Reading the output of a command from batch

devblogs.microsoft.com/oldnewthing/20050909-24

September 9, 2005

Raymond Chen

The `FOR` command has become the batch language's looping construct. If you ask for help via `FOR /?` you can see all the ways it has become overloaded. For example, you can read the output of a command by using the `for` command.

```
FOR /F “tokens=*” %i IN (‘ver’) DO echo %i
```

The /F switch in conjunction with the single quotation marks indicates that the quoted string is a command to run, whose output is then to be parsed and returned in the specified variable (or variables). The option `“tokens=*”` says that the entire line should be collected. There are several other options that control the parsing, which I leave you to read on your own.

The kludgy batch language gets even kludgier. Why is the batch language such a grammatical mess? Backwards compatibility.

Any change to the batch language cannot break compatibility with the millions of batch programs already in existence. Such batch files are burned onto millions of CDs (you'd be surprised how many commercial programs use batch files, particularly as part of their installation process). They're also run by corporations around the world to get their day-to-day work done. Plus of course the batch files written by people like you and me to do a wide variety of things. Any change to the batch language must keep these batch files running.

Of course, one could invent a brand new batch language, let's call it Batch^2 for the sake of discussion, and thereby be rid of the backwards compatibility constraints. But with that decision come different obstacles.

Suppose you have a 500-line batch file and you want to add one little feature to it, but that new feature is available only in Batch^2. Does this mean that you have to do a complete rewrite of your batch program into Batch^2? Your company spent years tweaking this batch file over the years. (And by “tweaking” I might mean “turning into a plate of spaghetti”.) Do you want to take the risk of introducing who-knows-how-many bugs and breaking various obscure features as part of the rewrite into Batch^2?

Suppose you decide to bite the bullet and rewrite. Oh, but Batch² is available only in more recent versions of Windows. Do you tell your customers, “We don’t support the older versions of Windows any more”? Or do you bite another bullet and say, “We support only versions of Windows that have Batch²”?

I’m not saying that it won’t happen. (In fact, I’m under the impression that there are already efforts to design a new command console language with an entirely new grammar. Said effort might even be presenting at the PDC in a few days.) I’m just explaining why the classic batch language is such a mess. Welcome to evolution.

[Raymond is currently away; this message was pre-recorded.]

Raymond Chen

Follow