# As random as I wanna be: Why cmd.exe's %RANDOM% isn't so random

devblogs.microsoft.com/oldnewthing/20100617-00

June 17, 2010

Raymond Chen

Somebody on my team reported that a particular script in our project's build process would fail with bizarre output maybe once in fifty tries. This script was run from a `makefile`, and the result was a failed build. Rerunning *make* fixed the problem, but that's not much consolation when the build lab encounters it approximately every other day. The strange thing about the bizarre output was that it appeared to contain a mix of two different runs. How could the output of two runs be mixed into one output file?

The script was a batch file, and it generated its output in a few different steps, storing the intermediate output in randomly-named temporary files, taking advantage of the `%RANDOM%` pseudovariable to generate the name of those temporary files. (They were `%TEMP%\%RANDOM%.tmp1`, `%TEMP%\%RANDOM%.tmp2`, you get the idea.)

Cutting to the chase: The reason for the mixed output was that the `%RANDOM%` pseudo-variable wasn't random enough. If two copies of the script are running at the same time, they will get the *same* "random" number and end up mixing their output together. (And running multiple builds at the same time is something the people in the build lab are wont to do.)

It turns out that the Windows command processor uses the standard naïve algorithm for seeding the random number generator:

```
srand((unsigned)time(NULL));
```

Since `time` has a resolution of one second, two command prompts launched in rapid succession have a good chance of seeding the random number generator with the same timestamp, which means that they will have the same random number stream.

```
C> copy con notsorandom.cmd
@pause
@echo %RANDOM%
^Z
        1 file(s) copied.
C> for /l %i in (1,1,3000) do @cmd /c notsorandom.cmd
// hold down the space bar
Press any key to continue . . .
14153
Press any key to continue . . .
14153
Press any key to continue . . .
14153
Press any key to continue . . .
14153
Press any key to continue . . .
14156
Press any key to continue . . .
14156
Press any key to continue . . .
14156
Press any key to continue . . .
14156
Press any key to continue . . .
14156
Press any key to continue . . .
14160
Press any key to continue . . .
14160
Press any key to continue . . .
14160
```

Notice that the `%RANDOM%` pseudovariable generates the same "random" number until the clock ticks over another second. (Notice also that the "random" numbers don't look all that random.)

We fixed the script so it generated its temporary file in the project's output directory rather than in the (shared) `%TEMP%` directory. That way, even if two copies of the project are building at the same time, they will generate their temporary files in different directories and not step on each other.

**Exercise**: There is much subtlety in that `for` command. Describe alternative formulations of the `for` command, both those that work and those that don't. To get you started: Explain the output of this variation:

```
for /l %i in (1,1,300) do @(pause&echo %RANDOM%)
```

**Obligatory batch file bashing**: Every time I write an entry about batch files, you can count on people complaining about how insane the batch programming language is. The batch language wasn't designed; it evolved. (And according to commenter Daev, it followed a form of parallel evolution from what most people are familiar with.) I doubt anybody actually enjoy writing batch files. At best you tolerate it. I'm just trying to make it slightly more tolerable. I bet these are the same people who complain to their tax preparer about the complexity of tax law.