NAME

PCRE2 - Perl-compatible regular expressions (revised API)

PCRE2 REGULAR EXPRESSION SYNTAX SUMMARY

The full syntax and semantics of the regular expressions that are supported by PCRE2 are described in the **pcre2pattern** documentation. This document contains a quick-reference summary of the syntax.

QUOTING

```
\x        where x is non-alphanumeric is a literal x
\Q...\E   treat enclosed characters as literal
```

ESCAPED CHARACTERS

This table applies to ASCII and Unicode environments. An unrecognized escape sequence causes an error.

```
\a        alarm, that is, the BEL character (hex 07)
\cx       "control-x", where x is any ASCII printing character
\e        escape (hex 1B)
\f        form feed (hex 0C)
\n        newline (hex 0A)
\r        carriage return (hex 0D)
\t        tab (hex 09)
\0dd      character with octal code 0dd
\ddd      character with octal code ddd, or backreference
\o{ddd..} character with octal code ddd..
\N{U+hh..} character with Unicode code point hh.. (Unicode mode only)
\xhh      character with hex code hh
\x{hh..}  character with hex code hh..
```

If PCRE2_ALT_BSUX or PCRE2_EXTRA_ALT_BSUX is set ("ALT_BSUX mode"), the following are also recognized:

```
\U        the character "U"
\uhhhh    character with hex code hhhh
\u{hh..}  character with hex code hh.. but only for EXTRA_ALT_BSUX
```

When \x is not followed by {, from zero to two hexadecimal digits are read, but in ALT_BSUX mode \x must be followed by two hexadecimal digits to be recognized as a hexadecimal escape; otherwise it matches a literal "x". Likewise, if \u (in ALT_BSUX mode) is not followed by four hexadecimal digits or (in EXTRA_ALT_BSUX mode) a sequence of hex digits in curly brackets, it matches a literal "u".

Note that \0dd is always an octal code. The treatment of backslash followed by a non-zero digit is complicated; for details see the section "Non-printing characters" in the **pcre2pattern** documentation, where details of escape processing in EBCDIC environments are also given. \N{U+hh..} is synonymous with \x{hh..} in PCRE2 but is not supported in EBCDIC environments. Note that \N not followed by an opening curly bracket has a different meaning (see below).

CHARACTER TYPES

```
.         any character except newline;
            in dotall mode, any character whatsoever
\C        one code unit, even in UTF mode (best avoided)
\d        a decimal digit
\D        a character that is not a decimal digit
\h        a horizontal white space character
```

\H a character that is not a horizontal white space character
\N a character that is not a newline
\p{*xx*} a character with the*xx* property
\P{*xx*} a character without the*xx* property
\R a newline sequence
\s a white space character
\S a character that is not a white space character
\v a vertical white space character
\V a character that is not a vertical white space character
\w a "word" character
\W a "non-word" character
\X a Unicode extended grapheme cluster

\C is dangerous because it may leave the current matching point in the middle of a UTF-8 or UTF-16 character. The application can lock out the use of \C by setting the PCRE2_NEVER_BACKSLASH_C option. It is also possible to build PCRE2 with the use of \C permanently disabled.

By default, \d, \s, and \w match only ASCII characters, even in UTF-8 mode or in the 16-bit and 32-bit libraries. However, if locale-specific matching is happening, \s and \w may also match characters with code points in the range 128-255. If the PCRE2_UCP option is set, the behaviour of these escape sequences is changed to use Unicode properties and they match many more characters.

GENERAL CATEGORY PROPERTIES FOR \p and \P

C Other
Cc Control
Cf Format
Cn Unassigned
Co Private use
Cs Surrogate

L Letter
Ll Lower case letter
Lm Modifier letter
Lo Other letter
Lt Title case letter
Lu Upper case letter
L& Ll, Lu, or Lt

M Mark
Mc Spacing mark
Me Enclosing mark
Mn Non-spacing mark

N Number
Nd Decimal number
Nl Letter number
No Other number

P Punctuation
Pc Connector punctuation
Pd Dash punctuation
Pe Close punctuation
Pf Final punctuation
Pi Initial punctuation
Po Other punctuation

Ps Open punctuation

S Symbol
Sc Currency symbol
Sk Modifier symbol
Sm Mathematical symbol
So Other symbol

Z Separator
Zl Line separator
Zp Paragraph separator
Zs Space separator

PCRE2 SPECIAL CATEGORY PROPERTIES FOR \p and \P

Xan Alphanumeric: union of properties L and N
Xps POSIX space: property Z or tab, NL, VT, FF, CR
Xsp Perl space: property Z or tab, NL, VT, FF, CR
Xuc Univerally-named character: one that can be represented by a Universal Character Name
Xwd Perl word: property Xan or underscore

Perl and POSIX space are now the same. Perl added VT to its space character set at release 5.18.

SCRIPT NAMES FOR \p AND \P

Adlam, Ahom, Anatolian_Hieroglyphs, Arabic, Armenian, Avestan, Balinese, Bamum, Bassa_Vah, Batak, Bengali, Bhaiksuki, Bopomofo, Brahmi, Braille, Buginese, Buhid, Canadian_Aboriginal, Carian, Caucasian_Albanian, Chakma, Cham, Cherokee, Chorasmian, Common, Coptic, Cuneiform, Cypriot, Cypro_Minoan, Cyrillic, Deseret, Devanagari, Dives_Akuru, Dogra, Duployan, Egyptian_Hieroglyphs, Elbasan, Elymaic, Ethiopic, Georgian, Glagolitic, Gothic, Grantha, Greek, Gujarati, Gunjala_Gondi, Gurmukhi, Han, Hangul, Hanifi_Rohingya, Hanunoo, Hatran, Hebrew, Hiragana, Imperial_Aramaic, Inherited, Inscriptional_Pahlavi, Inscriptional_Parthian, Javanese, Kaithi, Kannada, Katakana, Kayah_Li, Kharoshthi, Khitan_Small_Script, Khmer, Khojki, Khudawadi, Lao, Latin, Lepcha, Limbu, Linear_A, Linear_B, Lisu, Lycian, Lydian, Mahajani, Makasar, Malayalam, Mandaic, Manichaean, Marchen, Masaram_Gondi, Medefaidrin, Meetei_Mayek, Mende_Kikakui, Meroitic_Cursive, Meroitic_Hieroglyphs, Miao, Modi, Mongolian, Mro, Multani, Myanmar, Nabataean, Nandinagari, New_Tai_Lue, Newa, Nko, Nushu, Nyakeng_Puachue_Hmong, Ogham, Ol_Chiki, Old_Hungarian, Old_Italic, Old_North_Arabian, Old_Permic, Old_Persian, Old_Sogdian, Old_South_Arabian, Old_Turkic, Old_Uyghur, Oriya, Osage, Osmanya, Pahawh_Hmong, Palmyrene, Pau_Cin_Hau, Phags_Pa, Phoenician, Psalter_Pahlavi, Rejang, Runic, Samaritan, Saurashtra, Sharada, Shavian, Siddham, SignWriting, Sinhala, Sogdian, Sora_Sompeng, Soyombo, Sundanese, Syloti_Nagri, Syriac, Tagalog, Tagbanwa, Tai_Le, Tai_Tham, Tai_Viet, Takri, Tamil, Tangsa, Tangut, Telugu, Thaana, Thai, Tibetan, Tifinagh, Tirhuta, Toto, Ugaritic, Vai, Vithkuqi, Wancho, Warang_Citi, Yezidi, Yi, Zanabazar_Square.

CHARACTER CLASSES

[...] positive character class
[^...] negative character class
[x-y] range (can be used for hex characters)
[[:xxx:]] positive POSIX named set
[[:^xxx:]] negative POSIX named set

alnum alphanumeric
alpha alphabetic
ascii 0-127

```
blank      space or tab
cntrl      control character
digit      decimal digit
graph      printing, excluding space
lower      lower case letter
print      printing, including space
punct      printing, excluding alphanumeric
space      white space
upper      upper case letter
word       same as \w
xdigit     hexadecimal digit
```

In PCRE2, POSIX character set names recognize only ASCII characters by default, but some of them use Unicode properties if PCRE2_UCP is set. You can use \Q...\E inside a character class.

QUANTIFIERS

```
?          0 or 1, greedy
?+         0 or 1, possessive
??         0 or 1, lazy
*          0 or more, greedy
*+         0 or more, possessive
*?         0 or more, lazy
+          1 or more, greedy
++         1 or more, possessive
+?         1 or more, lazy
{n}        exactly n
{n,m}      at least n, no more than m, greedy
{n,m}+     at least n, no more than m, possessive
{n,m}?     at least n, no more than m, lazy
{n,}       n or more, greedy
{n,}+      n or more, possessive
{n,}?      n or more, lazy
```

ANCHORS AND SIMPLE ASSERTIONS

```
\b         word boundary
\B         not a word boundary
^          start of subject
             also after an internal newline in multiline mode
             (after any newline if PCRE2_ALT_CIRCUMFLEX is set)
\A         start of subject
$          end of subject
             also before newline at end of subject
             also before internal newline in multiline mode
\Z         end of subject
             also before newline at end of subject
\z         end of subject
\G         first matching position in subject
```

REPORTED MATCH POINT SETTING

```
\K         set reported start of match
```

From release 10.38 \K is not permitted by default in lookaround assertions, for compatibility with Perl. However, if the PCRE2_EXTRA_ALLOW_LOOKAROUND_BSK option is set, the previous behaviour is

re-enabled. When this option is set, \K is honoured in positive assertions, but ignored in negative ones.

ALTERNATION

expr|expr|expr...

CAPTURING

(...) capture group
(?<name>...) named capture group (Perl)
(?'name'...) named capture group (Perl)
(?P<name>...) named capture group (Python)
(?:...) non-capture group
(?|...) non-capture group; reset group numbers for capture groups in each alternative

In non-UTF modes, names may contain underscores and ASCII letters and digits; in UTF modes, any Unicode letters and Unicode decimal digits are permitted. In both cases, a name must not start with a digit.

ATOMIC GROUPS

(?>...) atomic non-capture group
(*atomic:...) atomic non-capture group

COMMENT

(?#....) comment (not nestable)

OPTION SETTING

Changes of these options within a group are automatically cancelled at the end of the group.

(?i) caseless
(?J) allow duplicate named groups
(?m) multiline
(?n) no auto capture
(?s) single line (dotall)
(?U) default ungreedy (lazy)
(?x) extended: ignore white space except in classes
(?xx) as (?x) but also ignore space and tab in classes
(?-...) unset option(s)
(?^) unset imnsx options

Unsetting x or xx unsets both. Several options may be set at once, and a mixture of setting and unsetting such as (?i-x) is allowed, but there may be only one hyphen. Setting (but no unsetting) is allowed after (?^ for example (?^in). An option setting may appear at the start of a non-capture group, for example (?i:...).

The following are recognized only at the very start of a pattern or after one of the newline or \R options with similar syntax. More than one of them may appear. For the first three, d is a decimal number.

(*LIMIT_DEPTH=d) set the backtracking limit to d
(*LIMIT_HEAP=d) set the heap size limit to d * 1024 bytes
(*LIMIT_MATCH=d) set the match limit to d
(*NOTEMPTY) set PCRE2_NOTEMPTY when matching
(*NOTEMPTY_ATSTART) set PCRE2_NOTEMPTY_ATSTART when matching
(*NO_AUTO_POSSESS) no auto-possessification (PCRE2_NO_AUTO_POSSESS)
(*NO_DOTSTAR_ANCHOR) no .* anchoring (PCRE2_NO_DOTSTAR_ANCHOR)
(*NO_JIT) disable JIT optimization
(*NO_START_OPT) no start-match optimization (PCRE2_NO_START_OPTIMIZE)

(*UTF) set appropriate UTF mode for the library in use
(*UCP) set PCRE2_UCP (use Unicode properties for \d etc)

Note that LIMIT_DEPTH, LIMIT_HEAP, and LIMIT_MATCH can only reduce the value of the limits set by the caller of **pcre2_match()** or **pcre2_dfa_match()**, not increase them. LIMIT_RECURSION is an obsolete synonym for LIMIT_DEPTH. The application can lock out the use of (*UTF) and (*UCP) by setting the PCRE2_NEVER_UTF or PCRE2_NEVER_UCP options, respectively, at compile time.

NEWLINE CONVENTION

These are recognized only at the very start of the pattern or after option settings with a similar syntax.

(*CR) carriage return only
(*LF) linefeed only
(*CRLF) carriage return followed by linefeed
(*ANYCRLF) all three of the above
(*ANY) any Unicode newline sequence
(*NUL) the NUL character (binary zero)

WHAT \R MATCHES

These are recognized only at the very start of the pattern or after option setting with a similar syntax.

(*BSR_ANYCRLF) CR, LF, or CRLF
(*BSR_UNICODE) any Unicode newline sequence

LOOKAHEAD AND LOOKBEHIND ASSERTIONS

(?=...))
(*pla:...)) positive lookahead
(*positive_lookahead:...))

(?!...))
(*nla:...)) negative lookahead
(*negative_lookahead:...))

(?<=...))
(*plb:...)) positive lookbehind
(*positive_lookbehind:...))

(?<!...))
(*nlb:...)) negative lookbehind
(*negative_lookbehind:...))

Each top-level branch of a lookbehind must be of a fixed length.

NON-ATOMIC LOOKAROUND ASSERTIONS

These assertions are specific to PCRE2 and are not Perl-compatible.

(?*...))
(*napla:...)) synonyms
(*non_atomic_positive_lookahead:...))

(?<*...))
(*naplb:...)) synonyms
(*non_atomic_positive_lookbehind:...))

SCRIPT RUNS

(*script_run:...)) script run, can be backtracked into
(*sr:...))

(*atomic_script_run:...)) atomic script run
(*asr:...))

BACKREFERENCES

\n reference by number (can be ambiguous)
\gn reference by number
\g{n} reference by number
\g+n relative reference by number (PCRE2 extension)
\g-n relative reference by number
\g{+n} relative reference by number (PCRE2 extension)
\g{-n} relative reference by number
\k<name> reference by name (Perl)
\k'name' reference by name (Perl)
\g{name} reference by name (Perl)
\k{name} reference by name (.NET)
(?P=name) reference by name (Python)

SUBROUTINE REFERENCES (POSSIBLY RECURSIVE)

(?R) recurse whole pattern
(?n) call subroutine by absolute number
(?+n) call subroutine by relative number
(?-n) call subroutine by relative number
(?&name) call subroutine by name (Perl)
(?P>name) call subroutine by name (Python)
\g<name> call subroutine by name (Oniguruma)
\g'name' call subroutine by name (Oniguruma)
\g<n> call subroutine by absolute number (Oniguruma)
\g'n' call subroutine by absolute number (Oniguruma)
\g<+n> call subroutine by relative number (PCRE2 extension)
\g'+n' call subroutine by relative number (PCRE2 extension)
\g<-n> call subroutine by relative number (PCRE2 extension)
\g'-n' call subroutine by relative number (PCRE2 extension)

CONDITIONAL PATTERNS

(?(condition)yes-pattern)
(?(condition)yes-pattern|no-pattern)

(?(n) absolute reference condition
(?(+n) relative reference condition
(?(-n) relative reference condition
(?(<name>) named reference condition (Perl)
(?('name') named reference condition (Perl)
(?(name) named reference condition (PCRE2, deprecated)
(?(R) overall recursion condition
(?(Rn) specific numbered group recursion condition
(?(R&name) specific named group recursion condition
(?(DEFINE) define groups for reference
(?(VERSION[>]=n.m) test PCRE2 version

(?(assert) assertion condition

Note the ambiguity of (?(R) and (?(Rn) which might be named reference conditions or recursion tests. Such a condition is interpreted as a reference condition if the relevant named group exists.

BACKTRACKING CONTROL

All backtracking control verbs may be in the form (*VERB:NAME). For (*MARK) the name is mandatory, for the others it is optional. (*SKIP) changes its behaviour if :NAME is present. The others just set a name for passing back to the caller, but this is not a name that (*SKIP) can see. The following act immediately they are reached:

(*ACCEPT) force successful match
(*FAIL) force backtrack; synonym (*F)
(*MARK:NAME) set name to be passed back; synonym (*:NAME)

The following act only when a subsequent match failure causes a backtrack to reach them. They all force a match failure, but they differ in what happens afterwards. Those that advance the start-of-match point do so only if the pattern is not anchored.

(*COMMIT) overall failure, no advance of starting point
(*PRUNE) advance to next starting character
(*SKIP) advance to current matching position
(*SKIP:NAME) advance to position corresponding to an earlier
(*MARK:NAME); if not found, the (*SKIP) is ignored
(*THEN) local failure, backtrack to next alternation

The effect of one of these verbs in a group called as a subroutine is confined to the subroutine call.

CALLOUTS

(?C) callout (assumed number 0)
(?Cn) callout with numerical data n
(?C"text") callout with string data

The allowed string delimiters are ‘ ’ " ^ % # $ (which are the same for the start and the end), and the starting delimiter { matched with the ending delimiter }. To encode the ending delimiter within the string, double it.

SEE ALSO

pcre2pattern(3), **pcre2api**(3), **pcre2callout**(3), **pcre2matching**(3), **pcre2**(3).

AUTHOR

Philip Hazel
Retired from University Computing Service
Cambridge, England.

REVISION

Last updated: 30 August 2021