Emacs support for Perl

Description	Keystroke	Function	Note
Perl Editing See: **Perl 5 Syntax** • **Activate Perl** ➡ • **Select major mode** ➡ ⚠ • ◲ **Speedbar** ➡ • ◲ **Indentation** control ➡	Emacs provides two major modes for Perl: **perl-mode** (Emacs default, simpler mode) and **cperl-mode**. 📦 The **HaraldJoerg/cperl-mode** external package has support for new Perl language features. ☑ PEL activates Perl support with the **pel-use-perl** user-options. When turned on: • PEL provides supports all of them: **pel-perl-mode** selects the mode, using 📦 **HaraldJoerg/cperl-mode** by default because it has the best support for Perl. • After using **HaraldJoerg/cperl-mode** if you want to revert to Emacs' own **cperl-mode**, remove the cperl-mode.el* files from ~/.emacs.d/utils • PEL activates minor modes for the perl major mode as specified by the **pel-perl-activates-minor-modes** user-option. • If **pel-use-speedbar** is on, speedbar support for Perl is activated. ⊞ Perl indentation for both **perl-mode** and **cperl-mode** is controlled by the following user-options: • **pel-general-perl-indent-level**: it defaults to 4. It is applied to **perl-indent-level** and **cperl-indent-level** as well as **tab-width.** The use of hard tabs is controlled by: • **pel-general-perl-use-tabs**: it defaults to nil, forcing use of spaces. It is applied to **indent-tabs-mode**.		
PEL Perl support improvements	• By default **cperl-mode** shows trailing whitespaces as underscores. Set **pel-cperl-show-trailing-whitespace-normally** user-option to **t** to use the trailing-whitespace face instead; by default this is a red background whitespace. • PEL improves **iedit** support for cperl-mode when pel-use-iedit is on. See ◲ **Search/Replace** . • **pel-open-at-point** can find Perl files in Perl directories. It supports the Perl package :: and ' syntax. • **pel-perl-critic** executes Perl Critic over Perl code and provides navigation between compilation-mode error message and code.		
Open this PDF file. See also: ◲ **Help/Info**	`<f11> SPC P <f1>` `<f12> <f1>`	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the ☲ **- Perl** local PDF. With prefix argument (like **C-u** or **M--**) opens the remote GitHub hosted raw PDF instead. If the **pel-flip-help-pdf-arg** user-option is set it's the other way around.
◲ **Customize** PEL Perl support	`<f11> SPC P <f2>` `<f12> <f2>`	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL Perl support. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
◲ **Customize** Emacs Perl support	`<f11> SPC P <f3>` `<f12> <f3>`	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs Perl support: perl, cperl, electricity, perl-repl. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
PCRE support	See **PCRE support** in ◲ **Search/Replace** for commands to activate Perl Compatible regular expression search and operations.		
Perl Doc	With Perl info and man files installed, Emacs ◲ **Help/Info** man support provides access to the Perl documentation available on your system. ☝ Inside Emacs **you can use man instead of perldoc**: for example, **man perlintro** provides the same information as given by the **perldoc perlintro** command. The Perl Documentation web page provides the complete information on-line: • **Overview** • **Operators** **Modules** • **History** • **Tutorials** • **Functions** **Utilities** • **Internals and C Language Interface** **Miscellaneous** • **Reference Manual** • **Variables** • **Language-specific** **Platform-specific**		
Open file at point (for Perl) ◲ **File mngt**	The following command opens the file identified by the text taken at point (the cursor location). It supports extracting Perl package file names from the Perl package name syntax which uses either :: or ' as path delimiter. • It searches the Perl package directories identified by Perl's **@INC** array and the directories identified by **pel-perl-extra-project-root-directories** user-option. ☝ Note that when using the Ido completion mode, it is possible to instruct Ido to use a file name at point as the basis for the file name to open. This Ido behaviour is controlled by the **ido-use-filename-at-point** user-option. With PEL you can control it globally or locally with `<f11> f M-.`.		
Open file or web-page whose name is at point ★★ Command is also specialized for: • **Ⅿ reStructuredText** • **☲ - C**, **☲ - C++** • **☲ - Emacs Lisp** • **☲ - Erlang** • **☲ - UNIX Shell** **Generic Delimiting characters** **Select target window** ☞ **N>20 : open the directory** ☞ See function docstring for more info.	• `M-<f6>` • `<f11> f .` • `6y`	(pel-open-at-point &optional N)	Open the file, library or the URL, named at point, with potential line & column #s. ⊞ Supports glob characters, partial directory path. When multiple files are found it prompts using the method selected by **pel-prompt-read-method** user-option. 📦☑ The **6y** key-chord is available if **pel-use-key-chord** is non-nil. See ◲ **Key-Chords**.

	This command works generically (for a normal file or directory name) and is specialized for Perl buffers: it finds Perl files in directories trees identified by Perl's **@INC** array. The command extracts the file or directory name, and possibly line and column numbers, from text at point and tries to open the file or directory.
	• The generic mode extraction works by identifying the beginning & end of the file/directory/library/URL name string by delimiter characters, one of: tab, newline and:
	"；`'|()[]{}<> '"" 「」 〔〕 〈〉 《》 【】 〖〗 «»‹›‹〉() ·。
	⊞ When finding several file names, the command lists them and prompts using the method selected by **pel-prompt-read-method** user-option.
	• The default is a very primitive function implemented by PEL. You can select a more powerful **ivy** prompting instead.
	  • With **ivy** selected, PEL will automatically set ☑ **pel-use-ivy** to **t** 📦 and **Ivy mode** will be installed automatically when you restart Emacs.
	• Note that the command shows all files found by the specified search method, it does not only use the first one found.
	  • ☝ Use this to detect potential duplication in package and module files.
	The command opens the file in the window selected by the following logic controlled by presence or absence of typed numerical prefix arguments:
	• **Select target window:**
	  • Without argument:
	    • If file or directory is already opened in a window, move point to that window and to the line column coordinates if specified following the file name at point.
	    • If no window holds that file, select the target window according to the number of editable windows in frame: if 1, split that window and use the new window, if 2: use the other window, if 3 or more, use the current window.
	  • With prefix numeric argument N:
	    • N < 0 : create a new window and use that.
	    • (abs N) > 20: then open the **directory** instead of the file. Interpret the window position from the N value adjusted: N-20 (or N+20 if N is negative)
	    • N = 0: use the 'other' (the next) window.
	    • N = 1, 3, 7 or above (excluding 8, 9 and 10): select the target window based on the number of editable windows in frame:
	      • if 1 window: split that window and use the new window,
	      • if 2 windows: use the other window,
	      • if 3 or more windows: use the current window.
	    • N is: 8: up, 2: down, 4:left, 5:current, 6:right.
	    • N is 9: force **opening the file in the OS associated application** (with N=29 or N=-29, open the file's directory with the OS associated application (eg. macOS Finder, Windows Explorer). If this is a URL, open it in the OS default web browser.
	• Selecting Minibuffer, inexistent or dedicated window is not allowed.		

Comments	Perl comments start with #		
Toggle display of comments in buffer or active region	`<f11> ; ;`	(hide/show-comments-toggle &optional START END)	Toggle hiding/showing of comments in the active region or whole buffer. • If the region is active then toggle in the region. Otherwise, in the whole buffer.
	📦 This requires the **hide-comnt.el** package (see ◲ **Comments**). ☑ PEL activates it when the **pel-use-hide-comnt** user option is **t**.		
Perl base	These commands are provided by Emacs basic Perl support.		
Move to beginning of function	• `C-M-a` • `<f12> <up>`	(perl-beginning-of-function &optional ARG)	Move backward to next beginning-of-function, or as far as possible. With argument, repeat that many times; negative args move forward.
To end of function	• `C-M-e` • `<f12> <down>`	(perl-end-of-function &optional ARG)	Move forward to next end-of-function. • The end of a function is found by moving forward from the beginning of one. • With argument, repeat that many times; negative args move backward.
Mark function	`C-M-h`	(perl-mark-function)	Put mark at end of Perl function, point at beginning
Indent	`C-M-q`	• (perl-indent-exp) • (cperl-indent-exp)	Indent each line of the Perl grouping following point.
cperl-mode	PEL supports 2 **cperl-mode** implementations: Emacs' **cperl-mode** and 📦 **HaraldJoerg/cperl-mode** (more complete) activated by the **pel-use-perl** user-option.		
Show command info	`C-c C-h F`	(cperl-info-on-command COMMAND)	Show documentation for Perl command COMMAND in other window. • If perl-info buffer is shown in some frame, uses this frame. • Customized by setting variables 'cperl-shrink-wrap-info-frame', 'cperl-max-help-size'.
perldoc at point	• `C-c C-h P` • `<f12> ?`	(cperl-perldoc-at-point)	Run a 'perldoc' on the word around point to show information about that Perl word.
	`C-c C-h p`	(cperl-perldoc WORD &optional SECTION)	Prompt for for (default to word at point). Show information about the selected word with 'perldoc'.
Show Perl doc	`C-c C-h f`	(cperl-info-on-current-command)	Show documentation for Perl command at point in other window.

Description	Keystroke	Function	Note
Help on symbol at point	`C-c C-h v`	(cperl-get-help)	Get one-line docs on the symbol at the point. • The data for these docs is a little bit obsolete and may be in fact longer than a line.
toggle Perl auto-help	`C-c C-h a`	(cperl-toggle-autohelp)	Toggle the state of Auto-Help on Perl constructs (put in the message area). • Delay of auto-help controlled by '**cperl-lazy-help-time**'.
Toggle auto-newline	`C-c C-a`	(cperl-toggle-auto-newline)	Toggle the state of '**cperl-auto-newline**'.
Toggle electric mode	`C-c C-e`	(cperl-toggle-electric)	Toggle the state of parentheses doubling in CPerl mode. When typing an opening parens character the closing one is automatically entered.
Toggle auto-fill mode	`C-c C-f` • `<f11> t f a` • `<f11> RET`	(**auto-fill-mode** &optional ARG)	Toggle automatic line breaking (Auto Fill mode). • With a prefix argument, enable Auto Fill mode if the prefix argument is positive, disable it otherwise. • When Auto Fill mode is enabled, inserting a space at a column beyond 'current-fill-column' automatically breaks the line at a previous space.
Toggle keyword expansion	`C-c C-k`	(cperl-toggle-abbrev)	Toggle the state of automatic keyword expansion in CPerl mode.
Toggle space fix	`C-c C-w`	(cperl-toggle-construct-fix)	Toggle whether '**indent-region**'/'**indent-sexp**' fix whitespace too.

Check code with perlcritic	`<f12> c`	(**pel-perl-critic** &optional VERBOSE)	Validate the Perl file visited in current buffer with **perlcritic**. Report error if it's not installed.
			• With optional VERBOSE prefix argument, print extra information:
			  • Full name of the Policy module that created the violation
			  • Full diagnostic discussion of each Perl Best Practice (PBP) violation.
			Show errors in compilation-mode buffer in a format that allows navigation.

Spell-check here-docs	`C-c C-d`	(cperl-here-doc-spell)	Spell-check HERE-documents in the Perl buffer. • If a region is highlighted, restricts to the region.
Find/fix missing whitespace code	`C-c C-b`	(cperl-find-bad-style)	Find places in the buffer where insertion of a whitespace may help. • Prompts user for insertion of spaces. Currently it is tuned to C and Perl syntax.
Insert Perl new line	`C-c C-j`	(cperl-linefeed)	Go to end of line, open a new line and indent appropriately. If in POD, insert appropriate lines.

It's a convenience replacement for typing return. It puts point in the next line with proper indentation, or if you type it inside the inline block of control construct, like

```
foreach (@lines) {print; print}
```

and you are on a boundary of a statement inside braces, it will transform the construct into a multiline and will place you into an appropriately indented blank line.
• Use `C-j` for usual 'newline-and-indent' behavior. See '**cperl-electric-linefeed**' documentation.

Description	Keystroke	Function	Note
Newline and indent	`C-j`	(newline-and-indent)	Insert a newline, then indent according to major mode. • Indentation is done using the value of 'indent-line-function'. In programming language modes, this is the same as TAB.
Indent Perl code	`TAB`	(cperl-indent-command &optional WHOLE-EXP)	Indent current line as Perl code, or in some cases insert a tab character. • If '**cperl-tab-always-indent**' is non-nil (the default), always indent current line. Otherwise, indent the current line only if point is at the left margin or in the line's indentation; otherwise insert a tab. • A numeric argument, regardless of its value, means indent rigidly all the lines of the expression starting after point so that this line becomes properly indented. The relative indentation among the lines of the expression are preserved.
Indent continued expression	`C-M-q`	(cperl-indent-exp)	Simple variant of indentation of continued-sexp. • Will not indent comment if it starts at '**comment-indent**' or looks like continuation of the comment on the previous line. • If '**cperl-indent-region-fix-constructs**', will improve spacing on conditional/loop constructs.
Indent region	`C-M-\`	(cperl-indent-region START END)	Simple variant of indentation of region in CPerl mode. • Should be slow. Will not indent comment if it starts at '**comment-indent**' or looks like continuation of the comment on the previous line. • Indents all the lines whose first character is between START and END inclusive. • If '**cperl-indent-region-fix-constructs**', will improve spacing on conditional/loop constructs.
Insert matching parens (toggle with C-c C-e)	• `(` • `<` • `[`	(cperl-electric-paren ARG)	Insert an opening parenthesis or a matching pair of parentheses. • Controlled by '**cperl-electric-parens**' and '**cperl-hairy**' user-options.
	• `)` • `]`	(cperl-electric-rparen ARG)	Insert a matching pair of parentheses if marking is active. • If not, or if we are not at the end of marking range, would self-insert. • Controlled by '**cperl-electric-parens**' and '**cperl-hairy**' user-options.
Insert : and indent	`:`	(cperl-electric-terminator ARG)	Insert character and correct line's indentation.
Insert ; and indent	`;`	(cperl-electric-semi ARG)	Insert character and correct line's indentation.
Insert { and indent	`{`	(cperl-electric-lbrace ARG &optional END)	Insert character, correct line's indentation, correct quoting by space.
Insert } and indent	`}`	(cperl-electric-brace ARG &optional ONLY-BEFORE)	Insert character and correct line's indentation. • If ONLY-BEFORE and '**cperl-auto-newline**', will insert newline before the place (even in empty line), but not after. If after ")" and the inserted char is "{", insert extra newline before only if '**cperl-extra-newline-before-brace**'.
Narrows to here-doc	`C-c C-n`	(cperl-narrow-to-here-doc &optional POS)	Narrows editing region to the HERE-DOC at POS. POS defaults to the point.
Spell-check POD documentation	`C-c C-p`	(cperl-pod-spell &optional DO-HERES)	Spell-check POD documentation. • If invoked with prefix argument, will do HERE-DOCs instead. • If a region is highlighted, restricts to the region.
Refactor: if (A) {B}' ⇒ 'B if A;'	`C-c C-t`	(cperl-invert-if-unless)	Change 'if (A) {B}' into 'B if A;' etc (or visa versa) if possible. • If the cursor is not on the leading keyword of the BLOCK flavor of construct, will assume it is the STATEMENT flavor, so will try to find the appropriate statement modifier.
Lining up Perl Code : ∑ Align			It's often best to **line up Perl code vertically**, arranging elements as tables, to help reader understand the code intent visually. • This technique is also promoted in <u>Damian Conway's Perl Best Practice book, Vertical Alignment section of chapter 2</u>. See **pel-align-words-vertically** in ∑ **Align**.
Lineup	• `C-M-|` • `<f12> |`	(cperl-lineup BEG END &optional STEP MINSHIFT)	Lineup construction in a region. • Beginning of region should be at the start of a construction. • All first occurrences of this construction in the lines that are partially contained in the region are lined up at the same column. • MINSHIFT is the minimal amount of space to insert before the construction. • STEP is the tabwidth to position constructions. • If STEP is nil, 'cperl-lineup-step' will be used (or 'cperl-indent-level', if 'cperl-lineup-step' is nil). • Will not move the position at the start to the left.
Move to next interpolated	`C-c C-v`	(cperl-next-interpolated-REx &optional SKIP BEG LIMIT)	Move point to next REx which has interpolated parts. • SKIP is a list of possible types to skip, BEG and LIMIT are the starting point and the limit of search (default to point and end of buffer). • SKIP may be a number, then it behaves as list of numbers up to SKIP; this semantic may be used as a numeric argument. • Types are 0 for / $rex /o (interpolated once), 1 for /$rex/ (if $rex is a result of qr//, this is not a performance hit), t for the rest
	`C-c C-x`	(cperl-next-interpolated-REx-0)	Move point to next REx which has interpolated parts without //o.
	`C-c C-y`	(cperl-next-interpolated-REx-1)	Move point to next REx which has interpolated parts without //o. • Skips RExes consisting of one interpolated variable. • Note that skipped RExen are not performance hits.
Deleted and possibly untabify	`DEL`	(cperl-electric-backspace ARG)	Backspace, or remove whitespace around the point inserted by an electric key. • Will untabify if '**cperl-electric-backspace-untabify**' is non-nil.

Description	Keystroke	Function	Note

Start Perl REPL
 	
See: ⅀ **start Shells/ REPLs**
	`<f11> z r P`
	
	`<f12> z`
		(perl-repl)	Run a Perl REPL in a *Perl-REPL* buffer.
			📦 Requires the **perl-repl** external package 🔗 activated by **perl-use-perl-repl** user-option.
			• The **perl-repl-file-path** user option specifies the name of the Perl REPL program, which may optionally specify the explicit file path.
			   • PEL provides the **perl-repl** shell script which uses the Perl command line.

Emacs & Perl — References

Document	Notes
Perl @ Wikipedia	
perl.org	
Learn Perl @ perl.org • **Perl Style Guide**	• Perl Tutorial - a gentle introduction to Perl with several examples over a browsable set of pages. • Perl Intro - a quick introduction to Perl • Online Perl books • Beginning Perl
Perl Reference Manuals	• Perl Keywords, • Perl Operators. Also see the Perl ABC Operator page: organizes the information in sections (but has some markup typos). • Perl Functions

Is Perl still relevant?
    Most probably.
	• What makes Perl relevant in 2022? @ Stackoverflow blog.
	• Perl is dying quick. Could be extinct by 2023. The HFT Guy, 2019. (which includes several invalid points).
	  • My point is that Perl was popular, there's a lot of Perl code still being used and it's worth knowing and being able to write and edit Perl code (which was certainly not the first programming language as state by the article). But anyway, the post represents a point of view (and has many people commenting on it).
	• Perl is making a comeback: 5 reasons why it's worth learning. Posted January 6, 2023 by Lucas Rees.
	• Why Perl Didn't Win on outspeaking.com, updated December 21, 2020

Perl and Secure Coding Practice and Tools	• Security Issues in Perl Scripts - by Jordan Dimov - Discussion of misused and overlooked features of Perl from the security point of view.
	• SEI CERT Perl Coding Standard @ Carnegie Mellon University
	  • The CERT Perl Secure Coding Standard, by David Svoboda , June 25, 2012
	• perlsec describes Perl security.
	Tools:
	• The **perlcritic script** uses Perl::Critic to scan Perl code and provides some listing advices.
	  • On some Linux distros you can install it with: `sudo dnf install perl-Perl-Critic`
	• The **perltidy** application reformats Perl code to the promoted format.
	• The **zarn** static analyzer, hosted on Github, is another static analyzer for Perl. (Quite immature as of Oct. 2024).

Perl File name extensions	• **.pl** Perl non executable libraries, also used for Perl scripts (but a file with no extension and a shebang line is also fine, and preferable, allowing the invocation of the script without having to type the '.pl'). • **.pm** Perl modules. • .plx Used by Active State implementation. Identifies executable Perl scripts. Also used elsewhere to distinguish from Prolog (which also uses the .pl file extension). • .pls • .xs • .t • .pod **Plain Old Documentation** files, a lightweight markup language used to document Perl code. • See also **perlpod**. • .ph • .cgi • .psgi
Perl programs	• Perl command line options
• **perl**	Perl language interpreter
• **perlbug** / **perlthanks**	Describes how to submit bug report on Perl
• **perldoc**	Print Perl Documentation, looking it up in the .pod format embedded in perl installation. Support following options: • -f : built-in functions • -q : FAQ keyword search • -v : variable • -a perl API
• **perlivp**	• Perl Installation Verification Procedure : checks Perl installation. • Part of **perl-level** package.
perlsec - Perl security	
Getting Perl Modules with CPAN: • **CPAN @ Wikipedia** • **The Zen of Comprehensive Archive Networks** • **CPAN** • **Search CPAN — meta::cpan** • **PAUSE - Perl Authors Upload Server**	**Command line tools** interacting with CPAN: • **cpan** : install on some Linux with: `sudo dnf install perl-CPAN` • **cpanplus** • cpanminus : **cpanm** : install on some Linux with: `sudo dnf install perl-App-cpanminus`