↖ PEL Topic Index

Perl 5 🚧

See also: 𝔓ℓ - Perl	Perl Tools	**Perl Style Guide**. **perlcritic script** uses **Perl::Critic** to scan Perl code. The **perltidy** application reformats Perl code.
• **Perl @ Wikipedia** • **perl.org** • **perldoc browser**	**Learning Perl**	• Perl Intro - a quick introduction to Perl • Online Perl books • Beginning Perl

Perl 5 Syntax 🚧

Perl 5 Operators	Perl has a large number of operators, listed below with their precedence and associativity. Note: • C Operators missing from Perl : unary &, unary * and (type) • Quote and Quote-like operators : in Perl quotes are operators and they provide various kind of interpolating and pattern matching capabilities.

Associativity: one of:
• right
• left
• NA : not associative: cannot use more than one of these operators in sequence.
• CH: chained
	left **terms and list operators (leftward)**
	left **Arrow Operator**: `->`
	NA **Auto-increment and Uato-decrement**: `++ --`
	right **Exponentiation**: `**`
	right **Symbolic Unary Operators**: `! ~ ~. \` and unary `+` and `-`
	left **Binding operators**: `=~ !~`
	left **Multiplicative Operators**: `* / % x`
	left **Additive Operators**: `+ - .`
	left **Shift Operators**: `<< >>`
	NA **named unary operators**
	NA **Class instance Operator**: `isa`
	CH **Relational Operators**: `< > <= >= lt gt le ge`
	CH/NA **Equality Operators**: `== != eq ne <=> cmp ~~`
	left. **Bitwise And**: `& &.`
	left **Bitwise Or and Exclusive Or**: `| |. ^ ^.`
	left **C-style Logical And**: `&&`
	left **Logical Defined-Or**: `|| ^^ //`
	NA **Range Operators**: `.. …`
	right **Conditional Operator**: `?:`
	right **Assignment Operators**: `=`
	`**= += *= &= &.= <<= &&=`
	`-= /= |= |.= >>= ||=`
	`.= %= ^= ^.= //=`
	`x=`
	`goto last next redo dump`
	Comma, fat-comma Operators: `, =>`
	left **list operators (rightward)**
	NA **Logical Not**: `not`
	right **Logical And**: `and`
	left **Logical or and Exclusive or**: `or xor`
	left

File test operators	It is possible to combine the file test operator with the AND operator as in the following example:
	`if (-e $fname && -f _ && -r _ ){`
	`  print("$fname exists and is readable\n");`
	`}`

The most important operators are shown here. They check if the file…	**-r** is readable **-w** is writable **-x** is executable **-o** is owned by effective uid. **-R** is readable **-W** is writable **-X** is executable **-O** file is owned by real uid. **-e** exists. **-z** is empty. **-s** has nonzero size (returns size in bytes). **-f** is a plain file. **-d** is a directory. **-l** is a symbolic link. **-p** is a named pipe (FIFO) or Filehandle is a pipe. **-S** is a socket. **-b** is a block special file. **-c** is a character special file. **-t** handle is opened to a tty. **-u** has setuid bit set. **-g** has setgid bit set. **-k** has sticky bit set. **-T** is an ASCII text file (heuristic guess). **-B** is a “binary” file (opposite of -T).
Perl Special Variables	☝To get information about a Perl special variable from the command line use the perldoc -v command. • To get information about **$<** use: **`perldoc -v '$<'`**

• **General variables**					
default input and pattern searching space	• $ARG • $_	subroutine parameters	• @ARG • @_		
list separator	• $LIST_SEPARATOR • $”	Subscript separator for multidimensional array emulation	• $SUBSCRIPT_SEPARATOR • $SUBSEP • $;		
Name of executed program	• $PROGRAM_NAME • $0	Name used to execute the current copy of Perl	• $EXECUTABLE_NAME • $^X		
Perl process ID	• $PROCESS_ID • $PID • $$				
Process real GID	• $REAL_GROUP_ID • $GID • $(	Process effective GID	• $EFFECTIVE_GROUP_ID • $EGID • $)		
Process real UID	• $REAL_USER_ID • $UIG • $<	Process effective UID	• $EFFECTIVE_USER_ID$ • $EUID • $>		
Special variables in sort	• $a • $b				
Current environment	%ENV	Environment variable accessed as an associative array (a hash). • See: Perl: How to access shell environment variables through Perl associative arrays.			
Perl interpreter revision, version and subversion	• $OLD_PERL_VERSION • $]	Perl interpreter revision, version and subversion	• $PERL_VERSION • $^V		
Maximum file descriptor	• $SYSTEM_FD_MAX • $^F				
Fields of each line when auto-split mode is on.	@F				
Include Directories	@INC	Included filenames	%INC	Hook localization (?)	$INC
inplace-edit extension value	• $INPLACE_EDIT • $^I				

Package's class parent classes	@ISA				
Emergency memory pool	$^M				
Maximum block nesting	${^MAX_NESTED_EVAL_BEGIN_BLOCKS}				
Name of OS where this Perl was built	• $OSNAME • $^O				
Signal handlers	%SIG				
Coderefs for various perl keywords	%{^HOOK}				
Time when program began running	• $BASETIME • $^T				
• **Variables related to regular expressions**					
captured sub-patterns	$<digit>($1, $2, …)				
Capture buffer content	@{^CAPTURE}				
String matched	• $MATCH • $&		String matched (compiled regexp)	${^MATCH}	
String preceding match	• $PREMATCH • $`		String preceding match (compiled regexp)	${^PREMATCH}	
String following match	• $POSTMATCH • $'		String following match (compiled regexp)	{^POSTMATCH}	
Last capture group	• $LAST_PAREN_MATCH • $+		Most recently closed capture group	• $LAST_SUBMATCH_RESULT • $^N	
Match capture key values	• %{^CAPTURE} • %LAST_PAREN_MATCH • %+				
Match start offsets	• @LAST_MATCH_START • @-	Match ends offsets	• @LAST_MATCH_END • @+	Named captured groups	• %{^CAPTURE_ALL} • %-
Last successful pattern	${^LAST_SUCESSFUL_PATTERN}				
Result of last successful regexp assertion	• $LAST_REGEXP_CODE_RESULT • $^R				
Maximum regexp nested group	${^RE_COMPILE_RECURSION_LIMIT}				
regexp debug flag	${^RE_DEBUG_FLAG}				
regexp internal optimization/memory	${^RE_TRIE_MAXBUF}				
• **Variables related to file handles**					
Name of current file read from <>	$ARGV	Command line arguments of the script	@ARGV	Number of arguments minus one	$#ARGV
Special file handle that iterates over command-line filenames in @ARGV	ARGV	Special file handle that points to currently open output file when doing edit-in-place processing	ARGVOUT		
Output field separator for the print operator	• IO::Handle->output_field_separator(EXPR) • $OUTPUT_FIELD_SEPARATOR • $OFS • $,		Current line number for the last file handled accessed	• HANDLE->input_line_number(EXPR) • $INPUT_LINE_NUMBER • $NR • $.	
Input record separator (newline by default)	• IO::Handle->input_record_separator(EXPR) • $INPUT_RECORD_SEPARATOR • $RS • $/		Output record separator	• IO::Handle->output_record_separator(EXPR) • $OUTPUT_RECORD_SEPARATOR • $ORS • $\	
Auto-flush control	• HANDLE->autoflush(EXPR) • $OUTPUT_AUTOFLUSH • $|		Last read file handle	${^LAST_FH}	
• **Variables related to format**					
• **Error Variables**					
• **Variables related to the interpreter state**					
• **Deprecated and removed variables**					