Thread affinity of user interface objects, part 2: Device contexts

devblogs.microsoft.com/oldnewthing/20051011-10

October 11, 2005

Raymond Chen

Last time, we discussed briefly the thread affinity rules that govern window handles.

Device contexts (DCs) also have a certain degree of thread affinity. The thread that calls functions such as `GetDC` must also be the one that calls `ReleaseDC`, but as with window handles, during the lifetime of the DC, any thread can use it. If you choose to use a DC in a multi-threaded manner, it's your responsibility to coordinate the consumers of that device context so that only one thread uses it at a time. For example, to host windowless controls across multiple threads, the host obtains a DC on the host thread, then asks each control in sequence to draw itself into that DC. Only one control draws into the DC at a time, even if the control happens to be on a different thread.

The thread affinity of DCs is much more subtle than that of window handles, because if you mess up and release a DC from the wrong thread, things will still seem to be running okay, but the window manager's internal bookkeeping will be messed up and you may get a bad DC from GetDC a little later down the line.

Next time, the remaining user interface elements.

Raymond Chen

Follow