Making Code Perfectly Clear
February 6, 2006

By Peter Coffee

As January drew to a close, Microsoft proposed to meet European courts' demands for documentation of key interfaces by disclosing source code to competitors. Company counsel Brad Smith called this "the ultimate documentation" of the technologies.

It's useful to understand why that is not so, because the same question could come up in your own environment—for example, in outsourcing arrangements or in any of several other plausible situations.

QUIZ QUESTION: When someone says, "The source code is the documentation," it's an example of (check all that apply):

(a) Accuracy  
(b) Irony  
(c) Laziness  
(d) Genius  
(e) Guilt

GRADING NOTES: When a program springs from someone's mind to take tangible form, the program itself is clearly meant to define what it does. Answer (a) is therefore right—but, if unqualified, is also dangerously wrong.

Your worst students will argue that (a) is a weak form of the only accurate answer—that the program is *ipso facto* a perfect definition of what it does. That is not correct. Exams submitted with (a) as the only checked answer should receive a grade of F, regardless of the answers to any other questions. If challenged on this by students or their tuition-paying parents, remind them that Ken Thompson's Turing Award lecture in 1984 showed how a programmer could manipulate system software to make programs carry out functions that were *completely absent* from the source code.

Thompson's scenario assumed either personal control or a successful conspiracy that would be hard to maintain. But, even so, no one should get away with saying, "Of course that's how it works! Read the code!" Answer (b) is therefore also correct, and no partial credit should be given to any student who does not include (b) among the checked responses.
Larry Seltzer wonders why Microsoft didn't make its source code available to competitors years ago. Click here to read his column.

Further, when code is written in a language that can be compiled for any of several hardware architectures or even for different operating systems on the same hardware, the same source code can yield different behaviors. The difference might be due to one compiler treating the "natural" size of a simple numeric value as a 16-bit quantity, while another considers it to be 32 bits; it might be due to one operating system allocating time across multiple threads of equal priority "fairly," while another tries to maximize efficiency by letting a task complete unless interrupted by another thread that's declared to be more important.

Documentation for a program should therefore indicate the programmer's intentions and, if possible, show that the desired behavior is actually specified to occur. A combination of the language specification and version number, the compiler version number, and the operating system specification and version number may all be needed in the documentation in addition to what's in the source code if the program behavior is to be accurately predicted—or its misbehavior quickly understood.

It's far more likely that the program will be tested on a single environment, debugged until it works correctly there, and released with "documentation" consisting of the dates and version numbers of the test suites thereby passed. Answer (c) is therefore correct; give no partial credit if this choice is not checked.

Students may argue that source code comments are the best documentation, since nothing else is certain to follow the code in its travels. Remind them of the infamous comment, RIPJSB, once found next to the value 1750. Only after much lost time was this deduced to mean "RIP Johann Sebastian Bach," 1750 being the year of the composer's death.

Comments, in short, say as much or as little about the behavior of the code as the coder's whims may dictate. Answer (d) may be correct, but code maintainers may nonetheless suffer: Give credit for either response.

Truth or falsity of (e) is beyond the scope of this course—but Microsoft is invited to explain its answer.

To read reader response to this article, click here.

Technology Editor Peter Coffee can be reached at peter_coffee@ziffdavis.com.

Check out eWEEK.com's Application Development Center for the latest news, reviews and analysis in programming environments and developer tools.

Copyright (c) 2006 Ziff Davis Media Inc. All Rights Reserved.