How does Explorer detect whether your program supports long file names?

devblogs.microsoft.com/oldnewthing/20041020-00

October 20, 2004

Raymond Chen

When you register your program with a file association, the shell needs to decide whether your program supports long file names so it can decide whether to pass you the long name (which may contains spaces! so make sure you put quotation marks around the “%1” in your registration) or the short name.

The rule is simple: The shell looks at your program’s EXE header to see what kind of program it is.

- If it is a 16-bit program, then the shell assumes that it supports long file names if it is marked as Windows 95-compatible. Otherwise, the shell assumes that it does not support long file anmes.
- If it is a 32-bit program (or 64-bit program for 64-bit systems), then the shell assumes that it supports long file names.
- If it **can’t find your program**, then the shell plays it safe and assumes that the program doesn’t support long file names.

Note that third case. If you mess up your program registration, then the shell will be unable to determine whether your program supports long file names and assumes not. Then when your program displays the file name in, say, the title bar, you end up displaying some icky short file name alias instead of the proper long file name that the user expects to see.

The most common way people mess up their program registration is by forgetting to quote spaces in the path to the program itself! For example, an erroneous registration might go something like this:

```
HKEY_CLASSES_ROOT
  litfile
    shell
      open
        command
          (default) = C:\Program Files\LitWare Deluxe\litware.exe “%1”
```

Observe that the spaces in the path “C:\Program Files\Litware Deluxe\litware.exe” are not quoted in the program registration. Consequently, the shell mistakenly believes that the program name is “C:\Program”, which it cannot find. The shell therefore plays it safe and assumes no LFN support.

Compatibility note: As part of other security work, the code in the shell that parses these command lines was augmented to chase down the “intended” path of the program. This presented the opportunity to fix that third case, so that the shell could find the program after all and see that it supported long file names, thereby saving the user the ignominy of seeing their wonderful file name turn into a mush of tildes.

And after we made the change, we had to take it out.

Because there were programs that not only registered themselves incorrectly, but were **relying on the shell not being smart enough** to find their real location, resulting in the program receiving the short name on the command line. Turns out these programs **wanted the short name**, and doing this fake-out was their way of accomplishing it.

(And to those of you who are already shouting, “Go ahead and break them,” that’s all fine and good as long as the thing that’s incompatible isn’t something you use. But if it’s your program, or a program your company relies on, I expect you’re going to change your tune.)

Raymond Chen

Follow