What happens to the control names in the IDE when my program is running?

devblogs.microsoft.com/oldnewthing/20100329-00

March 29, 2010

Raymond Chen

nick_journals demonstrates some confusion about names in source code and their relationship to runtime behavior.

> A topic I am particularly interested in is the naming of controls, how it works…
>
> Every control gets a name from a developer…via the IDE (e.g btnOK)
>
> When using this function: GetWindowLong(handle,GWL_ID) it doesn't return the name itself but mostly a number or nothing.
>
> What is GWL_ID, the documentation isn't very clear on this.
>
> How does this whole system work, what are these numbers and where are the 'real' names?

I'm going to answer the questions most technical first. That way you can stop reading when you understand the topic. The `GWL_ID` window long (or, more precisely, the `GWLP_ID` window pointer-sized long) returns the value you passed to the `CreateWindowEx` function as the child window identifier (overloaded as the `hMenu` parameter). The call to `CreateWindowEx` might have happened explicitly in your code, or it may have been the result of a call to `CreateWindowEx` made on your behalf by another component, such as the dialog manager, which takes the control identifier from the dialog template. (Note that only child windows have child window identifiers. Top-level windows don't have child window identifiers.) And if you passed zero as the child window identifier, then when you ask for the `GWLP_ID`, you'll get zero back. The name that appears in your source code is just a name you decided to use to talk about the control. It's just a convenience for yourself, so instead of saying "Control number 103" all over the place, you can say "Control number `IDC_ADD`". And sometimes even that is too much typing, so you shorten it to "control `btnAdd`." The window manager doesn't know what cute shortcut names you've created for your child windows; the window manager still calls it control number 103. By analogy, the phone company doesn't know which numbers you've programmed into your speed dial. That's just something you set up in your phone to make dialing more convenient. In other words, the “real name” is really just a fake name you created to make things easier to talk about. The actual “real name” is the child window identifier. When you look in the telephone book for Bob, you’ll see Bob’s phone number, not his speed-dial number on your phone. This is the same thing that happens to your variables after the code has been compiled. At runtime, your variable names don’t exist any more. They were just convenient mnemonic names you gave to computational values. The compiler uses those convenient names to determine what you’re talking about when you assign a variable or a fetch a variable’s value, but once that’s done, it has no need for the name any more. In other words, the name in your source code is just something you did to make things easier to write. The compiler’s job is to change your line `x = y;` into `mov eax, [ebp-05ch]; mov [00437180h], eax`, at which point the names `x` and `y` are no longer needed and are discarded. (Actually, it’s saved off in a separate file for the debugger to use, so that when you ask the debugger to show the value of the variable `x`, it knows to look in `00437180h`, but that happens outside of the execution environment.)

It’s like asking, “What happens to the phrase *1 cup sugar* when I eat my cookies?” The phrase *1 cup sugar* was part of the instructions for making the cookies. Once you finish following the recipe, the instructions aren’t needed any more. Or at least not for the process of enjoying cookies. (You might still want to hang onto them to debug your recipe!)

Raymond Chen

Follow