Why doesn't the Windows Vista copy progress dialog show the names of the files being copied?, redux

devblogs.microsoft.com/oldnewthing/20100601-01

June 1, 2010

Raymond Chen

As expected, everybody concluded that the Windows Vista copy progress dialog made every wrong decision possible. Let's look at some of the suggestions on making it suck less: Why not update the file name every five seconds to the file that is being copied at that time? Sure, you could do that, but the cost of getting the name is part of the problem. Retrieving the name is more than just "Remove everything after the last dot." You may have to look up localization information in order to display the name of the item in a manner appropriate for the user's preferred language. Since the operation being performed might be destructive (for example, delete or move), you need to retrieve the display name *before* you begin the operation. Since Microsoft Research hasn't yet finished their project to predict the future, we don't know at the time an operation starts whether it will take more than five seconds. Therefore, we have to play it safe and retrieve the name just in case. All you saved was the cost of the screen update; you didn't save the cost of retrieving the display names. That said, in Windows 7, when you expand the file copy progress dialog, it updates the fields at a maximum speed of four times per second. This is probably a decent balance between not updating faster than the screen refresh rate while still updating fast enough that you don't get the impression that it's skipping some files. Obviously the solution is to make getting localised names faster. Yup, work was done to make getting localized names faster. When the copy engine begins operating on a directory, it caches all the localized names at one go and reads the names out of the cache as it processes the directory. The reason 'updating the display is slow' is nonsense, as the progress bar is constantly updating anyway. I think Drak didn't click through the discussion of how updating the screen can be a significant cost when you are updating continuously. The operative word here is *continuously*. If you're copying a lot of small files, and you redraw each time you get to a new file, you may end up redrawing hundreds of times a second. On the other hand, the progress bar updates only around ten times a second, and the copy rate only once a second, and neither have to access the disk to decide what to display next. A changing progress bar or updating copy estimate does not come across as unstable; they appear as progress. But a hundred file names flying past per second? That looks unstable. Is it really necessary to display those names localized? Most users will never see them, and it'll confuse many of those that do see them. Imagine if you clicked to delete the file *Calculator.lnk* and the dialog box said "Deleting 計算器.lnk". Would

you be concerned? It's only confusing if you sometimes see the nonlocalized names, like from programs which just assume that the display name can be obtained by deleting everything after the last dot instead of using the `SHGetFileInfo` function. But Apple has the same problem, and everybody knows that Apple does everything right. It took me a while to realize that the complaint about disk-to-disk copy of several large files at once referred not to selecting multiple files and copying them in a single operation but rather starting one copy, then starting another copy, then starting another copy. If you copy them as a single operation, then they will queue up behind each other. But if you copy them as separate operations, then they will all compete with each other. The proposal to have a single queue for files copied to/from the same disk sounds interesting, but it quickly gets complicated:

- Suppose there is a copy in progress from volume A to volume B, and another copy from volume B to volume C, and another copy from volume D to itself. Presumably, the second copy should wait behind the first (since they both involve volume B); should the third copy "jump the queue" and start immediately?
- Is the queue implicit or explicit? In other words, is there a single dialog with a list box, one entry per queued-up operation? Or do you still keep separate dialog boxes, but just have the second dialog box wait until the first one is finished before it starts its operation? If you think about it, you can't have a single dialog with a list box: Suppose a program calls `SHFileOperation`, and its operation gets queued up. You disable the application window and do… what? Do you set focus to the queue dialog? That queue dialog is part of some other window hierarchy, so it can get stuck behind the caller's window. And even if you make sure it comes to the foreground (hooray for focus-stealing), you then have the problem that when the operation completes, focus goes to the queue dialog's owner, not to the program that triggered the focus change.
- Suppose I start copying a large file from volume A to volume B, and then a colleague asks, "Hey, can you copy that shortcut to volume B so I can take a look at it?" Under the old system, you would start a second copy operation, and it would run concurrently with the first copy operation but complete first (since shortcuts are relatively small). With the queueing design, you would need a way to let people adjust the priority of jobs in the queue so the second one could be bumped ahead of the first. Which also means that when a partially-completed operation gets pushed down the queue, you need to be able to pause the operation and resume it later. This is a lot of UI to design, implement, and test, especially since we already decided that you can't have a single dialog box with a list box.
- When the second operation queues up behind the first, its total estimated time needs to include the estimated times for all the jobs ahead of it in the queue. How do you get an estimate for a job that hasn't started yet? Do you do some preliminary calculations to come up with the estimate? But wait, that's going to access the hard drive and interfere with the first operation. Even if you have only one job ahead of you in the queue (and therefore have an estimate for that job), it means that copying a 1KB file might tell you "Estimated time to completion: 30 minutes" because it's waiting behind a 4GB file.

I'm not saying that these problems are unsolveable. But the simple suggestion of using a queue created a very large new problem. As we saw last time, you can't solve every problem; you have to decide which twenty problems you're going to solve and leave the others for next time. Windows 7 decided to address the problem by letting you trade off performance for information. The operation defaults to performance (no file name updates), but you can expand the dialog to say, "I'm willing for my file operation to be a smidge slower in exchange for more information about how it's going." I wonder if there are people would be willing to make the tradeoff "I'm willing to lose the progress bar in exchange for not having to wait for the operation to start while it calculates how big the progress bar should be." But I guess those people just use xcopy.

At any rate, the issue of the missing file names was fixed in Windows 7, so any further complaining just falls into the category of asking for a time machine. Though I did find it interesting that people suggested solutions that involved doing more work. Doing more work means writing more code, which means designing and testing more code, which means the product ships even later. I bet these are also the same people who complain that Windows always ships late. (What ever happened to the commenters who say that Microsoft should be deleting code more aggressively? Oh, wait, I know what happened to them. "Microsoft should be deleting code, but only the code for for features that I personally don't like." Good luck getting people to agree on what code to delete. It's like the people who argue that Microsoft should intentionally abandon backward compatibility, then change their tune when they realize that their favorite program stopped working.)