11.7 Extended Regular Expressions

Of all the clones, `vim` provides the richest set of regular expression matching facilities. Much of the descriptive text in the list below is borrowed from the `vim` documentation:

`\|`

Indicates alternation, `house\|home`.

`\+`

Matches one or more of the preceding regular expression.

`\=`

Matches zero or one of the preceding regular expression.

`\{n,m}`

Matches *n* to *m* of the preceding regular expression, as much as possible. *n* and *m* are numbers between 0 and 32000; `vim` requires only the left brace to be preceded by a backslash, not the right brace.

`\{n}`

Matches *n* of the preceding regular expression.

`\{n,}`

Matches at least *n* of the preceding regular expression, as much as possible.

`\{,m}`

Matches 0 to *m* of the preceding regular expression, as much as possible.

`\{}`

Matches 0 or more of the preceding regular expression, as much as possible (same as `*`).

`\{-n,m}`

Matches *n* to *m* of the preceding regular expression, as few as possible.

`\{-n}`

Matches *n* of the preceding regular expression.

`\{-n,}`

Matches at least *n* of the preceding regular expression, as few as possible.

`\{-,m}`

Matches 0 to *m* of the preceding regular expression, as few as possible.

`\i`

Matches any identifier character, as defined by the `isident` option.

`\I`

Like `\i`, but excluding digits.

`\k`

Matches any keyword character, as defined by the `iskeyword` option.

`\K`

Like `\k`, but excluding digits.

`\f`

Matches any filename character, as defined by the `isfname` option.

`\F`

Like `\f`, but excluding digits.

`\p`

Matches any printable character, as defined by the `isprint` option.

`\P`

Like `\p`, but excluding digits.

`\s`

Matches a whitespace character (exactly space and tab).

`\S`

Matches anything that isn't a space or a tab.

`\b`

Backspace.

`\e`

Escape.

`\r`

Carriage return.

`\t`

Tab.

`\n`

Reserved for future use. Eventually, it will be used for matching multi-line patterns. See the `vim` documentation for more details.

`~`

Matches the last given substitute (i.e., replacement) string.

`\(...\)`

Provides grouping for `*`, `\+`, and `\=`, as well as making matched sub-texts available in the replacement part of a substitute command (`\1`, `\2`, etc.).

`\1`

Matches the same string that was matched by the first sub-expression in `\(` and `\)`. For example: `\([a-z]\).\1` matches *ata*, *ehe*, *tot*, etc. `\2`, `\3`, and so on may be used to represent the second, third, and so forth subexpressions.

The `isident`, `iskeyword`, `isfname`, and `isprint` options define the characters that appear in identifiers, keywords, and filenames, and that are printable. Use of these options makes regular expression matching very flexible.

PREVIOUS 11.6 GUI Interfaces | HOME BOOK INDEX | NEXT 11.8 Improved Editing Facilities

LIBRARY HOME | UNIX POWER TOOLS | UNIX IN A NUTSHELL | LEARNING VI | SED & AWK | KORN SHELL | LEARNING UNIX