Dialog Wrapper

and Tony Rein
CIS 76276,2662

Preface:

This file uses a system called "DocCruiser™," used courtesy of Woody Leonhard of Pinecliffe International (author of **A Hacker's Guide to Word for Windows**). DocCruiser is simply a method of moving quickly from place to place in a document. It works like this: Text in **THIS COLOR** means "Double-click on me to jump." For example, double-click on the next line to do a practice jump now:

§

OK, are you back? Good! In the table of contents, each heading is a jump button. You can double-click on the text to go directly to the section of your choice, and at the end of each section there's a jump button to bring you back to the this page. Jump buttons are also used several other places in the document just look for anything **this color.**

Table of Contents

What Is Dialog Wrapper?

Dialog Wrapper is a way to let your WordBasic macros use the standard dialog boxes that Windows makes available to programs. Since you use Word for Windows you've seen these boxes — they're the ones you get when you save, rename, or open a file from WinWord's menus. Windows also makes standard dialog boxes available for color selection, font selection, search and replace functions, and printer functions. For the purposes of this document, let's call these dialogs the "**Windows Common Dialog Boxes,**" or "**Common Dialogs**."

It sure would be nice to be able to use these in WordBasic macros. After all, they're standard, and that's the whole idea behind using Windows in the first place, isn't it? A user who knows how to choose a file in any Windows program that uses them (and that's an awful lot) knows how to choose a file in any other. They may not be great art, but they're familiar. If your macros are to be used by other people, the use of the common dialogs will reduce the "hassle factor" for them.

At first glance it would seem that you could access the common dialogs from "stock" WordBasic the "**Declare**" statement is provided just for the purpose of getting at functions in **dynamic link libraries**, and the common dialogs are located in a DLL called COMMDLG.DLL[1], which you probably already have on your system. Unfortunately, though, the functions in COMMDLG.DLL require that you pass them a "**pointer to a structure**," and WordBasic can pass only strings and numbers. Pointers and structures by themselves are out, let alone pointers to structures. If you don't know what pointers and structures are, don't sweat it; you don't need to for WordBasic[2].

That, of course, is where Dialog Wrapper comes in. Dialog Wrapper is a dynamic link library file (DLGWRAP.DLL) containing functions which take and return only strings and numbers, and which act as "front ends" to the various functions in COMMDLG.DLL. To see it in action, select the "File Delete" option from the "File" menu of this document. Here is another quick example. Don't worry if you don't understand the details now for an explanation see **Overview and Installation** and **File-oriented Functions**.

```
REM "Declare" statements go here, outside of "Sub MAIN...End Sub".
REM They tell WordBasic the names of the external functions, where
REM to find them, what parameters are passed, and what kind of value
REM is returned:
Declare Sub cfnSetTitle Lib "DLGWRAP.DLL" (St$ as String)
Declare Sub cfnSetStartDirectory Lib "DLGWRAP.DLL" (St$ as String)
Declare Function cfnRun Lib "DLGWRAP.DLL" As Integer
Declare Function cfnGetFileName$ Lib "DLGWRAP.DLL"  As String

Sub MAIN
        cfnSetTitle("File to Copy:")
        cfnSetStartDirectory("C:\WINWORD\CLIPART\")
        Response = cfnRun    'REM Displays and runs the file selection
```

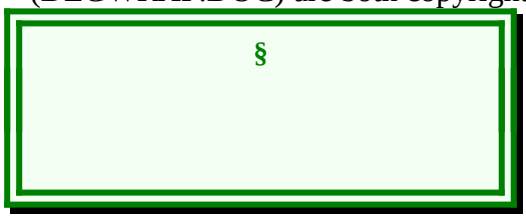
```
dialog box
        If Response = 0 Then
            goto Done       'REM User canceled or there was an error
        Else
            FiletoCopy$ = cfnGetFileName$
            'REM Process the file...
        End If
    Done:
    End Sub
```

It would probably be possible to "roll your own" functions in WordBasic that do everything that the common dialogs do list the files in a directory, allow the user to switch directories, filter file listings by extensions, warn your user if an existing file is about to be overwritten, etc. but I think it would take you a while. In contrast, COMMDLG.DLL is already written and debugged for you by Microsoft, and Dialog Wrapbh SENTRY!CPS#A1B2GUCÈ¶™µ55few steps:

1. Tell WordBasic which functions you plan to use ("Declare" statements)
2. (Optional) Use setup functions for your dialog box's title, etc.
3. Call "cfnRun" and check its return result for cancellation or error
4. Call cfnGetFileName and use the name it returns

Dialog Wrapper is shareware. That is, I (Tony Rein) and my company (Oriole Computing) hereby grant you a license to use it for a reasonable period for evaluation purposes. If, after that period, you'd like to continue using it, you can do so by registering it for a fee of $7.50. For details, see **Registration** and **Legalese**.

The Dialog Wrapper dynamic link library (DLGWRAP.DLL) and documentation (DLGWRAP.DOC) are both copyrighted © 1993 by Tony Rein and Oriole Computing.



Overview and Installation

Limitation:

Dialog Wrapper does not implement access to all the functions in COMMDLG.DLL, but only those I thought would be the most useful the file-oriented ones. I thought hard about the printer, color, font, and search-and-replace functions, but I didn't think there'd be that much application for them in a WordBasic context. If I'm wrong, and you'd be able to use them, please let me know.

Requirements:

Dialog Wrapper requires Windows to be running in "Standard" or "386 Enhanced" mode. It should work fine in Windows 3.0 (although we haven't tested it with anything except 3.1) but will not work in "Real" mode. If you don't know which version of Windows you're running, or what mode it's in, here's how to find out:

1. Press **CONTROL-ESCAPE** to display the list of running programs
2. Select **Program Manager**
3. Press **Alt-H**, and then **A (Help | About Program Manager)**
4. You'll see a box with the version listed near the top, and the mode near the bottom.

Dialog Wrapper does not actually require Word for Windows; it could theoretically be used by any Windows program that can call DLL functions. If you use such a program, all the functions should work fine. Please let me know if you'd like the documentation converted into some other form.

Installation:

Before you use Dialog Wrapper, you must, naturally, install it. To do this, copy DLGWRAP.DLL to your Windows System directory (probably "\WINDOWS\SYSTEM"), and DLGWRAP.DOC to anywhere that's convenient for you. While you're at it, check your Windows System directory for a copy of COMMDLG.DLL; if you don't have one, or the one included with Dialog Wrapper is newer than yours, copy COMMDLG.DLL to your Windows System directory, as well.

You will, of course, be delighted with Dialog Wrapper, and find it well worth the $7.50 registration fee, but if you do want to un-install it, all you need to do is delete DLGWRAP.DLL and DLGWRAP.DOC.

Overview:

Use of Dialog Wrapper involves two basic steps: 1).Tell WordBasic about the functions you're going to use by putting "**Declare**" statements in your macro. 2)Call the functions themselves.

Let's break those steps down a bit:

1) Declare

The purpose of the **Declare** statement is to tell WordBasic about "external functions" that a macro will use. An external function is one that's not part of WordBasic itself, and not defined in WordBasic macros. External functions are usually contained in dynamic link libraries. For details on the syntax of **Declare**, see the WordBasic manual, or on-line help. Briefly, **Declare** gives:

- The function name
- The library where the function can be found.
- The name(s) and type(s) of any parameters passed to the function, and
- The type of the return value, if any.

You actually don't have to understand **Declare** to use Dialog Wrapper the description of each function further down in this file has a **Declare** statement you can just copy and paste into your macro. If WinWord says that it can't find DLGWRAP.DLL, **make sure that DLGWRAP.DLL is in one of the following locations:** 1) The current directory; 2)A directory in your DOS PATH; 3)Your Windows directory; or 4)Your Windows System directory

Here are some examples:

Declare Sub cfnSetTitle Lib "dlgwrap.dll" (NewTitle$ As String)

says that "cfnSetTitle" is to be found in the library DLGWRAP.DLL, that it's a subroutine (as opposed to a function), that it takes one parameter, a string, and that it returns nothing.

Declare Function cfnRun Lib "dlgwrap.dll" As Integer

says that "cfnRun" is a function that takes no parameters, and returns an "integer." (For purposes of this document and the WordBasic **Declare** statement, an integer is the same as a "number.")

Declare Sub cfnResetDefaults Lib "dlgwrap.dll"

says that cfnResetDefaults is a subroutine that takes no parameters and returns nothing.

The **Declare** statements must appear outside of any **"Sub . . . End Sub"** or "**Function . . . End Function"** pairs. Traditionally, they're put at the very top of the macro, before **"Sub MAIN."**

2) Calling the functions

Function names: All functions in DLGWRAP.DLL start with the prefix **"cfn."** This indicates that they are file-oriented functions ("cfn" for "choose file name"). At some future point DLGWRAP.DLL may include functions to implement the font, color, printer, or search-and-replace areas of COMMDLG.DLL; those functions would have

other prefixes to indicate their purposes.

Function categories: Each function in DLGWRAP.DLL is either a setup function, an execute function, or a result return function. The setup functions control how the dialog box will appear or behave; the execute function actually displays and "runs" the dialog box on the screen; and the results functions get the user's choice(s) back into your macro. The setup functions often (but **not** always) have names similar to "**cfnSet...,"** e.g., "cfnSetTitle," "cfnSetStartDirectory," etc. The only execute function at this time is **"cfnRun,"** and the results functions generally have names similar to **"cfnGet...,"** e.g., "cfnGetFileName$," "cfnGetExtension$," etc.

At the point in your macro where you want to use Dialog Wrapper, you will call at least two and probably more functions from DLGWRAP.DLL, in this general order:

1. (Optional) Setup functions. If the defaults are fine, you won't need this step.
2. An execute function.
3. One or more result return functions.

For examples, see the sections "**File-oriented Functions**" and "**Real-World Example.**"

File-oriented Functions

Alphabetical List of File Functions:

§ 17 § 29

Functions by Category:

Setup:

Name:	Comments:
§	Specifies whether user will be allowed to select a read-only file
§	Specifies whether user will be allowed to specify a directory that isn't there
§	Specifies whether user will be allowed to select a file that isn't there
§	Sets several parameters for a default Open File dialog box
§	Sets several parameters for a default Save File dialog box
§	Undoes customization of the dialog box
§	Sets a default extension
§	Selects which files will be displayed when dialog box starts
§	Sets the starting directory

§ Sets the starting file name

§ Sets the title for the dialog box

§ Specifies whether or not user will be warned that the chosen name is the name of an already-existing file

Execution:

§ Displays and executes the dialog box

Result return:

Name: **Comments:**

§ Tells whether or not a given file or path exists.

§ Returns the file name (no path or extension)

§ Returns the directory (no drive or file name)

§ Returns the drive letter and colon.

§ Returns the extension

§ Returns the complete file specification (drive, path, name, and extension, if any)

§ Returns the drive and directory

Function Detail:

Subroutine cfnAllowReadOnly
Syntax:
Declare Sub cfnAllowReadOnly Lib "dlgwrap.dll" (D As Double)
Purpose:
Setup
Description:
Controls whether the next dialog box will allow the user to select or name a read-only file or not. If AllowReadOnly is "turned off," choosing a read-only file will cause the dialog box to display a message indicating that the file exists and is read-only; the user won't be allowed to end the dialog without choosing or naming a file that isn't read-only, or canceling.
Pass:
A "double" (number). Zero turns AllowReadOnly off (that is, the user will **not** be allowed to select a read-only file), and anything else will turn it on.
Returns:
Nothing
See Also:
cfnDirMustExist, cfnDoesItExist, cfnFileMustExist, cfnMakeOpenBox, cfnMakeSaveBox, cfnResetDefaults, cfnWarnOverwrite
Notes:
Example:

```
Declare Sub cfnAllowReadOnly Lib "dlgwrap.dll"  (D As Double)
Declare Function cfnRun Lib "dlgwrap.dll" As Integer
Declare Function cfnGetFileName$ Lib "dlgwrap.dll" As String

Sub MAIN
   cfnAllowReadOnly(1)                    'REM Says that returned file
name can be
   ReturnStatus = cfnRun                  'REM name of a read-only file.
   if  ReturnStatus  <> 0 then
         FileChoice$ = cfnGetFileName$
         'REM Process file name...
   else
         'REM User pressed CANCEL
   end if
End Sub

§
```

Subroutine cfnDirMustExist

Syntax:

Declare Sub cfnDirMustExist Lib "dlgwrap.dll" (D As Double)

Purpose:

Setup

Description:

Controls whether the next dialog box will allow the user to select a directory that doesn't exist. If DirMustExist is "turned off," choosing a non-existent directory will cause the dialog box to display a message indicating that the directory does not exist; the user will not be allowed to end the dialog without naming a directory that exists, or canceling.

Pass:

A "double" (number). Zero turns DirMustExist off (allows user to choose a directory that doesn't exist), and anything else will turn it on.

Returns:

Nothing

See Also:

cfnAllowReadOnly, cfnDoesItExist, cfnFileMustExist, cfnMakeOpenBox, cfnMakeSaveBox, cfnResetDefaults, cfnWarnOverwrite

Notes:

Example:

```
Declare Sub cfnDirMustExist Lib "dlgwrap.dll"  (D As Double)
Declare Function cfnRun Lib "dlgwrap.dll" As Integer
Declare Function cfnGetPath$ Lib "dlgwrap.dll" As String
Declare Function cfnDoesItExist Lib "dlgwrap.dll" (St$ As String) As Integer

Sub MAIN
   cfnDirMustExist(0)
   If cfnRun <> 0 Then
         path$ = cfnGetPath$
         isitthere = cfnDoesItExist(path$)
         'REM path$ may or may not exist.
         'REM If  not, then "isitthere" will be zero.
         If isitthere <> 0 then
               MsgBox path$, "This path exists"
         Else
               MsgBox path$, "This path does not exist"
         End If
   End If
   cfnDirMustExist(1)
   If cfnRun <> 0 Then
         path$ = cfnGetPath$
         'REM We know that path$ must exist.
   End If
```

```
End Sub

§
```

Function cfnDoesItExist

Syntax:

Declare Function cfnDoesItExist Lib "dlgwrap.dll" (S$ As String) As Integer

Purpose:

Return result

Description:

Tells whether the argument passed to it represents a directory of file that actually exists or not.

Pass:

A string the name of file or directory to check.

Returns:

0 if the directory or file in question does **not** exist.
-1 if it does.

See Also:

cfnAllowReadOnly, cfnDirMustExist, cfnDoesItExist, cfnFileMustExist, cfnMakeOpenBox, cfnMakeSaveBox, cfnResetDefaults, cfnWarnOverwrite

Notes:

This function appears unreliable in the following case:

- If you pass it the name of a directory on a CD-ROM drive. For example, on our system, drive E: is a CD-ROM drive, and this function won't work properly with "E:," "E:\BOOKS," etc. In this case it will give a false negative. It will, however, properly detect files that are on a CD.

Example:

```
Declare Function cfnDoesItExist Lib "dlgwrap.dll" (S$ As String) As Integer
Declare Function cfnRun Lib "dlgwrap.dll" As Integer
Declare Sub cfnFileMustExist Lib "dlgwrap.dll" (D As Double)
Declare Function cfnGetFileName$ Lib "dlgwrap.dll" As String

Sub MAIN
   cfnFileMustExist(0)
   If cfnRun = 0 Then
         Goto Done      'REM User canceled.
   End If
   choice$ = cfnGetfileName$
   If cfnDoesItExist(choice$) = 0 Then
         response = MsgBox(choice$, "This file doesn't exist. Should I create
it?", 36)
         If response = -1 Then   'REM User selected "Yes" button
               'REM Create file ...
         Else
               Goto Done
         End If
   'REM Process file....
   '....
```

```
    End If
Done:
End Sub
```

§

Subroutine cfnFileMustExist

Syntax:

Declare Sub cfnFileMustExist Lib "dlgwrap.dll" (D As Double)

Purpose:

Setup

Description:

Controls whether the next dialog box will allow the user to select a file that doesn't exist. If FileMustExist is "turned off," choosing a non-existent file will cause the dialog box to display a message indicating that the file does not exist; the user will not be allowed to end the dialog without naming one that exists, or canceling.

Pass:

A "double" (number). Zero turns FileMustExist off (allows user to choose a file that doesn't exist), and anything else will turn it on.

Returns:

Nothing

See Also:

cfnAllowReadOnly, cfnDirMustExist, cfnDoesItExist, cfnMakeOpenBox, cfnMakeSaveBox, cfnResetDefaults, cfnWarnOverwrite

Notes:

If cfnRun is the first Dialog Wrapper function or subroutine you call, FileMustExist will be "ON," since this is the default. FileMustExist will also be "ON" immediately after a call to cfnMakeOpenBox, cfnFileMustExist(X) with X being greater than or equal to 1, or cfnResetDefaults. FileMustExist will be "OFF" (i.e., allow selection of non-existent files) immediately after a call to cfnFileMustExist(0) or cfnMakeSaveBox.

Example:

```
Declare Sub cfnFileMustExist Lib "dlgwrap.dll"  (D As Double)
Declare Function cfnRun Lib "dlgwrap.dll" As Integer
Declare Function cfnDoesItExist Lib "dlgwrap.dll"(St$ As String) As Integer
Declare Function cfnGetFileName$ Lib "dlgwrap.dll" As String

Sub MAIN
   cfnFileMustExist(0)
   If cfnRun <> 0 Then
         fn$ = cfnGetFileName$
         isitthere = cfnDoesItExist(fn$)
         'REM fn$ may or may not exist.
         'REM If  not, then "isitthere" will be zero.
         If isitthere <> 0 Then
               MsgBox fn$, "This file exists"
         Else
               MsgBox fn$, "This file does not exist"
         End If
```

```
    End If
    cfnFileMustExist(1)
    If cfnRun <> 0 Then
            fn$ = cfnGetFileName$
            'REM We know that fn$ must exist.
    End If
End Sub

§
```

Function cfnGetBareFile$

Syntax:

Declare Function cfnGetBareFile$ Lib "dlgwrap.dll" As String

Purpose:

Result return

Description:

Use this function to find out what file name the user entered the last time cfnRun was called.

Pass:

Nothing

Returns:

A string. This will include only the file name (no drive, directory, or extension).

See Also:

cfnGetDrive$, cfnGetExtension$, cfnGetFileName$, cfnGetPath$, cfnRun

Notes:

This function will return a meaningless result if it is called before cfnRun, or if it is called after the user cancels a dialog box.

Example:

```
Declare Function cfnGetBareFile$ Lib "dlgwrap.dll" As String
Declare Function cfnRun Lib "dlgwrap.dll" As Integer

Sub MAIN
   If cfnRun <> 0 Then
         filechoice$ = cfnGetBareFile$
         MsgBox filechoice$, "This is the file chosen"
   Else
         MsgBox "User cancelled", "cfnRun"
   End If
End Sub

§
```

Function cfnGetDirectory$

Syntax:

Declare Function cfnGetDirectory$ Lib "dlgwrap.dll" As String

Purpose:

Return result

Description:

Use this after cfnRun to find the directory portion of the file specification chosen by the user (no drive, file name, or extension).

Pass:

Nothing

Returns:

A string. This will be meaningless if it is called before cfnRun, or if the user canceled the last time cfnRun was called.

See Also:

cfnGetBareFile$, cfnGetDrive$, cfnGetExtension$, cfnGetFileName$, cfnGetPath$

Notes:

The string returned by this function includes a leading and trailing backslash. That is, if the user enters "C:\JUNK\STUFF.FIL" you get "\JUNK\." For root directories, "\" is returned.

Example:

```
Declare Function cfnGetDirectory$ Lib "dlgwrap.dll" As String
Declare Function cfnRun Lib "dlgwrap.dll" As Integer

Sub MAIN
   If cfnRun <> 0 Then
         filechoice$ = cfnGetDirectory$
         MsgBox filechoice$, "Directory of chosen file:"
   Else
         MsgBox "User canceled", "cfnRun"
   End If
End Sub
```

§

Function cfnGetDrive$

Syntax:

Declare Function cfnGetDrive$ Lib "dlgwrap.dll" As String

Purpose:

Return result

Description:

Use this after cfnRun to find the drive of the file the user chose (no file name, directory, or extension information).

Pass:

Nothing

Returns:

A string. This will be meaningless if it is called before cfnRun, or if the user canceled the last time cfnRun was called.

See Also:

cfnGetBareFile$, cfnGetDirectory$, cfnGetExtension$, cfnGetFileName$, cfnGetPath$

Notes:

The string returned by this function includes a colon. That is, if the user enters "C:\JUNK\BOGUS.FIL," you get "C:"

Example:

```
Declare Function cfnGetDrive$ Lib "dlgwrap.dll" As String
Declare Function cfnRun Lib "dlgwrap.dll" As Integer

Sub MAIN
   If cfnRun <> 0 Then
         filechoice$ = cfnGetDrive$
         MsgBox filechoice$, "Drive of chosen file:"
   Else
         MsgBox "User canceled", "cfnRun"
   End If
End Sub

§
```

Function cfnGetExtension$

Syntax:

Declare Function cfnGetExtension$ Lib "dlgwrap.dll" As String

Purpose:

Return result

Description:

Use this after cfnRun to find the extension of the file the user chose.

Pass:

Nothing

Returns:

A string. This will be meaningless if it is called before cfnRun, or if the user canceled the last time cfnRun was called.

See Also:

cfnGetBareFile$, cfnGetDirectory$, cfnGetDrive$, cfnGetFileName$, cfnGetPath$

Notes:

The string returned by this function includes the initial dot. That is, if the user chooses "C:\BORLANDC\BIN\BCC.EXE," then ".EXE" will be returned. If the user chooses a file with no extension, then cfnGetExtension$ returns simply "."

Example:

```
Declare Function cfnGetExtension$ Lib "dlgwrap.dll" As String
Declare Function cfnRun Lib "dlgwrap.dll" As Integer

Sub MAIN
   If cfnRun <> 0 Then
         filechoice$ = cfnGetExtension$
         MsgBox filechoice$, "Extension of chosen file:"
   Else
         MsgBox "User canceled", "cfnRun"
   End If
End Sub

§
```

Function cfnGetFileName$

Syntax:

Declare Function cfnGetFileName$ Lib "dlgwrap.dll" As String

Purpose:

Return result

Description:

Use this after cfnRun to find the complete file specification chosen by the user (drive, directory, name, and extension).

Pass:

Nothing

Returns:

A string. This will be meaningless if it is called before cfnRun, or if the user canceled the last time cfnRun was called.

See Also:

cfnGetBareFile$, cfnGetDirectory$, cfnGetDrive$, cfnGetExtension$, cfnGetPath$

Notes:

The string returned by this function includes all components of the file specification drive, directory, file name, and extension (if any). To find the individual components of the file specification use the other "cfnGet..." functions.

Example:

```
Declare Function cfnGetFileName$ Lib "dlgwrap.dll" As String
Declare Function cfnRun Lib "dlgwrap.dll" As Integer

Sub MAIN
   If cfnRun <> 0 Then
         filechoice$ = cfnGetFileName$
         MsgBox filechoice$, "You chose:"
   Else
         MsgBox "User canceled", "cfnRun"
   End If
End Sub

§
```

Function cfnGetPath$

Syntax:

Declare Function cfnGetPath$ Lib "dlgwrap.dll" As String

Purpose:

Return result

Description:

Use this after cfnRun to find the drive and directory of the file the user chose (no file name or extension information).

Pass:

Nothing

Returns:

A string. This will be meaningless if it is called before cfnRun, or if the user canceled the last time cfnRun was called.

See Also:

cfnGetBareFile$, cfnGetDirectory$, cfnGetDrive$, cfnGetExtension$, cfnGetFileName$

Notes:

The string returned by this function includes a trailing backslash. That is, if the user enters "C:\JUNK\BOGUS.FIL," you get "C:\JUNK\."

Example:

```
Declare Function cfnGetPath$ Lib "dlgwrap.dll" As String
Declare Function cfnRun Lib "dlgwrap.dll" As Integer

Sub MAIN
   If cfnRun <> 0 Then
         filechoice$ = cfnGetPath$
         MsgBox filechoice$, "Path of chosen file:"
   Else
         MsgBox "User canceled", "cfnRun"
   End If
End Sub

§
```

Subroutine cfnMakeOpenBox

Syntax:

Declare Sub cfnMakeOpenBox Lib "dlgwrap.dll"

Purpose:

Setup

Description:

There are two basic types of file dialogs in COMMDLG.DLL: the "open" box and the "save" box. The only difference is that in the "save" box, the names of the files shown in the directory listings are grayed-out, since the assumption is that the user will give a name to be used for a new file, rather than an already-existing file. cfnMakeOpenBox switches the Dialog Wrapper dialog type to "open," and then sets several defaults as they would most likely be useful for a file open box: the user must name a file that exists, and will (of course) not be warned that the file already exists. The subroutine also sets the title of the dialog box to "File.."

Pass:

Nothing

Returns:

Nothing

See Also:

cfnAllowReadOnly, cfnDirMustExist, cfnFileMustExist, cfnMakeSaveBox, cfnResetDefaults, cfnSetTitle, cfnWarnOverwrite

Notes:

Example:

```
Declare Sub cfnMakeOpenBox Lib "dlgwrap.dll"
Declare Function cfnGetFileName$ Lib "dlgwrap.dll" As String
Declare Function cfnRun Lib "dlgwrap.dll" As Integer

Sub MAIN
   cfnMakeOpenBox
   'REM The situation now is exactly as if the macro had read:
   'REM  cfnFileMustExist(1)
   'REM  cfnDirMustExist(1)
   'REM  cfnWarnOverwrite(0)
   'REM  cfnSetTitle("File..")
   'REM  In addition, the dialog type is set at "open;" i.e.,
   'REM  the files in the directory listings are not grayed-out.
   If cfnRun <> 0 Then
         MsgBox cfnGetFileName$, "Name of chosen file:"
   Else
         MsgBox "User canceled", "cfnRun"
   End If
End Sub
```

§

Subroutine cfnMakeSaveBox

Syntax:

Declare Sub cfnMakeSaveBox Lib "dlgwrap.dll"

Purpose:

Setup

Description:

There are two basic types of file dialogs in COMMDLG.DLL: the "open" box and the "save" box. The only difference is that in the "save" box, the names of the files shown in the directory listings are grayed-out, since the assumption is that the user will give a name to be used for a new file, rather than an already-existing file. cfnMakeSaveBox switches the Dialog Wrapper dialog type to "save," and then sets several defaults as they would most likely be useful for a file save box: the user will be allowed to enter a file and/or directory name whether or not the file or directory exist, and a warning will be issued if the file chosen already exists. The subroutine also sets the dialog box title to "Save as.."

Pass:

Nothing

Returns:

Nothing

See Also:

cfnDirMustExist, cfnFileMustExist, cfnMakeOpenBox, cfnResetDefaults, cfnSetTitle, cfnWarnOverwrite

Notes:

Example:

```
Declare Sub cfnMakeSaveBox Lib "dlgwrap.dll"
Declare Function cfnGetFileName$ Lib "dlgwrap.dll" As String
Declare Function cfnRun Lib "dlgwrap.dll" As Integer

Sub MAIN
   cfnMakeSaveBox
   'REM The situation now is exactly as if the macro had read:
   'REM  cfnFileMustExist(0)
   'REM  cfnDirMustExist(0)
   'REM  cfnWarnOverwrite(1)
   'REM  cfnSetTitle("Save as..")
   'REM  In addition, the dialog type is set at "save;" i.e.,
   'REM  the files in the directory listings are grayed-out.
   If cfnRun <> 0 Then
         MsgBox cfnGetFileName$, "Name of chosen file:"
   Else
         MsgBox "User canceled", "cfnRun"
   End If
End Sub
```

§

Subroutine cfnResetDefaults

Syntax:

Declare Sub cfnResetDefaults Lib "dlgwrap.dll"

Purpose:

Setup

Description:

This subroutine undoes all the custom setup you may have done.

Pass:

Nothing

Returns:

Nothing

See Also:

cfnAllowReadOnly, cfnDirMustExist, cfnDoesItExist, cfnFileMustExist, cfnMakeOpenBox, cfnSetStartDirectory, cfnSetDefExt, cfnSetStartFile, cfnSetFilterIndex, cfnSetTitle

Notes:

This subroutine is simply a shortcut for:

```
cfnMakeOpenBox
cfnSetStartDirectory("")
cfnSetDefExt("")
cfnSetStartFile("")
cfnSetFilterIndex(1)
```

Example:

```
Declare Function cfnRun Lib "dlgwrap.dll" As Integer
Declare Function cfnGetFileName$ Lib "dlgwrap.dll" As String
Declare Sub cfnSetStartDirectory Lib "dlgwrap.dll" (S$ As String)
Declare Sub cfnSetTitle Lib "dlgwrap.dll" (S$ As String)
Declare Sub cfnSetStartFile Lib "dlgwrap.dll" (S$ As String)

Sub MAIN
    'REM First, a "plain vanilla" box:
    If cfnRun = 0 Then goto Done    'REM User canceled
    MsgBox cfnGetFileName$, "You chose:"

    'REM Now, a dialog that's customized "to the hilt:"
    cfnSetTitle("What're we looking for, anyway?")
    cfnSetStartDirectory("D:\BOGUS\")
    cfnSetStartFile("*.WB1")
    cfnFileMustExist(0)
    cfnDirMustExist(0)
    If cfnRun = 0 Then goto Done    'REM User canceled
    MsgBox cfnGetFileName$, "You chose:"

    'REM Now, "plain vanilla" once again:
```

```
    cfnResetDefaults
    If cfnRun = 0 Then goto Done    'REM User canceled
    MsgBox cfnGetFileName$, "You chose:"

Done:
End Sub
```

§

Function cfnRun
Syntax:
Declare Function cfnRun Lib "dlgwrap.dll" As Integer
Purpose:
Execution
Description:
This function runs the dialog box. That is, it displays it on the screen, processes the user's keystrokes, and returns the results.
Pass:
Nothing
Returns:
0 if the user clicks "CANCEL," presses ESCAPE, or closes the dialog box with the system icon or the "CLOSE" item on the system menu.
-1 if the user clicks "OK" or presses ENTER after choosing a valid file name.
See Also:
Notes:
Example:

```
Declare Function cfnRun Lib "dlgwrap.dll" As Integer
Declare Function cfnGetFileName$ Lib "dlgwrap.dll" As String

Sub MAIN
   If cfnRun = 0 Then
         goto Done    'REM User canceled
   Else
         choice$ = cfnGetFileName$
         MsgBox choice$, "You chose:"
   End If
Done:
End Sub

§
```

Subroutine cfnSetDefExt

Syntax:

Declare Sub cfnSetDefExt Lib "dlgwrap.dll" (S$ As String)

Purpose:

Setup

Description:

This subroutine sets a default extension; that is, an extension that will be added automatically if the user doesn't supply one.

Pass:

A string the extension to add.

Returns:

Nothing

See Also:

cfnResetDefaults, cfnSetStartDirectory, cfnSetStartFile

Notes:

- The default extension **is "sticky;"** that is, if you set one, it will be used until you explicitly call cfnSetStartDirectory("") to "unset" it.
- If you designate a default extension, but the user types a terminating period (for example "c:\batch\junk.") then the default extension will be ignored.
- If you call this subroutine, passing it simply a dot (cfnSetDefExt(".")), that's the same as calling cfnSetDefExt(""); that is, it "unsets" the default extension.
- It might be possible to confuse cfnSetDefExt("BAS") with cfnSetStartFile("*.BAS"), but **they are not the same.** cfnSetStartFile controls what's displayed when a dialog box opens, and has nothing to do with what the final result is. cfnSetDefExt, by contrast, has no effect on the dialog box as it's running, but only designates an extension to be added if the user doesn't supply one.

For example, see the next page.

Example:

```
Declare Function cfnRun Lib "dlgwrap.dll" As Integer
Declare Function cfnGetFileName$ Lib "dlgwrap.dll" As String
Declare Sub cfnSetDefExt Lib "dlgwrap.dll"(S$ As String)
Declare Sub cfnFileMustExist Lib "dlgwrap.dll"(I As Integer)
Declare Function cfnGetExtension$ Lib "dlgwrap.dll" As String

Sub MAIN
   cfnFileMustExist(0)
   'REM Above line only to make experimentation easier, since most of us don't
   'REM have a lot of files with no extension.

   'REM In the next call to cfnRun, if the user enters no extension,
   'REM the returned file name will have none.
   If cfnRun = 0 Then
         Goto Done    'REM User canceled
   Else
         MsgBox cfnGetFileName$, "You chose:"
         MsgBox cfnGetExtension$, "Extension:"
   End If

   cfnSetDefExt(".COM")
   'REM If user enters no extension, dialog box will supply ".COM". Note, by
the way,
   'REM that you can supply an extension either with or without the dot; if you
include it
   'REM the subroutine ignores it, anyway.
   If cfnRun = 0 Then
         Goto Done    'REM User canceled
   Else
         MsgBox cfnGetFileName$, "You chose:"
         MsgBox cfnGetExtension$, "Extension:"
   End If

   'REM In this next call to cfnRun, the default extension is still in force:
   If cfnRun = 0 Then
         Goto Done    'REM User canceled
   Else
         MsgBox cfnGetFileName$, "You chose:"
         MsgBox cfnGetExtension$, "Extension:"
   End If
   cfnSetDefExt("")
   'REM In this next call to cfnRun, there is no default extension:
```

```
        If cfnRun = 0 Then
        Goto Done    'REM User canceled
    Else
        MsgBox cfnGetFileName$, "You chose:"
        MsgBox cfnGetExtension$, "Extension:"
    End If
Done:
End Sub
```

§

Subroutine cfnSetFilterIndex

Syntax:

Declare Sub cfnSetFilterIndex Lib "dlgwrap.dll" (I As Integer)

Purpose:

Setup

Description:

At the lower-left corner of the file dialogs is a "combo box" labeled "List Files of Type:" which affects which files are displayed in the file listing window. By clicking on the arrow at the right-hand end of this box you can see a list of available "filters;" choose one of these to "filter out" files that don't have the extension of interest to you. By default, when cfnRun is first called, the first filter ["Doc/Template (*.DO?)] will be the active one. Use cfnSetFilterIndex to start with another filter.

Pass:

An integer the number of the desired filter, starting with 1.

Returns:

Nothing.

See Also:

cfnSetDefExt, cfnSetStartFile

Notes:

- The index set by this subroutine is **not "sticky;"** that is, it will only be in effect the next time cfnRun is called. The time after, the starting filter index will be whichever one cfnRun leaves off with.
- The combo box allows you to choose one of the following filters: "*.DO?," "*.TXT," "*.BMP," "*.PCX," "*.*," and "readme" files (READ*.*). If you want to use another one, use cfnSetStartFile("*.XYZ"), where "XYZ" is the desired extension or wild card set.
- If you pass a number that's less than one or greater than six, one will be used.
- If you pass a non-integer, it will be rounded.
- The filter index is reset to 1 when your macro ends. That is, the first time a macro calls cfnRun, unless that macro has called cfnSetFilterIndex, the starting index will be 1, even if a previous macro left it set to something else.

Example:

```
Declare Sub cfnSetFilterIndex Lib "dlgwrap.dll" (I As Integer)
Declare Function cfnRun Lib "dlgwrap.dll" As Integer

Sub MAIN
   cfnSetFilterIndex(2)          'REM Start the box showing "*.TXT" files.
   If cfnRun <> 0 Then
         ...
   End If
End Sub
```

§

Subroutine cfnSetStartDirectory

Syntax:

Declare Sub cfnSetStartDirectory Lib "dlgwrap.dll" (S$ As String)

Purpose:

Setup

Description:

This subroutine sets the starting directory; i.e., the one that will be displayed first when cfnRun is called.

Pass:

A string the name of the directory.

Returns:

Nothing

See Also:

cfnResetDefaults, cfnSetDefExt, cfnSetStartFile

Notes:

- If you don't use this subroutine, the starting directory will be the last one that was displayed the last time cfnRun was called. If cfnRun hasn't been called, the default starting directory is the DOS current directory.
- The trailing backslash ("\") is optional, unless you're passing a root directory. That is, cfnSetStartDirectory("D:\VIEWER") and cfnSetStartDirectory("D:\VIEWER\") do exactly the same thing.
- If you want to designate a root directory, you must include the trailing backslash; if you don't then the display will simply switch to the current directory of the given drive, rather than the root.
- The starting directory set by this subroutine is **not "sticky;"** that is, the directory that cfnRun leaves off with it will be the starting directory the next time, unless, of course, you call cfnSetStartDirectory again.
- If there's a default file name set (see cfnSetStartFile) calling this will unset it, on the assumption that the same file name isn't likely to exist in the other directory.

For example, see next page.

Example:

```
Declare Sub cfnDirMustExist Lib "dlgwrap.dll"(D As Double)
Declare Function cfnRun Lib "dlgwrap.dll" As Integer
Declare Function cfnGetFileName$ Lib "dlgwrap.dll" As String
Declare Sub cfnSetStartDirectory Lib "dlgwrap.dll"(S$ As String)

Sub MAIN
   'REM If the DOS current directory is "C:\WINWORD" then that will be the
first
   'REM directory displayed when the cfnRun is called in the next line:
   If cfnRun = 0 Then
         Goto Done    'REM User canceled
   Else
         choice$ = cfnGetFileName$
         MsgBox choice$, "You chose:"
   End If
   'REM If the last directory the user viewed when cfnRun ran was "C:\
BORLANDC\BIN," then
   'REM that will be the first one displayed in the next line:
   If cfnRun = 0 Then
         Goto Done    'REM User canceled
   Else
         choice$ = cfnGetFileName$
         MsgBox choice$, "You chose:"
   End If
   cfnSetStartDirectory("D:\WINBATCH\")
   'REM "D:\WINBATCH" will be the first directory seen
            'REM as the next line runs:
   If cfnRun = 0 Then
         Goto Done    'REM User canceled
   Else
         choice$ = cfnGetFileName$
         MsgBox choice$, "You chose:"
   End If
Done:
End Sub

§
```

Subroutine cfnSetStartFile

Syntax:

Declare Sub cfnSetStartFile Lib "dlgwrap.dll" (S$ As String)

Purpose:

Setup

Description:

This designates a file to be the starting file the next time cfnRun is called. That is, the name given here will appear in the file name edit field when the dialog box appearss.

Pass:

A string the name of the file

Returns:

Nothing

See Also:

cfnResetDefaults, cfnSetStartDirectory, cfnSetDefExt, cfnSetFilterIndex

Notes:

- If the string passed to this subroutine includes a path, that path will be set as the starting directory.
- You may include wild card characters ("*" or "?") in the name. If, for example, you want to see a listing of all Basic source files, you'd call cfnSetStartFile("*.BAS"). However, this is **not the same as** cfnSetDefExt("BAS"). That has nothing to do with what's displayed, only with which extension will be added if the user doesn't supply one. This, by contrast, has nothing to do with the final results of the dialog box, only with what will be displayed when it starts.
- Don't call this subroutine and then call cfnSetStartDirectory before running the dialog box, because cfnSetStartDirectory clears any default file name.
- If there is no starting file name given, but there is a filter in effect (see cfnSetFilterIndex), that filter will be shown in the file name edit field when the dialog box opens.
- The starting file name given by this subroutine is **not "sticky;"** that is, it only affects the next time cfnRun is called. The time after, unless this subroutine is used again, the starting file name will be whichever one cfnRun left off with.
- If you attempt to set a starting file name that's too long, and then call cfnRun, cfnRun won't display the dialog box, but will simply return zero.
- If you attempt to set a directory that's too long, it will be truncated.

Example:

```
Declare Sub cfnSetStartFile Lib "dlgwrap.dll"(S$ As String)
Declare Function cfnRun Lib "dlgwrap.dll" As Integer
Declare Function cfnGetFileName$ Lib "dlgwrap.dll" As String
Declare Sub cfnDirMustExist Lib "dlgwrap.dll"(D As Double)

Sub MAIN
   cfnDirMustExist(0)
```

```
    cfnSetStartFile("c:\util\touch.com")
    If cfnRun <> 0 Then
            MsgBox cfnGetFileName$, "You chose:"
    End If
End Sub

§
```

Subroutine cfnSetTitle

Syntax:

Declare Sub cfnSetTitle Lib "dlgwrap.dll" (S$ As String)

Purpose:

Setup

Description:

Sets the title displayed in the upper border of the dialog box.

Pass:

A string □the title.

Returns:

Nothing

See Also:

Notes:

- The maximum length allowed is 64 characters. Anything longer will be truncated.
- It's not possible to set the title to no title at all. If you call cfnSetTitle("") the title will be set to the default □either "File.." if the dialog box type is "open" or "Save as.." if the type is "save" (see cfnMakeOpenBox and cfnMakeSaveBox). If you want an invisible title, call cfnSetTitle(" ") (one space).

Example:

```
Declare Sub cfnSetTitle Lib "dlgwrap.dll" (S$ As String)
Declare Function cfnRun Lib "dlgwrap.dll" As Integer

Sub MAIN
    cfnSetTitle("What file yak want, Boss?")
    If cfnRun <> 0 Then
            '...
    End If
End Sub
```

§

Subroutine cfnWarnOverwrite

Syntax:

Declare Sub cfnWarnOverwrite Lib "dlgwrap.dll" (D As Double)

Purpose:

Setup

Description:

Controls whether the next dialog box will issue a warning if the user selects a file that already exists. If this option is "turned on," choosing an already-existing file will cause the dialog box to display a warning message, and you'll have a chance to change your mind.

Pass:

A "double" (number). Zero turns WarnOverwrite off (allows user to choose an existing file with no warning), and anything else turns it on.

Returns:

Nothing

See Also:

cfnAllowReadOnly, cfnDirMustExist, cfnDoesItExist, cfnFileMustExist, cfnMakeOpenBox, cfnMakeSaveBox

Notes:

WarnOverwrite on is the default for a save box (set up by cfnMakeSaveBox) and off is the default for an open box (cfnMakeOpenBox).

Example:

```
Declare Sub cfnWarnOverwrite Lib "dlgwrap.dll"  (D As Double)
Declare Function cfnRun Lib "dlgwrap.dll" As Integer
Declare Function cfnGetFileName$ Lib "dlgwrap.dll" As String
Declare Function cfnDoesItExist Lib "dlgwrap.dll" (St$ As String) As Integer

Sub MAIN
   cfnMakeSaveBox
   'REM WarnOverwrite is now on, since this is one of the things that
cfnMakeSaveBox does.
   'REM The user will not be allowed to choose an already-existing file without
being warned.
   If cfnRun <> 0 Then
         MsgBox cfnGetFileName$, "You chose:"
   End If
   cfnWarnOverwrite(0)
   If cfnRun <> 0 Then
         MsgBox cfnGetFileName$, "You chose:"
         'REM The user may have named a file that already exists. Don't
overwrite it                    'REM without giving the user a chance to
cancel.
   End If
End Sub
```

§

Real-World Example

Here is the macro that prompted us to start this project in the first place. As the name implies, it allows the user to choose and delete a file.

There are various macros available from Compuserve that allow one to do this from Word for Windows without jumping to the File Manager, but none of the ones we were familiar with used the common dialog boxes very effectively, or ran into problems when the same file name existed in more than one directory.

We named this macro "FileDelete" and attached it to the File menu.

```
'Macro "FileDelete"
'To be used from within Word for Windows to choose and delete a disk file.
'Will not delete read-only files or a currently open file.'
'Uses the functions in DLGWRAP.DLL, the Dialog Wrapper library.
'Dialog Wrapper is copyright © 1993 by Tony Rein (Compuserve 76276,2662)
and Oriole Computing.

Declare Sub cfnAllowReadOnly Lib "dlgwrap.dll"(D As Double)
Declare Function cfnRun Lib "dlgwrap.dll" As Integer
Declare Function cfnGetFileName$ Lib "dlgwrap.dll" As String
Declare Sub cfnMakeOpenBox Lib "dlgwrap.dll"
Declare Sub cfnSetTitle Lib "dlgwrap.dll"(S$ As String)

Sub MAIN
    On Error Goto AnError
    cfnMakeOpenBox              'REM Won't allow selection of non-existent
file.
    cfnAllowReadOnly(0)                 'REM Don't allow selection of read-
only file.
    cfnSetTitle("File to Delete:")
Loop1:
    result = cfnRun
    If result = 0 Then                  'REM User canceled.
            Goto Done
    Else
            fn$ = cfnGetFileName$       'REM Which file did the user choose?
    End If
    'REM Confirm with the user:
    DotheDeed =  MsgBox(fn$ + "?", "Shall I delete this file?", 32 + 3)
    If DotheDeed = - 1 Then                     'REM User chose "Yes."
            Kill fn$
            Goto Done
    ElseIf DotheDeed = 0 Then           'REM User chose "No."
            Goto Loop1
```

```
Else                                    'REM User chose "Cancel."
        Goto Done
End If

'REM Continued on next page.
```

```
    'REM Example macro FileDelete, continued.

AnError:
    If Err = 55 Then
            MsgBox "Can't delete the file that WinWord is editing.", fn$, 48
            Err = 0
            Goto Loop1
    ElseIf Err = 53 Then                        'REM We shouldn't see this one.
            MsgBox "File not found", fn$, 48
            Err = 0
            Goto Loop1
            Err = 0
    Else
            'REM Not our error it came from somewhere
            'REM else. Let WinWord and the user take care of it:
            Error Err
            Goto Done
    End If
Done:
End Sub
```

Registration

Dialog Wrapper is copyrighted. The dynamic link library (DLGWRAP.DLL) and documentation (DLGWRAP.DOC) are copyright © 1993 by Tony Rein and Oriole Computing.

Dialog Wrapper is shareware. That is, it's "try before you buy" software. Please use it as much as you'd like to in order to see if it's useful for you. After this evaluation period, you must either register Dialog Wrapper or discontinue using it. How long is the evaluation period? We'll leave that to your judgment If you've decided Dialog Wrapper is useful to you, you'll know it.

Whether or not your register, you are welcome indeed, urged to distribute copies of Dialog Wrapper to anyone you think may be interested. The only restrictions on this are:

- You must distribute DLGWRAP.DOC and DLGWRAP.DLL together, unchanged. You may also distribute COMMDLG.DLL, as long as no representation is made that COMMDLG.DLL is a product of Oriole Computing or Tony Rein.
- You may not charge anything for Dialog Wrapper, although you may charge for your costs, as long as this charge does not exceed $5.00.
- You may distribute other files with Dialog Wrapper, as long as no representation is made that these other files are part of Dialog Wrapper. To register, send $7.50 to:

  Tony Rein
  Oriole Computing
  3110 Bishop Street
  Cincinnati, OH 45220

To register, send your name, address, and a check for $7.50 (US) to the above address. Please let us know the version of Dialog Wrapper you have if there's been an upgrade or a bug fix we'll send you the latest. (Please specify a disk preference; if you don't, we'll assume you want a 5.25 in, 360K disk.)

To find the version, use the "About Dialog Wrapper" item on the "Help" menu of this document, or simply write and run the following macro:

```
Declare Sub dwAbout Lib "dlgwrap.dll"
Sub MAIN
        dwAbout
End Sub
```

Also, please let us know what software you're using Dialog Wrapper with, if it's not Word for Windows.

The easy ways to register are to choose "Register Dialog Wrapper" from this document's "Help" menu or double-click on this button:

Register

§

Technical Support

If you encounter bugs in Dialog Wrapper, please, please let us know! If we don't know about a problem, we can't fix it.

If you have questions and/or suggestions for improvements to Dialog Wrapper, please contact us at one of these two addresses:

US Mail:

Tony Rein
Oriole Computing
3110 Bishop Street
Cincinnati, OH 45220

Compuserve Mail:

ID #76276,2662

If you contact us, please send as much of the following information as you can:

- The steps that led to the problem. Was the macro "running free" or were you tracing its execution with the macro debugger? What was the exact sequence of macro commands (not just Dialog Wrapper stuff, but other macro commands as well; especially system-oriented ones like SendKeys, On Time, etc.)?
- The version of Dialog Wrapper you have. If you don't know this, use the "About Dialog Wrapper" item on the "Help" menu of this document, or simply write and run the following macro:

  ```
  Declare Sub dwAbout Lib "dlgwrap.dll"
  Sub MAIN
      dwAbout
  End Sub
  ```
- What kind of processor (386SX, 486, etc.) does your computer have?
- How much memory is installed?
- How much free space is there on the drive Windows uses as its "temp" storage space? To find out which one this is, get to the DOS command line, type "SET" and press the ENTER key. You'll see a display of so-called "environment variables;" one line will look like

  ```
  TEMP=D:\WINTEMP
  ```

  The drive and directory listed there are the ones we're asking about.
- Which version of Windows are you running, and which mode is it running in? How much memory is free, and what percentage of the "system resources?" To find out, go to the Program Manager, type Alt-H to drop down its "Help" menu, and select "About Program Manager." You'll see a message box giving this information. If possible, do this **while the problem is actually occurring**, since the percentage of system resources and amount of memory free changes whenever almost anything happens in Windows.

Legalese

We (Tony Rein and Oriole Computing) have taken reasonable care in writing and testing DLGWRAP.DLL. However, we can make no guarantees that DLGWRAP.DLL will work properly on your system. In particular, we can't guarantee that it won't cause problems on your system, including (but not limited to) loss of data.

"WinWord" or "Word for Windows" mean Microsoft® Word for Windows™.

"Windows" means Microsoft® Windows™.

§

Bugs

cfnDoesItExist

- If you pass this function the name of a directory on a CD-ROM drive it will give a false negative. For example, on our system, drive E: is a CD-ROM drive, and this function won't work properly with "E:," "E:\BOOKS," etc. It will, however, properly detect files that are on a CD.

If you find any others, **please, please let us know.** See "Technical Support" for the address.

§

Acknowledgments

Thanks to Woody Leonhard and Pinecliffe International for the use of DocCruiser™. The idea originated with WOPR ("Woody's Office Power Pack"), which Mr. Leonhard describes as "the number-one add-on to Word for Windows." Mr. Leonhard and Vincent Chen are the authors of Hacker's Guide to Word for Windows, (Addison-Wesley Publishing Company, Copyright © 1993 by Pinecliffe International and Vincent Chen). You ought to get this book if you're interested in doing much work with WordBasic.

Thanks to Charles Petzold, Ray Duncan and Richard Hale Shaw for writing books and articles that have helped us to understand dynamic link libraries to the (limited) extent that we do.

And Tony Rein would like to extend special thanks to his wife, for allowing him to spend time messing with the computer instead of doing something useful!

§

Notes

[1] A dynamic link library is a file containing functions available to any Windows program running on the computer where the library is installed. "DLL" is the most common extension for a DLL file, but many DLL's on your system have "EXE" or some other extension. COMMDLG.DLL is currently distributed with Windows, and has been since Windows 3.1 came out. If you're using version 3.0 you didn't get it with Windows, but Microsoft allows free distribution of the file with programs that use it. You may already have it on your system if you've installed some such program. If not, you probably have it now: if you got Dialog Wrapper as an archived file named DLGWRP.ZIP, COMMDLG.DLL is part of it.

By the way, for the purposes of this document, "function" means a WordBasic function or subroutine, or any similar construct in any other programming language.

[2] If you're curious about these topics, I recommend pretty much any book on Pascal by Tom Swan.

§

Welcome to the Target of the Practice Jump! We hope you will enjoy your stay. When you're ready to go home, double-click on the button below. And thanks again for visiting!

§