Why are there both TMP and TEMP environment variables, and which one is right?

devblogs.microsoft.com/oldnewthing/20150417-00

April 17, 2015

Raymond Chen

If you snoop around your environment variables, you may notice that there are two variables that propose to specify the location of temporary files. There is one called `TMP` and another called `TEMP`. Why two? And if they disagree, then who's right?

Rewind to 1973. The operating system common on microcomputers was CP/M. The CP/M operating system had no environment variables. That sounds like a strange place to start a discussion of environment variables, but it's actually important. Since it had no environment variables, there was consequently neither a `TMP` nor a `TEMP` environment variable. If you wanted to configure a program to specify where to put its temporary files, you needed to do some sort of program-specific configuration, like patching a byte in the executable to indicate the drive letter where temporary files should be stored.

(My recollection is that most CP/M programs were configured via patching. At least that's how I configured them. I remember my WordStar manual coming with details about which bytes to patch to do what. There was also a few dozen bytes of patch space set aside for you to write your own subroutines, in case you needed to add custom support for your printer. I did this to add an "Is printer ready to accept another character?" function, which allowed for smoother background printing.)

Move forward to 1981. The 8086 processor and the MS-DOS operating system arrived on the scene. The design of both the 8086 processor and the MS-DOS operating system were strongly inspired by CP/M, so much so that it was the primary design goal that it be possible to take your CP/M program written for the 8080 processor and machine-translate it into an MS-DOS program written for the 8086 processor. Mind you, the translator assumed that you didn't play any sneaky tricks like self-modifying code, jumping into the middle of an instruction, or using code as data, but if you played honest, the translator would convert your program.

(The goal of allowing machine-translation of code written for the 8080 processor into code written for the 8086 processor helps to explain some of the quirks of the 8086 instruction set. For example, the H and L registers on the 8080 map to the BH and BL registers on the

8086, and on the 8080, the only register that you could use to access a computed address was HL. This is why of the four basic registers AX, BX, CX, and DX on the 8086, the only one that you can use to access memory is BX.)

One of the things that MS-DOS added beyond compatibility with CP/M was environment variables. Since no existing CP/M programs used environment variables, none of the first batch of programs for MS-DOS used them either, since the first programs for MS-DOS were all ported from CP/M. Sure, you could set a `TEMP` or `TMP` environment variable, but nobody would pay attention to it.

Over time, programs were written with MS-DOS as their primary target, and they started to realize that they could use environment variables as a way to store configuration data. In the ensuing chaos of the marketplace, two environment variables emerged as the front-runners for specifying where temporary files should go: `TEMP` and `TMP`.

MS-DOS 2.0 introduced the ability to pipe the output of one program as the input of another. Since MS-DOS was a single-tasking operating system, this was simulated by redirecting the first program's output to a temporary file and running it to completion, then running the second program with its input redirected from that temporary file. Now all of a sudden, MS-DOS needed a location to create temporary files! For whatever reason, the authors of MS-DOS chose to use the `TEMP` variable to control where these temporary files were created.

Mind you, the fact that `COMMAND.COM` chose to go with `TEMP` didn't affect the fact that other programs could use either `TEMP` or `TMP`, depending on the mood of their original author. Many programs tried to appease both sides of the conflict by checking for both, and it was up to the mood of the original author which one it checked first. For example, the old `DISKCOPY` and `EDIT` programs would look for `TEMP` before looking for `TMP`.

Windows went through a similar exercise, but for whatever reason, the original authors of the `GetTempFileName` function chose to look for `TMP` before looking for `TEMP`.

The result of all this is that the directory used for temporary files by any particular program is at the discretion of that program, Windows programs are likely to use the `GetTempFileName` function to create their temporary files, in which case they will prefer to use `TMP`.

When you go to the Environment Variables configuration dialog, you'll still see both variables there, `TMP` and `TEMP`, still duking it out for your attention. It's like Adidas versus Puma, geek version.

Raymond Chen

Follow