NAME

PCRE2 - Perl-compatible regular expressions (revised API)

DIFFERENCES BETWEEN PCRE2 AND PERL

This document describes some of the differences in the ways that PCRE2 and Perl handle regular expressions. The differences described here are with respect to Perl version 5.32.0, but as both Perl and PCRE2 are continually changing, the information may at times be out of date.

1. PCRE2 has only a subset of Perl's Unicode support. Details of what it does have are given in the **pcre2unicode** page.

2. Like Perl, PCRE2 allows repeat quantifiers on parenthesized assertions, but they do not mean what you might think. For example, (?!a){3} does not assert that the next three characters are not "a". It just asserts that the next character is not "a" three times (in principle; PCRE2 optimizes this to run the assertion just once). Perl allows some repeat quantifiers on other assertions, for example, \b* (but not \b{3}, though oddly it does allow ^{3}), but these do not seem to have any use. PCRE2 does not allow any kind of quantifier on non-lookaround assertions.

3. Capture groups that occur inside negative lookaround assertions are counted, but their entries in the offsets vector are set only when a negative assertion is a condition that has a matching branch (that is, the condition is false). Perl may set such capture groups in other circumstances.

4. The following Perl escape sequences are not supported: \F, \l, \L, \u, \U, and \N when followed by a character name. \N on its own, matching a non-newline character, and \N{U+dd..}, matching a Unicode code point, are supported. The escapes that modify the case of following letters are implemented by Perl's general string-handling and are not part of its pattern matching engine. If any of these are encountered by PCRE2, an error is generated by default. However, if either of the PCRE2_ALT_BSUX or PCRE2_EXTRA_ALT_BSUX options is set, \U and \u are interpreted as ECMAScript interprets them.

5. The Perl escape sequences \p, \P, and \X are supported only if PCRE2 is built with Unicode support (the default). The properties that can be tested with \p and \P are limited to the general category properties such as Lu and Nd, script names such as Greek or Han, and the derived properties Any and L&. Both PCRE2 and Perl support the Cs (surrogate) property, but in PCRE2 its use is limited. See the **pcre2pattern** documentation for details. The long synonyms for property names that Perl supports (such as \p{Letter}) are not supported by PCRE2, nor is it permitted to prefix any of these properties with "Is".

6. PCRE2 supports the \Q...\E escape for quoting substrings. Characters in between are treated as literals. However, this is slightly different from Perl in that $ and @ are also handled as literals inside the quotes. In Perl, they cause variable interpolation (but of course PCRE2 does not have variables). Also, Perl does "double-quotish backslash interpolation" on any backslashes between \Q and \E which, its documentation says, "may lead to confusing results". PCRE2 treats a backslash between \Q and \E just like any other character. Note the following examples:

Pattern	PCRE2 matches	Perl matches
\Qabc$xyz\E	abc$xyz	abc followed by the contents of $xyz
\Qabc\$xyz\E	abc\$xyz	abc\$xyz
\Qabc\E\$\Qxyz\E	abc$xyz	abc$xyz
\QA\B\E	A\B	A\B
\Q\\E	\	\\E

The \Q...\E sequence is recognized both inside and outside character classes by both PCRE2 and Perl.

7. Fairly obviously, PCRE2 does not support the (?{code}) and (??{code}) constructions. However, PCRE2 does have a "callout" feature, which allows an external function to be called during pattern matching. See the **pcre2callout** documentation for details.

8. Subroutine calls (whether recursive or not) were treated as atomic groups up to PCRE2 release 10.23, but

from release 10.30 this changed, and backtracking into subroutine calls is now supported, as in Perl.

9. In PCRE2, if any of the backtracking control verbs are used in a group that is called as a subroutine (whether or not recursively), their effect is confined to that group; it does not extend to the surrounding pattern. This is not always the case in Perl. In particular, if (*THEN) is present in a group that is called as a subroutine, its action is limited to that group, even if the group does not contain any | characters. Note that such groups are processed as anchored at the point where they are tested.

10. If a pattern contains more than one backtracking control verb, the first one that is backtracked onto acts. For example, in the pattern A(*COMMIT)B(*PRUNE)C a failure in B triggers (*COMMIT), but a failure in C triggers (*PRUNE). Perl's behaviour is more complex; in many cases it is the same as PCRE2, but there are cases where it differs.

11. There are some differences that are concerned with the settings of captured strings when part of a pattern is repeated. For example, matching "aba" against the pattern /^(a(b)?)+$/ in Perl leaves $2 unset, but in PCRE2 it is set to "b".

12. PCRE2's handling of duplicate capture group numbers and names is not as general as Perl's. This is a consequence of the fact the PCRE2 works internally just with numbers, using an external table to translate between numbers and names. In particular, a pattern such as (?|(?<a>A)|(?<b>B)), where the two capture groups have the same number but different names, is not supported, and causes an error at compile time. If it were allowed, it would not be possible to distinguish which group matched, because both names map to capture group number 1. To avoid this confusing situation, an error is given at compile time.

13. Perl used to recognize comments in some places that PCRE2 does not, for example, between the (and ? at the start of a group. If the /x modifier is set, Perl allowed white space between (and ? though the latest Perls give an error (for a while it was just deprecated). There may still be some cases where Perl behaves differently.

14. Perl, when in warning mode, gives warnings for character classes such as [A-\d] or [a-[:digit:]]. It then treats the hyphens as literals. PCRE2 has no warning features, so it gives an error in these cases because they are almost certainly user mistakes.

15. In PCRE2, the upper/lower case character properties Lu and Ll are not affected when case-independent matching is specified. For example, \p{Lu} always matches an upper case letter. I think Perl has changed in this respect; in the release at the time of writing (5.32), \p{Lu} and \p{Ll} match all letters, regardless of case, when case independence is specified.

16. From release 5.32.0, Perl locks out the use of \K in lookaround assertions. From release 10.38 PCRE2 does the same by default. However, there is an option for re-enabling the previous behaviour. When this option is set, \K is acted on when it occurs in positive assertions, but is ignored in negative assertions.

17. PCRE2 provides some extensions to the Perl regular expression facilities. Perl 5.10 included new features that were not in earlier versions of Perl, some of which (such as named parentheses) were in PCRE2 for some time before. This list is with respect to Perl 5.32:

(a) Although lookbehind assertions in PCRE2 must match fixed length strings, each alternative toplevel branch of a lookbehind assertion can match a different length of string. Perl requires them all to have the same length.

(b) From PCRE2 10.23, backreferences to groups of fixed length are supported in lookbehinds, provided that there is no possibility of referencing a non-unique number or name. Perl does not support backreferences in lookbehinds.

(c) If PCRE2_DOLLAR_ENDONLY is set and PCRE2_MULTILINE is not set, the $ meta-character matches only at the very end of the string.

(d) A backslash followed by a letter with no special meaning is faulted. (Perl can be made to issue a warning.)

(e) If PCRE2_UNGREEDY is set, the greediness of the repetition quantifiers is inverted, that is, by default they are not greedy, but if followed by a question mark they are.

(f) PCRE2_ANCHORED can be used at matching time to force a pattern to be tried only at the first matching position in the subject string.

(g) The PCRE2_NOTBOL, PCRE2_NOTEOL, PCRE2_NOTEMPTY and PCRE2_NOTEMPTY_ATSTART options have no Perl equivalents.

(h) The \R escape sequence can be restricted to match only CR, LF, or CRLF by the PCRE2_BSR_ANYCRLF option.

(i) The callout facility is PCRE2-specific. Perl supports codeblocks and variable interpolation, but not general hooks on every match.

(j) The partial matching facility is PCRE2-specific.

(k) The alternative matching function (**pcre2_dfa_match()** matches in a different way and is not Perl-compatible.

(l) PCRE2 recognizes some special sequences such as (*CR) or (*NO_JIT) at the start of a pattern. These set overall options that cannot be changed within the pattern.

(m) PCRE2 supports non-atomic positive lookaround assertions. This is an extension to the lookaround facilities. The default, Perl-compatible lookarounds are atomic.

18. The Perl /a modifier restricts /d numbers to pure ascii, and the /aa modifier restricts /i case-insensitive matching to pure ascii, ignoring Unicode rules. This separation cannot be represented with PCRE2_UCP.

19. Perl has different limits than PCRE2. See the **pcre2limit** documentation for details. Perl went with 5.10 from recursion to iteration keeping the intermediate matches on the heap, which is ˜10% slower but does not fall into any stack-overflow limit. PCRE2 made a similar change at release 10.30, and also has many build-time and run-time customizable limits.

AUTHOR

Philip Hazel
Retired from University Computing Service
Cambridge, England.

REVISION

Last updated: 30 August 2021