Manejo de Archivos en visual basic - parte 7

Volver al índice

En esta parte se describen los conceptos básicos para el tratamiento y acceso a los archivos en visual basic

---

-> > Contenido

---

1 -introducción

En esta sección se ve un repaso básico de las tres formas de trabajar con diferentes tipos de archivos en visual basic: Archivos secuenciales, archivos aleatorios y archivos binarios.

Por último en el final de la sección, se describe como utilizar sentencias y funciones propias de visual basic para trabajar con archivos y directorios del sistema operativo, como por ejemplo la sentencia Dir para buscar ficheros, Filecopy para copiar , ChDir, Mkdir para crear carpetas y algunas otras funciones relacionadas.

---

2 - Sentencia Open

Esta sentencia es la encargada de abrir o crear un archivo, ya sea para leer datos del mismo, sobre escribirlos y/o para agregarle datos.

Esta sentencia tiene una serie de parámetros que varían de acuerdo a lo que queramos hacer.

Por ejemplo :

```
Open "c:\prueba.txt" For Input As #1
```

Este ejemplo abre un archivo para poder acceder a los datos que contenga en su interior. cuando decimos abrir, estamos diciendo que se encuentra actualmente abierto y alojado en la memoria ram para poder manipularlo.

En el primer parámetro indicamos el **path del archivo** en el que queremos trabajar.

Luego la palabra "As #1" indica el número de archivo que estamos utilizando. Este número lo asignamos nosotros y va desde el rango 1 al 511. El "número de archivo" se utiliza para poder diferenciar al archivo en el código. Cada archivo que se encuentra abierto no se puede asignar un número de archivo igual, ya que nos daría un error en tiempo de ejecución.

Por ejemplo no podríamos hacer esto:

```
Open "archivo1.txt" For Input As #1
Open "otroarchivo1.txt" For Input As #1
```

Una vez que terminamos de trabajar con un archivo que se encuentra abierto y que no lo vamos a utilizar mas, debemos cerrarlo. para ello se utiliza la **sentencia Close** seguido del número de archivo que lo identifica.

Por ejemplo:

```
Close #2
```

Esto cerrará al archivo abierto que anteriormente le asignamos el número de archivo 2. También si tengo 3 archivos abiertos podría hacer lo siguiente:

```
Close #1, #2, #3
```

Si utilizamos la sentencia Close sin ningún parámetro, o mejor dicho ningún número de archivo, se cerrarán todos los archivos cargados en memoria por nuestra aplicación (los que nosotros abrimos con Open).

Por lo general, si trabajamos con varios archivos abiertos simultáneamente, es aconsejable utilizar la sentencia Close sin ningún parámetro, de este modo cuando termine de ejecutarse el procedimiento se cerrarán todos los archivos abiertos.

Hay una función en Visual basic llamada FreeFile. Esta función lo que hace es darnos un número

de archivo que esté libre y que se pueda usar, para que de este modo no intentamos abrir uno esté siendo utilizado, y evitar un error en tiempo de ejecución. Para usar la función es muy fácil, por ejemplo:

```
Dim NumeroArchivo As Integer

NumeroArchivo = FreeFile

Open "La ruta de un archivo" For Input As #NumeroArchivo
```

3 - Archivos secuenciales

Los archivos secuenciales se denominan de esta manera por que la forma de escribir y leer los datos en un archivo es, desde el principio hasta el fin del archivo, es decir, si yo quisiera acceder a un determinado dato del archivo y este dato se encuentra en la mitad del archivo, para llegar a ese dato necesito pasar por todos los demás datos, de forma secuencial.

Por lo general se suelen utilizar los archivos secuenciales, para trabajar con archivos que contengan una estructura de datos no muy compleja. por que por ejemplo, si utilizáramos un archivo para almacenar 50000 nombres con sus respectivos datos (apellido, teléfono, dirección etc...) , este mecanismo resultaría ineficiente, ya que si quisiera recuperar por ejemplo el registro n° 3650, para leer este dato tendría que pasar previamente por los 3649 registros anteriores a este, haciendo mas lento el acceso y por lo tanto desperdiciando recursos del sistema.

4 - Diferentes tipos de acceso

Append: esta sentencia se utiliza para agregar información a un archivo de texto.

Ejemplo :

```
Open "c:\miarchivo.txt" For Append As #1
```

Si el archivo ya contiene datos, se le agregarán al mismo al final del archivo, si no contenía datos los agrega igual. Si el archivo no existe, lo crea y le agrega los datos.

Input: la sentencia Input se utiliza para leer datos de un archivo de texto, ejemplo:

```
Open "c:\miarchivo.txt" For Input As #2
```

Este ejemplo abre un archivo para leer los datos del mismo. En las próximas líneas veremos ejemplos de como leer los datos , escribir y guardar.

Una cosa importante con respecto a leer datos de un archivo con Input es que, si el archivo que queremos abrir, no existe, se producirá un error en tiempo de ejecución al intentar abrir un archivo que no existe, por eso debemos verificar la ruta del mismo siempre que esté bien escrita en la sentencia Open y evitarnos dolores de cabeza, y simpre añadir algún manejador de error para este caso

Output: esta sentencia se utiliaza para crear un archivo de texto y grabar datos. Esta es igual que

Append salvo por un gran detalle:

Output crea el archivo y le agrega información, pero si el archivo existía y contenía información previa, **sobre escribe** todos los datos del archivo por los datos nuevos, **perdiendo los anteriores datos**.

Si accedemos con Output a un archivo que no existe, no se produce un error, si no que se crea dicho archivo.

Conclusión : si vamos a añadir datos a un archivo existente (para actualizarlo por ejemplo) hay que utilizar la sentencia **Append**. si vamos a crear un archivo vacío y nuevo para grabar nuevos datos, hay que utilizar la **sentencia Output**. si vamos a abrir un archivo para **leer datos** utilizamos la sentencia **Input**.

5 - Leer datos de un archivo utilizando Line Input

C omo se comentó antes , para leer datos de un archivo se utiliza la sentencia Input.
pero para leer los datos del archivo y poder utilizarlos en nuestro programa se utilizan 2 sentencias o instrucciones: **Line Input** e **Input**.

Ejemplo:

```
Dim mivariable As String

Open "c:\nombres.txt" For Input As #1

While Not EOF(1)

Line Input #1, mivariable

Wend

Close #1
```

En el ejemplo anterior aparece una función nueva llamada **EOF**. Esta función significa End Of File o fin de archivo.

Cuando abrimos un archivo para leer información con Input, debemos saber en que momento llegamos al **final del archivo**, por que de lo contrario la sentencia Line Input o Input seguiría leyendo líneas donde no existen datos después del fin del archivo, y por lo tanto se produciría **un error en tiempo de ejecución por querer leer un dato que no existe**.

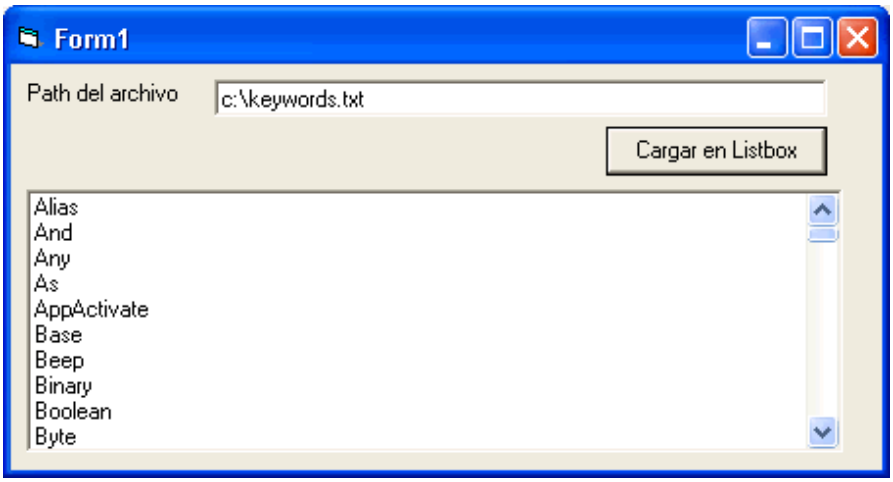
Mediante la condición Not Eof(1) leeremos los datos hasta que llegue al final del archivo, y en cada pasada del bucle While se almacenará en la variable mivariable, línea por línea los datos del mismo. Es evidente que como está planteado el ejemplo habría que manipular los datos leídos y agregar otras instrucciones, por que en este ejemplo la variable mivariable solo almacenaría los datos de la línea que está leyendo y borrando los datos de las líneas que almacenó anteriormente. Mas adelante veremos como solucionar esto.

conclusión : para leer datos de un archivo se utiliza la sentencia Line Input y luego de la coma debemos poner una variable nuestra para almacenar los datos que se van leyendo de la línea actual

```
Line Input #1, nuestravariable
```

6 - Ejemplo para leer un archivo de texto utilizando Line Input

Lo siguiente, lee un archivo de texto línea por línea, y carga el contenido en un control ListBox. colocar un control List1, un Command1 y un Text1 para indicar el path del archivo a leer:




```
Option Explicit


Private Sub Command1_Click()
On Error GoTo errSub

Dim n_File As Integer
Dim Linea As String

    'Elimina el contenido del listbox
    List1.Clear

    'Número de archivo libre
    n_File = FreeFile

    'Abre el archivo para leer los datos
    Open text1.Text For Input As n_File

    'Recorre linea a linea el mismo y añade las lineas al control List
    Do While Not EOF(n_File)

        'Lee la linea
        Line Input #n_File, Linea
        List1.AddItem Linea
    Loop

Exit Sub
errSub:
'error
MsgBox "Número de error: " & Err.Number & vbNewLine & _
       "Descripción del error: " & Err.Description, vbCritical
End Sub

Private Sub Form_Load()
```

```
Command1.Caption = " Cargar en Listbox "

End Sub
```

Este otro ejemplo, lo que hace es leer un archivo de texto para contar las líneas del mismo

Colocar un command1 en el formulario y pegar el siguiente código fuente

```
Public Function Contar_Lineas(ByVal strTextFile As String) As Long

    Dim F As Integer
    Dim Lineas As Long
    Dim str_Linea As String

    ' Número de archivo libre
    F = FreeFile

    ' Abre el archivo de texto
    Open strTextFile For Input As #F

    'Recorre todo el archivo de texto _
     linea por linea hasta el final
    Do
        'Lee una línea
        Line Input #F, str_Linea

        ' Incrementa la cantidad de lineas leidas
        Lineas = Lineas + 1

    ' Leerá hasta que llegue al fin de archivo
    Loop While Not EOF(F)

    ' Cierra el archivo de texto abierto
    Close #F

    ' Retorna a la función el número de lineas del fichero
    Contar_Lineas = Lineas

End Function

Private Sub Command1_Click()

    Dim Path As String

    Path = InputBox(" Ingrese la ruta de un archivo de texto ", _
                    " Contar líneas ")

    If Path = vbNullString Then Exit Sub

    MsgBox " Cantidad de lineas: " & Contar_Lineas(Path)

End Sub
```

7 - Leer el contenido de un archivo de texto con la función Input

La función input, a diferencia de Line Input que Lee por línea, Input lee todo el contenido del archivo de una sola ves, es decir no necesitamos, como en el ejemplo anterior, utilizar un bucle.

El modo de usarlo es el siguiente:

Primero se abre el archivo con Open, por ejemplo:

```
Open Path_Archivo For Input As #1
```

Luego se asigna a una variable, el contenido que devolverá la función Input:

Contenido = Input(LOF(1), #1)

Un ejemplo:

Lo siguiente lee el contenido de un archivo txt y lo carga en un textBox multilínea

Colocar un TextBox llamado txt_Path (para especificar la ruta del archivo).

El Textbox que mostrará los datos del fichero, colocarle el nombre txt_Contenido

Código en un botón:

```
On Error GoTo Err_Sub

Dim n_File As Integer
Dim Contenido As String


    'Número de archivo libre
    n_File = FreeFile

    'Abre el archivo indicado
    Open txt_Path For Input As n_File

    'Lee todo los datos del archivo y lo almacena en la variable
    Contenido = Input$(LOF(n_File), #n_File)

    'Cierra el archivo abierto
    Close n_File

    'Carga el contenido de la variable en el TextBox
    txt_Contenido = Contenido

    Exit Sub

Err_Sub:

    MsgBox Err.Description, vbCritical
```

8 - Escribir datos en archivos de texto

Para escribir datos en archivos de texto se utiliza la sentencia Print y Write.

Ejemplo:

```
Dim nombre As String
                Dim edad As Integer
                Dim telefono As String

Open "c:\miarchivo.txt" For Append As #1
Print #1, "esta es una línea"
                  Print #1, "esta es otra línea"
                  Print #1, nombre, edad, telefono
Close #1
```

En este ejemplo agregamos datos a un archivo existente llamado miArchivo.txt con la **sentencia Print**.

Primero abrimos el archivo con Open. luego la sentencia Print lleva 2 parámetros. el primero indica el archivo que estamos manipulando (el archivo n°1, dentro del código), el segundo parámetro indica los datos que se agregarán en el mismo. En el segundo parámetro podemos poner un dato directamente (una cadena, número, etc...),, por último cerramos el archivo.

Escribir datos con la sentencia Write

A diferencia de Print, la **sentencia Write** escribe datos en un archivo separados por comas.

Ejemplo:

```
Dim nombre As String
Dim apellido As String

apellido = "Peres"

nombre = "Carlos"

Open App.Path & "\Archivo.txt" For Output As #1
```

```
Write #1, nombre, apellido

Close #1
```

Como podemos ver la sentencia Write escribe los datos en el archivo separados por coma. En la primera línea escribirá el contenido de las variables nombre y apellido . También se pueden pasar los datos directamente a Write sin usar variables, por ejemplo:

```
Write #1, "Un dato", "Otro dato", "Otro mas"
```

Nota importante: cuando escribimos datos con la sentencia Print se utiliza la sentencia Line Input para leer los datos del archivo. En cambio cuando escribimos datos separados por comas con la **sentencia Write** debemos utilizar la **sencia Input**

Para que esta página no sea tan extensa, he armado 10 ejercicios que utilizan todo lo visto hasta aquí, o sea el tema relacionado a los archivos secuenciales. Dentro del código se encuentra detallado cada paso y lo podés descargar al final de esta página.

---

9 - Archivos aleatorios o directos

A diferencia de los archivos secuenciales, **los archivos aleatorios** almacenan datos en forma de registros. Como habíamos dicho en el capítulo anterior para leer datos de un archivo secuencial había que leer todo el archivo, es decir que no podíamos leer por ejemplo los datos que estuviesen en la línea 35 del mismo sin antes pasar por todos los datos anteriores, por eso su nombre de archivo secuencial.

En cambio los archivos aleatorios, también llamados archivos directos, almacenan los datos con una estructura diferente. Los datos se guardan en registros mediante una estructura definida de tipo Type (**estructura definida por nosotros**) también llamada UDT

Por ejemplo si tuviesemos 25 registros, cada uno con datos (apellido, email, telefono,etc..), y quisiera acceder al registro 17, puedo leer los datos del registro 17 sin tener que leer los 16 registros anteriores, ganando con ello mas velocidad y teniendo una estructura de datos definida.

Para **abrir un archivo aleatorio** para trabajar con él, se utiliza la sentencia Open con algunos cambios en lo que se refiere a los archivos secuenciales

Ejemplo:

```
open "elarchivo.dat" for random as #1 len = len(mivariable)
```

Como podemos ver para abrir un archivo de acceso aleatorio se utiliza la palabra **Random** (aleatorio). Luego debemos indicar el **número de archivo** para identificarlo, y por último una opción nueva : **Len**.

cada registro en el archivo, que es una estructura de datos Type, tiene que tener una longitud fija. Mediante la función Len de visual Basic debemos indicar el tamaño de la estructura de datos que vamos a utilizar. para que quede mas claro un ejemplo:

Primero definimos una estructura de datos:

```
Private Type t_clientes

nombre As String * 20
apellido As String * 15
dirección As String * 30
edad As Integer

End Type
```

Después creamos una variable llamada clientes de tipo t_clientes

```
Dim clientes As t_clientes
```

ahora abrimos nuestro archivo, y en el parámetro de la función Len le pasamos la variable para que visual basic calcule el tamaño de la estructura t_clientes

```
Open "datos.dat" For Random As #1 Len = Len(clientes)
```

La estructura de datos debe tener un tamaño fijo, no datos variables como en el caso de los archivos secuenciales, de ahí que se indicara en el Type en las variables mediante el asterisco en el caso de los String.

En cada campo de la estructura de datos debemos indicar el tamaño de la variable. En el caso del campo nombre de tipo string le hemos asignado una longitud fija de 20 bytes, en el apellido 15 y en la dirección 30 bytes. La variable o el campo edad que es de tipo integer, no debemos indicar el tamaño de la variable, ya que sabemos que un número integer ocupa 2 bytes.

En total esta estructura t_clientes tiene una longitud de 67 bytes (30 + 20 + 15 + 2).

Si no indicáramos en la sentencia Open el tamaño de la estructura, visual basic, por defecto asumiría la estructura de un tamaño de 128 bytes, y si nuestra estructura tendría mayor tamaño ocurriría un error en tiempo de ejecución, por lo cual siempre debemos utilizar la función len para calcular el tamaño de la misma.

10 - Grabar datos en archivos aleatorios

Para grabar datos en un archivo de acceso aleatorio se utiliza la **sentencia Put**.

Un ejemplo paso a paso:

primero declaramos una estructura de datos que contendrá la información de cada registro:

```
Private Type t_empleados
nombre As String * 15
apellido As String * 15
dirección As String * 30
edad As Integer
End Type
```

Luego creamos una variable que sea de tipo **t_empleados**.

```
Dim empleados As t_empleados
```

Ahora abrimos el archivo e indicamos en la función Len el tamaño de la estructura

```
Open "datos.dat" For Random As #1 Len = Len(clientes)
```

Ahora le asignamos un valor a cada campo de la estructura de esta forma:

```
empleados.nombre = "Carlos"
empleados.apellido = "Martinez"
empleados.dirección = "Avenida 9 de julio n° 2054"
empleados.edad = 45
```

Ahora grabamos en el registro n°1 del archivo los datos de esta forma:

```
Put #1, 1, empleados
```

Si queremos grabar mas datos, por ejemplo en el registro n° 2 del archivo hacemos lo siguiente:

```
empleados.nombre = "María"
empleados.apellido = "Gonzales"
empleados.dirección = "Avenida 13 n° 1100"
empleados.edad = 35

Put #1, 2, empleados

Close #1
```

como vemos ver la sentencia Put lleva 3 parámetros. El primero indica el numero de archivo abierto en el cual estamos trabajando. en el segundo debemos indicar el número de registro en el que se grabarán los datos. Si no ponemos el número de registro, los datos se grabarán en el último registro. Por último en el tercer parámetro le pasamos la variable asociada con la estructura de datos.

11 - Leer datos en archivos aleatorios

Para leer los registros o datos de un archivo aleatorio se utiliza la **sentencia Get**. Esta sentencia es exactamente igual que Put, pero la diferencia es que en vez de grabar los datos **los recupera** para poder utilizarlos luego en el programa.

12 - Posicionarse en un registro determinado

Supongamos que tenemos un archivo aleatorio que contiene 452 registros. Ahora queremos recuperar los datos del registro 258. Para posicionarse en un determinado registro hay una sentencia , hay una **sentencia llamada Seek**. Ejemplo:

```
Seek #1, 258
Get #1, , mivariable
```

si queremos posicionarnos en un registro determinado, pero en vez de leer, queremos grabar datos, hacemos la misma operación pero con la sentencia Put:

```
Seek #1, 258
Put #1, , mivariable
```

Nota: Desde este enlace podés ver un simple **código fuente de una Agenda de contactos utilizando Archivos Directos - Aleatorios**

---

13 - Funciones propias de vb para el manejo de ficheros y directorios

Visual basic posee varias funciones para manejar archivos y directorios de nuestro sistema. Las principales son.

14 - Función Dir

La función Dir se utiliza o para buscar archivos y devuelve una cadena que representa el nombre de un archivo o directorio de acuerdo a un determinado patrón de búsqueda. La sintaxis de la función es:

```
Dir (ruta, atributos)
```

En el parámetro ruta debemos indicar el path de un archivo, directorio o unidad. Si el path no existe, la función Dir devuelve una cadena vacía.

En el parámetro atributos podemos especificar los siguientes:

- vbnormal : cualquier atributo de archivo.
- vbreadonly : archivos de solo lectura.
- vbhidden: archivos ocultos
- vbsystem : archivos de sistema
- vbvolume : etiqueta de volumen de una unidad
- vbdirectory : directorios

Ejemplo:

Si el archivo existe Dir devuelve "autoexec.bat"

```
Archivoabuscar = Dir("c:\autoexec.bat")
```

También podemos utilizar los comodines * y ? para filtrar búsquedas.

En este ejemplo devuelve el primer archivo exe que encuentra, si es que existe.

```
Archivoabuscar = Dir("c:\windows\*.exe")
```

En este ejemplo la función Dir devuelve directorios además de archivos

```
Archivoabuscar = Dir("c:\, vbdirectory")
```

El único inconveniente de esta función es que siempre devuelve el primer archivo que encuentra. Para que continúe buscando los demás archivos debemos poner la función sin parámetros. Ejemplo

```
Dim archivo As String

archivo = Dir("c:\*.exe")

While archivo <> ""
archivo = Dir
Wend
```

En el ejemplo anterior buscará todos los archivos exe en el directorio c:\. cuando ya no encuentre mas devolverá una cadena vacía y saldrá del bucle While.

Este otro ejemplo verifica si un archivo existe:

```
Private Function Verificar_Existe(path) As Boolean

    If Len(Trim$(Dir$(path))) Then
        Verificar_Existe = True
    Else
        Verificar_Existe = False
    End If

    MsgBox Verificar_Existe

End Function

Private Sub Form_Load()

Call Verificar_Existe("c:\autoexec.bat")

End Sub
```

El siguiente enlace muestra un ejemplo de como utilizar la función Dir para **buscar archivos y también ordenarlos enforma alfabética** y luego agregar la lista en un control Listbox

Este otro enlace muestra como **buscar ficheros mediante el Api de windows**

15 - Función FileCopy

FileCopy nos permite copiar archivos. Esta función es muy fácil de usar. ejemplo:

FileCopy "origen", **"destino"**

Nota: si el archivo existe la función lo sobre escribe.

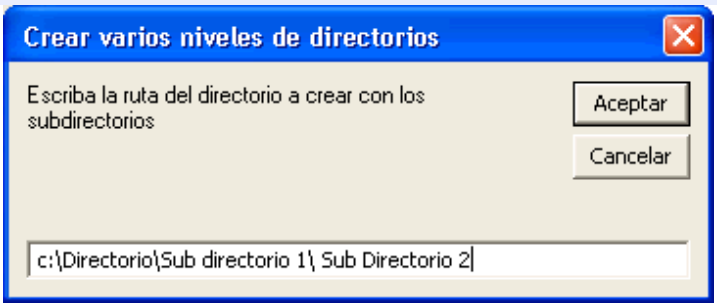
16 - Función MkDir

Esta función se utiliza para crear directorios, pero crea un directorio por vez, es decir que no crea varios niveles al mismo tiempo. La sintaxis es:

MkDir "Ruta directorio"

para poder crear un Directorio que contiene varios subdirectorios, podemos hacer lo siguiente:

El ejemplo despliega un InputBox para ingresar el path a crear:




```
Private Sub Command1_Click()

Dim i As Integer
Dim Array_Dir As Variant
Dim Sub_Dir As String
Dim El_Path As String

El_Path = InputBox("Escriba la ruta del directorio a crear" & _
          "con los subdirectorios", " Crear varios niveles de directorios")

If El_Path = vbNullString Then Exit Sub

    'Desglosa el path
    Array_Dir = Split(El_Path, "\")

    El_Path = vbNullString

    'Recorre el array anterior para ir creando uno por uno
    For i = LBound(Array_Dir) To UBound(Array_Dir)

        Sub_Dir = Array_Dir(i)

        If Len(Sub_Dir) > 0 Then
           El_Path = El_Path & Sub_Dir & "\"

           If Right$(Sub_Dir, 1) <> ":" Then
              ' Verificamos que no exista
              If Dir$(El_Path, vbDirectory) = vbNullString Then
                 'Crea la carpeta
                 Call MkDir(El_Path)
              End If
           End If
        End If
    Next i

End Sub
```

17 - Función kill

La función kill se utiliza para eliminar archivos. También es muy fácil de utilizar, y el único parámetro que lleva es la ruta del archivo que queremos eliminar. Si queremos eliminar varios archivos de un directorio podemos utilizar el comodín"*", y si queremos excluir ciertos archivos utilizamos el comodín "?".

```
kill "c:\*.txt"
```

18 - Función FileLen

E sta función nos devuelve el **tamaño en bytes** de un archivo.

Ejemplo:

```
Dim tamaño As Long

tamaño = FileLen("c:\windows\system32\control.exe")

MsgBox tamaño & " bytes"
```

FileLen es muy útil para trabajar con archivos aleatorios, ya que si queremos conocer la cantidad de registros que tiene el archivo, debemos dividir el tamaño del archivo por el tamaño de la estructura.

Ejemplo:

```
Private Type t_personas

nombre As String * 20
apellido As String * 20

End Type

Dim cant_reg As Long

Private Sub Form_Load()

cant_reg = FileLen("c:\miarchivo.dat") / Len(t_personas)

End Sub
```

19 - Función Name

Name se utiliza para renombrar archivos y directorios. Esta función no se puede utilizar para renombrar archivos abiertos.

Ejemplo:

Name "path del archivo viejo a renombrar" as "path y nombre nuevo del archivo"

También con esta sentencia podemos mover archivos y directorios a otra ubicación. para ello debemos cambiar en el segundo parámetro la ruta del archivo.

20 - Función GetAttr y SetAttr

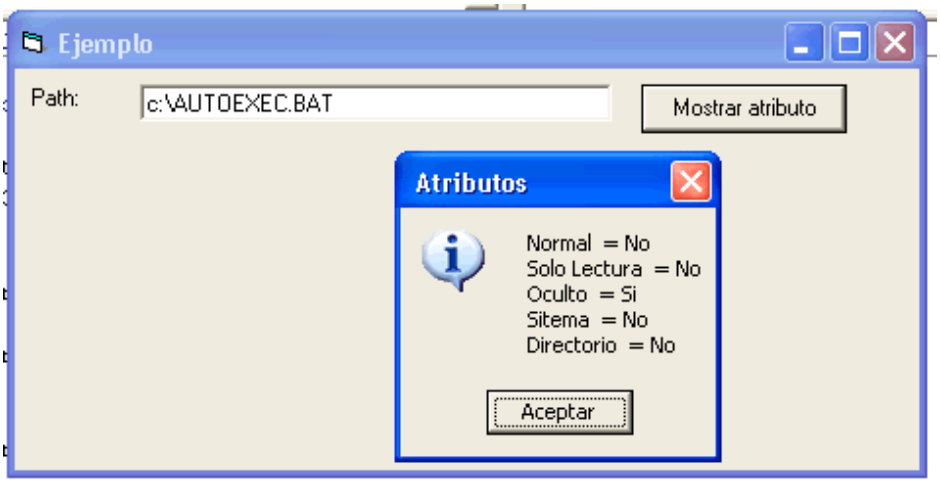
La función Getattr permite recuperar información sobre los atributos de un archivo o directorio. Ejemplo:

```
GetAttr "ruta del archivo"
```

los valores devueltos por la función pueden ser los siguientes:

- 0 - normal
- 1 - solo lectura
- 2 - oculto
- 4 - archivo de sistema
- 16 - directorio
- 32 - archivo modificado

El siguiente ejemplo lo que hace es mostrar en un MsgBox los atributos de un archivo. Colocar un TextBox llamado Text1 y un Command1.



Código en el formulario:

```
Option Explicit

Private Sub Command1_Click()
Dim ret As Long
Dim Atributos  As String
    ret = GetAttr(Text1.Text)

    If ret And vbNormal Then
        Atributos = " Normal  = Si" & vbNewLine
    Else
```

```
        Atributos = " Normal  = No" & vbNewLine
    End If
    If ret And vbReadOnly Then
        Atributos = Atributos & " Solo Lectura  = Si" & vbNewLine
    Else
        Atributos = Atributos & " Solo Lectura  = No" & vbNewLine
    End If
    If ret And vbHidden Then
        Atributos = Atributos & " Oculto  = Si" & vbNewLine
    Else
        Atributos = Atributos & " Oculto  = No" & vbNewLine
    End If
    If ret And vbSystem Then
        Atributos = Atributos & " Sitema  = Si" & vbNewLine
    Else
        Atributos = Atributos & " Sitema  = No" & vbNewLine
    End If
    If ret And vbDirectory Then
        Atributos = Atributos & " Directorio  = Si" & vbNewLine
    Else
        Atributos = Atributos & " Directorio  = No" & vbNewLine
    End If

    'Muestra los atributos del archivo elegido
    MsgBox Atributos, vbInformation, " Atributos "

End Sub
```

La función SetAttr lo que hace es establecer los atributos de un archivo .

Ejemplo:

```
SetAttr "ruta del archivo", valores de los atributos
```

21 - Función RmDir

RmDir elimina directorios o carpetas. Pero antes de eliminar directorios tenemos que estar seguros que la carpeta **no contiene archivos**, de lo contrario nos dará un error en tiempo de ejecución. Si el directorio contiene archivos debemos eliminarlos previamente con la sentencia kill.

Ejemplo:

```
RmDir "path del directorio a remover"
```

Descarga de ejemplos sobre el manejo de archivos en vb - código fuente

Ejemplos sobre el manejo de Archivos secuenciales y aleatorios en Visual Basic

---

Volver

Buscar en Recursos vb

Buscar
