

Section 8 - Regular expressions

8. Regular expressions

Exim supports the use of regular expressions in many of its options. It uses the PCRE regular expression library; this provides regular expression matching that is compatible with Perl 5. The syntax and semantics of regular expressions is discussed in many Perl reference books, and also in Jeffrey Friedl's *Mastering Regular Expressions*, which is published by O'Reilly (see <http://www.oreilly.com/catalog/regex2/>).

The documentation for the syntax and semantics of the regular expressions that are supported by PCRE is included in plain text in the file `doc/pcrpattern.txt` in the Exim distribution, and also in the HTML tarbundle of Exim documentation. It describes in detail the features of the regular expressions that PCRE supports, so no further description is included here. The PCRE functions are called from Exim using the default option settings (that is, with no PCRE options set), except that the `PCRE_CASELESS` option is set when the matching is required to be case-insensitive.

In most cases, when a regular expression is required in an Exim configuration, it has to start with a circumflex, in order to distinguish it from plain text or an "ends with" wildcard. In this example of a configuration setting, the second item in the colon-separated list is a regular expression. `domains = a.b.c : ^\d{3} : *.y.z : ...`

The doubling of the backslash is required because of string expansion that precedes interpretation – see section 11.1 for more discussion of this issue, and a way of avoiding the need for doubling backslashes. The regular expression that is eventually used in this example contains just one backslash. The circumflex is included in the regular expression, and has the normal effect of "anchoring" it to the start of the string that is being matched.

There are, however, two cases where a circumflex is not required for the recognition of a regular expression: these are the match condition in a string expansion, and the matches condition in an Exim filter file. In these cases, the relevant string is always treated as a regular expression; if it does not start with a circumflex, the expression is not anchored, and can match anywhere in the subject string.

In all cases, if you want a regular expression to match at the end of a string, you must code the `$` metacharacter to indicate this. For example: `domains = ^\d{3}\\.example`

matches the domain `123.example`, but it also matches `123.example.com`. You need to use: `domains = ^\d{3}\\.example\\$`

if you want `example` to be the top-level domain. The backslash before the `$` is needed because string expansion also interprets dollar characters.

8.1 Testing regular expressions

A program called `pcrtest` forms part of the PCRE distribution and is built with PCRE during the process of building Exim. It is primarily intended for testing PCRE itself, but it can also be used for experimenting with regular expressions. After building Exim, the binary can be found in the build directory (it is not installed anywhere automatically). There is documentation of various options in `doc/pcrtest.txt`, but for simple testing, none are needed. This is the output of a sample run of `pcrtest`:

```
re> /^([@]+)@.+.(ac|edu)\.(?!kr)[a-z]{2}$/
data> x@y.ac.uk
0: x@y.ac.uk
1: x
2: ac
data> x@y.ac.kr
No match
data> x@y.edu.com
No match
data> x@y.edu.co
0: x@y.edu.co
1: x
2: edu
```

Input typed by the user is shown in bold face. After the "re>" prompt, a regular expression enclosed in delimiters is expected. If this compiles without error, "data>" prompts are given for strings against which the expression is matched. An empty data line causes a new regular expression to be read. If the match is successful, the captured substring values (that is, what would be in the variables `$0`, `$1`, `$2`, etc.) are shown. The above example tests

for an email address whose domain ends with either “ac” or “edu” followed by a two-character top-level domain that is not “kr”. The local part is captured in \$1 and the “ac” or “edu” in \$2.