Annotating function parameters and using PREfast to check them

devblogs.microsoft.com/oldnewthing/20100614-00

June 14, 2010

Raymond Chen

Via the suggestion box, Sys64738 asks, whether I think is a good C/C++ programming style to add IN and OUT to function prototypes. Remember, this is my opinion. Your opinion may validly differ. I would go beyond just IN and OUT and step up to SAL annotations, which describe the semantics of function parameters in more detail than simply IN and OUT. For example, the `__out_ecount` annotation lets you specify not only that the buffer is an output parameter, but also lets you specify how big the buffer is. This added expressiveness is used by tools like PREfast and its user-mode equivalent the C/C++ Code Analysis tool. These tools study your source code and attempt to find errors like writing past the end of an output buffer, passing the wrong buffer size for an output parameter, or writing to an input buffer. One question that often comes up when people start adding SAL annotations to their code is "I have a function whose parameters are used in a manner that isn't neatly covered by an existing annotation. How do I annotate it?" If you ask them to describe the parameters they are having trouble with, you often find that they don't have an easy annotation because they are too complicated. "Well, this is a null-terminated input string buffer, unless the `SET` flag set, in which case it's an output buffer with length specified by the `cchBuf` parameter, and the buffer size is specified in `CHAR` s unless the `FN_UNICODE` flag is set in the `dwFlags` parameter, in which case the size is in `WCHAR` s."

One lesson we learned in Windows is that if your function is so complicated that the annotation language has trouble expressing it, then that might be a clue that your function is so complicated that human beings can't understand it either. The answer to "How do I annotate it?" is therefore "Don't try. Simplify your interface instead." (Of course, if the function must be kept around for compatibility reasons, then you are stuck with a complicated annotation.)