Perl 5 🚧

See also: 𝔓𝔏 - Perl • **Perl @ Wikipedia** • **perl.org** • **perldoc browser**	**Perl Tools**	**Perl Style Guide**. **perlcritic script** uses **Perl::Critic** to scan Perl code. The **perltidy** application reformats Perl code.
	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`
left	**Comma, fat-comma Operators:**	`, =>`
NA	**list operators (rightward)**	
right	**Logical Not:**	`not`
left	**Logical And:**	`and`
left	**Logical or and Exclusive or:**	`or xor`

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	**-e**	exists.	**-b**	is a block special file.
-w	is writable	**-z**	is empty.	**-c**	is a character special file.
-x	is executable	**-s**	has nonzero size (returns size in bytes).	**-t**	handle is opened to a tty.
-o	is owned by effective uid.	**-f**	is a plain file.	**-u**	has setuid bit set.
-R	is readable	**-d**	is a directory.	**-g**	has setgid bit set.
-W	is writable	**-l**	is a symbolic link.	**-k**	has sticky bit set.
-X	is executable	**-p**	is a named pipe (FIFO) or Filehandle is a pipe.	**-T**	is an ASCII text file (heuristic guess).
-O	file is owned by real uid.	**-S**	is a socket.	**-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**					
Current value of the write() accumulator for format() lines.	• $ACCUMULATOR • $^A				
Form feed format. defaults to \f	• IO::Handle->format_formfeed(EXPR) • $FORMAT_FORMFEED • $^L	Set of characters after which a string may be broken to fill continuation fields	• IO::Handle->format_line_break_characters EXPR • $FORMAT_LINE_BREAK_CHARACTERS • $:		
Number of lines left on the page on currently selected output channel	• HANDLE->format_lines_left(EXPR) • $FORMAT_LINES_LEFT • $-	Current page length of current output channel	• HANDLE->format_lines_per_page(EXPR) • $FORMAT_LINES_PER_PAGE • $=		
Name of current top-page format of output channel	• HANDLE->format_top_name(EXPR) • $FORMAT_TOP_NAME • $^	Report format name of output channel	• HANDLE->format_name(EXPR) • $FORMAT_NAME • $~		
• **Error Variables**	The variables $@, $!, $^E, and $? contain information about different types of error conditions that may appear during execution of a Perl program. They correspond to errors detected by the Perl interpreter, C library, operating system, or an external program, respectively.				
Perl error from the last eval operator	• $EVAL_ERROR • $@	Current state of interpreter	• $EXCEPTIONS_BEING_CAUGHT • $^S		
Current value of C errno integer variable	• $OS_ERROR • $ERRNO • $!	Hash of error names to 0 or 1, set to 1 if current error is this error.	• %OS_ERROR • %ERRNO • %!		
OS detected error	• $EXTENDED_OS_ERROR • $^E				

Status returned by last pipe close, backtick command, wait, waited, or system() call.	• $CHILD_ERROR • $?	native status returned by last pipe close , backtick command, wait() or wiatpid() or system() call	${^CHILD_ERROR_NATIVE}
Current value of warning switch	• $WARNING • $^W	Current set of warning checks enabled by the use warnings pragma	${^WARNING_BITS}
• **Variables related to the interpreter state**	These variables provide information about the current interpreter state.		
Flag associated with the -c switch	• $COMPILING • $^C	The current value of the debugging flags	• $DEBUGGING • $^D
Current phase of the perl interpreter	${^GLOBAL_PHASE}		
Compile-time hints for the perl interpreter. Internal use only	$^H	Values of compiled statements	%^H
Input/Output Layers. Internal use by PerlIO only.	${^OPEN}		
Debugging support. Internal variable.	• $PERLDB • $^P		
Taint mode	${^TAINT}	Safe locale operations availability	${^SAFE_LOCALES}
Unicode Settings of Perl	${^UNICODE}		
Internal UTF-8 offset caching code state	${^UTF8CACHE}	State of UTF-8 locale detected by perl at startup.	${^UTF8LOCALE}
• **Deprecated and removed variables:**	$# $* $[${^ENCODING} ${^WIN32_SLOPPY_STAT}		