How do I indicate that I want my window to follow right-to-left layout rules?

devblogs.microsoft.com/oldnewthing/20100611-00

June 11, 2010

Raymond Chen

There are many ways, depending on how wide a scope you want. If there is one specific window that you want to follow right-to-left layout rules, then you pass the `WS_EX_LAYOUT-RTL` extended window style when you create the window. This extended style is inherited by child windows, so once you set a top-level window to have right-to-left layout, all child windows will have it, too. To block the `WS_EX_LAYOUTRTL` extended style from being inherited by child windows, pass the `WS_EX_NOINHERITLAYOUT` style when you create the parent window. **Sidebar**: If you're calling the `MessageBox` function, then you don't directly control the styles of the top-level window. But there's a weird back-channel way to specify that you want the message box dialog to have the `WS_EX_LAYOUTRTL` extended style: Begin the `lpText` string with two U+200F characters. Then again, instead of `MessageBox` you should be using the `TaskDialogIndirect` function which not only lets you customize the text on the buttons, but also lets you pass the `TDF_RTL_LAYOUT` flag to indicate that you want the dialog to be laid out according to RTL rules. (And as an aid to porting, the `Task-Dialog` and `TaskDialogIndirect` functions implicitly turn on the `TDF_RTL_LAYOUT` flag if they find that `pszContent` is a pointer to a string—not a `MAKEINTRESOURCE`—which begins with two U+200F characters.) **End sidebar**.[1] If you want right-to-left layout rules to apply to all top-level windows in your process, you have two choices. You can either do it programmatically or declaratively. (Similar to how you can specify DPI-awareness either programmatically or declaratively.) The programmatic way is to call `SetProcessDefault-Layout(LAYOUT_RTL)` from your application. The declarative way is to insert two left-to-right marks (U+200E) at the beginning of the *FileDescription* version resource string of the executable. Note that the caveats which apply to changing the process DPI awareness programmatically also apply to changing the default process layout programmatically: Code which calls `GetProcessDefaultLayout` will see the default at the time of the call, even if some code later on calls `SetProcessDefaultLayout` to change it. Note also that it really is the application's call whether its default layout is left-to-right or right-to-left. A DLL shouldn't decide on its own to change the process default layout, at least not without coöperation from that process. If you are a DLL and you want to create a specific window with right-to-left layout, you should use the `WS_EX_LAYOUTRTL` method so that your decision applies only to your DLL's windows. (Otherwise you're using a global setting to manage a local problem.) **Bonus chatter**: Why isn't the default layout specified in the manifest like DPI-awareness? Because RTL support was added in Windows 2000, which predated application manifests by several years. If the feature were invented today, the manifest would be a much better place for declaring it. **Update** ¹ Commenter SCB pointed out that there is indeed a flag to specify that you want RTL layout on your message box: `MB_RTLREADING`. If that flag exists, then why also have the U+200F back-channel?

Answer: So that translators can mark a string as requiring RTL treatment without having to go back and make code changes.