Even if a function doesn't do anything, you still have to call it if the documentation says so, because it might do something tomorrow

devblogs.microsoft.com/oldnewthing/20080925-00

September 25, 2008

Raymond Chen

If the documentation says that you have to call a function, then you have to call it. It may be that the function doesn't do anything, but that doesn't prevent it from doing something in the future.

Today's example is the function `GetEnvironmentStrings`, which returns you all the environment variables of the current process in a single block, which you can then study at your leisure. When you're finished, you're supposed to call `FreeEnvironmentStrings`. That's what the documentation says, and if you did that, then you're in good shape.

However, some people noticed that on Windows NT 4, the Unicode version of the `FreeEnvironmentStrings` function didn't do anything. In other words, the Unicode environment block didn't need to be freed. When you called `GetEnvironmentStrings`, the kernel just returned you a raw pointer to the *real live* environment strings (which, since this is Windows NT, are kept in Unicode internally). Since nothing was allocated, there was nothing to free.

The problem with this technique was that if somebody called `SetEnvironmentVariable` in the meantime, the environment block changed out from under the caller of `GetEnvironmentStrings`.

Oops.

To fix this, the `GetEnvironmentStrings` function was changed to return a copy of the environment block *even if you call the Unicode version*. The corresponding Unicode `FreeEnvironmentStrings` function frees that environment copy.

Programs that followed the specification and called `FreeEnvironmentStrings` (even though it didn't do anything up until now) were in good shape. Their call to `FreeEnvironmentStrings` now frees the memory, and all is right with the world.

Programs that coded to the implementation rather than the specification are now in a world of hurt. If they simply skipped the "useless" call to `FreeEnvironmentStrings`, they will now find themselves leaking memory. On the other hand, if they gave lip service to `FreeEnvironmentStrings` by calling it, but using the memory anyway, they will find themselves accessing invalid heap memory, and all sorts of havoc can ensue.

There's sometimes a reason for the rules that seem stupid at first glance. ("Call this function that doesn't do anything.") Changes to the implementation may make them less stupid in the future.

(Credit goes to my colleague Neill Clift for providing the information that led to today's article.)

Raymond Chen

Follow