If one program blocks shutdown, then all programs block shutdown, revisited

devblogs.microsoft.com/oldnewthing/20250331-00

Ian Boyd March 31, 2025

Raymond Chen

Read next

April 8, 2025

The Goldilocks zone of software stability

Raymond Chen

Some time ago, I noted that if one program blocks shutdown, then *all* programs block shutdown. Each program is asked in turn if they have any final words, and only after each program says that it's finished do we ask the next program.

But what if we asked *all* the programs for their final words? After all, it's frustrating to go through two shutdown confirmations from two programs, only to find that the question from the third program is one that you realize "Oh, right, I should send that unfinished email. I need to cancel shutdown." If you could see all the questions at once, then you would see the third question right away and cancel shutdown without having to deal with the first two programs.

The problem with this design is that the current shutdown mechanism doesn't have a way for the system to tell programs, "I know I asked you to say your final good-byes, and you're in the middle of a dialog with the user, but I need you to cancel out of it right now because we're not shutting down after all."

That's not to say that such a mechanism couldn't be invented, but you'd have to deal with the adoption curve, because the feature doesn't work until *everybody* supports it, and in practice, there will always be programs that don't support it, either because it's too hard to implement,

or because the programs are no longer being maintained by their vendors.

Windows has instead been encouraging programs to dispense with the farewells entirely.

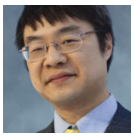
Programs written to the Universal Windows Platform, for example, never even get told that the system is shutting down. They are expected to autosave any relevant state whenever they receive a notification from the system that they are about to be suspended, and once suspended, the system is permitted to terminate the program without any further warning. And programs that adhere to the classic application model are encouraged to employ their own autosave/autorecover logic so that the user can shut down with a minimum of fuss.

I mean, when you power off your phone, you don't get a barrage of apps asking you if you want to save. They just go away quietly, and when you launch them next time, they try to pick up where they left off.

In other words, instead of trying to make the shutdown problem easier to deal with, Windows is trying to make the shutdown problem go away.

Author

Raymond Chen



Raymond has been involved in the evolution of Windows for more than 30 years. In 2003, he began a Web site known as The Old New Thing which has grown in popularity far beyond his wildest imagination, a development which still gives him the heebie-jeebies. The Web site spawned a book, coincidentally also titled The Old New Thing (Addison Wesley 2007). He occasionally appears on the Windows Dev Docs Twitter account to tell stories which convey no useful information.

12 comments

Join the discussion.

Sign in

Newest

- IB

  From all the various comments, we see how hard it is to choose a color to paint the barn.

- JD

John Dyer 1 week ago

Great post! Next question, how is Sleep handled (aka closing the lid)? One of the commenters mentions having SQL management studio open with an open transaction. Seems a very rare case. From what I understand on a Mac when the lid is closed apps aren’t ask because the user closed the lid. Not sure if they’re closed or just suspended. Windows is weird in this area. Something is happening that usually sleep works but a non-trivial number of times you open the backpack and it’s hot in there.

- JH

Joshua Hudson

I liked it better when QM_QUERYENDSESSION wasn't sent to more than one program at a time.

Microsoft claims the main design change was due to misbehaving programs; in my experience this was *not* true but rather the following:

1) Disk I/O saturation
2) Antivirus dragging down the I/O Subsystem
3) Single active program not pumping messages; this was always a case of "oops shouldn't have pressed shutdown". While sometimes the program really was misbehaving usually it was underlying libraries that were misbehaving. I'm looking at you gethostbyname(). My experience actually is "programs crash randomly on Windows 98" due to programs that I...

Read more

- RT

  Richard Thompson
  The phone comparison is disingenuous, though.

  Phone and UWP apps are primarily designed for content consumption, or to access a remote server where the user-generated state is stored.
  On phones, there's only one, possibly two active apps.
  They thus have very little local state - eg WSL2 host terminals have basically nothing, it's all in the VM.

  The Windows platform is primarily designed for content creation, thus there is an extremely large amount of user-generated local state that must be saved to disk.
  Many, perhaps most PC users regularly have multiple gigabytes of local user data open across multiple applications - far...

  Read more

  - Ω

    ω 6 days ago
    “The Windows **platform** is primarily designed for content creation” (emphasis added)

    Bold assertion I have to disagree with.

- 

  Martin Ba 1 week ago
  Hogwash!?! :-)

  What do we have here:
  * Outlook - autosave anyway
  * 2 MS Visual Studios with a 1050 Project Solution - minor bytes to save on close (sourcecode only), all other state is autosave
  * Word/Excel - yeah, might get big, but is autosave 50% of the time nowaways anyways, and if it isn't the current edit state can easily be saved to disk without hassle.
  * Thunderbird/Firefox ... all autosave / savestate

  If I have SQLDeveloper open with an uncommitted DB connection, then I want to be *asked*, but the act of commit/rollback is < seconds.

  What I do agree...

  Read more

  - MF

    Marc Fauser 6 days ago
    Firefox, okay.
    But Thunderbird has a stupid bug.
    I open all the mails I want to finish in the next hours in new windows.
    When you close TB, it won’t open the windows like Firefox.
    It’s a known bug and they don’t want to fix it.

  - CL

    Christopher Lee April 1, 2025
    Long-running operations (like writing files to a compact disc, or a terminal window computation process) are a problem. I'd like to see more options or alternatives for these, like shifting the in-progress terminal window computation to a Task Scheduler task that can continue running in an independent session after logoff, or having the database session continue running on the database server and letting you reconnect to it (like reconnecting to a disconnected Remote Desktop session). Maybe Xbox consoles can innovate in this area with cross-device Quick Resume and have that trickle down to Windows; they already have multi-user...

    Read more

- JL

GL 1 week ago · Edited
I think the one-by-one design is only for compatibility with programs that rely on a specific order.

>I know I asked you to say your final good-byes, and you’re in the middle of a dialog with the user, but I need you to cancel out of it right now because we’re not shutting down after all.

I'm not sure how to parse this. The documentation of WM_QueryEndSession and WM_EndSession says the following.

1. WM_QES is sent to every window when a session ending is initiated.
2. Applications should respond to WM_QES immediately and defer clean-up to processing WM_ES.
3. If a window responds...

Read more

Raymond Chen Author 1 week ago

The current recommendation is to return immediately from WM_QUERYENDSESSION, but the past, the common pattern was for programs to display a message like "Save changes before exiting?" If the user says "Save", then they save. If the user says "Discard", then they don't save. If the user says "Cancel", then they cancel shutdown. The problem is how to tell the program that has not yet responded to QM_QUERYENDSESSION, "Hey, sorry, I'm withdrawing my query. Please cancel your active "Save changes before exiting?" dialog box because we're not exiting."

The alternative would be to have the user say "Yes, save before exiting",...

Read more

- NR

  Neil Rashbrook 1 week ago

  As well as the problem of the programs that save and exit in response to WM_QUERYENDSESSION, there are also programs with the reverse problem – they set a save state and register with the restart manager, but nothing happens because the shutdown was cancelled; they then ignore the real shutdown, and so don’t get restarted. (But fortunately the save state is still there, so I was able to extract the data.)

- JL

  GL

  Thanks for the elaborate explanation. I guess one thing is that most programs never moved on from the Windows XP shutdown design ("save and exit, discard and exit, cancel shutdown" dialog).

  Preemptively, I think writing the correct shutdown logic is a software tax, so I don't expect much to be done. For a theoretical discussion... It's actually good to not dismiss the dialog if the user cancels shutdown, on the presumption that the user cancels BSDR to deal with the offending programs then shutdown again. Under that assumption, it's also good if the program saves and exits. Re the other comment...

  Read more

Stay informed

Get notified when new posts are published.