What are these dire multithreading consequences that the GetFullPathName documentation is trying to warn me about?

devblogs.microsoft.com/oldnewthing/20210816-00

August 16, 2021

Raymond Chen

The documentation for the `GetFullPathName` function contains this dire warning:

> Multithreaded applications and shared library code should not use the **GetFullPathName** function and should avoid using relative path names. The current directory state written by the **SetCurrentDirectory** function is stored as a global variable in each process, therefore multithreaded applications cannot reliably use this value without possible data corruption from other threads that may also be reading or setting this value. This limitation also applies to the **SetCurrentDirectory** and **GetCurrentDirectory** functions. The exception being when the application is guaranteed to be running in a single thread, for example parsing file names from the command line argument string in the main thread prior to creating any additional threads. Using relative path names in multithreaded applications or shared library code can yield unpredictable results and is not supported.

What is this warning trying to say? It seems to suggest that the current directory global variable is not thread-safe. Does this mean that all calls to `SetCurrentDirectory` and `GetCurrentDirectory` need to be serialized by the application?

No, that's not what it's saying.

What it's trying to say is that the meaning of a relative path depends on the current value of the current directory. The value of the current directory can be changed by any thread at any time, so make sure that you understand that the result of the `GetFullPathName` function is a "moment in time" calculation. Resolving the same relative path in consecutive calls to the `GetFullPathName` function could result in different results if the current directory changed in between.

If you find yourself with a relative path, you have a few choices.

One option is to pass it as a relative path to a function like `CreateFile`, but only once, and as soon as possible. Don't assume that a second `CreateFile` will open the same file. You want to use it as soon as possible because the user entered the relative path based on some underlying assumptions about the current directory, and the longer you wait, the more likely those assumptions are going to be wrong.

Another option is to convert it to an absolute path as soon as possible, and use the absolute path (at your leisure) from then on. Again, you want to convert it as soon as possible to reduce the likelihood of changes to the conditions under which the user entered the relative path.

If the relative path didn't come from the user but rather from, say, a configuration file or another process, then things are kind of sketchy. The relative path in the configuration file is probably intended to be interpreted relative to some anchor point (such as the configuration file itself), not relative to something as fickle as the process's current directory. And a relative path received from another process is even more sketchy, because that other process has no idea what your current directory is.

The concept of the current directory was inherited from MS-DOS, which in turn got it from the concept of the current drive in CP/M (and probably influenced by a similar concept in Unix). CP/M was a single-threaded operating system, so there were no race conditions related to the current directory. And at the time, Unix supported only one thread per process, so the issue never arose there either.

The idea of a current directory in today's multithreaded world is a bit of an anachronism, as if there's only one "place" a process can be at a time. Standard handles were also designed in a single-threaded world. I think the convention for the current directory should be that only the main thread of the main process can change the current directory: Background threads and helper libraries should keep their hands off.

Bonus chatter: Don't forget to pass the `OFN_NOCHANGEDIR` flag when you use the common file dialogs, to tell them not to change the current directory.

Bonus reading: The curse of the current directory.

Raymond Chen

Follow