How-to: Match filenames with Wildcards

```
The * wildcard will match any sequence of characters
              (0 or more, including NULL characters)

The ? wildcard will match a single character
              (or a NULL at the end of a filename)
```

A few quirks affect the operation of wildcards which are best illustrated by example:

To match the filename BAR.TXT any of the following patterns will match:
```
?AR.TXT
BAR.*
??R.TXT
B?R.???
BA?.TXT
BA??.TXT
```

However the following will fail to match with BAR.TXT
```
??AR.TXT
?BAR.TXT
B??AR.TXT
```

Wildcard matching rules

`*` Generally matches any 0 or more characters, with one exception (see next rule). The non-greedy wildcard is free to match as many or as few characters as are necessary for the remainder of the mask to match.

`*.` At end of mask matches any 0 or more characters except for {dot}. In actuality, the rule applies with any number of {dot} and {space} characters between the * and terminal {dot}. The regular expression for this term is `"[*][. ]*[.]$"`

`?` Match 0 or one character, except for {dot}.
The only time it matches 0 characters is when it matches the end of the name, or the position before a {dot}.
The question mark can also be used more than once to match more than one character.

Wildcards match both the Short and Long filename

The command DIR /X will reveal short filenames if they exist, where many similar names exist in the same folder the short file name (SFN) will not always be an obvious contraction of the long name. e.g.

```
DIR /X

      2019-05-12  01:12    96 DIABLO~1 diablo1640
      2019-05-12  01:12    96 DIABLO~2 diablo1641
      2019-05-12  01:12    96 DIABLO~3 diablo1642
      2019-05-12  01:12    96 DIABLO~4 diablo1643
      2019-05-12  01:12    96 DIE359~1 diablo1644
      2019-05-12  01:12    96 DIC49C~1 diablo1648
      2019-05-12  01:12    96 DIF2E9~1 diablo1740
      2019-05-12  01:12    96 DIE2EF~1 diablo1649
```

As you can see, the first four short filenames follow the usual numeric tails (~1, ~2, ~3, ~4).
Then, from fifth and more files with similar beginning, the short names have four hex digits in the middle. This is actually a hex checksum of the long filename. (If you need a true file checksum look at `CertUtil -hashfile`)

Wildcards are supported by the following commands:
ATTRIB, CACLS, CIPER, COMPACT, COPY, DEL, DIR, EXPAND, EXTRACT, FIND, FINDSTR, FOR, FORFILES, FTP, ICACLS, IF EXIST, MORE, MOVE, MV, NET (*=Any Drive), PERMS, PRINT, REN, REPLACE, ROBOCOPY, ROUTE, TAKEOWN, TYPE, WHERE, XCACLS, XCOPY

The commands COPY and REN accept two sets of wildcards, there are some subtle differences between how these are treated, see the REN page for details.

The wildcards used by FORFILES are non-standard, but are similar to the wildcards used in PowerShell.

Undocumented Wildcards

The two undocumented wildcards, < and > can be used with commands like DIR and COPY, or to supply a command name but only if quoted: `DIR /b "<demo<"`

`<` Matches any 0 or more characters in either the base name or the extension, but never both.
Unlike the * wildcard, a single < cannot match characters in both the base name and the extension.
The {dot} is considered to be part of the base name, not the extension. There is one exception - If the name consists solely of an extension, without a base name, then the {dot} is considered to be part of the extension. This non-greedy wild card is free to match as many or as few characters as are necessary for the remainder of the mask to match.

`>` Is identical to `?.` The only difference is that it can be placed after a {dot} to prevent the {dot} from matching the end of the name.

Examples at Dostips

The < and > wildcards work with the following commands: CACLS, CIPHER, COPY, DEL, DIR, FINDSTR, IF EXIST, MOVE, TYPE

Numeric Comparisons

There are several contexts where CMD.EXE will parse a string as a numeric expression:

- IF comparisons - EQU, NEQ, LSS, LEQ, GEQ, GTR
- SET /A
- variable substring expansion - %var:~*n,m*%
- FOR /F "TOKENS=*n*"
- FOR /F "SKIP=*n*"
- FOR /L %%A in (*n1 n2 n3*)

For many purposes a 4 byte signed integer value ranging from -2,147,483,648 to 2,147,483,647 will suffice, but in the above contexts it is also possible to express the numbers in hexadecimal or octal notation.

e.g. Octal: 00, 07 Hex: 0x00, 0xFF

There are a number of subtle differences (Negative numbers, command, version of Windows) which affect how these numbers are parsed and these are described in the DosTips forum thread Rules for how CMD.EXE parses numbers.

"We usually see only the things we are looking for, so much that we sometimes see them where they are not" ~ Eric Hoffer

Related commands

FINDSTR
REN - Rename files.
How-to: Long and short filename issues
How did wildcards work in MS-DOS? - Raymond Chen