What happens if I define one environment variable in terms of the value of another environment variable?

devblogs.microsoft.com/oldnewthing/20231212-00

December 12, 2023

Raymond Chen

In Windows, you can call up the Environment Variables dialog box to edit both the System environment variables and the User environment variables. What happens if you use the `%` sign to reference one environment variable from another?

Environment variable processing occurs in several steps.[1]

- Core system environment variables: `ALLUSERSPROFILE`, `ProgramData`, `PUBLIC`, `SystemDrive`, `SystemRoot`.
- System environment variables of type `REG_SZ`.
- System environment variables of type `REG_EXPAND_SZ`.
- Core user environment variables: `APPDATA`, `COMPUTERNAME`, `LOCALAPPDATA`, `ProgramFiles`, `USERPROFILE`.
- User environment variables of type `REG_SZ`.
- User environment variables of type `REG_EXPAND_SZ`.
- Account environment variables: `USERDNSDOMAIN`, `USERDOMAIN`, `USERNAME`.

The rule is that each step can redefine variables set by a previous step, and variables of type `REG_EXPAND_SZ` can depend on variables set by a previous step, but cannot depend on variables set elsewhere within the same step or by a future step.

This means, for example, that you can use `%USERPROFILE%` in the definition of a User environment variable of type `REG_EXPAND_SZ`,, but not in the definition of a System environment variable.

There is a bonus special rule for the `PATH` environment variable: The User definition of the `PATH` environment variable is *appended to* the System definition, rather than replacing it.[2]

If you have a `REG_EXPAND_SZ` between variables at the same step, it is unspecified whether the expansion is the old value or the new value, so don't do that.[3]

[1] This is a simplified discussion for the purpose of exposition.

[2] This special rule also applies to the vestigial `LibPath` and `Os2LibPath` environment variables.

[3] To avoid people taking dependencies on implementation details, I would have enforced the rule more strictly and consistently and said that all `REG_EXPAND_SZ` expansions use the value of the variable as defined by previous steps, and any changes made in the same step are not visible. But of course the implementation was written before the realization that people would try to create dependencies among variables in the same group.