NAME

glob - globbing pathnames

DESCRIPTION

Long ago, in UNIX V6, there was a program */etc/glob* that would expand wildcard patterns. Soon afterward this became a shell built-in.

These days there is also a library routine glob(3) that will perform this function for a user program.

The rules are as follows (POSIX.2, 3.13).

Wildcard matching

A string is a wildcard pattern if it contains one of the characters ?, * or [. Globbing is the operation that expands a wildcard pattern into the list of pathnames matching the pattern. Matching is defined by:

A ? (not between brackets) matches any single character.

A * (not between brackets) matches any string, including the empty string.

Character classes

An expression *[...]* where the first character after the leading [is not an ! matches a single character, namely any of the characters enclosed by the brackets. The string enclosed by the brackets cannot be empty; therefore] can be allowed between the brackets, provided that it is the first character. (Thus, *[][!]* matches the three characters [,] and !.)

Ranges

There is one special convention: two characters separated by - denote a range. (Thus, *[A-Fa-f0-9]* is equivalent to *[ABCDEFabcdef0123456789]*.) One may include - in its literal meaning by making it the first or last character between the brackets. (Thus, *[]-]* matches just the two characters] and -, and *[--0]* matches the three characters -, ., 0, since / cannot be matched.)

Complementation

An expression *[!...]* matches a single character, namely any character that is not matched by the expression obtained by removing the first ! from it. (Thus, *[!]a-]* matches any single character except], a and -.)

One can remove the special meaning of ?, * and [by preceding them by a backslash, or, in case this is part of a shell command line, enclosing them in quotes. Between brackets these characters stand for themselves. Thus, *[[?*]* matches the four characters [, ?, * and .

Pathnames

Globbing is applied on each of the components of a pathname separately. A / in a pathname cannot be matched by a ? or * wildcard, or by a range like *[.-0]*. A range cannot contain an explicit / character; this would lead to a syntax error.

If a filename starts with a ., this character must be matched explicitly. (Thus, *rm * * will not remove .profile, and *tar c ** will not archive all your files; *tar c .* is better.)

Empty lists

The nice and simple rule given above: expand a wildcard pattern into the list of matching pathnames was the original UNIX definition. It allowed one to have patterns that expand into an empty list, as in

xv -wait 0 *.gif *.jpg

where perhaps no *.gif files are present (and this is not an error). However, POSIX requires that a wildcard pattern is left unchanged when it is syntactically incorrect, or the list of matching pathnames is empty. With*b ash* one can force the classical behavior using this command:

shopt -s nullglob

(Similar problems occur elsewhere. For example, where old scripts have

rm `find . -name *~`

new scripts require

rm -f nosuchfile `find . -name *~`

to avoid error messages from *rm* called with an empty argument list.)

NOTES

Regular expressions

Note that wildcard patterns are not regular expressions, although they are a bit similar. First of all, they match filenames, rather than text, and secondly, the conventions are not the same: for example, in a regular expression * means zero or more copies of the preceding thing.

Now that regular expressions have bracket expressions where the negation is indicated by a ^, POSIX has declared the effect of a wildcard pattern *[^...]* to be undefined.

Character classes and internationalization

Of course ranges were originally meant to be ASCII ranges, so that *[-%]* stands for *[!#%]* and *[a-z]* stands for any lowercase letter. Some UNIX implementations generalized this so that a range X-Y stands for the set of characters with code between the codes for X and for Y. However, this requires the user to know the character coding in use on the local system, and moreover, is not convenient if the collating sequence for the local alphabet differs from the ordering of the character codes. Therefore, POSIX extended the bracket notation greatly, both for wildcard patterns and for regular expressions. In the above we saw three types of items that can occur in a bracket expression: namely (i) the negation, (ii) explicit single characters, and (iii) ranges. POSIX specifies ranges in an internationally more useful way and adds three more types:

(iii) Ranges X-Y comprise all characters that fall between X and Y (inclusive) in the current collating sequence as defined by the **LC_COLLATE** category in the current locale.

(iv) Named character classes, like

[:alnum:] [:alpha:] [:blank:] [:cntrl:]
[:digit:] [:graph:] [:lower:] [:print:]
[:punct:] [:space:] [:upper:] [:xdigit:]

so that one can say *[[:lower:]]* instead of *[a-z]*, and have things work in Denmark, too, where there are three letters past z in the alphabet. These character classes are defined by the **LC_CTYPE** category in the current locale.

(v) Collating symbols, like *[.ch.]* or *[.a-acute.]*, where the string between *[.* and *.]* is a collating element defined for the current locale. Note that this may be a multicharacter element.

(vi) Equivalence class expressions, like *[=a=]*, where the string between *[=* and *=]* is any collating element from its equivalence class, as defined for the current locale. For example, *[[=a=]]* might be equivalent to *[a]*, that is, to *[a[.a-acute.][.a-grave.][.a-umlaut.][.a-circumflex.]]*.

SEE ALSO

sh(1), fnmatch(3), glob(3), locale(7), regex(7)

COLOPHON

This page is part of release 3.74 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at http://www.kernel.org/doc/man-pages/.