

NAME

PCRE - Perl-compatible regular expressions

PCRE REGULAR EXPRESSION SYNTAX SUMMARY

The full syntax and semantics of the regular expressions that are supported by PCRE are described in the **pcregex** 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

CHARACTERS

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

Note that `\Odd` is always an octal code, and that `\8` and `\9` are the literal characters "8" and "9".

CHARACTER TYPES

`.` any character except newline;
in dotall mode, any character whatsoever
`\C` one data 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

<code>\S</code>	a character that is not a white space character
<code>\v</code>	a vertical white space character
<code>\V</code>	a character that is not a vertical white space character
<code>\w</code>	a "word" character
<code>\W</code>	a "non-word" character
<code>\X</code>	a Unicode extended grapheme cluster

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 `PCRE_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`

<code>C</code>	Other
<code>Cc</code>	Control
<code>Cf</code>	Format
<code>Cn</code>	Unassigned
<code>Co</code>	Private use
<code>Cs</code>	Surrogate
<code>L</code>	Letter
<code>Ll</code>	Lower case letter
<code>Lm</code>	Modifier letter
<code>Lo</code>	Other letter
<code>Lt</code>	Title case letter
<code>Lu</code>	Upper case letter
<code>L&</code>	Ll, Lu, or Lt
<code>M</code>	Mark
<code>Mc</code>	Spacing mark
<code>Me</code>	Enclosing mark
<code>Mn</code>	Non-spacing mark
<code>N</code>	Number
<code>Nd</code>	Decimal number
<code>Nl</code>	Letter number
<code>No</code>	Other number
<code>P</code>	Punctuation
<code>Pc</code>	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

PCRE 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	Universally-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 and PCRE changed at release 8.34.

SCRIPT NAMES FOR \p AND \P

Arabic, Armenian, Avestan, Balinese, Bamum, Bassa_Vah, Batak, Bengali, Bopomofo, Brahmi, Braille, Buginese, Buhid, Canadian_Aboriginal, Carian, Caucasian_Albanian, Chakma, Cham, Cherokee, Common, Coptic, Cuneiform, Cypriot, Cyrillic, Deseret, Devanagari, Duployan, Egyptian_Hieroglyphs, Elbasan, Ethiopic, Georgian, Glagolitic, Gothic, Grantha, Greek, Gujarati, Gurmukhi, Han, Hangul, Hanunoo, Hebrew, Hiragana, Imperial_Aramaic, Inherited, Inscriptional_Pahlavi, Inscriptional_Parthian, Javanese, Kaithi, Kannada, Katakana, Kayah_Li, Kharoshthi, Khmer, Khojki, Khudawadi, Lao, Latin, Lepcha, Limbu, Linear_A, Linear_B, Lisu, Lycian, Lydian, Mahajani, Malayalam, Mandaic, Manichaean, Meetei_Mayek, Mende_Kikakui, Meroitic_Cursive, Meroitic_Hieroglyphs, Miao, Modi, Mongolian, Mro, Myanmar, Nabataean, New_Tai_Lue, Nko, Ogham, Ol_Chiki, Old_Italic, Old_North_Arabian, Old_Permic, Old_Persian,

Old_South_Arabian, Old_Turkic, Oriya, Osmanya, Pahawh_Hmong, Palmyrene, Pau_Cin_Hau, Phags_Pa, Phoenician, Psalter_Pahlavi, Rejang, Runic, Samaritan, Saurashtra, Sharada, Shavian, Siddham, Sinhala, Sora_Sompeng, Sundanese, Syloti_Nagri, Syriac, Tagalog, Tagbanwa, Tai_Le, Tai_Tham, Tai_Viet, Takri, Tamil, Telugu, Thaana, Thai, Tibetan, Tifinagh, Tirhuta, Ugaritic, Vai, Warang_Citi, Yi.

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 PCRE, POSIX character set names recognize only ASCII characters by default, but some of them use Unicode properties if PCRE_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

<code>+?</code>	1 or more, lazy
<code>{n}</code>	exactly n
<code>{n,m}</code>	at least n, no more than m, greedy
<code>{n,m}+</code>	at least n, no more than m, possessive
<code>{n,m}?</code>	at least n, no more than m, lazy
<code>{n,}</code>	n or more, greedy
<code>{n,}+</code>	n or more, possessive
<code>{n,}?</code>	n or more, lazy

ANCHORS AND SIMPLE ASSERTIONS

<code>\b</code>	word boundary
<code>\B</code>	not a word boundary
<code>^</code>	start of subject also after internal newline in multiline mode
<code>\A</code>	start of subject
<code>\$</code>	end of subject also before newline at end of subject also before internal newline in multiline mode
<code>\Z</code>	end of subject also before newline at end of subject
<code>\z</code>	end of subject
<code>\G</code>	first matching position in subject

MATCH POINT RESET

<code>\K</code>	reset start of match
-----------------	----------------------

`\K` is honoured in positive assertions, but ignored in negative ones.

ALTERNATION

`expr|expr|expr...`

CAPTURING

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

ATOMIC GROUPS

(?>...) atomic, non-capturing group

COMMENT

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

OPTION SETTING

(?i) caseless
 (?J) allow duplicate names
 (?m) multiline
 (?s) single line (dotall)
 (?U) default ungreedy (lazy)
 (?x) extended (ignore white space)
 (?-...) unset option(s)

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.

(*LIMIT_MATCH=d) set the match limit to d (decimal number)
 (*LIMIT_RECURSION=d) set the recursion limit to d (decimal number)
 (*NO_AUTO_POSSESS) no auto-possessification (PCRE_NO_AUTO_POSSESS)
 (*NO_START_OPT) no start-match optimization (PCRE_NO_START_OPTIMIZE)
 (*UTF8) set UTF-8 mode: 8-bit library (PCRE_UTF8)
 (*UTF16) set UTF-16 mode: 16-bit library (PCRE_UTF16)
 (*UTF32) set UTF-32 mode: 32-bit library (PCRE_UTF32)
 (*UTF) set appropriate UTF mode for the library in use
 (*UCP) set PCRE_UCP (use Unicode properties for \d etc)

Note that LIMIT_MATCH and LIMIT_RECURSION can only reduce the value of the limits set by the caller of pcre_exec(), not increase them.

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

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

(?=...) positive look ahead

(?!...) negative look ahead

(?<=...) positive look behind

(?<!...) negative look behind

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

BACKREFERENCES

\n reference by number (can be ambiguous)

\gn reference by number

\g{n} reference by number

\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 subpattern by absolute number

(?+n) call subpattern by relative number

(?-n) call subpattern by relative number

(?&name) call subpattern by name (Perl)

(?P>name) call subpattern by name (Python)

\g<name> call subpattern by name (Oniguruma)

\g'name' call subpattern by name (Oniguruma)

\g<n> call subpattern by absolute number (Oniguruma)

\g'n' call subpattern by absolute number (Oniguruma)

\g<+n> call subpattern by relative number (PCRE extension)

\g'+n' call subpattern by relative number (PCRE extension)

\g<-n> call subpattern by relative number (PCRE extension)

\g'-n' call subpattern by relative number (PCRE 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 (PCRE)
 (? (R) ... overall recursion condition
 (? (Rn) ... specific group recursion condition
 (? (R&name) ... specific recursion condition
 (? (DEFINE) ... define subpattern for reference
 (? (assert) ... assertion condition

BACKTRACKING CONTROL

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
 (*PRUNE:NAME) equivalent to (*MARK:NAME)(*PRUNE)
 (*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
 (*THEN:NAME) equivalent to (*MARK:NAME)(*THEN)

CALLOUTS

(?C) callout
 (?Cn) callout with data n

SEE ALSO

pcrepattern(3), pcreapi(3), pcrecallout(3), pcrematching(3), pcre(3).

AUTHOR

Philip Hazel
University Computing Service
Cambridge CB2 3QH, England.

REVISION

Last updated: 08 January 2014
Copyright (c) 1997-2014 University of Cambridge.