey = "Tesis"
retval = RegOpenKeyEx(HKEY_CURRENT_USER, subkey, 0, KEY_WRITE, hregkey)
retval = RegSetValueEx(hregkey, "Hola a todos", 0, REG_DWORD, tipo, Len(tipo))
End Sub
```

Aquí es muy parecido al caso anterior, solo que primero le tenemos que indicar que dirección, o mejor dicho dentro de que llave utilizaremos nuevamente la API RegOpenKey, una vez la abierta y encontrada creamos la variable y su parámetros dentro de ella, tipo palabra, el tamaño, y su nombre, que son declarados un poco antes y asignados su valor.

Ahora bien, este es un caso de grabar el tipo de variable palabra, pero una cadena como le hacemos para grabarla, bien, utilizando la misma función pero con las constantes y casos que están un par de paginas atrás se utilizaran de acuerdo a lo recurrido, a continuación muestro un caso de variable tipo cadena pero utilizando la misma secuencia de programación solo adaptándolo a el caso a mostrar ahora y utilizando la siguiente función para evitar posibles errores.

3.3.4 RegCloseKey

Declare Function RegCloseKey Lib "advapi32.dll" Alias "RegCloseKey" (ByVal hKey As Long) As Long

Información
Esta API cierra llave identificada

Sistema Operativo
Windows NT 3.1, Windows 9x

Parámetros
hKey. Identifica que tipo de llave a cual corresponde

Código
Este código está en Visual Basic 6

- *Crear un proyecto nuevo*
- *Seleccionar Visual Basic*
- *Aplicación de Windows*
- *Crear un botón simple en la forma, llamarlo "poner valor 2 y cerrándolo" y un modulo también, agregar el código correspondiente*

```
' dentro del modulo
Public Const HKEY_CURRENT_USER = &H80000001
Public Const KEY_WRITE = &H20006
Public Const REG_SZ = 1

Public Declare Function RegSetValueEx Lib "advapi32.dll" Alias "RegSetValueExA"
(ByVal hKey As Long, ByVal lpValueName As String, ByVal Reserved As Long, ByVal
dwType As Long, lpData As Any, ByVal cbData As Long) As Long
Public Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyExA"
(ByVal hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal
samDesired As Long, phkResult As Long) As Long
Public Declare Function RegCloseKey Lib "advapi32.dll" (ByVal hKey As Long) As
Long

Private Sub poner_el_valor_2_y_cerrandolo_Click()
Dim hregkey As Long ' variable que toma el valor de la llave
Dim subkey As String  ' variable que toma la direccion
Dim retval As Variant ' variable que toma el valor de regreso
Dim stringbuffer As String

' siempre que inicie con windows
subkey = "Tesis"
retval = RegOpenKeyEx(HKEY_CURRENT_USER, subkey, 0, KEY_WRITE, hregkey)

stringbuffer = "Valor tipo cadena"
retval = RegSetValueEx(hregkey, "Cadena", 0, REG_SZ, ByVal stringbuffer,
Len(stringbuffer))
' cierro la llave
RegCloseKey hregkey
End Sub
```

Bien, en este caso se abre primero la llave, o lee mejor dicho, se agrega el tipo de variable y su valor y se cierra correctamente. Yo recomiendo utilizar la función para cerrar la variable en el registro, por que de no hacerlo, quedara basura en el sistema de la maquina mientras se esté ejecutando la acción y tal vez podría colapsar el sistema en caso de alterar una variable del sistema operativo.

3.3.5 RegDeleteKey

Declare Function RegDeleteKey Lib "advapi32.dll" Alias "RegDeleteKeyA" (ByVal hKey As Long, ByVal lpSubKey As String) As Long

Información
Esta API elimina la llave identificada

Sistema Operativo
Windows NT 3.1, Windows 9x

Parámetros
hKey. Identifica que tipo de llave es o a cual corresponde
lpSubKey. Coloca el reemplazo de la llave anterior por la actual, no puede tener como valor Null

Código
Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma, llamarlo "borrar_llave" y un módulo también, agregar el código correspondiente

```
' dentro del modulo
Public Const HKEY_CURRENT_USER = &H80000001
Public Const KEY_WRITE = &H20006

Public Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyExA"
(ByVal hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal
samDesired As Long, phkResult As Long) As Long
Public Declare Function RegCloseKey Lib "advapi32.dll" (ByVal hKey As Long) As
Long
Declare Function RegDeleteKey Lib "advapi32.dll" Alias "RegDeleteKeyA" (ByVal
hKey As Long, ByVal lpSubKey As String) As Long

Private Sub borrar_llave_Click()
Dim hregkey As Long ' variable que toma el valor de la llave
Dim subkey As String  ' variable que toma la direccion
Dim retval As Variant ' variable que toma el valor de regreso

    ' abro la llave
    subkey = "Tesis\Sub carpeta"
    retval = RegOpenKeyEx(HKEY_CURRENT_USER, subkey, 0, KEY_WRITE, hregkey)
    ' aqui se crea una subcarpeta
    RegDeleteKey hregkey, ""
    RegCloseKey hregkey
End Sub
```

Bien, este código es intuitivo, y demostrativo, fácil de entender una vez ya hechos los casos de APIS anteriores, igualmente, primero se identifica la llave a eliminar, después se pasa el parámetro y se elimina reemplazándola con doble comillas, una vez hecho esto, se puede ver el registro manualmente y actualizarlo con F5 para ver los cambios.

3.3.6 RegDeleteValue

Declare Function RegDeleteValue Lib "advapi32.dll" Alias "RegDeleteValueA" (ByVal hKey As Long, ByVal lpValueName As String) As Long

Información
Esta API elimina la variable identificada

Sistema Operativo
Windows NT 3.1, Windows 9x

Parámetros
hKey. Identifica que tipo de llave a cual corresponde
lpValueName. Coloca el reemplazo de la variable

Código
Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma, llamarlo "borrar_valor" y un módulo también, agregar el código correspondiente

```
' dentro del modulo
Public Const HKEY_CURRENT_USER = &H80000001
Public Const KEY_WRITE = &H20006

Public Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyExA"
(ByVal hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal
samDesired As Long, phkResult As Long) As Long
Public Declare Function RegCloseKey Lib "advapi32.dll" (ByVal hKey As Long) As
Long
Declare Function RegDeleteValue Lib "advapi32.dll" Alias "RegDeleteValueA"
(ByVal hKey As Long, ByVal lpValueName As String) As Long

Private Sub borrar_valor_Click()
Dim hregkey As Long ' variable que toma el valor de la llave
Dim subkey As String  ' variable que toma la direccion
Dim retval As Variant ' variable que toma el valor de regreso

' abro la llave
subkey = "Tesis"
retval = RegOpenKeyEx(HKEY_CURRENT_USER, subkey, 0, KEY_WRITE, hregkey)

RegDeleteValue hregkey, "hola a todos"
RegCloseKey hregkey
End Sub
```

Esta función es sencilla de utilizar, basta con abrir primeramente la llave o carpeta donde se encuentra para poder pasarle la variable long y la identifique, seguida del nombre de la variable entre comillas tipo cadena, para que sepa cual es, la elimina, cierro la llave y es todo.

Existen más APIS para manejar el registro, mas especificas y de función única del mismo modo que tipos de variables, valores y llaves, etc, por ejemplo si queremos que nuestro programa que realizamos cargue cada vez que se inicie Windows se deberá colocar una variable tipo cadena en el registro con el valor de la ruta del ejecutable de nuestro programa y en la llave del registro "Software\Microsoft\Windows\CurrentVersion\Run" y listo, cada vez que inicie Windows, automáticamente nuestro programa arrancara, en fin, a continuación pondré la imagen de lo que se ha tratado en este capítulo.

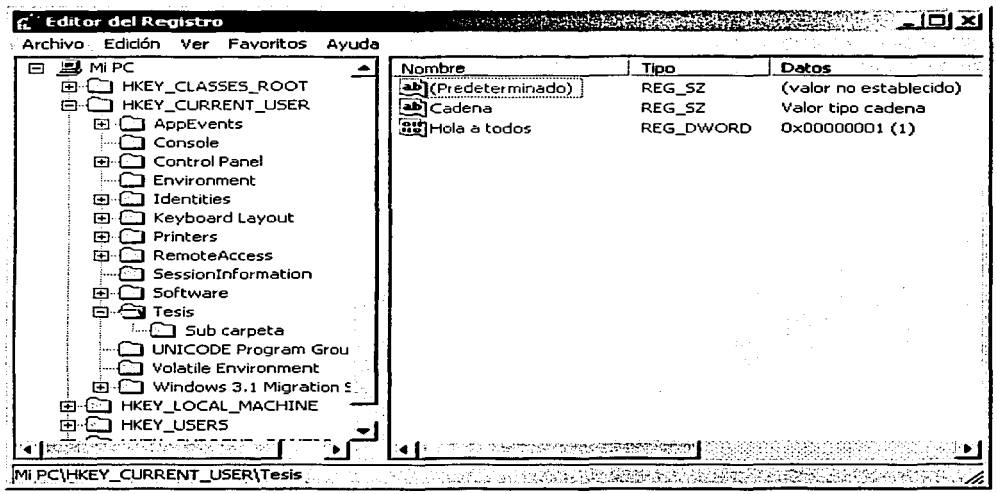

Figura 3.10 Ventana del registro mostrando la llave "Tesis" dentro de la clave principal Hkey_Current_User y los dos tipos de variables, cadena REG_SZ, palabra REG_DWORD y la sub llave "Sub carpeta"

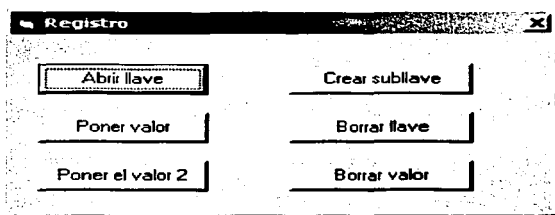

Figura 3.11 Ventana en Visual Basic 6 del programa "Registro" construido con el código visto

3.3.7 RegQueryValueEx

Declare Function RegQueryValueEx Lib "advapi32.dll" Alias "RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName As String, ByVal lpReserved As Long, lpType As Long, lpData As Any, lpcbData As Long) As Long

Información
Esta API obtiene la información necesitada de valor o tipo para una llave requerida en el registro

Sistema Operativo
Windows NT 3.1, Windows 9x

Parámetros
hKey. Identifica que tipo de llave es o a cual corresponde
lpValueName. Tiene en forma cadena sub llave a la cual se hace referencia
lpReserved. Reservada, debe ser cero
lpType. Apunta a la variable que recibirá el tipo de valor devuelto
lpData. Apunta al buffer que recibirá el valor
lpcbData. Apunta el tamaño en bytes del búfer apuntado por lpData

Código
Este código está en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma, llamarlo "info", agregar el código correspondiente

```
' dentro de la forma [8]
' librerias apis a utilizar
Private Declare Function RegOpenKeyEx Lib "advapi32" Alias "RegOpenKeyExA"
(ByVal hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal
samDesired As Long, ByRef phkResult As Long) As Long
Private Declare Function RegQueryValueEx Lib "advapi32" Alias "RegQueryValueExA"
(ByVal hKey As Long, ByVal lpValueName As String, ByVal lpReserved As Long,
ByRef lpType As Long, ByVal lpData As String, ByRef lpcbData As Long) As Long
Private Declare Function RegCloseKey Lib "advapi32" (ByVal hKey As Long) As Long

Option Explicit
' Opciones de seguridad de clave del Registro
Const KEY_ALL_ACCESS = &H2003F

' Tipos ROOT "llaves principales" de claves del Registro
Const HKEY_LOCAL_MACHINE = &H80000002
Const ERROR_SUCCESS = 0
Const REG_SZ = 1
Const REG_DWORD = 4

'Constantes de la direccion del programa de la info
Const gREGKEYSYSINFOLOC = "SOFTWARE\Microsoft\Shared Tools Location"
Const gREGVALSYSINFOLOC = "MSINFO"
```

[8] http://msdn.microsoft.com/

```
Const gREGKEYSYSINFO = "SOFTWARE\Microsoft\Shared Tools\MSINFO"
Const gREGVALSYSINFO = "PATH"

Public Sub StartSysInfo()
    On Error GoTo SysInfoErr

        Dim rc As Long
        Dim SysInfoPath As String

        ' obtener el nombre y la ruta del programa en el Registro
        If GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFO, gREGVALSYSINFO,
SysInfoPath) Then
        ' obtener sólo la ruta del programa en el Registro
        ElseIf GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFOLOC,
gREGVALSYSINFOLOC, SysInfoPath) Then
                ' Validar la existencia de versión conocida del archivo
                If (Dir(SysInfoPath & "\MSINFO32.EXE") <> "") Then
                        SysInfoPath = SysInfoPath & "\MSINFO32.EXE"

                ' Error: no se encuentra el archivo
                Else
                        GoTo SysInfoErr
                End If
        ' Error: no se encuentra la entrada del Registro
        Else
                GoTo SysInfoErr
        End If

        Call Shell(SysInfoPath, vbNormalFocus)

        Exit Sub
SysInfoErr:
        MsgBox "La información del sistema no está disponible", vbOKOnly
End Sub

Public Function GetKeyValue(KeyRoot As Long, KeyName As String, SubKeyRef As
String, ByRef KeyVal As String) As Boolean
        Dim i As Long
        Dim rc As Long
        Dim hKey As Long
        Dim hDepth As Long
        Dim KeyValType As Long
        Dim tmpVal As String
        Dim KeyValSize As Long

        ' Abrir RegKey
                rc = RegOpenKeyEx(KeyRoot, KeyName, 0, KEY_ALL_ACCESS, hKey)

        If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError

        tmpVal = String$(1024, 0)
        KeyValSize = 1024

        rc = RegQueryValueEx(hKey, SubKeyRef, 0, KeyValType, tmpVal, KeyValSize)
        If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError

        tmpVal = VBA.Left(tmpVal, InStr(tmpVal, VBA.Chr(0)) - 1)
```

```
        Select Case KeyValType
        Case REG_SZ
                KeyVal = tmpVal
        Case REG_DWORD
                For i = Len(tmpVal) To 1 Step -1
                        KeyVal = KeyVal + Hex(Asc(Mid(tmpVal, i, 1)))
                Next
                KeyVal = Format$("&h" + KeyVal)
        End Select

        GetKeyValue = True
        rc = RegCloseKey(hKey)
        Exit Function

GetKeyError:
        KeyVal = ""
        GetKeyValue = False
        rc = RegCloseKey(hKey)
End Function
Private Sub info_Click()
    Call StartSysInfo
End Sub
```

Perfecto, una vez ya tecleado el código en Visual Basic y ejecutado, nos mostrará el programa que tiene Windows por de falta para mostrar todas las características técnicas y de software de Windows, la "Información del Sistema" mejor conocida, esta pequeña aplicación es utilizada por la mayoría de programas robustos para Windows como lo es el mismo Office y Corel, por mencionar algunos, ahora bien, desglosaremos el código.

Primeramente es la declaración de las APIS a utilizar, después la declaración de las constantes, cabe mencionar que en esta parte ya se están dando las llaves en el registro que se necesitan, pues previamente ya sabemos la dirección de en donde se encuentra el programa, y de no ser así tenemos el nombre, pues este mismo pequeño programa funcionará tanto para Windows 98 como para el mismo XP, sin necesidad de cambiar una sola línea de código.

Esta es la ventaja de trabajar con el registro de Windows, de esta manera quiero decir que si se tienen los datos básicos de otras aplicaciones, se pueden alterar para interactuar con nuestro propio programa, por ejemplo, agregar un icono al Internet Explorer, cambiar propiedades de Office y barras de Windows, agregar menús, protegerlos, etc. y muchas cosas más.

Continuando con el código en la propiedad clic del botón "info" mando llamar a StartSysInfo() sin paso de parámetros, inmediatamente creo una etiqueta para captar posibles errores, declaro las variables y después dentro de un if llamo la función GetKeyValue con los parámetros de las constantes ya antes declaradas, ahora brincamos a la función, aquí de igual modo declaro las variables a utilizar después abro el registro con la dirección predefinida, en caso de error que seria que la llave no existiera, mando llamar GetKeyError la etiqueta para el error, la cierro y regreso a la función, nuevamente el if resulta ser falso y regreso mando a SysInfoErr la etiqueta que manda un mensaje de error para alertar al usuario, este error seria probable de que el programa de información del sistema de Windows fuera borrado con o sin intenciones de hacerlo.

Regresando a la función, una vez que se abrió la llave y resulto ser exitosa, creo la variable tipo cadena y de tamaño 1024, aquí hago una observación, en este punto se esta creando, anteriormente se declaro mas no se creo, ahora bien, teniendo la dirección y la variable, utilizo la api RegQueryValueEx para obtener la información que deseamos, en caso de haber nuevamente error se repiten los pasos anteriores con las etiquetas de error, pero de no serlo a la variable tmpVal, la corto en su longitud y solo dejo la información exacta en cadena.

Después realizo un if para seleccionar el caso de valor devuelto, si es tipo REG_SZ o REG_DWORD en caso de ser la primera solo necesito pasarle el valor a KeyVal, de ser la segunda, entro a un ciclo para cortar la cadena devuelta y dejarla en un formato compatible para trabajarlo, dicho de este modo GetKeyValue la pongo a verdadero con bandera cierro la llave y salgo de la función, una vez de regreso en StartSysInfo termino el if principal, utilizo call para abrir un programa ejecutable o *.exe, que en este caso en el del info de sistema con todo y su dirección, se carga y es todo, por medio del if.

En caso de que éste hubiera sido un resultado negativo, valido la existencia del archivo desde su nombre ejecutable, y lo sumo o encadeno a la dirección devuelta de donde se encuentra, aquí existe un if para atrapar un error mas en caso de que el archivo no se encuentre, por si fue borrado o cambiado o eliminado, en fin, esta es la manera de trabajar del código, pero sin duda la mejor manera de hacerlo, modificarlo al antojo y utilizarlo para otras cosas es tecleándolo uno mismo.

En la siguiente página se muestra la figura del programa de información de Windows, que se encuentra instalado con el sistema operativo como componente del mismo, en el caso de Windows XP, cuenta con información del Hardware instalado y descripciones específicas del mismo, la configuración a Internet como programas de Internet Explorer, Office de Microsoft y componentes de dispositivos físicos, por que ejemplo el cd – rom, versión del controlador, fabricante, velocidad y características únicas de cada versión, entre otras muchas mas.

En la imagen se demuestra el resultado del código anterior, que mandara llamar este programa, en este caso es Windows XP, pero podría ser Windows 2000, 98 u otros, varias de las características que este mismo programa es que muestra sobre información de la computadora y sistema operativo, usuario, entre otras, serán vistas utilizando otras APIS más adelante en otros temas.

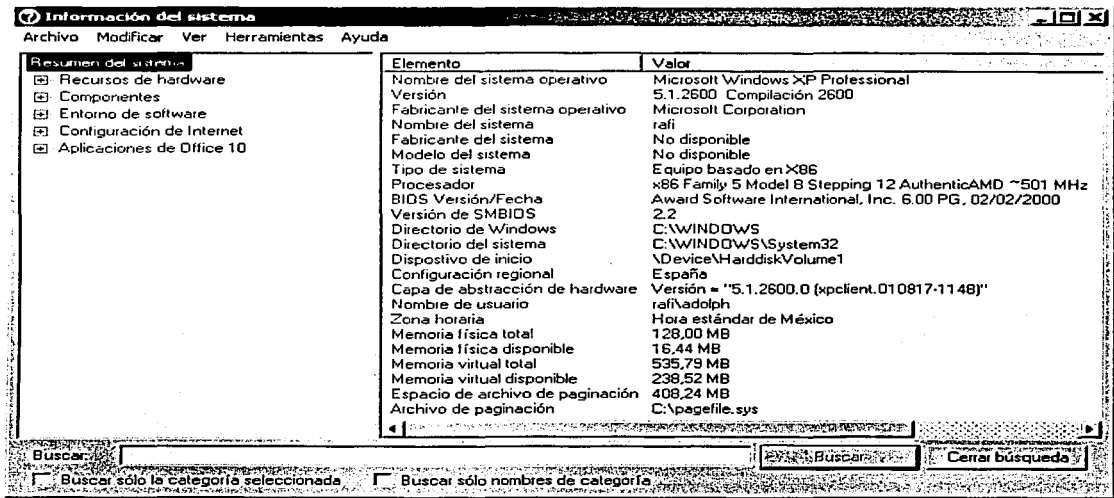

Figura 3.12 Ventana del programa Información del Sistema de Windows XP

A manera de conclusión de este tema cabe resaltar que estas APIs son muy utilizadas en los programas de actualidad, ya sea para instalarse, desinstalarse, corregir o guardar datos o parámetros que necesiten o interactuen con el sistema operativo, como lo es el caso del registro concretamente.

Otros son los que utilizan el internet y las redes como pan de cada día, y la mayoria accede a las caracteristicas de Windows y su información, desde parámetros y carpetas hasta información detallada de dispositivos para detectar conflictos de hardware entre otras posibilidades.

FUNCIONES AVANZADAS DE WINDOWS APIS

CAPITULO IV

4.1 CAJAS DE DIALOGO

Las cajas de dialogo son ventanas especializadas y universales en el Sistema Operativo Windows, son variadas y para diferentes usos, desde configuraciónes de páginas, impresoras, listas para imprimir, hasta abrir archivos y visualizarlos si son archivos tipo avi, (formato de película), guardar, colores, carpetas, etc.

Las siguientes APIS muestran las más utilizadas de ellas, sus constantes para sus diferentes usos y la manera de ser portables para los sistemas operativos Windows sin necesidad de emigrar y reescribir el código como se tendría que hacer con componentes extras, tomando las características de configuración que tiene Windows en esa determinada computadora, sin afectar cambios permanentes ni configuraciones extras, y ayudado de estas grandes herramientas para los programadores bajo esta plataforma, ya que sin estas herramientas se tendría que programar toda una caja de dialogo para hacer la tarea determinada.

4.1.1 GetOpenFileName

Declare Function GetOpenFileName Lib "comdlg32.dll" Alias "GetOpenFileNameA" (pOpenfilename As OPENFILENAME) As Long

Información
Esta función crea una caja de dialogo para permitir al usuario escoger entre discos duros, carpetas y archivos para seleccionar cual abrir

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
Lpofn. Apuntador a la estructura de Openfilename que contiene la información usada por la caja de diálogo

Código
Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "abrir" y agregar el código correspondiente

```
' en la forma
Option Explicit

' estructura para abrir archivo necesaria
Private Type OPENFILENAME
    lStructSize As Long
    hwndOwner As Long
    hInstance As Long
    lpstrFilter As String
    lpstrCustomFilter As String
```

```
    nMaxCustFilter As Long
    nFilterIndex As Long
    lpstrFile As String
    nMaxFile As Long
    lpstrFileTitle As String
    nMaxFileTitle As Long
    lpstrInitialDir As String
    lpstrTitle As String
    flags As Long
    nFileOffset As Integer
    nFileExtension As Integer
    lpstrDefExt As String
    lCustData As Long
    lpfnHook As Long
    lpTemplateName As String
End Type

Dim OFName As OPENFILENAME

' declaracion de las apis
Private Declare Function GetOpenFileName Lib "comdlg32.dll" Alias
"GetOpenFileNameA" (pOpenfilename As OPENFILENAME) As Long

Private Function ShowOpen() As String
    ' poner el tamaño de la estructura
    OFName.lStructSize = Len(OFName)
    ' poner la ventana propia
    OFName.hwndOwner = Me.hWnd
    ' poner la instancia de la aplicacion
    OFName.hInstance = App.hInstance
    ' poner las extenciones
    OFName.lpstrFilter = "Text Files (*.txt)" + Chr$(0) + "*.txt" + Chr$(0) +
"All Files (*.*)" + Chr$(0) + "*.*" + Chr$(0)
    ' crear el buffer
    OFName.lpstrFile = Space$(254)
    ' decir el tamaño maximo de caracteres
    OFName.nMaxFile = 255
    ' crearle el buffer
    OFName.lpstrFileTitle = Space$(254)

    ' decir el tamaño maximo de caracteres
    OFName.nMaxFileTitle = 255
    ' poner un directorio inicial
    OFName.lpstrInitialDir = "C:\"
    ' poner el titulo a la caja de dialogo
    OFName.lpstrTitle = "Abrir archivo"
    ' sin banderas
    OFName.flags = 0

    ' mostrar la caja de dialogo
    If GetOpenFileName(OFName) Then
        ShowOpen = Trim$(OFName.lpstrFile)
    Else
        ShowOpen = ""
    End If
End Function
```

```
Private Sub abrir_Click()
 Dim sFile As String
    sFile = ShowOpen
    If sFile <> "" Then
        MsgBox "Seleccionaste este archivo: " + sFile
    Else
        MsgBox "Seleccionaste Cancelar"
    End If
End Sub
```

Este código se podría decir que es casi idéntico al de la API para abrir la caja de dialogo de guardar, pero con sus diferencias, comenzando con la explicación, primeramente se tiene una estructura llamada OPENFILENAME, que creará la forma para recibir y mandar los parámetros en la interacción con la caja de dialogo, después la definición de la API, ahora bien viene una función llamada ShowOpen(), en la cual mandaremos cada parámetro necesario para la visualización de la caja de dialogo, desde el nombre de la ventana, el directorio raíz con el cual comenzar, el filtrado de los archivos para ver, para que no muestre todos, etc. ahora se coloco en if con la condición de que si en la caja no se selecciono ningún archivo, la cadena de resultado sea "" y para que no marque error, este valor se pasa al evento del botón, que no hará mas que recibir la cadena con la dirección del archivo seleccionado para abrir y lo pondrá dentro de un mensaje.
De caso contrario hará lo mismo indicando que no se selecciono ninguno, en caso de arrepentimiento del usuario, como modo preventivo al programador para atrapar el error generado al pulsar este botón.

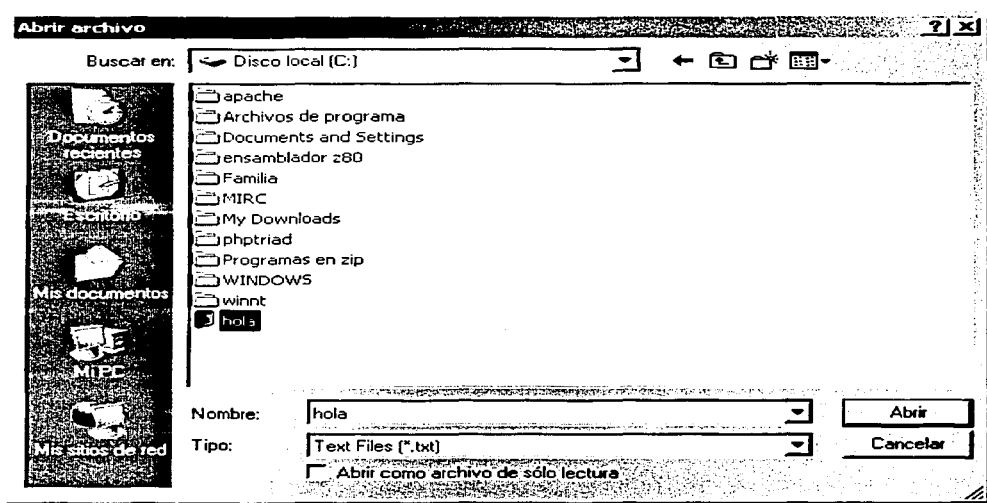

Figura 4.1 Ventana de la Caja de dialogo "Abrir archivo"

4.1.2 GetSaveFileName

Declare Function GetSaveFileName Lib "comdlg32.dll" Alias "GetSaveFileNameA" (pOpenfilename As OPENFILENAME) As Long

Información

Esta función crea una caja de dialogo para permitir al usuario escoger entre discos duros, carpetas y archivos los cuales seleccionar para poder guardar

Sistema Operativo

Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros

Lpofn. Apuntador a la estructura de Openfilename que contiene la información usada por la caja de dialogo. Cuando GetSaveFileName regresa esta contiene la información seleccionada por el usuario.

Código

Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "guardar" y agregar el código correspondiente

```
' dentro de la forma
Option Explicit

' estrcuturas
Private Type OPENFILENAME
     lStructSize As Long
     hwndOwner As Long
     hInstance As Long
     lpstrFilter As String
     lpstrCustomFilter As String
     nMaxCustFilter As Long
     nFilterIndex As Long
     lpstrFile As String
     nMaxFile As Long
     lpstrFileTitle As String
     nMaxFileTitle As Long
     lpstrInitialDir As String
     lpstrTitle As String
     flags As Long
     nFileOffset As Integer
     nFileExtension As Integer
     lpstrDefExt As String
     lCustData As Long
     lpfnHook As Long
     lpTemplateName As String
End Type

' declaracion de la api
Private Declare Function GetSaveFileName Lib "comdlg32.dll" Alias
"GetSaveFileNameA" (pOpenfilename As OPENFILENAME) As Long
```

```
Dim OFName As OPENFILENAME

Private Function ShowSave() As String
  ' poner el tamaño de la estructura
     OFName.lStructSize = Len(OFName)
     ' poner la ventana propia
     OFName.hwndOwner = Me.hWnd
     ' poner la instancia de la aplicacion
     OFName.hInstance = App.hInstance
     ' poner las extenciones
     OFName.lpstrFilter = "Text Files (*.txt)" + Chr$(0) + "*.txt" + Chr$(0) +
"All Files (*.*)" + Chr$(0) + "*.*" + Chr$(0)
     ' crear el buffer
     OFName.lpstrFile = Space$(254)
     ' decir el tamaño maximo de caracteres
     OFName.nMaxFile = 255
     ' crearle el buffer
     OFName.lpstrFileTitle = Space$(254)
     ' decir el tamaño maximo de caracteres
     OFName.nMaxFileTitle = 255
     ' poner un directorio inicial
     OFName.lpstrInitialDir = "C:\"
     ' poner el titulo a la caja de dialogo
     OFName.lpstrTitle = "Guardar archivo como..."
     ' sin banderas
     OFName.flags = 0

     ' mostrar la caja de dialogo
     If GetSaveFileName(OFName) Then
          ShowSave = Trim$(OFName.lpstrFile)
     Else
          ShowSave = ""
     End If
End Function

Private Sub guardar_Click()
     Dim sFile As String
     sFile = ShowSave

     If sFile <> "" Then
          MsgBox "Seleccionaste guardar el archivo: " + sFile
     Else
          MsgBox "Seleccionaste cancelar"
     End If
End Sub
```

Bien, la primera parte de la explicación de este código es muy parecido al de la API para abrir un archivo, la declaración de la constante, la estructura igual de OPENFILENAME (ya que son exactas las propiedades de las cajas de dialogo abrir y guardar) y la API, una vez declaradas viene la función ShowSave() que es prácticamente idéntica a la de abrir archivo, en la cual se dan los parámetros necesarios para abrir la caja de dialogo, cada línea es comentada dentro del mismo código.

Por último el evento clic del botón, que abrirá como su acción dicha caja de dialogo, para dar como resultado un mensaje indicando la cadena completa del archivo seleccionado, que se guardaría, y en caso contrario la cancelación del botón "cancelar".

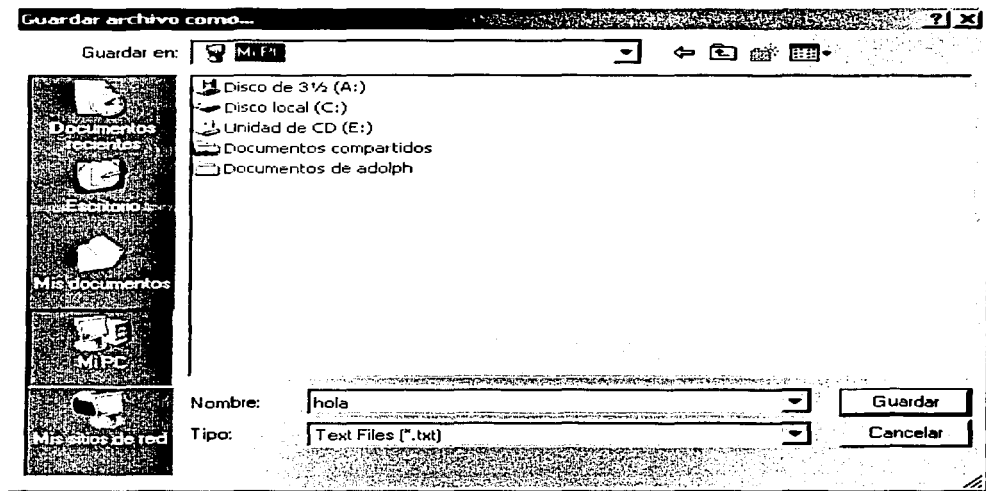

Figura 4.2 Ventana de la Caja de dialogo "Guardar archivo como"

4.1.3 CHOOSECOLOR

```
Declare Function ChooseColor Lib "comdlg32.dll" Alias "ChooseColorA" (pChoosecolor As CHOOSECOLOR) As Long
```

Información
Esta función crea una caja de dialogo para permitir al usuario escoger un color en particular o personalizado.

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
lpcc. Apuntador a la estructura de CHOOSECOLOR que contiene la información usada para la inicialización de la caja de dialogo.

Código
Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows

- Crear un botón simple en la forma llamado "color", un objeto Picture1 y poner su propiedad de apariencia a tipo 0-flat y agregar el código correspondiente

```
' en la forma
Option Explicit

' estrcutura para el color
Private Type CHOOSECOLOR
    lStructSize As Long
    hwndOwner As Long
    hInstance As Long
    rgbResult As Long
    lpCustColors As String
    flags As Long
    lCustData As Long
    lpfnHook As Long
    lpTemplateName As String
End Type

' declaracion de apis
Private Declare Function CHOOSECOLOR Lib "comdlg32.dll" Alias "ChooseColorA"
(pChoosecolor As CHOOSECOLOR) As Long
Dim CustomColors() As Byte


Private Sub Form_Load()
' pequeño ciclo para segurarnos que al escojer un color
' este no blocke a los restantes en caso de escojer otro nuevamente
' utilizando el LBond y Ubond para unirlos y actualizarlos
ReDim CustomColors(0 To 16 * 4 - 1) As Byte
Dim i As Integer
    For i = LBound(CustomColors) To UBound(CustomColors)
        CustomColors(i) = 0
    Next i
End Sub
```

Primeramente se declara la estructura tipo CHOOSECOLOR y la API, ahora hecho esto al iniciar la forma es necesario colocar un pequeño ciclo para actualizar y limpiar, ya que de no hacerlo una vez que seleccionemos un color y volvamos a intentar abrir la caja de dialogo de color marcara error.

```
Private Sub color_Click()
Dim NewColor As Long
NewColor = ShowColor
' en caso de no seleccionar color no entra
If NewColor <> -1 Then
    Picture1.BackColor = NewColor
Else
    Exit Sub
End If
End Sub

Private Function ShowColor() As Long
    Dim cc As CHOOSECOLOR
    Dim Custcolor(16) As Long
    Dim lReturn As Long
```

```
    ' pone el tamaño de la estrcutura
    cc.lStructSize = Len(cc)
    ' se coloca el mismo padre
    cc.hwndOwner = Me.hWnd
    ' pone la instancia de la aplicacion
    cc.hInstance = App.hInstance
    ' colocar los colores
    cc.lpCustColors = StrConv(CustomColors, vbUnicode)
    ' sin banderas extra
    cc.flags = 0

    ' mostrar la caja de dialogo de colores
    If CHOOSECOLOR(cc) <> 0 Then
        ShowColor = cc.rgbResult
        CustomColors = StrConv(cc.lpCustColors, vbFromUnicode)
    Else
        ShowColor = -1
    End If
End Function
```

La función ShowColor() es mandada llamar por el botón color(), dentro de esta función se necesita primeramente colocar los valores necesarios para poder visualizar la caja de dialogo como lo es el handle de la forma padre, la propia instancia, entre otras, cada línea es comentada en el código. Una vez hecho esto se llama la función para abrir la caja de dialogo y se capturan los cambios hechos por el usuario, se pasa el valor del color escogido para la comparación del if, en caso de no haber escogido uno, o dar al botón cancelar se pone un -1, para que dentro del if del botón cancelar se pueda atrapar el error y se termine sin hacer cambios de lo contrario, el if del botón, aplicará el cambio al objeto Picture1, cabe mencionar que lo he querido hacer de un modo simple para su entendimiento, pero se puede pasar este valor del color seleccionado a múltiples objetos como lo es la misma forma, o texto dentro de una caja, entre muchas mas.

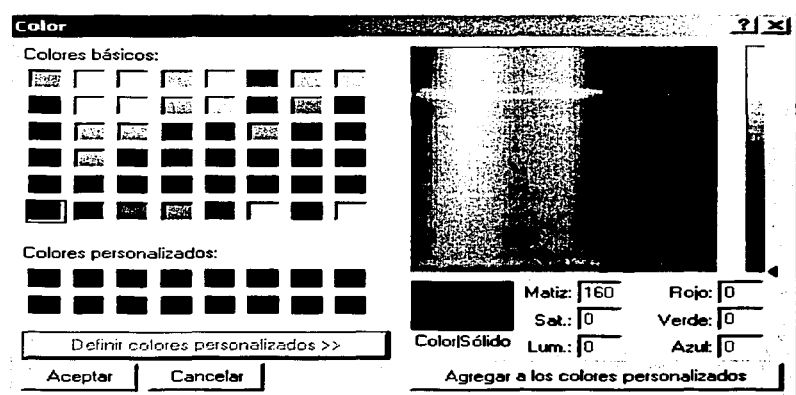

Figura 4.3 Ventana de la Caja de dialogo "Color"

4.1.4 PAGESETUPDLG

Declare Function PageSetupDlg Lib "comdlg32.dll" Alias "PageSetupDlgA" (pPagesetupdlg As PAGESETUPDLG) As Long

Información

Esta función crea la caja de diálogo de propiedades de página para impresión, para permitir al usuario modificar, el tipo, orientación tamaño de hoja antes de imprimir su trabajo.

Sistema Operativo

Windows NT 5.1, Windows 9x, Windows 2000/XP

Parámetros

Lppsd. Apunta a la estructura que contiene la información utilizada para inicializar la caja de diálogo.

Código

Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "Command1" y agregar el código correspondiente

```
' en la forma
Option Explicit


' estructuras para la propiedades de la pagina necesarias
Private Type POINTAPI
    x As Long
    y As Long
End Type


Private Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
End Type


Private Type PAGESETUPDLG
    lStructSize As Long
    hwndOwner As Long
    hDevMode As Long
    hDevNames As Long
    flags As Long
    ptPaperSize As POINTAPI
    rtMinMargin As RECT
    rtMargin As RECT
    hInstance As Long
    lCustData As Long
```

```
    lpfnPageSetupHook As Long
    lpfnPagePaintHook As Long
    lpPageSetupTemplateName As String
    hPageSetupTemplate As Long
End Type

' declaracion d ela api
Private Declare Function PAGESETUPDLG Lib "comdlg32.dll" Alias
"PageSetupDlgA" (pPagesetupdlg As PAGESETUPDLG) As Long

Private Sub Command1_Click()
    ' mandara llamar la funcion tipo long que regresara el valor entero
    If ShowPageSetupDlg = 0 Then
        MsgBox "Se han hecho cambios"
    Else
        MsgBox "Se ha presionado cancelar"
    End If
End Sub

Private Function ShowPageSetupDlg() As Long
Dim m_PSD As PAGESETUPDLG
' poner el tamaño dela estructura
m_PSD.lStructSize = Len(m_PSD)
' colocarla o abrirla el la propia ventana
m_PSD.hwndOwner = Form1.hWnd
' poner la instancia de la aplicacion
m_PSD.hInstance = App.hInstance
' sin banderas extra
m_PSD.flags = 0

' mostrar la caja de dialogo
If PAGESETUPDLG(m_PSD) Then
   ' hacer cambios o dar al boton aceptar
   ShowPageSetupDlg = 0
Else
   ' no hacer cambios o dar al boton cancelar
   ShowPageSetupDlg = -1
End If
End Function
```

Una vez declaradas las estructuras que se necesitarán para abrir la caja de diálogo y su API correspondiente, dentro del botón, se manda llamar ShowPageSetupDlg() y si su resultado es cero entonces el usuario ha hecho cambios y serán aplicados, de lo contrario se atrapara el error si se ha dado clic al botón de cancelar.

Una vez dentro de ShowPageSetupDlg() se declaran las variables, he comentado casi línea por línea para un mejor entendimiento, pero sigue las estructuras comunes o patrones de las cajas de diálogo antes vistas que son, la creación de la instancia y el emparentarlo con la forma desde la que se abrió, para dar paso a las propiedades con las que serán llenadas y después su apertura y llenado de la variable que regresara los valores que el usuario haya modificado.

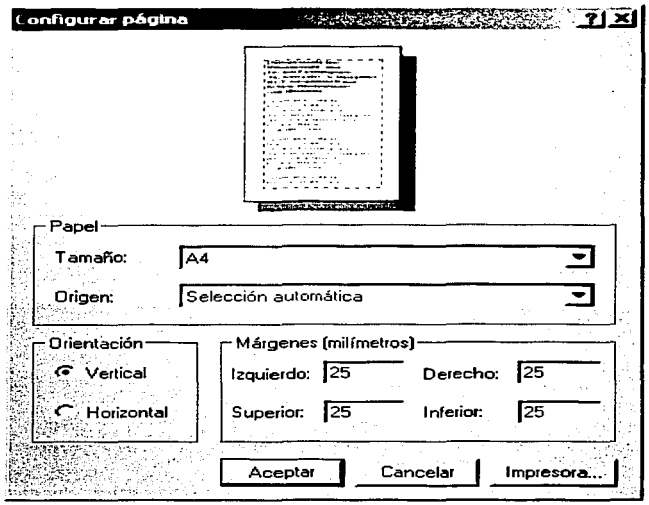

Figura 4.4 Ventana de la Caja de dialogo "Configurar página"

4.1.5 CHOOSEFONT

```
Declare Function ChooseFont Lib "comdlg32.dll" Alias "ChooseFontA" (pChoosefont As CHOOSEFONT) As Long
```

Información
Esta función crea una caja de dialogo para permitir al usuario escoger un tipo de fuente, tamaño, color, y otras caracteristicas en particular.

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
lpcf. Apuntador a la estructura de CHOOSEFONT que contiene la información usada para la inicialización de la caja de diálogo.

4.1.6 SHBrowseForFolder

```
Declare Function SHBrowseForFolder Lib "shell32" (lpbi As BrowseInfo) As Long
```

Información
Esta función muestra la caja de dialogo que permite al usuario seleccionar una carpeta en el sistema, a modo de búsqueda

Sistema Operativo
Windows NT 4, Windows 9x, Windows 2000/XP

Parámetros
lpbi. Apunta a una estructura tipo BROWSEINFO, que contiene información utilizada para mostrar la caja de diálogo

4.2 MEMORIA

La memoria una parte muy importante de toda computadora tan importante como el procesador en si, sin ella no se podrían cargar los programas, las siguientes APIS son unas de las cuales manejan la memoria de una manera directa con el Sistema Operativo y de manera eficiente, estas APIS no son todas, pero son las necesarias para poder explicar el funcionamiento de las memoria en un ejercicio acompañadas de la API CHOOSEFONT, para poder mostrar la caja de dialogo de la fuente y modificar así un texto a modo de ejemplo, este se complementa con la API de fuente en cajas de diálogo.

4.2.1 CopyMemory
Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (pDst As Any, pSrc As Any, ByVal ByteLen As Long)

Información
Esta función copia un bloque de memoria de una locación a otra.

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
Destination. Apunta al comienzo de la dirección del bloque copiado
Source. Apunta al comienzo de la dirección del bloque de memoria a copiar
Length. Especifica el tamaño en bytes del bloque de memoria a copiar

4.2.2 GlobalAlloc
Declare Function GlobalAlloc Lib "kernel32" Alias "GlobalAlloc" (ByVal wFlags As Long, ByVal dwBytes As Long) As Long

Información
Esta función aloja un número especificado de bytes a memoria.

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
uFlags. Especifica como alojar la memoria
dwBytes. Especifica el número de bytes para alojar

4.2.3 GlobalFree
Declare Function GlobalFree Lib "kernel32" Alias "GlobalFree" (ByVal hMem As Long) As Long

Información
Esta función libera una locación específica de un objeto en la memoria e inhabilita su handle.

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hMem. Identifica el objeto en memoria

4.2.4 GlobalLock
Declare Function GlobalLock Lib "kernel32" Alias "GlobalLock" (ByVal hMem As Long) As Long

Información
Esta función encierra un objeto en la memoria y regresa un apuntador del primer byte del objeto encerrado. Este objeto no puede ser movido de la memoria o desechado.

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hMem. Identifica el objeto en memoria.

4.2.5 GlobalUnlock
Declare Function GlobalUnlock Lib "kernel32" Alias "GlobalUnlock" (ByVal hMem As Long) As Long

Información
Esta función decrementa el contador asociado con el objeto a memoria que fue alojado con la bandera o constante GMEM_MOVEABLE.

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hMem. Identifica el objeto de memoria global.

Una vez conocidas las APIS para el manejo de la memoria de una manera básica crear el proyecto y su código, para explicarlo de manera fraccionada.

Código
Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "Command1", un objeto Text1 y poner su propiedad de multilínea activada y de scrolls a tipo 3-both y agregar el código correspondiente

```
' en la forma [9]
Option Explicit

' constantes
Const FW_NORMAL = 400
Const DEFAULT_CHARSET = 1
Const OUT_DEFAULT_PRECIS = 0
Const CLIP_DEFAULT_PRECIS = 0
Const DEFAULT_QUALITY = 0
Const DEFAULT_PITCH = 0
Const FF_ROMAN = 16
Const GMEM_MOVEABLE = &H2
Const GMEM_ZEROINIT = &H40
Const CF_PRINTERFONTS = &H2
Const CF_SCREENFONTS = &H1
Const CF_BOTH = (CF_SCREENFONTS Or CF_PRINTERFONTS)
Const CF_EFFECTS = &H100&
Const CF_FORCEFONTEXIST = &H10000
Const CF_INITTOLOGFONTSTRUCT = &H40&
Const CF_LIMITSIZE = &H2000&
Const REGULAR_FONTTYPE = &H400
Const LF_FACESIZE = 32

' estrcutura de la fuente
Private Type LOGFONT
        lfHeight As Long
        lfWidth As Long
        lfEscapement As Long
        lfOrientation As Long
        lfWeight As Long
        lfItalic As Byte
        lfUnderline As Byte
        lfStrikeOut As Byte
        lfCharSet As Byte
        lfOutPrecision As Byte
        lfClipPrecision As Byte
```

[9] http://www.vbworld.com/

```
        lfQuality As Byte
        lfPitchAndFamily As Byte
        lfFaceName As String * 31
End Type

' estrcutura de la variable tipo fuente
Private Type CHOOSEFONT
        lStructSize As Long
        hwndOwner As Long
        hDC As Long
        lpLogFont As Long
        iPointSize As Long
        flags As Long
        rgbColors As Long
        lCustData As Long
        lpfnHook As Long
        lpTemplateName As String
        hInstance As Long
        lpszStyle As String
        nFontType As Integer
        MISSING_ALIGNMENT As Integer
        nSizeMin As Long
        nSizeMax As Long
End Type

' apis utilizadas
Private Declare Function CHOOSEFONT Lib "comdlg32.dll" Alias "ChooseFontA"
(pChoosefont As CHOOSEFONT) As Long
Private Declare Function GlobalUnlock Lib "kernel32" (ByVal hMem As Long) As
Long
Private Declare Function GlobalAlloc Lib "kernel32" (ByVal wFlags As Long,
ByVal dwBytes As Long) As Long
Private Declare Function GlobalFree Lib "kernel32" (ByVal hMem As Long) As
Long
Private Declare Function GlobalLock Lib "kernel32" (ByVal hMem As Long) As
Long
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (hpvDest
As Any, hpvSource As Any, ByVal cbCopy As Long)

' variables globales
Dim nom As String, tam As Integer, color As Long
```

Aquí primeramente como siempre se tienen que definir las constantes, variables y estructuras que se utilizaran en el transcurso del programa, las primeras constantes son sumamente necesarias para utilizar las posibilidades de las características que requerirá la caja de dialogo, seguidas de la estructura de la fuente y variable tipo fuente, después las API a utilizar y variables globales necesarias en todo el programa.

```
Private Sub Command1_Click()
nom = ShowFont ' paso el nombre de la fuente
If nom = "" Then Exit Sub  ' en caso de error
' pongo el nombre, tamaño y color a la caja de texto en sus parametros
Text1.Font = nom
Text1.FontSize = tam / 10 ' dividir entre 10
Text1.ForeColor = color
End Sub
```

Este es el botón de la caja, primero le asigno a una variable el resultado de la llamada de la función ShowFont que regresa un tipo cadena, si esta vacía termino saliéndome, ya que este es una detección de error en caso de que el usuario presione el botón cancelar de la caja de diálogo, después asigno a la caja de texto las propiedades obtenidas de las selecciones que realizo el usuario en la caja.

```
' funcion de la fuente que regresara en cadena el nombre de la fuente
Private Function ShowFont() As String
    Dim cf As CHOOSEFONT
    Dim lfont As LOGFONT, hMem As Long, pMem As Long
    Dim fontname As String, retval As Long
    ' colocar las caracteristicas de la fuente en la ventana por defalta
    lfont.lfHeight = 0   ' determinar el peso
    lfont.lfWidth = 0   ' determinar el ancho
    lfont.lfEscapement = 0   ' angulo de la base de linea
    lfont.lfOrientation = 0   ' angulo del vector
    lfont.lfWeight = FW_NORMAL   ' modo normal del tipo
    lfont.lfCharSet = DEFAULT_CHARSET   ' usar el caracter por defalta
    lfont.lfOutPrecision = OUT_DEFAULT_PRECIS   ' valor de falta
    lfont.lfClipPrecision = CLIP_DEFAULT_PRECIS   ' valor de falta
    lfont.lfQuality = DEFAULT_QUALITY   ' valor de falta
    lfont.lfPitchAndFamily = DEFAULT_PITCH Or FF_ROMAN   ' valor de falta
    lfont.lfFaceName = "Times New Roman" & vbNullChar   ' la cadena debe
terminar el nulo

    ' Crear el blocke de memoria el cual actuara como LOGFONT en la
estructura del buffer
    hMem = GlobalAlloc(GMEM_MOVEABLE Or GMEM_ZEROINIT, Len(lfont))
    pMem = GlobalLock(hMem)   ' capturar y encerrar el puntero
    CopyMemory ByVal pMem, lfont, Len(lfont)   ' copiar el conternido de la
estrctura en un blocke

    ' Inicializar la caja de dialogo
    cf.lStructSize = Len(cf)   ' tamaño dela estrcutura
    cf.hwndOwner = Form1.hWnd   ' en la ventana Form1 es abierta esta caja de
dialogo fuente
    cf.hDC = Printer.hDC   ' contexto por defalta
    cf.lpLogFont = pMem   ' apuntador a LOGFONT blocke de buffer de memoria
    cf.iPointSize = 120   ' 12 puntos
    cf.flags = CF_BOTH Or CF_EFFECTS Or CF_FORCEFONTEXIST Or
CF_INITTOLOGFONTSTRUCT Or CF_LIMITSIZE
    cf.rgbColors = RGB(0, 0, 0)   ' color negro
    cf.nFontType = REGULAR_FONTTYPE   ' fuente normal, no negrita, etc
    cf.nSizeMin = 10   ' tamaño minimo
    cf.nSizeMax = 72   ' tamaño maximo

    ' Llamada a la funcion, si es exitosa copia la estructura LOGFONT
    ' y colocar los atriburos que el usuario selecciono
    retval = CHOOSEFONT(cf)   ' abrir la caja de dialogo
    If retval <> 0 Then   ' si es exitosa, entra
        CopyMemory lfont, ByVal pMem, Len(lfont)   ' copia la memoria
        ' copia el nombre de la cadena del nombre de la fuente
        ShowFont = Left(lfont.lfFaceName, InStr(lfont.lfFaceName, vbNullChar)
- 1)
```

```
        tam = cf.iPointSize ' poner el tamaño de la fuente
        color = cf.rgbColors ' poner el color del a fuente seleccionada
    End If
    ' eliminar la memoria hehca anteriormente, eliminarla con o sin ser
exitosa
    retval = GlobalUnlock(hMem)   ' destruye el puntero
    retval = GlobalFree(hMem)   ' libera la memoria
End Function
```

Este código se ve un poco complicado pero es sencillo hasta cierto punto, la función ShowFont la documente casi por línea para un mejor entendimiento, esta función esta dividida por partes en forma de secuencia, primero la declaración de las variables, algo lógico, seguida de un bloque de instrucciones que no son mas que las características de la caja de diálogo, utilizando las partes de la estructura y trabajando con las constantes, que no son mas que valores enteros, una vez hecho esto se utilizan las APIS de memoria para crear las estructuras temporales que necesitaremos para las propiedades de la caja.

Una vez creados los punteros y bloques de memoria, que están emparentados con la forma1 en este caso y las variables de estructura, se dan las características de la variable cf con la que abrirá la caja de diálogo y colocara sus valores, una vez que esta disponible al usuario este hará modificaciones, hechas estas se asignaran al cerrar la caja nuevamente a la variable cf, adoptando las características que el usuario deseo, para terminar si la caja fue exitosa, se copiara de memoria y transformara a cadena el nombre de la fuente y sus características serán pasadas a las variables globales, para poder ser aplicadas en el código del botón, ya obtenidas las características seleccionadas por el usuario se eliminará de memoria el objeto creado anteriormente, entre o no entre a visualizar la caja, ya que mas atrás en el código fue creado.

La siguiente API será utilizada en el ejemplo de encontrar o verificar que un archivo o carpeta exista o no, más adelante en la sección miscelánea.

4.2.6 CoTaskMemFree

```
Declare Sub CoTaskMemFree Lib "ole32.dll" (ByVal hMem As Long)
```

Información
Esta función libera un bloque de memoria previamente colocado a través de una llamada de las funciones CoTaskMemAlloc o CoTaskMemRealloc.

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
Pv. Apunta al bloque de memoria ha ser liberado

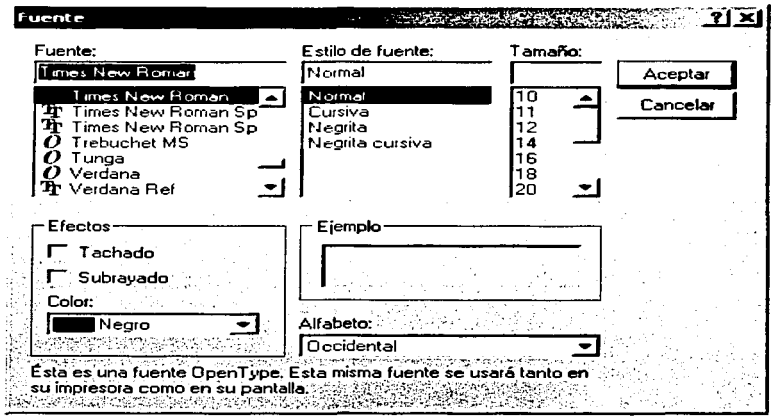

Figura 4.5 Ventana de la Caja de dialogo "Fuente"

4.3 ARCHIVOS

Los archivos son sumamente importantes en todo Sistema Operativo, hay muy variados y de diferentes características, desde los específicos para base de datos, de tipo audio, video, binarios para imágenes, y de tipo texto, en diferentes formatos.

En las siguientes APIS se mostrará como abrir, crear, leer y guardar un archivo tipo texto sencillo, para el conocimiento de estas APIS, existen muchas más, pero estas son las más utilizadas y necesarias.

Cabe mencionar que existen muchos controles tipo ActiveX, módulos y otros que están a la venta, para una manera mucho mas fácil de utilizarlos y entenderlos, el mismo Visual Basic en sus diferentes versiones incluye estos controles para poder manejarlos rápidamente, pero con una gran desventaja.

No son modificables directamente a la conveniencia del programador y consumen muchos mas ciclos del procesador, haciendo la aplicación que ocupe mas tiempo, mas espacio y memoria, queda abierto a consideración del programador, pero recomiendo que si se desea hacer una aplicación comercial y robusta se utilicen las librerías de Windows directamente, que utilizar controles lentos, pesados y que además se cobra a parte por su uso.

4.3.1 CreateFile

Declare Function CreateFile Lib "kernel32" Alias "CreateFileA" (ByVal lpFileName As String, ByVal dwDesiredAccess As Long, ByVal dwShareMode As Long, lpSecurityAttributes As SECURITY_ATTRIBUTES, ByVal dwCreationDisposition As Long, ByVal dwFlagsAndAttributes As Long, ByVal hTemplateFile As Long) As Long

Información
Esta función crea o abre diferentes objetos y regresa el handle que es usada para acceder estos objetos, como lo pueden ser, carpetas (abrirlas), lotes de correo, archivos, pipas, dispositivos de disco (bajo NT).

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
lpFileName. Apunta a una cadena terminada en nulo que especifica el tipo de objeto a crear o abrir
dwDesiredAccess. Especifica el tipo de acceso al objeto
dwShareMode. Coloca un bit de bandera que especifica como el objeto puede ser compartido
lpSecurityAttributes. Apunta a la estructura SECURITY_ATTRIBUTES que determina si el handle regresado puede ser heredado por un proceso hijo, el andel no puede ser heredado.
dwCreationDistribution. Especifica que accion tomar si el archivo existe o no existe tal
dwFlagsAndAttributes. Especifica los atributos del archivo y sus banderas
hTemplateFile. Especifica el handle con GENERIC_READ para el acceso al archivo, para Windows 95 este debe de ser nulo.

4.3.2 ReadFile

Declare Function ReadFile Lib "kernel32" Alias "ReadFile" (ByVal hFile As Long, lpBuffer As Any, ByVal nNumberOfBytesToRead As Long, lpNumberOfBytesRead As Long, lpOverlapped As OVERLAPPED) As Long

Información
Esta función lee datos de un archivo comenzando en la posición indicada por el puntero del archivo

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hFile. Identifica el archivo ha ser leeido
lpBuffer. Apunta al buffer que recibe la información leída del archivo
nNumberOfBytesToRead. Especifica el numero de bytes ha ser leídos del archivo
lpNumberOfBytesRead. Apunta al número de bytes leídos
lpOverlapped. Apunta a una estructura OVERLAPPED

4.3.3 WriteFile

Declare Function WriteFile Lib "kernel32" Alias "WriteFile" (ByVal hFile As Long, lpBuffer As Any, ByVal nNumberOfBytesToWrite As Long, lpNumberOfBytesWritten As Long, lpOverlapped As OVERLAPPED) As Long

Información
Esta función escribe información a un archivo y la comienza a escribir donde indique el puntero del archivo

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hFile. Identifica el archivo ha ser escrito
lpBuffer. Apunta al buffer que contiene la información ha ser escrita en el archivo
nNumberOfBytesToWrite. Especifica el numero de bytes ha ser escritos en el archivo
lpNumberOfBytesWritten. Apunta al número de bytes escritos por la llamada de esta función
lpOverlapped. Apunta a una estructura OVERLAPPED

4.3.4 SetFilePointer

Declare Function SetFilePointer Lib "kernel32" Alias "SetFilePointer" (ByVal hFile As Long, ByVal lDistanceToMove As Long, lpDistanceToMoveHigh As Long, ByVal dwMoveMethod As Long) As Long

Información
Esta función mueve el puntero de archivo a un archivo abierto

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hFile. Identifica al archivo al cual su puntero es movido
lDistanceToMove. Especifica el número de bytes a mover del puntero
lpDistanceToMoveHigh. Apunta a un orden alto de palabra para mover
dwMoveMethod. Especifica el punto de comienzo para mover el puntero

4.3.5 CloseHandle

Declare Function CloseHandle Lib "kernel32" Alias "CloseHandle" (ByVal hObject As Long) As Long

Información
Esta función cierra el handle de un objeto abierto

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hObject. Identifica el handle de un objeto abierto

Código
Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear una caja de texto llamada "Text1", poner propiedades de multilínea, activado, scrollbar 3-both y agregar el código correspondiente

```
' en la forma [10]
Option Explicit
' estructura
Private Type OVERLAPPED
   Internal As Long
   InternalHigh As Long
   Offset As Long
   OffsetHigh As Long
   hEvent As Long
End Type

'SetFilePointer dwMoveMethod sus constantes
Private Const FILE_BEGIN = 0
Private Const FILE_CURRENT = 1
Private Const FILE_END = 2
'CreateFile dwDesiredAccess sus constantes
Private Const GENERIC_READ = &H80000000
Private Const GENERIC_WRITE = &H40000000
'CreateFile dwShareMode sus constantes
Private Const FILE_SHARE_READ = &H1
Private Const FILE_SHARE_WRITE = &H2
'CreateFile dwCreationDisposition sus constantes
Private Const CREATE_ALWAYS = 2
Private Const CREATE_NEW = 1
Private Const OPEN_ALWAYS = 4
Private Const OPEN_EXISTING = 3
Private Const TRUNCATE_EXISTING = 5

'CreateFile dwFlagsAndAttributes sus constantes
Private Const FILE_ATTRIBUTE_ARCHIVE = &H20
Private Const FILE_ATTRIBUTE_HIDDEN = &H2
Private Const FILE_ATTRIBUTE_NORMAL = &H80
Private Const FILE_ATTRIBUTE_READONLY = &H1
Private Const FILE_ATTRIBUTE_SYSTEM = &H4
Private Const FILE_FLAG_DELETE_ON_CLOSE = &H4000000
Private Const FILE_FLAG_NO_BUFFERING = &H20000000
Private Const FILE_FLAG_OVERLAPPED = &H40000000
Private Const FILE_FLAG_POSIX_SEMANTICS = &H1000000
Private Const FILE_FLAG_RANDOM_ACCESS = &H10000000
Private Const FILE_FLAG_SEQUENTIAL_SCAN = &H8000000
Private Const FILE_FLAG_WRITE_THROUGH = &H80000000
```

[10] http://www.vbworld.com/

```
' apis de archivos
Private Declare Function CreateFile Lib "kernel32.dll" Alias "CreateFileA"
(ByVal lpFileName As String, ByVal dwDesiredAccess As Long, ByVal dwShareMode
As Long, lpSecurityAttributes As Any, ByVal dwCreationDisposition As Long,
ByVal dwFlagsAndAttributes As Long, ByVal hTemplateFile As Long) As Long

Private Declare Function ReadFile Lib "kernel32.dll" (ByVal hFile As Long,
lpBuffer As Any, ByVal nNumberOfBytesToRead As Long, lpNumberOfBytesRead As
Long, lpOverlapped As Any) As Long

Private Declare Function WriteFile Lib "kernel32.dll" (ByVal hFile As Long,
lpBuffer As Any, ByVal nNumberOfBytesToWrite As Long, lpNumberOfBytesWritten
As Long, lpOverlapped As Any) As Long

Private Declare Function SetFilePointer Lib "kernel32.dll" (ByVal hFile As
Long, ByVal lDistanceToMove As Long, lpDistanceToMoveHigh As Long, ByVal
dwMoveMethod As Long) As Long

Private Declare Function CloseHandle Lib "kernel32.dll" (ByVal hObject As
Long) As Long

Declaración de las constantes y su uso, como de la estructura OVERLAPPED
y APIs utilizadas en este ejemplo de archivos.

Private Sub Form_Load()
   Dim Retval As Long, hFile As Long
   Dim Puffer As String * 1024, RealReaded As Long

   ' abrir el archivo
   hFile = CreateFile("c:\hola.txt", GENERIC_READ, FILE_SHARE_READ, ByVal 0&,
OPEN_ALWAYS, FILE_ATTRIBUTE_ARCHIVE, 0&)
   If hFile = 0 Then
      MsgBox "Error de lectura"
      Exit Sub
   End If
   ' comenzar la caja limpia
   Text1.Text = ""
   ' cilo de repeticion para escribir los caracteres
   Do
      Retval = ReadFile(hFile, ByVal Puffer, Len(Puffer), RealReaded, ByVal
0&)
      Text1.Text = Text1.Text & Left$(Puffer, RealReaded)
      DoEvents
   Loop Until RealReaded < Len(Puffer)
   ' cerrar
   CloseHandle hFile
End Sub
```

Al inicio del programa manda llamar la forma, es necesario crear si se quiere un archivo tipo texto en el directorio raíz C:\ , sino se creara al iniciar el programa llamado "hola.txt", una vez abierto el archivo o creado limpia la caja de texto, en caso de no haber existido ningún error de tipo de lectura del archivo, hecho esto entra a un ciclo el cual escribirá del buffer de lectura y donde se encuentre el puntero, cuando termine saldrá y terminará con la carga de la forma sin antes destruir el handle utilizado poco antes.

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
   Dim Retval As Long, hFile As Long, MsgRet As Long
   Dim Puffer As String * 1024, RealWritten As Long, i As Integer

   ' preguntar si se desea guardar
   MsgRet = MsgBox("¿Guardar el archivo?", vbQuestion + vbYesNo, "Pregunta")
   If MsgRet = vbNo Then Exit Sub

   ' escribir el archivo
   hFile = CreateFile("c:\hola.txt", GENERIC_WRITE, FILE_SHARE_READ, ByVal
0&, TRUNCATE_EXISTING, FILE_ATTRIBUTE_ARCHIVE, 0&)
   If hFile = 0 Then
      MsgBox "Error de escritura"
      Exit Sub
   End If

   ' mover el puntero del archivo al inicio
   Retval = SetFilePointer(hFile, 0&, ByVal 0&, FILE_BEGIN)

   ' escribir el archivo
   If Len(Text1.Text) > 1024 Then
      For i = 0 To Len(Text1.Text) / 1024
         Puffer = Left$(Text1.Text, 1024)
         Text1.Text = Mid$(Text1.Text, 1025)
         Retval = WriteFile(hFile, Puffer, Len(Puffer), RealWritten, ByVal
0&)
      Next i
   End If
   Retval = WriteFile(hFile, ByVal Text1.Text, Len(Text1.Text), RealWritten,
ByVal 0&)

   ' cerrar
   CloseHandle hFile
End Sub
```

Una vez ya abierto el archivo el usuario puede escribir directamente sobre él (caja de texto multilínea), cuando termine, al cerrar el programa se mandara llamar Form_QueryUnload, una vez dentro abra un if con mensaje condicionado de que si se desea guardar el archivo abierto y sus cambios; si se selecciona no, termina el programa y se cierra, de lo contrario se abre nuevamente el archivo hola.txt, se coloca el puntero al inicio del archivo y se entra en una condición del contenido de la caja de texto ha sido modificada entra y escribe en el archivo desde la última posición de carácter que tenga el archivo hola.txt, para no encimar lo que ya se tenía y estropearlo, una vez almacenado en el buffer se escribe directamente al archivo y cierra el handle.

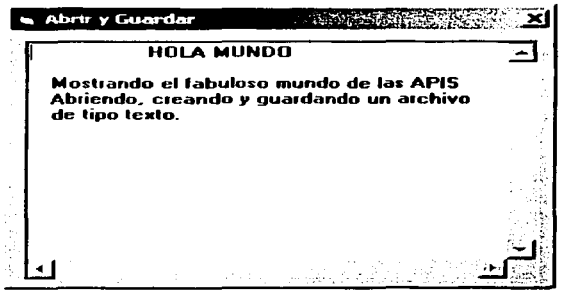

Figura 4.6 Ventana del Programa que abre y guarda archivos en formato tipo texto

La siguiente API será utilizada en el ejemplo de encontrar o verificar que un archivo o carpeta exista o no, más adelante en la sección miscelánea.

4.3.6 OpenFile

```
Declare Function OpenFile Lib "kernel32" Alias "OpenFile" (ByVal lpFileName As String, lpReOpenBuff As OFSTRUCT, ByVal wStyle As Long) As Long
```

Información
Esta función crea, abre, reabre, o borra un archivo.

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
lpFileName. Apunta a una cadena con terminación en nulo que contiene el nombre del archivo ha ser abierto
lpReOpenBuff. Apunta a una estructura tipo OFSTRUCT, que recibe la información de un archivo que será por primera vez abierto
uStyle. Especifica la acción a realizar o tomar

4.4 PORTAPAPELES

"Clipboard" llamado así en idioma inglés y "portapapeles" en español, no es mas que una locación de memoria destinada a utilizarse para realizar movimientos de datos o información en Windows.

Este puede variar en su tamaño y formato, ya que se puede usar de un modo tan sencillo como simplemente seleccionar lo que se desea trabajar, dar clic a la opción copiar, cortar o mover y después pegar, dentro de un mismo programa o entre diferentes, de este modo facilita mucho el trabajo en este Sistema Operativo, pues un texto que seleccione específicamente desde una página de Internet lo puedo copiar fácilmente a un editor de texto como puede ser el bloc de notas o el Word Office de Microsoft, imágenes a un programa como Photoshop de Adobe, archivos, fuentes, muchas cosas, y formatos específicos como elementos de un programa en concreto como Flash mx de Macromedia, para copiar figuras de un archivo a otro abiertos al mismo tiempo y trabajando en ellos.

Las siguientes APIs trabajan el portapapeles, son las mas utilizadas, no son todas, pero en este ejemplo se demuestra como usar estas APIs para la carga de una imagen desde un archivo hasta nuestro programa con ayuda de una API de gráficos, de este modo se puede programar el trabajo especifico y concreto para las necesidades del programa a realizar.

4.4.1 OpenClipboard
Declare Function OpenClipboard Lib "user32" Alias "OpenClipboard" (ByVal hwnd As Long) As Long

Información
Esta función abre el portapapeles para examinarlo y prever que otras aplicaciones modifiquen su contenido

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hWndNewOwner. Identifica la ventana que será asociada con la apertura del portapapeles

4.4.2 EmptyClipboard
Declare Function EmptyClipboard Lib "user32" Alias "EmptyClipboard" () As Long

Información
Esta función vacía el portapapeles y libera el handle

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
No tiene parámetros

4.4.3 CloseClipboard
Declare Function CloseClipboard Lib "user32" Alias "CloseClipboard" () As Long

Información
Esta función cierra el portapapeles

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
No tiene parámetros

4.4.4 SetClipboardData
Declare Function SetClipboardData Lib "user32" Alias "SetClipboardDataA" (ByVal wFormat As Long, ByVal hMem As Long) As Long

Información
Esta función coloca información de cierto formato en el portapapeles

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
uFormat. Especifica el formato del portapapeles
hMem. Identifica la información en el formato especificado

4.4.5 IsClipboardFormatAvailable
Declare Function IsClipboardFormatAvailable Lib "user32" Alias "IsClipboardFormatAvailable" (ByVal wFormat As Long) As Long

Información
Esta función determina si el portapapeles contiene información en el formato especificado

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
format. Especifica un estándar o registro en el formato del portapapeles

4.5 GRÁFICOS

Los gráficos, una parte muy importante de este Sistema Operativo, ya que su fama la obtuvo de los mismos y su nombre, "ventanas" por su apariencia.

Su forma visual y manejo han ayudado mucho al Sistema Operativo para ser del agrado de la mayoría de la población en el mundo con acceso al trabajo en las computadoras, pues facilita el trabajo y lo hace de una apariencia más "bonita" y de una manera más entendible con ayuda del mouse o ratón.

Por lo dicho anteriormente existen muchas APIS para el manejo de las graficas en Windows, desde la carga de ellas, hasta el formato de las mismas, la creación, eliminación, guardado y características únicas, en fin, las siguientes son unas de las APIS más utilizadas en este apartado.

4.5.1 LoadImage

Declare Function LoadImage Lib "user32" Alias "LoadImageA" (ByVal hInst As Long, ByVal lpsz As String, ByVal un1 As Long, ByVal n1 As Long, ByVal n2 As Long, ByVal un2 As Long) As Long

Información

Esta función carga un icono, cursor o puntero y una imagen en formato bmp (llamado bitmap)

Sistema Operativo

Windows NT 4, Windows 9x, Windows 2000/XP

Parámetros

hinst. Identifica la instancia del módulo que contiene la imagen para cargarla
lpszName. Identifica la imagen a cargar
uType. Especifica el tipo de imagen a cargar
cxDesired. Especifica el ancho, en píxeles del icono o cursor
cyDesired. Especifica el alto en píxeles del icono o cursor
fuLoad. Especifica la combinación de valores

Código

Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Colocar un botón llamado "cargar", un objeto llamado "Image1" y agregar el código correspondiente

```
' en la forma
Option Explicit

' declaracion de las constantes
Const LR_LOADFROMFILE = &H10
Const IMAGE_BITMAP = 0
Const IMAGE_ICON = 1
Const IMAGE_CURSOR = 2
Const IMAGE_ENHMETAFILE = 3
Const CF_BITMAP = 2
```

```
' declaracion de las apis
Private Declare Function LoadImage Lib "user32" Alias "LoadImageA" (ByVal
hInst As Long, ByVal lpsz As String, ByVal dwImageType As Long, ByVal
dwDesiredWidth As Long, ByVal dwDesiredHeight As Long, ByVal dwFlags As Long)
As Long

Private Declare Function CloseClipboard Lib "user32" () As Long
Private Declare Function OpenClipboard Lib "user32" (ByVal hwnd As Long) As
Long
Private Declare Function EmptyClipboard Lib "user32" () As Long
Private Declare Function SetClipboardData Lib "user32" (ByVal wFormat As
Long, ByVal hMem As Long) As Long

Private Declare Function IsClipboardFormatAvailable Lib "user32" (ByVal
wFormat As Long) As Long

Private Sub cargar_Click()
    Dim hDC As Long, hBitmap As Long
    ' cargar el bitmap en la memoria
    hBitmap = LoadImage(App.hInstance, "C:\Familia\Tesis\imagenes\cargar
clipboard\girl30.bmp", IMAGE_BITMAP, 640, 480, LR_LOADFROMFILE)
    ' en caso de error, como no encontrar el archivo o formato diferente
    If hBitmap = 0 Then
        MsgBox "Ocurrio un error enla carga de la imagen"
        Exit Sub
    End If

    ' abrir el portapapeles
    OpenClipboard Me.hwnd
    ' limpiar el portapapeles
    EmptyClipboard
    ' colocar en el portapapeles la imagen
    SetClipboardData CF_BITMAP, hBitmap
    ' verificar que la imagen se encuentra en el portapaleles
    If IsClipboardFormatAvailable(CF_BITMAP) = 0 Then
        MsgBox "Ocurrio un error al pasar la imagen al portapapeles"
    End If
    ' cerrar el portapapeles
    CloseClipboard
    ' obtener la imagen del portapapeles
    Image1.Picture = Clipboard.GetData(vbCFBitmap)
End Sub
```

La primera parte son las declaraciones de las APIS y las constantes necesarias, la segunda es el botón cargar, una vez declaradas las variables, lo primero será cargar la imagen a memoria, la segunda es abrir el portapapeles con la identificación de la forma, tercero, limpiar el portapapeles para no tomar basura anterior, cuarto colocar en el portapapeles la imagen ya cargada en memoria, quinto verificar que la imagen ha sido colocada exitosamente, sexto cerrar el portapapeles y por último tomar la imagen del portapapeles con la opciones de Visual Basic 6 directamente al objeto Image1.

Figura 4.7 Ventana del Programa que carga un archivo de imagen tipo mapa de bits (*.bmp) y lo abre por medio del portapapeles

4.6 MOUSE

El mouse o ratón, tan importante para los Sistemas Operativo Gráficos como lo es Windows, existen varias APIS para su único uso, algunas de ellas se muestran a continuación, sin dejar de mencionar que estas son solo algunas de ellas, existen más, para tareas específicas en el uso de este dispositivo tan utilizado por todos en el mundo de la computación.

4.6.1 SetCapture

```
Declare Function SetCapture Lib "user32" Alias "SetCapture" (ByVal hwnd As Long) As Long
```

Información
Esta función captura el mouse a una forma perteneciente o ventana, cuando se captura el mouse, este no puede salir de los límites de la ventana, solamente se puede capturar una vez al mouse por ventana.

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hWnd. Identifica la ventana que captura el mouse

4.6.2 ReleaseCapture

Declare Function ReleaseCapture Lib "user32" Alias "ReleaseCapture" () As Long

Información
Esta función libera el mouse que ha sido capturado por una ventana y lo regresa a su estado normal.

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
No hay parámetros

4.6.3 GetCursorPos

Declare Function GetCursorPos Lib "user32" Alias "GetCursorPos" (lpPoint As POINTAPI) As Long

Información
Esta función obtiene las coordenadas de la posición actual del mouse

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
lpPoint. Apunta a una estructura tipo POINT que recibe las coordenadas del cursor del mouse

Código
Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows

```
' en la forma
Option Explicit

' estrcutura
Private Type POINTAPI
    X As Long
    Y As Long
End Type
```

```
' declaracion d elas apis
Private Declare Function SetCapture Lib "user32" (ByVal hwnd As Long) As Long
Private Declare Function ReleaseCapture Lib "user32" () As Long
Private Declare Function GetCursorPos Lib "user32" (lpPoint As POINTAPI) As
Long
Dim Pt As POINTAPI

Private Sub Form_Load()
    ' capturar
    SetCapture Me.hwnd
End Sub

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single,
Y As Single)
    ' liberar la captura
    ReleaseCapture
    SetCapture Me.hwnd
End Sub

Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y
As Single)
    ' liberar el mouse
    ReleaseCapture
    SetCapture Me.hwnd
End Sub
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)
    ' obtener la posicion del mouse en toda la pantalla
    GetCursorPos Pt
    Me.CurrentX = 0
    Me.CurrentY = 0
    ' limpiar la pantalla
    Me.Cls
    Me.Print ""
    Me.Print " Posicion del cursor: "
    ' imprimir las coordenadas del cursor en la pantalla
    Me.Print " X:" + Str$(Pt.X) + " y " + " Y:" + Str$(Pt.Y)
    Me.Print ""
    Me.Print "     (Teclea ALT+F4 para descargar la forma)"
    SetCapture Me.hwnd
End Sub
```

Este código muestra la manera en que se captura al mouse en la pantalla y que no se le deja hacer nada hasta no descargar el programa, esto es muy visto en programas de demostración e interactivos que mientras se muestra algo no se desea que el usuario pierda la atención necesaria y realice alguna otra tarea, como también en video juegos, aunque hay otras maneras y APIS como la de convertir el puntero del mouse en el reloj de arena con el cual no se puede hacer nada mientras se encuentre en este estado.

Primeramente como se hace normalmente se declaran las constantes, estructuras y APIS que se utilizaran en el código.

Ahora este código se activará al iniciar la ventana o form1, que capturara el mouse a la ventana cargada, los eventos de Form_MouseDown, Form_MouseUp, atraparan al mouse en caso de querer minimizar, maximizar y cerrar la ventana desde sus iconos, para forzar a cerrar el programa con ALT+F4, y al evento de Form_MouseMove obtengo la posición del mouse, y después actualizo la forma o limpio, para imprimir los valores de "x" y "y" obtenidos anteriormente, de una manera sencilla de hacerlo.

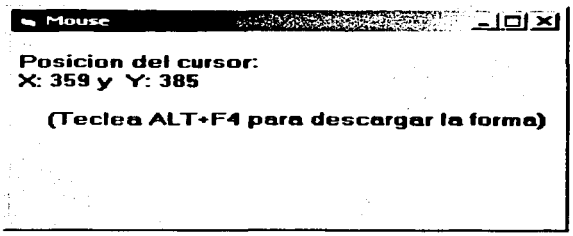

Figura 4.8 Ventana del Programa que muestra la posición del cursor

4.7 VENTANAS

Una parte muy importante de Windows ya que lleva el mismo nombre este Sistema Operativo, todo Windows es a base de gráficos, iconos, ventanas y otros elementos como texto, pero el trabajo con las ventanas es primordial ya que aquí se abren los programas, menús, comentarios, y muchas cosas más; son como recipientes en los cuales se coloca toda la información que se desee, botones, cajas de texto, videos, dibujos, etc. todo.

Y como todo llevan un orden jerárquico, de parentesco de padres e hijos, y diferencias que las distinguen unas de otras, ya que ninguna es igual aunque se tengan instancias del mismo programa abierto. Una vez aclaradas las ventanas las siguientes APIS muestran lo comentado anteriormente, nos dicen si tienen hijos o no, y nos dan parámetros de las mismas, como su handle, nombre, y clase a la que pertenecen, si están o no activas en ese momento, etc.

Utilizando las siguientes APIS y con ayuda de un más en miscelánea se ha hecho un programa que hace la misma función que el famoso CRTL+ALT+SUPR de Windows o administrador de tareas,

4.7.1 EnumWindows

```
Declare Function EnumWindows Lib "user32.dll" (ByVal lpEnumFunc As Long, ByVal lParam As Long)
As Long
```

Información
Esta función enumera todos los niveles de las ventanas de Windows en pantalla, pasando su enlace a quien lo llama, este continua ciclando hasta que la última ventana de nivel mas alto ha sido enumerado o la función regreso el valor de falso

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
lpEnumFunc. Apunta a la función de llamada de regreso
lParam. Especifica un número de 32 bits de retorno por la función

4.7.2 GetWindowText
Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" (ByVal hwnd As Long, ByVal lpString As String, ByVal cch As Long) As Long

Información
Esta función copia el texto específico de una ventana, (el nombre de la ventana) y lo coloca en un buffer

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hWnd. Identifica la ventana o control que contiene el texto
lpString. Apunta al buffer que recibe el texto
nMaxCount. Especifica el número máximo de caracteres que se copian al buffer incluyendo a Null, si se pasa del límite se trunca

4.7.3 GetWindowTextLength
Declare Function GetWindowTextLength Lib "user32" Alias "GetWindowTextLengthA" (ByVal hwnd As Long) As Long

Información
Esta función recibe la longitud en caracteres de l texto de la ventana

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hWnd. Identifica la ventana o control

4.7.4 IsWindowVisible

Declare Function IsWindowVisible Lib "user32" Alias "IsWindowVisible" (ByVal hwnd As Long) As Long

Información
Esta función recibe el estado de visibilidad de la ventana especificada, si es o no visible, o como se encuentra

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hWnd. Identifica la ventana

4.7.5 GetWindow

Declare Function GetWindow Lib "user32" Alias "GetWindow" (ByVal hwnd As Long, ByVal wCmd As Long) As Long

Información
Esta función recibe el estado de parentesco con la ventana especificada, si es propia o hija entre otras

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hWnd. Identifica la ventana
uCmd. Muestra la relación entre la ventana especificada y la ventana a quien entregara la correspondencia en parentesco

4.7.6 GetClassName

Declare Function GetClassName Lib "user32" Alias "GetClassNameA" (ByVal hwnd As Long, ByVal lpClassName As String, ByVal nMaxCount As Long) As Long

Información
Esta función regresa el nombre de la clase a la cual la ventana especificada pertenece

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hWnd. Identifica la ventana e indirectamente la clase a la cual la ventana pertenece
lpClassName. Apunta al buffer que recibe la cadena del nombre de la clase
nMaxCount. Especifica el tamaño en caracteres del buffer apuntado por el parámetro, este es truncado en caso de excederse

4.7.7 FindWindow

Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName As String, ByVal lpWindowName As String) As Long

Información
Esta función regresa el enlace del nivel al cual pertenece la ventana especificadas por las cadenas del nombre y la clase, esta función no busca ventanas hijas

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
lpClassName. Apunta a una cadena que especifica el nombre de la clase que la identifica
lpWindowName. Apunta a una cadena que especifica el nombre del titulo de la ventana, si el valor el Null, todas las ventanas encajan

Código
Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un menú simple en la forma llamado "mnuCerrar", un modulo y agregar el código correspondiente

```
' dentro del modulo
' declaracion de las apis
Declare Function EnumWindows Lib "user32" (ByVal lpEnumFunc As Long, ByVal
lParam As Long) As Boolean
Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" (ByVal
hwnd As Long, ByVal lpString As String, ByVal cch As Long) As Long
Declare Function GetWindowTextLength Lib "user32" Alias
"GetWindowTextLengthA" (ByVal hwnd As Long) As Long
Declare Function IsWindowVisible Lib "user32" (ByVal hwnd As Long) As Long
Declare Function GetWindow Lib "user32" (ByVal hwnd As Long, ByVal wCmd As
Long) As Long
Private Declare Function PostMessage Lib "user32" Alias "PostMessageA" (ByVal
hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, lParam As Any) As
Long
Private Declare Function GetClassName Lib "user32" Alias "GetClassNameA"
(ByVal hwnd As Long, ByVal lpClassName As String, ByVal nMaxCount As Long) As
Long
Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal
lpClassName As String, ByVal lpWindowName As String) As Long
```

```
' constantes
Public Const GW_OWNER = 4
Public Const WM_CLOSE = &H10
Const GW_CHILD = 5
Const GW_HWNDNEXT = 2

Public Function EnumWindowsProc(ByVal hwnd As Long, ByVal lParam As Long) As
Boolean
    Dim sSave As String, Ret As Long, visible As Long
    Dim WinWnd As Long, WinWnd2 As Long, ven As Long
    Dim lpClassName As String, RetVal As Long

    ' tomar el texto de la ventana
    Ret = GetWindowTextLength(hwnd)
    sSave = Space(Ret)
    visible = IsWindowVisible(hwnd)

    ' solo pongo las ventanas con texto
    If sSave <> "" Then
       If visible = 1 Then
           ven = GetWindow(hwnd, GW_OWNER)
           If ven = 0 Then
                ' brincarme estas dos ventanas o programas
                ' el manejador de windows y el propio programa proyecto1
                GetWindowText hwnd, sSave, 256
                 If sSave = "Program Manager" Or sSave = "Proyecto1" Then
                    GoTo etiqueta
                End If
                WinWnd2 = FindWindow(vbNullString, sSave)
                ' crear un bufer
                lpClassName = Space(256)
                ' obtener el nombre de la clase
                RetVal = GetClassName(WinWnd2, lpClassName, 256)
                cerrar.Print sSave
                ' cerrar el programa
                PostMessage WinWnd2, WM_CLOSE, 0&, 0&
           End If
       End If
    End If
etiqueta:
    'continuar con la enumeracion
    EnumWindowsProc = True
End Function
```

Primeramente veremos el código del módulo, que no es mas que la declaración de las APIS y un par de constantes, además de la función EnumWindowsProc que es la más importante ya que esta será la que cierre los programas activos en la barra de tareas, sin importar que acciones se estén realizando, este tipo de ejercicios usando las ventanas y procesos son muy comunes e importantes dentro de la programación para Windows.

Analizando la función primeramente se toma el texto de la ventana cualquiera que encuentre dentro de los procesos de Windows, ya que la función toma estos parámetros y regresa un tipo booleano, al final se declarara nuevamente en la etiqueta verdadero para ciclar y comenzar con la siguiente ventana en los procesos de Windows.

En la primera opción se eliminan las ventanas que no sean visibles o sea que se encuentren en la barra de tareas, es la primera condición, una vez que alguna cumpla, entrará y eliminará aquellas ventanas que tengan padres, o sea que dependan de una ventana más como los menús y listas, después hago una condición de que si el texto de la ventana padre, y visible es "proyecto1" o "Program Manager" se saldrá sin hacer cambios; la primera por que es el nombre de nuestro programa y no queremos que se cierre también y el manejador de programas para que no cierre Windows.

Ya que se han cumplido estas características ahora se necesita identificar el nombre de la clase a la cual pertenece la ventana; teniendo estas propiedades entonces se escribe en la forma a manera de indicar que ventana se esta cerrando y con la API PostMessage se manda el mensaje de cerrar la ventana requerida o encontrada, es como si se le diera la "x" de cerrar al programa principal y que se encuentre en la barra de tareas, si se esta haciendo un trabajo y aun no se guarda por ejemplo en el Bloc de notas nos preguntará si deseo guardar los cambios con si, no o cancelar, de este modo no es de pronto el cierre, pero ayuda mucho como administrador que con unos cambios este programa se puede ejecutar desde un servidor y monitorear programas no deseados en máquinas cliente, cuando estas se activen se cerraran de manera automática o con muchas otras variantes para el uso de estos procesos y ventanas con las APIS.

```
' dentro de la forma
Option Explicit

Private Sub Form_Load()
   ' para poder escribir en la forma
    Me.AutoRedraw = True
End Sub

Private Sub mnuCerrar_Click()
    ' mandar llamar la funcion principal
    EnumWindows AddressOf EnumWindowsProc, ByVal 0&
End Sub
```

La última parte esta en la forma, cuando se cargue esta se activa AutoRedraw para no tener que teclearlo manualmente y poder escribir sobre la forma, el segundo es el menú cerrar que manda llamar la función EnumWindows, que se ciclara hasta que todos los procesos de Windows que sean ventanas sean revisados.

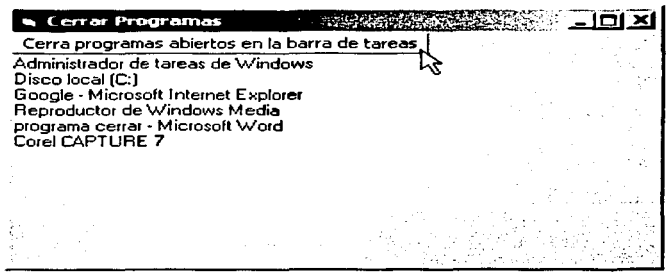

Figura 4.9 Ventana del Programa que trabaja como un Administrador de Tareas de Windows Cerrando todos los programas abiertos en la barra de tareas que se encuentren en ese momento

4.8 ICONOS

Los iconos son algo esencial para el Sistema Operativo Windows, desde el inicio del mismo los ha utilizado y hasta la fecha lo sigue haciendo, mejorando su aspecto, haciéndolos mas bonitos visualmente y cambiándolos de tamaño siguen siendo iconos, utilizados principalmente para identificar los diferentes tipos de archivos y el para que sirven, diferenciar los programas de los archivos que manejan estos programas y programas con su propio icono para identificación, acceso directo desde el escritorio de Windows tanto como en la barra de tareas e iconos mas pequeños para indicar tareas realizadas de fondo, junto al reloj, como las conexiones a Internet o red, antivirus, entre muchos otros.

Las siguientes APIS son claras y relativamente sencillas de utilizar, principalmente para la obtención de iconos referentes a archivos y el manejo de los mismos dentro del programa a realizar deseado.

4.8.1 ExtractAssociatedIcon

```
Declare Function ExtractAssociatedIcon Lib "shell32.dll" Alias "ExtractAssociatedIconA" (ByVal
hInst As Long, ByVal lpIconPath As String, lpiIcon As Long) As Long
```

Información
Esta función regresa el handle de un icono indexado encontrado en un archivo común, y en un ejecutable o programa

Sistema Operativo
Windows NT 3.5, Windows 9x, Windows 2000/XP

Parámetros
hInst. Especifica la instancia de la aplicación llamando a la función
lpIconPath. Apunta a una cadena que especifica la dirección completa y nombre del archivo al cual el icono esta asociado
lpiIcon. Apunta a una palabra que especifica el index del icono del cual será obtenido

4.8.2 DrawIcon

Declare Function DrawIcon Lib "user32" Alias "DrawIcon" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal hIcon As Long) As Long

Información
Esta función dibuja un icono en el área cliente de la ventana especificada

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hDC. Identifica la ventana
X. Especifica la coordenada superior izquierda donde será dibujado el icono
Y. Especifica la coordenada superior izquierda donde será dibujado el icono
hIcon. Identifica el icono ha ser dibujado

Nota. Para Windows NT y 95 el icono a dibujar debe haber sido previamente cargado con la API LoadIcon o LoadImage

Código
Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón llamado "abrir", otro "copiar", una caja de texto "Text1", un objeto "Picture1" y un "Image1", además de agregar un "CommonDialog1"

Para agregar este componente es necesario hacer lo siguiente:
Proyecto -> componentes -> Microsoft Common Dialog 6.0 (SP3) -> Aceptar

```
' en la forma
Option Explicit

' declaracion d elas apis
Private Declare Function ExtractAssociatedIcon Lib "shell32.dll" Alias
"ExtractAssociatedIconA" (ByVal hInst As Long, ByVal lpIconPath As String,
lpiIcon As Long) As Long

Private Declare Function DrawIcon Lib "user32" (ByVal hdc As Long, ByVal x As
Long, ByVal y As Long, ByVal hIcon As Long) As Long
```

```
Private Sub copiar_Click()
    ' se pasa el valor de picture1 a image1 de modo normal
    Image1.Picture = Picture1.Picture
End Sub
```

Esta primera parte solo da de alta las APIS a utilizar y el botón copiar demuestra que se puede pasar el objeto icono de un objeto Imagen a caja de imagen y trabajarlo en otros aspectos una vez que ya ha sido creado y cargado por el programa.

```
Private Sub abrir_Click()
Dim lIcon As Long
On Error GoTo LoadError

' utilizamos la caja de dialogo comun de abrir
With CommonDialog1
   ' filtramos las extenciones a todas
   .Filter = "All Files|*.*"
   ' se atrapan posibles errores y se muestra la caja
   .CancelError = True
   .ShowOpen

   ' mandamos el nombre del archivo a la caja
   Text1.Text = .FileName

   ' utilizamos la api para obtener el icono
   lIcon = ExtractAssociatedIcon(App.hInstance, .FileName, -1)
   ' se lo colocamos a picture
   Picture1 = LoadPicture()
   Picture1.AutoRedraw = True
   ' lo dibujamos el picture
   Call DrawIcon(Picture1.hdc, 0, 0, lIcon)
   Picture1 = Picture1.Image
End With

LoadError:
    ' simplemente terminar
End Sub
```

Esta segunda parte es el código del botón abrir, que lo que hace es tomar el icono asociado al archivo abierto por el CommonDialog1 y con ExtractAssociatedIcon y DrawIcon seleccionamos su handle para después dibujarlo, se colocan un par de propiedades para el objeto Picture1 para su correcta visualización y no hacerlo manualmente de forma directa en sus propiedades, el código es explicado por línea para su mayor entendimiento.

Cabe mencionar que en este ejemplo se utiliza una de las librerías de Visual Basic con el propósito de agilizar la muestra del programa, aunque esta misma caja de dialogo es un ejercicio mostrado en este mismo capítulo.

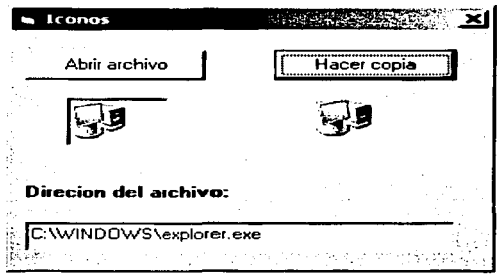

Figura 4.10 Ventana del Programa que muestran como seleccionar cualquier archivo y que carga su icono, al igual que su dirección completa

TECLADO

El teclado, tan importante como lo es ahora el mouse, este fue el primer dispositivo de entrada y por lo tanto para Windows es importantísimo ya que sin el no habría entendimiento directamente con el usuario y los caracteres de entrada, por ello existen una variedad de APIS específicas, desde la captura, estado, tiempo, secuencia de teclas, entre otras. La siguiente API muestra la captura y secuencia de teclas para ejecutar una tarea determinada.

4.9.1 keybd_event

```
Declare Sub keybd_event Lib "user32.dll" (ByVal bVk As Byte, ByVal bScan As Byte, ByVal dwFlags
As Long, ByVal dwExtraInfo As Long)
```

Información
Esta función sintetiza un conjunto de teclas tecleadas

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
bVk. Especifica un código, debe de estar entre 1 y 254
bScan. Especifica el escaneo del hardware y las teclas dadas
dwExtraInfo. Especifica un valor asociado con el código de teclas dado

Código
Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic

- *Aplicación de Windows*
- *Crear un botón simple en la forma llamado "Command1", hacerlo indexado y crear 5 botones mas de 0 a 5 y agregar el código correspondiente*

```
' en la forma
Option Explicit

' constantes
Private Const VK_LWIN = &H5B
Private Const KEYEVENTF_KEYUP = &H2
Private Const VK_APPS = &H5D

' declaracion api
Private Declare Sub keybd_event Lib "user32" (ByVal bVk As Byte, ByVal bScan
As Byte, ByVal dwFlags As Long, ByVal dwExtraInfo As Long)

Private Sub Command1_Click(Index As Integer)
Dim VK_ACTION As Long

' hacer un case para el index del boton
   Select Case Index
      Case 0: '&H45 es codigo hexadecimal
              'para la letra 'E' (ascii 69)
               VK_ACTION = &H45

      Case 1: '&H46 es codigo hexadecimal
              'para la letra 'F' (ascii 70)
               VK_ACTION = &H46

      Case 2: '&H4D es codigo hexadecimal
              'para la letra 'M' (ascii 77)
               VK_ACTION = &H4D

      Case 3: '&H52 es codigo hexadecimal
              'para la letra 'R' (ascii 82)
               VK_ACTION = &H52

      Case 4: '&H5B es codigo hexadecimal
              'para el boton de menu de inicio
               VK_ACTION = &H5B

      Case 5: '&H70 es codigo hexadecimal
              'for the carcter chr (ascii 112)
               VK_ACTION = &H70

   End Select

   ' llamada a la funcion pasando parametros por de falta y el obtenido en el
case
   Call keybd_event(VK_LWIN, 0, 0, 0)
   ' aqui se realiza la accion
   Call keybd_event(VK_ACTION, 0, 0, 0)
   Call keybd_event(VK_LWIN, 0, KEYEVENTF_KEYUP, 0)
End Sub
```

El código anterior es sencillo de entender, y como todos los demás continua con el mismo orden, declaración de las constante globales, las APIS, y las funciones, dentro de esta se coloca un case para los valores indexados del botón y no tener que haber creado 6 botones diferentes, cada valor del case se coloca el valor correspondiente al juego de teclas que desde el teclado se pueden dar, para realizar la acción, como el botón de ventanita Windows, que despliega el menú inicio, tecleando este botón en el programa también lo desplegara, este tipo de comandos son muy usados en programas bien estructurados, como por mencionar aquellos a los que toman dictados y se les puede indicar moverse en el escritorio de Windows, ¿por que no?, podrían estar utilizando este tipo de APIS.

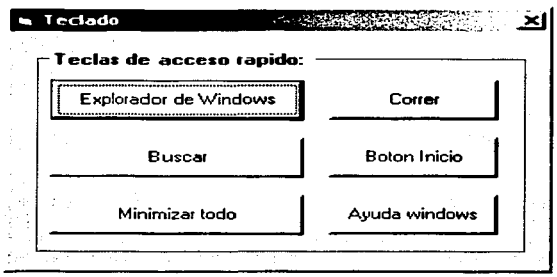

Figura 4.11 Ventana del Programa que muestra como se puede simular una secuencia de teclas presionando un botón del programa

4.10 SHELL

Shell (Cáscara) no se podría poner una traducción exacta para este término de acuerdo a su uso, pero en el Sistema Operativo Windows es una manera de trabajar con los programas directamente o de hasta crear toda una programación avanzada para dar de alta usuarios a la hora de dar el login en Windows o toda una barra de tareas de Windows para ciertos programas que los requieran, como por ejemplo centinelas o avanzados de administración y seguridad.

Las siguientes APIS son unas de las más utilizadas, para aquel que trabaja con Windows forzosamente cada vez que enciende su computadora utiliza estas APIS, así que estas son las más comunes, el programa muestra el ejemplo de lo ya mencionado.

4.10.1 ShellExecute

```
Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal hwnd As Long, ByVal
lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory As
String, ByVal nShowCmd As Long) As Long
```

Información
Esta función abre o imprime un archivo especificado, este puede ser ejecutable o de tipo archivo.

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hwnd. Especifica la ventana padre
lpOperation. Apunta a una cadena terminada en nulo que especifica la operación a realizar
lpFile. Apunta a una cadena terminada en nulo que especifica el archivo a abrir, ejecutar, imprimir o el fólder a explorar o abrir
lpParameters. Si lpFile especifica un ejecutable punta a una cadena terminada en nulo que especifica la operación a realizar, si es un archivo de documento lpParameters, debe ser nulo
lpDirectory. Apunta a una cadena terminada en nulo que especifica la carpeta por defalca
nShowCmd. Si lpFile especifica un archivo ejecutable, nShowCmd especifica como será mostrada la aplicación o abierta

4.10.2 ShellExecuteEx

```
Declare Function ShellExecuteEx Lib "shell32.dll" Alias "ShellExecuteEx" (SEI As SHELLEXECUTEINFO) As Long
```

Información
Esta función abre un archivo, este puede ser ejecutable o de tipo documento, si así lo es, será abierto por su correspondiente programa

Sistema Operativo
Windows NT 4, Windows 9x, Windows 2000/XP

Parámetros
lpExecInfo. Apunta a una estructura tipo SHELLEXECUTEINFO que contiene y recibe información acerca de la aplicación para comenzar

Código
Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear dos botones simples en la forma llamados "Command1", "Command2" y agregar el código correspondiente

```
' en la forma
Option Explicit


' constantes
Const SEE_MASK_INVOKEIDLIST = &HC
Const SEE_MASK_NOCLOSEPROCESS = &H40
```

```
Const SEE_MASK_FLAG_NO_UI = &H400
Const SW_SHOWNORMAL = 1

' estructura
Private Type SHELLEXECUTEINFO
     cbSize As Long
     fMask As Long
     hwnd As Long
     lpVerb As String
     lpFile As String
     lpParameters As String
     lpDirectory As String
     nShow As Long
     hInstApp As Long
     lpIDList As Long
     lpClass As String
     hkeyClass As Long
     dwHotKey As Long
     hIcon As Long
     hProcess As Long
End Type

' apis
Private Declare Function ShellExecuteEx Lib "shell32.dll" (SEI As
SHELLEXECUTEINFO) As Long

Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA"
(ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String,
ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As
Long) As Long
Sub ShowProps(FileName As String, OwnerhWnd As Long)
Dim SEI As SHELLEXECUTEINFO
Dim r As Long

With SEI
     'colocar el tamaño de la estrcutura
     .cbSize = Len(SEI)
     'colocar la mascara
     .fMask = SEE_MASK_NOCLOSEPROCESS Or _
              SEE_MASK_INVOKEIDLIST Or SEE_MASK_FLAG_NO_UI
     'ponerlo en la propia ventana
     .hwnd = OwnerhWnd
     'mostrar las propiedades
     .lpVerb = "properties"
     'pones el nombre dle archivo
     .lpFile = FileName
     .lpParameters = vbNullChar
     .lpDirectory = vbNullChar
     .nShow = 0
     .hInstApp = 0
     .lpIDList = 0
End With
' ejecutar la API con la variable SEI y sus propiedades dadas
r = ShellExecuteEx(SEI)
End Sub
```

```
Private Sub Command1_Click() [11]
Dim nombre As String

' obtenr la direccion y nombre del archivo
CommonDialog1.ShowOpen
nombre = CommonDialog1.FileName
' mandar llamar la funcion
ShowProps nombre, Me.hwnd
End Sub

Private Sub Command2_Click()
Dim nombre As String

' obtenr la direccion y nombre del archivo
CommonDialog1.ShowOpen
nombre = CommonDialog1.FileName
' ejecutar el archivo si el programa y si no su
' programa relacionado, abrirlo
ShellExecute Me.hwnd, vbNullString, nombre, vbNullString, "C:\",
SW_SHOWNORMAL
End Sub
```

En esta primera parte se encuentran las declaraciones de constantes, APIS y estructuras como es costumbre y deben de ir por la lógica del programa, después los dos eventos de ambos botones, el Command2 es mas sencillo que el primero, pues una vez obtenido la cadena del nombre del archivo se pasa a la API ShellExecute para la apertura correspondiente del archivo y el manda llamar la función ShowProps en la cual serán dadas las características de la estructura primeramente para mandar llamar la API ShellExecuteEx y colocar las propiedades según sea el archivo abierto, si es una imagen o tipo texto u otro.

La imagen mostrada en la figura 4.12 para este ejercicio es de las propiedades de un archivo tipo imagen, aunque con el botón Command2 será ejecutado el visor de imágenes para poder visualizarla, mas sin embargo con command1 muestra sus características; en caso de ser un archivo ejecutable, las características las mostrara, pero en cambio al ejecutarlo este abrirá el programa al que corresponde.

[11] http://www.allapi.net/

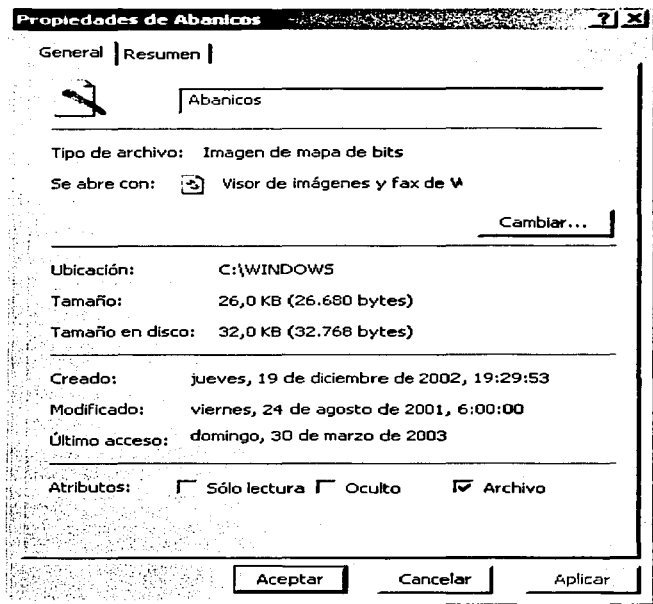

Figura 4.12 Ventana del Programa que ejecuta un archivo para visualizar sus características o propiedades

4.11 MISCELANEA

En esta sección he colocado APIS variadas, que pertenecen a diferentes apartados y librerías de Windows, desde procesos, ventanas, graficas, sonido, mensajes, cadenas, objetos entre otras, con ejercicios para mostrar cada una de ellas junto con otras funcionando como ha sido a lo largo de toda esta tesis.

4.11.1 GetDC

```
Declare Function GetDC Lib "user32" Alias "GetDC" (ByVal hwnd As Long) As Long
```

Información
Esta función obtiene el handle de un objeto en un área de cliente de una ventana específica

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hWnd. Identifica la ventana que el objeto regresa

4.11.2 ReleaseDC
Declare Function ReleaseDC Lib "user32" Alias "ReleaseDC" (ByVal hwnd As Long, ByVal hdc As Long) As Long

Información
Esta función libera un objeto creado por GetDC, dejándolo libre para otra aplicación

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP
Parámetros
hWnd. Identifica la ventana que será liberada
hDC. Identifica el objeto DC ha ser liberado

4.11.3 GetWindowRect
Declare Function GetWindowRect Lib "user32" Alias "GetWindowRect" (ByVal hwnd As Long, lpRect As RECT) As Long
Información
Esta función recibe las dimensiones del rectángulo de una ventana específica, dadas por las coordenadas de la pantalla

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hWnd. Identifica la ventana
lpRect. Apunta a una estructura tipo RECT que recibe las coordenadas de la pantalla superior izquierda e inferior derecha.

4.11.4 GetDesktopWindow
Declare Function GetDesktopWindow Lib "user32" Alias "GetDesktopWindow" () As Long

Información
Esta función regresa el handle de la ventana del escritorio de Windows, (la pantalla entera).

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
No tiene parámetros

4.11.5 BitBlt

Declare Function BitBlt Lib "gdi32" Alias "BitBlt" (ByVal hDestDC As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As Long) As Long

Información
Esta función trabaja con bloques de bits, en donde los transfiere de un objeto fuente a uno destino, pasando sus características y valores como si fuera una forma de copiar

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
hdcDest. Identifica el objeto destino
nXDest. Especifica la coordenada x superior izquierda del objeto destino
nYDest. Especifica la coordenada y superior izquierda del objeto destino
nWidth. Especifica el ancho del objeto fuente y destino
nHeight. Especifica el alto del objeto fuente y destino
hdcSrc. Identifica el objeto fuente
nXSrc. Especifica la coordenada x superior izquierda del objeto fuente
nYSrc. Especifica la coordenada y superior izquierda del objeto fuente
dwRop. Especifica el rastreo del color origen al destino

Código
Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un menú llamado "mnuCapturar", un objeto "Picture1" y colocar el codigo correspondiente

```
' en la forma
Option Explicit

' declaracion de la estructura
Private Type RECT
Left As Long
Top As Long
Right As Long
Bottom As Long
End Type
```

```
' declaracion de las apis

Private Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal x
As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal
hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As Long)
As Long

Private Declare Function GetDC Lib "user32" (ByVal hwnd As Long) As Long
Private Declare Function ReleaseDC Lib "user32" (ByVal hwnd As Long, ByVal
hdc As Long) As Long
Private Declare Function GetWindowRect Lib "user32" (ByVal hwnd As Long,
lpRect As RECT) As Long
Private Declare Function GetDesktopWindow Lib "user32" () As Long

Private Sub mnuCapturar_Click()
Dim DhWnd As Long, DhDC As Long,. Img As RECT
Dim ancho As Integer
Dim alto As Integer

' Tomo la imagen de todo el escritorio
DhWnd = GetDesktopWindow
' obtengo su handle
DhDC = GetDC(DhWnd)
GetWindowRect DhWnd, Img
' La pongo dentro de la caja usando la api
BitBlt Picture1.hdc, 0&, 0&, Screen.Width, Screen.Height, DhDC, 0&, 0&,
&HCC0020
Picture1 = Picture1.Image
' Libero
ReleaseDC DhWnd, DhDC
End Sub
```

Este código captura la imagen de todo el escritorio y la coloca dentro de un objeto Picture en la forma, aunque puede ser directamente en la forma, una vez cargado se puede trabajar con él, hacerlo a una escala ajustable, invertir colores, deformarlo, hacer acercamientos (zoom), etc. todo con las APIS apropiadas a realizar lo deseado.

En este caso con ayuda de las APIS de contexto una vez obtenida el escritorio con GetDesktopWindow y pasado su handle con BitBlt puedo colocarlo dentro del objeto Picture el nuestro programa, después liberamos el handle, el código es poco y sencillo aunque utiliza un par de APIS, que las coloque en el apartado de miscelánea.

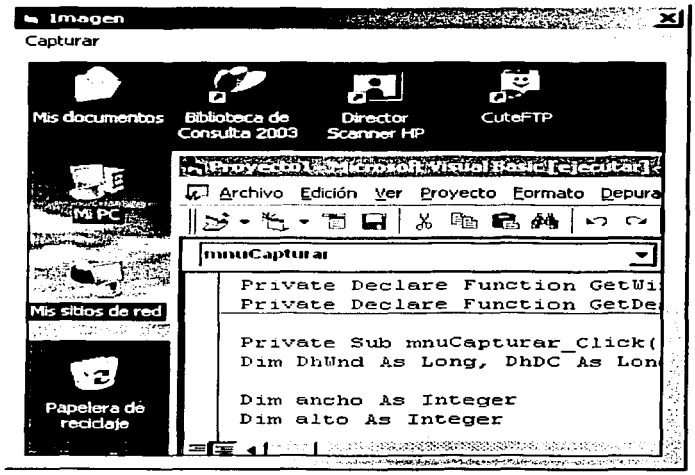

Figura 4.13 Ventana del Programa que captura el escritorio y lo coloca dentro de un objeto tipo Picture como imagen

4.11.6 GetTickCount

Declare Function GetTickCount Lib "kernel32" Alias "GetTickCount" () As Long

Información

Esta función recibe el número de milisegundos que has transcurrido desde que Windows inicio

Sistema Operativo

Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros

Sin parámetros

Código

Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "Command1" y agregar el código correspondiente

```
' en la forma
Option Explicit

' declaracion de la api
Private Declare Function GetTickCount& Lib "kernel32" ()

Private Sub Command1_Click()
Dim lngTickCount As Long

' mando llamar la funcion y el valor se lo asigno a lngTickCount
lngTickCount = GetTickCount
' simplemente el resultado se pone en un mensaje
Call MsgBox("Se ha utilizado esta computadora por: " & vbNewLine & _
vbNewLine & " * " & CStr(lngTickCount) & " millisegundos, o" & _
vbNewLine & " * " & CStr(Round(lngTickCount / 1000)) & " segundos, o" & _
vbNewLine & " * " & CStr(Round((lngTickCount / 1000) / 60)) & " minutos o " &
_
vbNewLine & " * " & CStr((((lngTickCount / 1000) / 60) / 60)) & _
" horas ", vbInformation, "Tiempo transcurrido")

' CStr lo convierte en tipo cadena
' Round redondea el valor, si pasa de .5 coloca el siguiente
End Sub
```

Sin lugar a dudas esta API es una de las mas sencillas ya que no trabaja con muchos parámetros, pero esto no quiere decir que sea no útil sino que todo lo contrario, con esta que es como un contador de segundos desde que Windows inicia se le puede sacar mucho jugo en programas que utilicen tiempos como necesidad primaria, como por ejemplo desde programas de prueba que a un tiempo determinado se cierren, etc.

Aquí la explicación es rápida, se declara la API, dentro del botón de manda llamar un mensaje y se le agregan las divisiones que obtuvimos de la variable en milisegundos del tiempo total de haber cargado Windows.

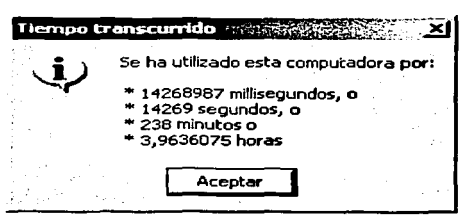

Figura 4.14 Ventana del Programa que muestra el tiempo transcurrido desde que Windows inicio

4.11.7 Lstrcat

Declare Function lstrcat Lib "kernel32" Alias "lstrcatA" (ByVal lpString1 As String, ByVal lpString2 As String) As Long

Información
Esta función añade una cadena a otra

Sistema Operativo
Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros
lpString1. Apunta a una cadena terminada en nulo
lpString2. Apunta a una cadena terminada en nulo para ser añadida la cadena especificada por lpString1

4.11.8 SHGetPathFromIDList

Declare Function SHGetPathFromIDList Lib "shell32" (ByVal pidList As Long, ByVal lpBuffer As String) As Long

Información
Esta función convierte un elemento identificador de listas a una direccion de archivo en el sistema.

Sistema Operativo
Windows NT 4, Windows 9x, Windows 2000/XP

Parámetros
pdl. Apunta a un elemento identificador de listas que especifica la dirección de un archivo o directorio raíz o del escritorio
pszPath. Apunta a un buffer que recibe la dirección del archivo

Código
Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón llamado "folder" y colocar el código correspondiente

```
' en la forma
Option Explicit

' constantes
Const BIF_RETURNONLYFSDIRS = 1
Const MAX_PATH = 260
Const OFS_MAXPATHNAME = 128
Const OF_EXIST = &H4000
Const FILE_NOT_FOUND = 2
```

```
' estructura para la informacion de las carpetas
Private Type BrowseInfo [12]
     hWndOwner As Long
     pIDLRoot As Long
     pszDisplayName As Long
     lpszTitle As Long
     ulFlags As Long
     lpfnCallback As Long
     lParam As Long
     iImage As Long
End Type

Private Type OFSTRUCT
     cBytes As Byte
     fFixedDisk As Byte
     nErrCode As Integer
     Reserved1 As Integer
     Reserved2 As Integer
     szPathName(OFS_MAXPATHNAME) As Byte
End Type

' declaracion delas apis
Private Declare Sub CoTaskMemFree Lib "ole32.dll" (ByVal hMem As Long)
Private Declare Function lstrcat Lib "kernel32" Alias "lstrcatA" (ByVal
lpString1 As String, ByVal lpString2 As String) As Long
Private Declare Function SHBrowseForFolder Lib "shell32" (lpbi As BrowseInfo)
As Long
Private Declare Function SHGetPathFromIDList Lib "shell32" (ByVal pidList As
Long, ByVal lpBuffer As String) As Long
Private Declare Function OpenFile Lib "kernel32" (ByVal lpFileName As String,
lpReOpenBuff As OFSTRUCT, ByVal wStyle As Long) As Long

Private Sub folder_click()
Dim iNull As Integer, lpIDList As Long, lResult As Long
Dim sPath As String, udtBI As BrowseInfo
Dim Ret As Integer

With udtBI
' ponerla en la propia ventana
.hWndOwner = Form1.hWnd
' la funcion lstrcat regresa la direccion de memoria de dos cadenas
.lpszTitle = lstrcat("C:\", "")
' regresa solo si el usuario selecciono una carpeta
.ulFlags = BIF_RETURNONLYFSDIRS
End With

' mostrar el cuadro de dialogo de carpetas
lpIDList = SHBrowseForFolder(udtBI)
If lpIDList Then
     sPath = String$(MAX_PATH, 0)
     ' obtener la direciconcion o path
     SHGetPathFromIDList lpIDList, sPath
     ' liberar el blocke de memoria
     CoTaskMemFree lpIDList
     iNull = InStr(sPath, vbNullChar)
```

[12] http://www.allapi.net/

```
        If iNull Then
            sPath = Left$(sPath, iNull - 1)
        End If
End If

' mando llamar la funcion FileExists y mando la direccion
' obtenida anteriormente, si es vacia no entra
If sPath <> "" Then
    Ret = FileExists(sPath)
    ' si Ret = -1 entra y es valida
    If Ret Then MsgBox "La carpta existe"
Else
   MsgBox "La carpeta no existe"
End If
End Sub

Function FileExists(FileName As String) As Integer
Dim RetCode As Integer
Dim OpenFileStructure As OFSTRUCT

' paso los parametros de la api para aberiguar si el archivo existe
RetCode = OpenFile(FileName$, OpenFileStructure, OF_EXIST)
If OpenFileStructure.nErrCode = FILE_NOT_FOUND Then
   FileExists = False
Else
   FileExists = True
End If
End Function
```

Este ejemplo es uno de los mas difíciles, no por su grado de complejidad en la forma en que se ejecuta ni por entenderlo, si no por las diferentes APIS que utiliza y su variedad tan solo para lograr un objetivo en concreto, el saber si una carpeta u archivo existe o no, además de permitir al usuario seleccionar manualmente la dirección y el estarlo buscando, este tipo de pequeños programas son comúnmente usados para localizar drives o controladores para el Sistema Operativo.

Una vez ya declaradas las constantes, APIS, y estructuras a necesitar, existe una función llamada FileExists, que pasa como parámetro la cadena del nombre del archivo, se coloca un if para que el resultado de la API nos diga si este existe o no, si lo encuentra es verdadero como resultado de la función de lo contrario es falso.

El botón lleva la carga de todo el ejemplo, cuando se activa despliega una caja de dialogo para seleccionar un archivo o carpeta, pasando los parámetros de la variable udtBI de tipo estructura, después se crea la caja de dialogo, y se espera al usuario que modifique los parámetros, para obtenerlos como resultado, para pasar a un if y mandar llamar la función FileExists, con la característica de que si regresa una cadena vacía mandara el mensaje de que no se ha encontrado tal carpeta de lo contrario será exitosa la búsqueda de la carpeta por el usuario.

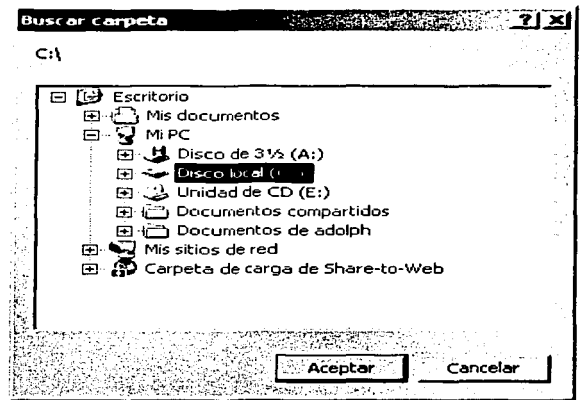

Figura 4.15 Ventana del Programa que permite al usuario escoger una carpeta

4.11.9 Beep

Declare Function Beep Lib "kernel32" Alias "Beep" (ByVal dwFreq As Long, ByVal dwDuration As Long) As Long

Información

Esta función genera simples tonos en la bocina de la computadora.

Sistema Operativo

Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros

dwFreq. Especifica la frecuencia en hertz del sonido

dwDuration. Especifica la duración del sonido en milisegundos

Nota. En Windows 95 ambos parámetros son ignorados

Código

Este código esta en Visual Basic 6

- Crear un proyecto nuevo
- Seleccionar Visual Basic
- Aplicación de Windows
- Crear un botón simple en la forma llamado "sonar", dos cajas de texto Text1 y Text2, dos etiquetas y agregar el código correspondiente

```
' en la forma
Option Explicit

' declaracion de la api
Private Declare Function Beep Lib "kernel32" (ByVal dwFreq As Long, ByVal
dwDuration As Long) As Long
Public tope As Integer
Public paso As Integer

Private Sub sonar_Click()
Dim Cnt As Long

' asignacion d elas variables
tope = Text1.Text
paso = Text2.Text
' un for desde cero a 4000 cada 20
For Cnt = 0 To tope Step 20
    ' toca un tono de Cnt en hertz, por 50 millisegundos
    Beep Cnt, paso
    Me.Caption = Cnt
Next Cnt
End Sub
```

El código es primeramente sus declaración de API, y al evento clic del botón sonar se realiza un ciclo de cero a tope mandando llamar la función Beep, con sus paso, cuando termine el ciclo sale y termina.

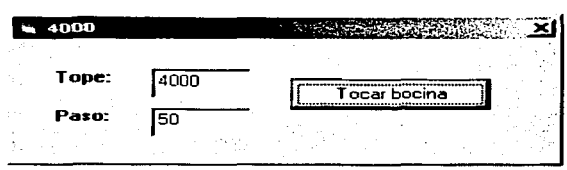

Figura 4.16 Ventana del Programa que mustra el boton para tocar la bocina

Figura realizadas por el autor de esta obra

La siguiente API será utilizada para complementar el ejemplo de encontrar y verificar que una ventana se encuentre activa en la barra de tareas para cerrarla de manera automática, junto con otras APIS en la sección de ventanas.

4.11.10 PostMessage

Declare Function PostMessage Lib "user32" Alias "PostMessageA" (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long

Información

Esta función coloca un mensaje que esta asociado con la tarea creada de la ventana especifica y que retorna sin esperar por el proceso del mensaje inicial

Sistema Operativo

Windows NT 3.1, Windows 9x, Windows 2000/XP

Parámetros

hWnd. Identifica la ventana del procedimiento que recibirá el mensaje
wMsg. Especifica el mensaje a ser colocado
wParam. Especifica características adicionales de información al mensaje
lParam. Especifica información adicional al mensaje

Finalizando el cuarto capítulo, cabe mencionar que no son todas las APIs en general y que no son todas las APIs pertenecientes a cada familia, se tomaron en cuenta solo las que se requerian para realizar estos ejemplos de ayuda y soporte, algunas tienen su ejemplo propio, otras se complementan con otras APIs para realizar un ejercicio mas completo y darlas a mostrar cada una en su manera de utilizarlas, sin dejar de lado que algunas pueden usar constantes variadas en base a la opción que puedan tomar dentro de la ejecución del programa.

Conclusión

Por todo lo descrito y código realizado en los capítulos anteriores se confirma que el objetivo principal propuesto ha quedado cubierto satisfactoriamente, dando un conocimiento al lector de los que son las APIs, como nacieron, como podrían ser, como están estructuradas, como se manejan, lo que es posible hacer con ellas, como están divididas en familias, como buscarlas en caso de necesitar alguna de ellas y un conocimiento práctico de los ejercicios mostrados en esta tesis, que son reales y se ejecutan sin problemas, que son de uso gratuito y que mejoran eficientemente el software a desarrollar para un sistema operativo Windows, que son difíciles de actualizar y modificar de una en una personalizada, que eliminan costos de elaboración de Software en el sentido de que ya no se necesitan comprar elementos extras como controles y programas externos para la aplicación a desarrollar.

Se demuestra que existen diferentes tipos y modos de utilizarlas de acuerdo al programa que se utilice como lenguaje de programación, que estas, el programador no las puede modificar o crear nuevas, ya que son dadas por las librerías en el propio sistema operativo Windows.

Se da por concluido que no son todas las APIs existentes pero que las explicadas son fácilmente utilizables y entendibles por las características en la forma de mostrar la información y estructuras. Se demuestra de este modo que las APis son una parte muy importante para el ingeniero en Computación ya que Windows es el sistema operativo mas utilizado del mundo actualmente, de modo que si se desea desarrollar software es necesario conocer y saber trabajar las APIs, aunque no se tenga el conocimiento de cada una de ellas pero si las bases para saber buscarlas de acuerdo a las necesidades que se puedan presentar para el programador.

De este modo queda entendido que se ha hecho una recolección de las APIs mas utilizadas, se han creado ejemplos prácticos y efectivos, no únicos y de difícil creación sin una explicación sólida, cabe mencionar que para la elaboración de esta tesis se busco información bibliográfica en diferentes ciudades de México como el Distrito Federal, Querétaro, Celaya, sin encontrar una fuente sólida, solo en la American Book Store en el Distrito Federal con fecha del 2000, la mayoría de la información fue obtenida de Internet de Sitios como MSDN del mismo Microsoft, foros de ayuda y expertos, fue una ardua tarea en la recopilación de la información y puesta en práctica ya que casi todo lo encontrado era en marco teórico o ejemplos muy censillos, pero que sin lugar a dudas servirán de mucho a los lectores programadores que trabajen por primera vez con las APIs o que las conozcan, de modo que el nivel de la tesis es para programadores con básica y media experiencia en el campo, para expertos en la materia recomiendo buscar fuentes de información directamente con Microsoft para su correcto desarrollo y especialización.

Bibliografía

Libros:

GRIMES Galen A. y APPLEMAN Dan, ***Visual Basic Programmer's Guide to the Win32API***, Sam, Estados Unidos, Diciembre 1999, 1548 p.

STEVEN Roman, ***Win32 API Programming with Visual Basic,*** O'Reilly & Associates, Estados Unidos, Noviembre 1999, 534 p.

JASON Bock, ***Visual Basic 6 Win32 API Tutorial,*** Wrox Press Inc, Estados Unidos, Diciembre 1998, 368 p.

Otras fuentes:

- Microsoft Encarta 2003 español
- http://guille.costasol.net/vb_api.htm
- http://es.wikipedia.org/wiki/Historia_de_Windows
- http://www.fortunecity.com/skyscraper/fatbit/607/winstory/storydetails.html
- http://www.entrebits.com/php/diccionario/?Page=3&letra=a
- http://www.unixsup.com/unixlinux/historiax1.html
- http://216.26.168.92/vbapi/ref/funcc.html#registry
- http://www.allapi.net/
- http://www.aisect.org/VB_resource_links.htm
- http://www.ifrance.com/mrjabrane/
- http://is-it-true.org/nt/hottips.shtml
- http://www.fw.cz/jankudr/tt.htm
- http://www.vbworld.com/
- http://msdn.microsoft.com/