

Syntax

Paul Wankadia edited this page on 17 Jun · 12 revisions

This page lists the regular expression syntax accepted by RE2.

It also lists some syntax accepted by PCRE, PERL, and VIM.

kinds of single-character expressions	examples
any character, possibly including newline (s=true)	.
character class	[xyz]
negated character class	[^xyz]
Perl character class (link)	\d
negated Perl character class	\D
ASCII character class (link)	[[:alpha:]]
negated ASCII character class	[[:^alpha:]]
Unicode character class (one-letter name)	\pN
Unicode character class	\p{Greek}
negated Unicode character class (one-letter name)	\PN
negated Unicode character class	\P{Greek}


	Composites
xy	x followed by y
x y	x or y (prefer x)


	Repetitions
x*	zero or more x , prefer more
x+	one or more x , prefer more
x?	zero or one x , prefer one
x{n,m}	n or n +1 or ... or m x , prefer more
x{n,}	n or more x , prefer more
x{n}	exactly n x
x*?	zero or more x , prefer fewer
x+?	one or more x , prefer fewer
x??	zero or one x , prefer zero
x{n,m}?	n or n +1 or ... or m x , prefer fewer
x{n,}?	n or more x , prefer fewer
x{n}?	exactly n x
x{ }	(≡ x*) (NOT SUPPORTED) VIM
x{ - }	(≡ x*?) (NOT SUPPORTED) VIM
x{ - n }	(≡ x{n}?) (NOT SUPPORTED) VIM
x=	(≡ x?) (NOT SUPPORTED) VIM

▼ Pages 7

- [Home](#)
- [Contribute](#)
- [CplusplusAPI](#)
- [Glossary](#)
- [Install](#)
- [Syntax](#)
- [WhyRE2](#)

Clone this wiki locally

<https://github.com/google/r> 

 Clone in Desktop

Implementation restriction: The counting forms `x{n,m}`, `x{n,}`, and `x{n}` reject forms that create a minimum or maximum repetition count above 1000. Unlimited repetitions are not subject to this restriction.

	Possessive repetitions
<code>x*+</code>	zero or more <code>x</code> , possessive (NOT SUPPORTED)
<code>x++</code>	one or more <code>x</code> , possessive (NOT SUPPORTED)
<code>x?+</code>	zero or one <code>x</code> , possessive (NOT SUPPORTED)
<code>x{n,m}+</code>	<code>n</code> or ... or <code>m</code> <code>x</code> , possessive (NOT SUPPORTED)
<code>x{n,}+</code>	<code>n</code> or more <code>x</code> , possessive (NOT SUPPORTED)
<code>x{n}+</code>	exactly <code>n</code> <code>x</code> , possessive (NOT SUPPORTED)

	Grouping
<code>(re)</code>	numbered capturing group (submatch)
<code>(?P<name>re)</code>	named & numbered capturing group (submatch)
<code>(?<name>re)</code>	named & numbered capturing group (submatch) (NOT SUPPORTED)
<code>(?'name're)</code>	named & numbered capturing group (submatch) (NOT SUPPORTED)
<code>(?:re)</code>	non-capturing group
<code>(?flags)</code>	set flags within current group; non-capturing
<code>(?flags:re)</code>	set flags during re; non-capturing
<code>(?#text)</code>	comment (NOT SUPPORTED)
<code>(? x y z)</code>	branch numbering reset (NOT SUPPORTED)
<code>(?>re)</code>	possessive match of <code>re</code> (NOT SUPPORTED)
<code>re@></code>	possessive match of <code>re</code> (NOT SUPPORTED) VIM
<code>%(re)</code>	non-capturing group (NOT SUPPORTED) VIM

	Flags
<code>i</code>	case-insensitive (default false)
<code>m</code>	multi-line mode: <code>^</code> and <code>\$</code> match begin/end line in addition to begin/end text (default false)
<code>s</code>	let <code>.</code> match <code>\n</code> (default false)
<code>u</code>	ungreedy: swap meaning of <code>x*</code> and <code>x*?</code> , <code>x+</code> and <code>x+?</code> , etc (default false)

Flag syntax is `xyz` (set) or `-xyz` (clear) or `xy-z` (set `xy`, clear `z`).

	Empty strings
<code>^</code>	at beginning of text or line (<code>m=true</code>)
<code>\$</code>	at end of text (like <code>\z</code> not <code>\Z</code>) or line (<code>m=true</code>)
<code>\A</code>	at beginning of text
<code>\b</code>	at ASCII word boundary (<code>\w</code> on one side and <code>\w</code> , <code>\A</code> , or <code>\z</code> on the other)
<code>\B</code>	not at ASCII word boundary
<code>\g</code>	at beginning of subtext being searched (NOT SUPPORTED) PCRE
<code>\G</code>	at end of last match (NOT SUPPORTED) PERL
<code>\Z</code>	at end of text, or before newline at end of text (NOT SUPPORTED)

<code>\z</code>	at end of text
<code>(?=re)</code>	before text matching <code>re</code> (NOT SUPPORTED)
<code>(?!re)</code>	before text not matching <code>re</code> (NOT SUPPORTED)
<code>(?<=re)</code>	after text matching <code>re</code> (NOT SUPPORTED)
<code>(?<!re)</code>	after text not matching <code>re</code> (NOT SUPPORTED)
<code>re&</code>	before text matching <code>re</code> (NOT SUPPORTED) VIM
<code>re@=</code>	before text matching <code>re</code> (NOT SUPPORTED) VIM
<code>re@!</code>	before text not matching <code>re</code> (NOT SUPPORTED) VIM
<code>re@<=</code>	after text matching <code>re</code> (NOT SUPPORTED) VIM
<code>re@<!</code>	after text not matching <code>re</code> (NOT SUPPORTED) VIM
<code>\zs</code>	sets start of match (= <code>\K</code>) (NOT SUPPORTED) VIM
<code>\ze</code>	sets end of match (NOT SUPPORTED) VIM
<code>\%^</code>	beginning of file (NOT SUPPORTED) VIM
<code>\%\$</code>	end of file (NOT SUPPORTED) VIM
<code>\%V</code>	on screen (NOT SUPPORTED) VIM
<code>\%#</code>	cursor position (NOT SUPPORTED) VIM
<code>\%'m</code>	mark <code>m</code> position (NOT SUPPORTED) VIM
<code>\%23l</code>	in line 23 (NOT SUPPORTED) VIM
<code>\%23c</code>	in column 23 (NOT SUPPORTED) VIM
<code>\%23v</code>	in virtual column 23 (NOT SUPPORTED) VIM

	Escape sequences
<code>\a</code>	bell (≡ <code>\007</code>)
<code>\f</code>	form feed (≡ <code>\014</code>)
<code>\t</code>	horizontal tab (≡ <code>\011</code>)
<code>\n</code>	newline (≡ <code>\012</code>)
<code>\r</code>	carriage return (≡ <code>\015</code>)
<code>\v</code>	vertical tab character (≡ <code>\013</code>)
<code>*</code>	literal <code>*</code> , for any punctuation character <code>*</code>
<code>\123</code>	octal character code (up to three digits)
<code>\x7F</code>	hex character code (exactly two digits)
<code>\x{10FFFF}</code>	hex character code
<code>\C</code>	match a single byte even in UTF-8 mode
<code>\Q... \E</code>	literal text ... even if ... has punctuation
<code>\1</code>	backreference (NOT SUPPORTED)
<code>\b</code>	backspace (NOT SUPPORTED) (use <code>\010</code>)
<code>\cK</code>	control char <code>^K</code> (NOT SUPPORTED) (use <code>\001</code> etc)
<code>\e</code>	escape (NOT SUPPORTED) (use <code>\033</code>)
<code>\g1</code>	backreference (NOT SUPPORTED)
<code>\g{1}</code>	backreference (NOT SUPPORTED)
<code>\g{+1}</code>	backreference (NOT SUPPORTED)

<code>\g{-1}</code>	backreference (NOT SUPPORTED)
<code>\g{name}</code>	named backreference (NOT SUPPORTED)
<code>\g<name></code>	subroutine call (NOT SUPPORTED)
<code>\g' name '</code>	subroutine call (NOT SUPPORTED)
<code>\k<name></code>	named backreference (NOT SUPPORTED)
<code>\k' name '</code>	named backreference (NOT SUPPORTED)
<code>\lX</code>	lowercase <code>x</code> (NOT SUPPORTED)
<code>\ux</code>	uppercase <code>x</code> (NOT SUPPORTED)
<code>\L...E</code>	lowercase text ... (NOT SUPPORTED)
<code>\K</code>	reset beginning of <code>\$0</code> (NOT SUPPORTED)
<code>\N{name}</code>	named Unicode character (NOT SUPPORTED)
<code>\R</code>	line break (NOT SUPPORTED)
<code>\U...E</code>	upper case text ... (NOT SUPPORTED)
<code>\X</code>	extended Unicode sequence (NOT SUPPORTED)
<code>\%d123</code>	decimal character 123 (NOT SUPPORTED) VIM
<code>\%xFF</code>	hex character FF (NOT SUPPORTED) VIM
<code>\%o123</code>	octal character 123 (NOT SUPPORTED) VIM
<code>\%u1234</code>	Unicode character 0x1234 (NOT SUPPORTED) VIM
<code>\%U12345678</code>	Unicode character 0x12345678 (NOT SUPPORTED) VIM

	Character class elements
<code>x</code>	single character
<code>A-Z</code>	character range (inclusive)
<code>\d</code>	Perl character class
<code>[:foo:]</code>	ASCII character class <code>foo</code>
<code>\p{Foo}</code>	Unicode character class <code>Foo</code>
<code>\pF</code>	Unicode character class <code>F</code> (one-letter name)

	Named character classes as character class elements
<code>[\d]</code>	digits (\equiv <code>\d</code>)
<code>[^\d]</code>	not digits (\equiv <code>\D</code>)
<code>[\D]</code>	not digits (\equiv <code>\D</code>)
<code>[^\D]</code>	not not digits (\equiv <code>\d</code>)
<code>[[:name:]]</code>	named ASCII class inside character class (\equiv <code>[:name:]</code>)
<code>[^[:name:]]</code>	named ASCII class inside negated character class (\equiv <code>[^name:]</code>)
<code>[\p{Name}]</code>	named Unicode property inside character class (\equiv <code>\p{Name}</code>)
<code>[^\p{Name}]</code>	named Unicode property inside negated character class (\equiv <code>\P{Name}</code>)

	Perl character classes (all ASCII-only)
<code>\d</code>	digits (\equiv <code>[0-9]</code>)
<code>\D</code>	not digits (\equiv <code>[^0-9]</code>)

<code>\s</code>	whitespace (≡ <code>[\t\n\f\r]</code>)
<code>\S</code>	not whitespace (≡ <code>[^\t\n\f\r]</code>)
<code>\w</code>	word characters (≡ <code>[0-9A-Za-z_]</code>)
<code>\W</code>	not word characters (≡ <code>^[^0-9A-Za-z_]</code>)
<code>\h</code>	horizontal space (NOT SUPPORTED)
<code>\H</code>	not horizontal space (NOT SUPPORTED)
<code>\v</code>	vertical space (NOT SUPPORTED)
<code>\V</code>	not vertical space (NOT SUPPORTED)

	ASCII character classes
<code>[:alnum:]</code>	alphanumeric (≡ <code>[0-9A-Za-z]</code>)
<code>[:alpha:]</code>	alphabetic (≡ <code>[A-Za-z]</code>)
<code>[:ascii:]</code>	ASCII (≡ <code>[\x00-\x7F]</code>)
<code>[:blank:]</code>	blank (≡ <code>[\t]</code>)
<code>[:cntrl:]</code>	control (≡ <code>[\x00-\x1F\x7F]</code>)
<code>[:digit:]</code>	digits (≡ <code>[0-9]</code>)
<code>[:graph:]</code>	graphical (≡ <code>[!-~]</code> ≡ <code>[A-Za-z0-9!"#\$%&'()*+,-./:;<=>?@[\\]^_`{ }~]</code>)
<code>[:lower:]</code>	lower case (≡ <code>[a-z]</code>)
<code>[:print:]</code>	printable (≡ <code>[-~]</code> ≡ <code>[:graph:]</code>)
<code>[:punct:]</code>	punctuation (≡ <code>[!-/:-@[-` {~]</code>)
<code>[:space:]</code>	whitespace (≡ <code>[\t\n\v\f\r]</code>)
<code>[:upper:]</code>	upper case (≡ <code>[A-Z]</code>)
<code>[:word:]</code>	word characters (≡ <code>[0-9A-Za-z_]</code>)
<code>[:xdigit:]</code>	hex digit (≡ <code>[0-9A-Fa-f]</code>)

	Unicode character class names--general category
C	other
Cc	control
Cf	format
Cn	unassigned code points (NOT SUPPORTED)
Co	private use
Cs	surrogate
L	letter
LC	cased letter (NOT SUPPORTED)
L&	cased letter (NOT SUPPORTED)
Ll	lowercase letter
Lm	modifier letter
Lo	other letter
Lt	titlecase letter
Lu	uppercase letter
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	math symbol
So	other symbol
Z	separator
Zl	line separator
Zp	paragraph separator
Zs	space separator

Unicode character class names--scripts
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
Common
Coptic
Cuneiform
Cypriot
Cyrillic
Deseret
Devanagari
Dogra
Duployan
Egyptian_Hieroglyphs
Elbasan
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
Khmer
Khojki
Khudawadi
Lao
Latin
Lepcha
Limbu
Linear_A
Linear_B
Lisu
Lycian
Lydian
Mahajani
Makasar
Malayalam
Mandaic
Manichaeen
Marchen
Masaram_Gondi
Medefaidrin
Meetei_Mayek
Mende_Kikakui
Meroitic_Cursive
Meroitic_Hieroglyphs
Miao
Modi
Mongolian
Mro
Multani
Myanmar
Nabataean
New_Tai_Lue
Newa
Nko
Nushu

Ogham
Ol_Chiki
Old_Hungarian
Old_Italic
Old_North_Arabian
Old_Permic
Old_Persian
Old_Sogdian
Old_South_Arabian
Old_Turkic
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
Tangut
Telugu
Thaana
Thai
Tibetan
Tifinagh
Tirhuta
Ugaritic
Vai
Warang_Citi
Yi
Zanabazar_Square

	Vim character classes
\i	identifier character (NOT SUPPORTED) VIM
\I	\i except digits (NOT SUPPORTED) VIM
\k	keyword character (NOT SUPPORTED) VIM
\K	\k except digits (NOT SUPPORTED) VIM
\f	file name character (NOT SUPPORTED) VIM
\F	\f except digits (NOT SUPPORTED) VIM
\p	printable character (NOT SUPPORTED) VIM
\P	\p except digits (NOT SUPPORTED) VIM
\s	whitespace character (≡ [\t]) (NOT SUPPORTED) VIM
\S	non-white space character (≡ [^ \t]) (NOT SUPPORTED) VIM
\d	digits (≡ [0-9]) VIM
\D	not \d VIM
\x	hex digits (≡ [0-9A-Fa-f]) (NOT SUPPORTED) VIM
\X	not \x (NOT SUPPORTED) VIM
\o	octal digits (≡ [0-7]) (NOT SUPPORTED) VIM
\O	not \o (NOT SUPPORTED) VIM
\w	word character VIM
\W	not \w VIM
\h	head of word character (NOT SUPPORTED) VIM
\H	not \h (NOT SUPPORTED) VIM
\a	alphabetic (NOT SUPPORTED) VIM
\A	not \a (NOT SUPPORTED) VIM
\l	lowercase (NOT SUPPORTED) VIM
\L	not lowercase (NOT SUPPORTED) VIM
\u	uppercase (NOT SUPPORTED) VIM
\U	not uppercase (NOT SUPPORTED) VIM

<code>_x</code>	<code>\x</code> plus newline, for any <code>x</code> (NOT SUPPORTED) VIM
<code>\c</code>	ignore case (NOT SUPPORTED) VIM
<code>\C</code>	match case (NOT SUPPORTED) VIM
<code>\m</code>	magic (NOT SUPPORTED) VIM
<code>\M</code>	nomagic (NOT SUPPORTED) VIM
<code>\v</code>	verymagic (NOT SUPPORTED) VIM
<code>\V</code>	verynomagic (NOT SUPPORTED) VIM
<code>\Z</code>	ignore differences in Unicode combining characters (NOT SUPPORTED) VIM

	Magic
<code>(?{code})</code>	arbitrary Perl code (NOT SUPPORTED) PERL
<code>(??{code})</code>	postponed arbitrary Perl code (NOT SUPPORTED) PERL
<code>(?n)</code>	recursive call to regexp capturing group <code>n</code> (NOT SUPPORTED)
<code>(?+n)</code>	recursive call to relative group <code>+n</code> (NOT SUPPORTED)
<code>(?-n)</code>	recursive call to relative group <code>-n</code> (NOT SUPPORTED)
<code>(?C)</code>	PCRE callout (NOT SUPPORTED) PCRE
<code>(?R)</code>	recursive call to entire regexp (\equiv <code>(?0)</code>) (NOT SUPPORTED)
<code>(?&name)</code>	recursive call to named group (NOT SUPPORTED)
<code>(?P=name)</code>	named backreference (NOT SUPPORTED)
<code>(?P>name)</code>	recursive call to named group (NOT SUPPORTED)
<code>(?(cond)true false)</code>	conditional branch (NOT SUPPORTED)
<code>(?(cond)true)</code>	conditional branch (NOT SUPPORTED)
<code>(*ACCEPT)</code>	make regexps more like Prolog (NOT SUPPORTED)
<code>(*COMMIT)</code>	(NOT SUPPORTED)
<code>(*F)</code>	(NOT SUPPORTED)
<code>(*FAIL)</code>	(NOT SUPPORTED)
<code>(*MARK)</code>	(NOT SUPPORTED)
<code>(*PRUNE)</code>	(NOT SUPPORTED)
<code>(*SKIP)</code>	(NOT SUPPORTED)
<code>(*THEN)</code>	(NOT SUPPORTED)
<code>(*ANY)</code>	set newline convention (NOT SUPPORTED)
<code>(*ANYCRLF)</code>	(NOT SUPPORTED)
<code>(*CR)</code>	(NOT SUPPORTED)
<code>(*CRLF)</code>	(NOT SUPPORTED)
<code>(*LF)</code>	(NOT SUPPORTED)
<code>(*BSR_ANYCRLF)</code>	set <code>\R</code> convention (NOT SUPPORTED) PCRE
<code>(*BSR_UNICODE)</code>	(NOT SUPPORTED) PCRE