How did copying and renaming with wildcards work in MS-DOS?

devblogs.microsoft.com/oldnewthing/20210713-00

July 13, 2021

Raymond Chen

Some time ago, I described how wildcards worked in MS-DOS, specifically how wildcards participate in pattern matching. Today, I'll look at how wildcards participate in copying and renaming.

First, the source and destination patterns are expanded into the eleven-character FCB format by the algorithm I described in that earlier article.

Next, the directory is searched for files that match the source pattern.

Once such a match is found, the fun begins: Applying the rename pattern.

The way it works is that the rename pattern is used to produce the resulting file name, except that if a question mark is encountered in the rename pattern, then the corresponding character from the source file name is copied.

Here's an example with renaming, although the same exercise also applies to copying:

```
REN ABC*.D?F GHIJ*.KL?
```

The source and destination patterns are

	Human-readable	Parsed pattern										
Source	`ABC*.D?F`	A	B	C	?	?	?	?	?	D	?	F
Destination	`GHIJ*.KL?`	G	H	I	J	?	?	?	?	K	L	?

Suppose we have a file `ABC12345.D6F` that matches the source pattern. How do we transform it according to the destination pattern?

Simple: Stack the original file name and the destination pattern on top of each other. For each character of the output, take the corresponding character from the destination pattern, unless it is a question mark, in which case you take the corresponding character from the original file name.

	Human-readable	Parsed pattern										
Actual	ABC12345.D6F	A	B	C	1	2	3	4	5	D	6	F
Destination	GHIJ*.KL?	G	H	I	J	?	?	?	?	K	L	?
Result	GHIJ2345.KLF	G	H	I	J	2	3	4	5	K	L	F

One way of thinking of this is that you treat the destination pattern as a stencil, with holes punched out where the question marks are. You then overlay the stencil on top of the original file name. The characters from the original file name show through the holes, and what you see is the result.

G	H	I	J	2	3	4	5	K	L	F

This algorithm enabled some simple rename patterns, like changing a file extension:

```
REN FRED*.TXT *.DOC
```

Suppose there is a file called `FRED123.TXT`. Let's see what happens:

	Human-readable	Parsed pattern										
Source	FRED*.*	F	R	E	D	?	?	?	?	?	?	?
Match	FRED123.TXT	F	R	E	D	1	2	3	·	T	X	T
Destination	*.DOC	?	?	?	?	?	?	?	?	D	O	C
Result	FRED123.DOC	F	R	E	D	1	2	3	·	D	O	C

Observe that we didn't have to repeat `FRED` in the replacement pattern. The asterisk (which parses into question marks) just copies the existing file name, which includes the `FRED`.

Does this make sense? Because it does carry its own consequences.

If you're not expecting the "copies the existing file name" behavior, and the question marks in the destination don't match the question marks in the source, the results can be somewhat surprising:

```
ren FRED*.* WILMA*.*
```

Suppose there is a file called `FRED123.TXT`, and you were hoping to rename it to `WILMA123.TXT`. Let's see what happens:

	Human-readable	Parsed pattern										
Source	`FRED*.*`	F	R	E	D	?	?	?	?	?	?	?
Match	`FRED123.TXT`	F	R	E	D	1	2	3	·	T	X	T
Destination	`WILMA*.*`	W	I	L	M	A	?	?	?	?	?	?
Result	`WILMA23.TXT`	W	I	L	M	A	2	3	·	T	X	T

Since `WILMA` is one character longer than `FRED`, the question marks don't line up. After copying `WILMA` to the result, we reach the first question mark in the destination, which lines up with the *second* question mark in the source, not the first. The character that is copied is the sixth character from the source, which is a `2`. The `1` from the source is not copied because it was overwritten by the `A` in `WILMA`.

Wildcards are just question marks, and question marks match or copy a single corresponding character. They don't "go looking around for their buddy question mark". Computers weren't that fancy back then.

We had agreed that the `REN FRED*.TXT *.DOC` made sense in that it didn't rename `FRED123.TXT` to `123.DOC`. But that rule that made sense then doesn't seem to make sense in this more complicated case where we are doing what is more like a search/replace in the filename.

It's important to understand the MS-DOS wildcard copy/rename algorithm because Windows remains compatible with it, so as not to break existing batch files. We'll look at this some more next time.

Raymond Chen

Follow