Thread affinity of user interface objects, part 5: Object clean-up

devblogs.microsoft.com/oldnewthing/20051014-19

October 14, 2005

Raymond Chen

The window manager and GDI objects as a general rule will automatically destroy objects created by a process when that process terminates. (The window manager also destroys windows when their owner threads exit.) Note, however, that this is a safety net and not an excuse for you to leak resources in your own program with the attitude of "Oh, it doesn't matter, the window manager will clean it up for me eventually." Since it's a safety net, you shouldn't use it as your primary means of protection.

For one thing, leaving junk behind to be cleaned up is just plain sloppy. It suggests that your program is too lazy (or stupid) to keep track of its own resources and has abdicated this to the safety net. It's like throwing your clothes on the floor because you know your mother will eventually come by to pick it up and put it away.

For another thing, this clean-up happens inside the window manager and no other window manager activity will occur until the clean-up is complete. If you leaked hundreds or thousands of objects, the system will seem visually unresponsive because the window manager is busy. (The system is still running, though. Operations that do not rely on the user interface, such as computation-intensive operations or network activity will still proceed normally while the window manager is cleaning up.)

Why didn't the window manager optimize the "massive clean-up" scenario? Because when you design a system, you focus on optimizing the case where people are using your system responsibly and in the manner intended. You don't want to reward the people who are abusing you. Imagine what kind of message you'd be sending if you designed the system so that people who abuse the system get better performance than people who follow the rules!