What does the executable timestamp really mean?

devblogs.microsoft.com/oldnewthing/20110713-00

July 13, 2011

Raymond Chen

A customer was looking for some information on the executable timestamp:

> I would like my program to display the date it was linked. The `IMAGE_FILE_HEADER.TimeDateStamp` looks like what I need. Is there an easy way to retrieve this information so I don't have to parse the EXE header myself? Also, what functions exist for formatting this timestamp into something human-readable?

The customer didn't explain why they needed this information, but presumably it was for diagnostic purposes. For example, the program might display the information in the *About* box to help the product support team identify which version of the program the end-user is running. We'll answer the questions in reverse order, and then answer a question that wasn't even asked. The timestamp is a Unix-style `time_t` timestamp; therefore, you can use the `ctime` function to convert it to text. If there is a particular format you like, you can use the appropriate time formatting function (though you may have to convert it first). If you want to retrieve this value, you can use helper functions in the `imagehlp` library; the one most applicable here appears to be `ImageNtHeader` or even `GetTimestampForLoadedLibrary`.

The unasked question is "Does this in fact give me the date and time that the image was linked?" Fortunately, I don't have to write out the answer to this question, because I answered it last year. The name *timestamp* is misleading. Its real purpose is to act as a signature so that the operating system can determine whether a DLL against which one set of values was precalculated matches the DLL physically on the system. A better name for it would have been `UniqueId`.

Raymond Chen

Follow