La Filosofía de Windows: El Paradigma del Paso de Mensajes

Ing. Eduardo Vega Alvarado
Jefe del Departamento de Laboratorios Ligeros del CINTEC-IPN

Dado su gran éxito comercial, Windows se ha convertido en el estándar de facto para el desarrollo de aplicaciones dirigidas al usuario general. Varias han sido las razones para este suceso, destacando entre las más importantes, su interfaz de tipo gráfico con el usuario, el manejo multitarea, y la relativa independencia de los programas de aplicación en relación al hardware sobre el que se ejecutarán.

Sin duda alguna, el ambiente de trabajo amigable ("friendly") de Windows constituye la característica más apreciada por el usuario, aunque esto plantea al programador nuevos y más complejos problemas para el desarrollo de aplicaciones. En este artículo se presenta una descripción general del entorno Windows y de los modelos tanto de desarrollo de aplicaciones como de ejecución de las mismas; en artículos posteriores se ejemplificarán las técnicas específicas para el diseño de programas enfocados a este sistema.

Windows como un Sistema Operativo

Windows ofrece al programador más de 1500 funciones o llamadas al sistema (todas ellas diferentes), conocidas colectivamente como Interface para Programación de Aplicaciones (API, "Application Programming Interface"), y que se encuentran disponibles a través de los archivos de biblioteca KERNEL.EXE, USER.EXE, y GDI.EXE. Estas bibliotecas son del tipo de encadenamiento dinámico (DLL**,** "Dynamic Link Library), por lo que su acceso se efectúa al momento de ejecución del programa de aplicación, evitándose así el desperdicio de espacio de almacenamiento en disco ocasionado por la presencia de código objeto duplicado.

El archivo KERNEL proporciona los servicios del sistema, tales como el manejo de recursos, la ejecución de aplicaciones, la administración de memoria, y la multitarea; por su parte, la biblioteca GDI ("Graphics Device Interface", Interface de Dispositivo Gráfico) dirige las operaciones para crear gráficos tanto en el monitor como en la impresora, interactuando con el manejador de video del sistema. Finalmente, la función de USER es crear y mantener a las ventanas de aplicación, así como procesar los comandos del teclado y el ratón para mover, cambiar de tamaño o cerrar ventanas.

Por otra parte, Windows lleva a efecto la administración de los recursos del sistema, incluyendo no solo aquellos referentes al hardware (principalmente dispositivos de entrada/salida), sino también los suministrados por el Windows mismo, tales como el menú de sistema, las abreviaturas o aceleradores del teclado, los íconos y mapas de bits, entre otros, así como las ventanas mismas. Dada esta gestión, las aplicaciones no deben pretender el acceso directo a los dispositivos de entrada/salida ni a las unidades de memoria.

No obstante ser Windows un sistema de explotación multitarea, la interrupción de un programa para pasar el control a otro programa se rige por un esquema de tipo no prioritario. Esto significa que un proceso no es interrumpido, sino que se interrumpe así mismo para permitir que otros programas se puedan ejecutar; todo en base al llamado **paradigma de paso de mensajes**. Para un programa Windows, los mensajes constituyen las unidades de tratamiento, las cuales producen la verdadera repartición de tiempo en periodos en los cuales la aplicación es realmente ejecutada. En la versión Windows95, si bien el sistema presenta un método diferente para manejo de multitarea (ver recuadro), también se sigue permitiendo el esquema de paso de mensajes, con la finalidad de conservar compatibilidad con todos los programas de aplicación desarrollados para versiones de Windows anteriores a esta.

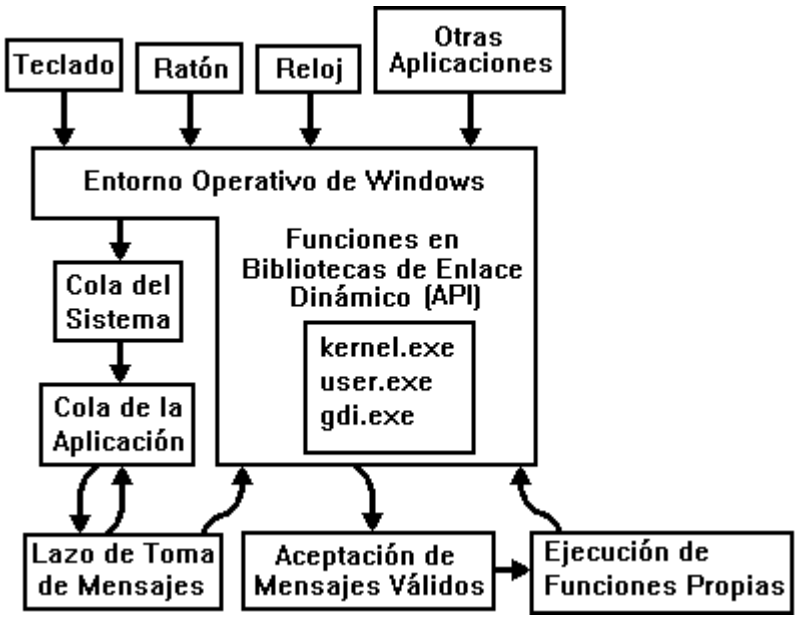

Figura 1. Paradigma del Paso de Parámetros

Paradigma del Paso de Mensajes

. Para el manejo y ejecución de los programas de aplicación, Windows se diseñó sobre el modelo conocido como **paradigma de paso de mensajes**. El esquema de este modelo se muestra en la **figura 1**.

En general, se dice que un mensaje (en el contexto Windows) es la notificación de la ocurrencia de un evento, el cual puede requerir que se produzca alguna acción específica. Como se observa en la figura, los mensajes son recibidos primeramente por Windows, quien los toma de la cola del sistema y los coloca en la cola de la aplicación requerida, informando a la misma de la presencia de un evento.

Cada vez que Windows envía un mensaje a una aplicación, esta se activa y recibe tiempo del procesador; de hecho, la aplicación solo dispone de atención del procesador al recibir un mensaje. Por ello, normalmente se dice que la tarea principal de cualquier aplicación para Windows es procesar mensajes. Sin embargo, es importante considerar que la aplicación debe compartir el tiempo del procesador y no intentar tomar el control exclusivo, ya que aunque Windows es un sistema multitarea, no puede obtener el control del procesador a menos que la aplicación lo libere después de ejecutar algún mensaje.

Básicamente, existen cuatro fuentes para la generación de mensajes, que son: el usuario (por medio de los dispositivos de entrada/salida), el sistema Windows, la aplicación misma, y otras aplicaciones en ejecución.

Las Ventanas y los Menús

Cada vez que se inicia una aplicación se crea una interface visual entre el usuario y el programa, denominada **ventana** (de ahí el nombre de Windows); toda aplicación tiene una ventana principal que sirve como su referencia. Cualquier ventana es tratada por Windows como un objeto, por lo que debe existir una clase **(class)** para modelarla (en programación orientada a objetos, un objeto es una particularización sobre un modelo constituido por una estructura de datos y funciones diversas que actúan sobre esa estructura, denominándose clase a dicho modelo).

¿Qué es Windows95?

Quizá la característica más importante de Windows95 **(W95)** es ser un sistema operativo completo en 32 bits, lo que elimina la necesidad de contar con una copia separada de MS-DOS; sin embargo, aún proporciona compatibilidad con aplicaciones escritas en 16 bits para versiones anteriores de Windows. W95 puede ejecutar cuatro tipos diferentes de programas: aquellos escritos para MS-DOS, los generados para Windows 3.1 **(W3.1)**, los escritos para Windows NT y los diseñados específicamente para W95, creándose automáticamente el medio ambiente adecuado para el tipo de programa presente.

W95 y W3.1

Desde el punto de vista del usuario, W95 proporciona una interface gráfica mejorada, basada en una organización centralizada para manejo de documentos **(folders)**. Así, el estilo de las ventanas ha sido modificado y se han agregado nuevos elementos de control, con la finalidad de permitir que cualquier usuario pueda ejecutar cualquier aplicación, aún sin un entrenamiento previo sobre el manejo del sistema.

Dado el enfoque visual de Windows, la ventana constituye el dispositivo primario para entrada/salida al medio ambiente de operación. Windows maneja las características estándar de las ventanas tales como el marco, las barras de desplazamiento y las barras de título, mientras que la aplicación se encarga del resto, destacando en ello la llamada área del cliente **(client area)**.

Un menú es una lista de comandos a desarrollar por la aplicación; aunque el programador determina los comandos específicos, el medio ambiente Windows se encarga de controlar la barra de menú (el seleccionador de las distintas opciones). Así mismo, existe un menú especial conocido como menú del sistema, el cual ya está determinado (con comandos fijos) como uno más de los recursos que ofrece Windows a sus usuarios.

Desarrollo de Aplicaciones

La filosofía de trabajo de Windows implica el que sus aplicaciones presenten características muy particulares en relación a los programas diseñados para MS-DOS. El desarrollo e implantación de aplicaciones para Windows debe contemplar el manejo de nuevas herramientas tales como los editores y los compiladores de recursos. Usualmente, una aplicación utiliza sus propios recursos, los cuales deben predefinirse en un archivo especial denominado **archivo de recursos**. Para la creación y complementación de estos archivos se emplean los editores de recursos (por ejemplo, WorkShop de Borland); posteriormente estos archivos se compilan utilizando un **compilador de recursos** (tal como RC, del mismo Borland), Los archivos de recursos no se usan directamente por el programa, sino que se encadenan al archivo .EXE de la aplicación.

Además del archivo de recursos (.RES), las aplicaciones también incluyen los siguientes archivos:

- Archivo de Definición del Módulo (.DEF),
- Archivo Fuente de la Aplicación (.C o .CPP) y,
- Archivo(s) de Encabezado (.H).

El archivo de definición del módulo es un archivo tipo texto (ASCII) que especifica, entre otras opciones, el nombre de la aplicación y los nombres de las funciones que servirán como punto de entrada a la misma, así como información al programa

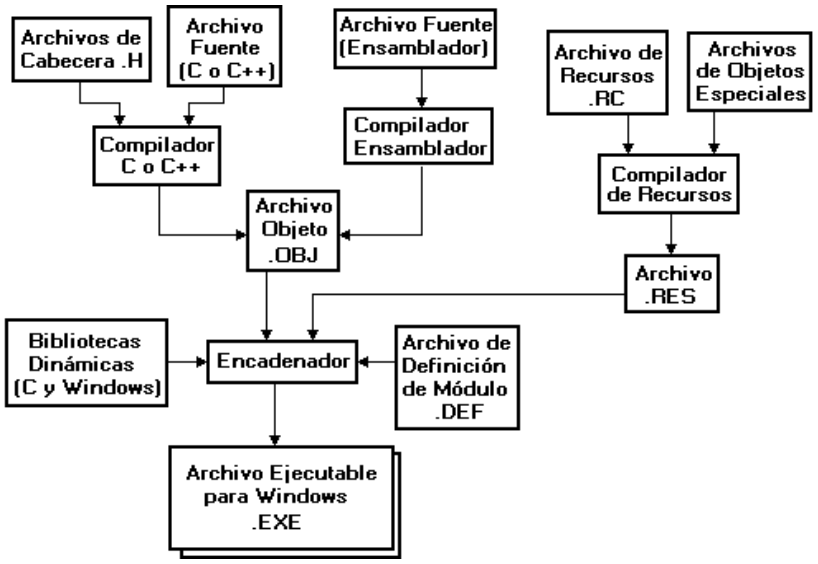

Figura 2. Procesamiento de archivos

Desde el punto de vista del programador, W95 presenta dos grandes mejoras. La primera de ellas la constituye el uso de memoria virtual y el direccionamiento completo a 32 bits, lo que permite a una aplicación disponer de hasta 4 GB de memoria virtual lineal. La segunda innovación corresponde a la multitarea. En W3.1 las aplicaciones corren concurrentemente por medio de un método conocido como multitarea cooperativa, en el que la aplicación verifica periódicamente la cola de mensajes para continuar o regresar control al sistema operativo. W95 también soporta esta multitarea en base a procesos, usando el paradigma de paso de mensajes, aunque normalmente se utiliza un segundo método, denominado multitarea por bloques **(thread-based multitasking)**.

Un bloque es una entidad ejecutable que pertenece a un proceso simple, que incluye además de su código un contador de programa, una pila para modo usuario, otra pila para modo kernel, y un conjunto de valores de los registros; todos los bloques en un proceso tienen igual acceso al espacio de direcciones del procesador, a los manejadores de objetos, y a los demás recursos del sistema. En W95 estos bloques se implementan como objetos, y se ejecutan e interrumpen en base a tiempo, correspondiendo al sistema operativo dicha temporización; de esta forma, es el sistema directamente quién determina el intercambio, ejecución e interrupción de tareas.

encadenador sobre los atributos de los segmentos de código y de datos, y la declaración del tamaño de las áreas de heap y de stack. El archivo fuente, por su parte, contiene el código para el manejo de la aplicación, mientras que en el archivo de encabezado se declaran las definiciones de variables, las listas de parámetros y los prototipos de las funciones empleadas en el archivo fuente. Como ya se mencionó anteriormente, en el siguiente artículo de esta serie se explicarán con mayor detalle estos archivos, principalmente en lo referente a la estructuración del archivo fuente, y la descripción de recursos propios en el archivo de definición de los mismos..

En la **figura 2** se ilustran los 4 archivos y el procesamiento necesario para obtener el programa .EXE final.

Windows maneja un mecanismo especial para encadenar a las aplicaciones con las funciones en los archivos de biblioteca, todo ello a tiempo de ejecución. Para conseguirlo, las aplicaciones están formuladas en un nuevo formato de archivo .EXE, conocido como **nuevo ejecutable**.

Bibliografía

]1] Adams, Lee. *"Programación Avanzada de Gráficos en C para Windows"*. Windcrest/McGraw-Hill. 1993.

[2] Ezzell, Ben. *"Windows 3.1 Graphics Programming"*. ZD Press. 1992.

[3] Farrell, Tim & Runnoe, Connally. *"Programming in Windows 3.1"*. QUE Corporation. 1992.

[4] Klein, Mike. *"Windows Programmer's Guide to DLLs and Memory Management"*. SAMS. 1992.

[5] Microsoft. *"Microsoft Windows95 Resource Kit"*. Microsoft Press. 1995.

[6] Murray, William H. & Pappas, Chris H.. *"Windows 3.1 Programming"*. Osborne/Mc Graw-Hill. 1992.

[7] Schildt, Herbert. *"Schildt's Windows 95 Programming in C and C++"*. Osborne/Mc Graw-Hill. 1995.