What's the difference between PathIsSystemFolder and Protected Operating System Files?

devblogs.microsoft.com/oldnewthing/20150325-00

March 25, 2015

Raymond Chen

The way to detect weird directories that should be excluded from the user interface is to check for the `FILE_ATTRIBUTE_HIDDEN` and `FILE_ATTRIBUTE_SYSTEM` attributes being set simultaneously. This is the mechanism used when you uncheck *Hide protected operating system files* in the Folder Options dialog. (Programmatically, you detect whether the user wants to see protected operating system files by checking the `fShowSuperHidden` member of the `SHELLSTATE` structure.) Michael Dunn suggested using `PathIsSystemFolder` to detect these special directories, but that is not quite right. `PathIsSystemFolder` is for marking a directory as "This directory has a nondefault UI behavior attached to it. Please consult the `desktop.ini` file for more information." You do this when your directory is, say, the root of a namespace extension, or it has been subjected to folder customization. Windows uses it to indicate that the directory has a localized name, as well as other funky internal state. There are two ways to mark a folder as having nondefault UI. One is to set the `FILE_ATTRIBUTE_READONLY` attribute, and the other is to set the `FILE_ATTRIBUTE_SYSTEM` attribute. Either one works, and `PathIsSystemFolder` checks for both, returning a nonzero value if either attribute is set. In its default configuration, Windows uses the read-only flag to mark folders with nondefault UI. However, some applications mistakenly believe that if a directory is marked read-only, then files within the directory cannot be modified. As a result, these applications refuse to let you save your documents onto the desktop, for example. To work around this, you can use the `UseSystemForSystemFolders` to tell Windows to use the `FILE_ATTRIBUTE_SYSTEM` attribute instead. Of course, if you do that, you will run into problems with applications which mistakenly believe that if a directory is marked system, then the directory is inaccessible. So you get to pick your poison. Programmers who wish to mark a folder as having nondefault UI should use the `PathMakeSystemFolder` function to set the appropriate attribute. That function consults the system policy and sets the attribute that the policy indicates should be used to mark folders with nondefault UI.

Going back to the original question, then: The difference between `PathIsSystemFolder` and checking for folders that are marked hidden+system is that they check different things and have different purposes.

Function	Test
`PathIsSystemFolder`	ReadOnly or System
path is protected operating system folder	Hidden and System

Raymond Chen

Follow