A discovered quirk is just few steps away from becoming a feature

devblogs.microsoft.com/oldnewthing/20140425-00

April 25, 2014

Raymond Chen

Commenter Cherry wonders who invented all those strange syntaxes, like `set "` to show all environment variables, including the hidden ones.

An interesting historical note is the origin of the convention in unix that files whose names begin with a dot are hidden by default (here's the relevant portion). That article highlights how a discovered quirk is just a few steps away from becoming a feature.

As Master Yoda might put it: Discovery leads to dissemination. Dissemination leads to adoption. Adoption leads to entrenchment. Entrenchment creates a compatibility constraint.

As I've noted many times, the batch language was not designed. It simply evolved out of the old CP/M program `SUBMIT`, which was an even more rudimentary batch processor. (The original `SUBMIT.COM` didn't have conditional branches. It merely ran every line in your batch file one after another.)

One of the consequences of something that old is that any quirk, once discovered, can turn into a feature, and from there it becomes a support burden and compatibility constraint. We've seen this many times before: Counting the number of lines in a file by exploiting a buffer underflow bug in `FIND.COM`. Update the last-modified time of a file by using a magic sequence of punctuation marks. Echoing a blank line by typing `ECHO.`. All of these were accidental discovered behaviors (just like unix dot files) which became entrenched. Even when the underlying program was completely rewritten, these special quirks had to be specifically detected and painstakingly reproduced because so many programs (*i.e.*, batch files) relied on them.

For `set "`, it's a case of taking advantage of two quirks in the implementation: The first quirk is that a missing close-quotation mark is forgiven. That means that `set "` is logically equivalent to `set ""`.

You are therefore asking for a filtered list of environment variables, but passing the logical equivalent of no filter. Specifically, you're asking for all environment variables which begin with the empty string, and it so happens that every string begins with the empty string. The second quirk is that when an explicit filter is applied, the `set` command disables its default filter of "Hide environment variables whose names begin with an equals sign."

In other words, the code goes like this:

```
foreach (var entry in Environment.GetEnvironmentVariables()) {
 if (prefixFilter != null ?
     entry.Key.StartsWith(prefixFilter) :
     !entry.Key.StartsWith("=")) {
  Console.WriteLine("{0}={1}", entry.Key, entry.Value);
 }
}
```

Perhaps this is a bug, and it should have been written like this:

```
foreach (var entry in Environment.GetEnvironmentVariables()) {
 if (!entry.Key.StartsWith("=") &&
     (prefixFilter == null || entry.Key.StartsWith(prefixFilter))) {
  Console.WriteLine("{0}={0}", entry.Key, entry.Value);
 }
}
```

But it's too late to fix it now. People have discovered the quote trick, so it's now a feature and therefore a compatibility constraint.

Raymond Chen

Follow