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: it best supports Perl and perltidy. PEL instals the upstream copy of perl-mode from the **HaraldJoerg/cperl-mode** repo (which is what Emacs 30 uses) and the master copy of perl-tidy-ediff. • After using **HaraldJoerg/cperl-mode** if you want to revert to Emacs' own **cperl-mode**, remove the cperl-mode.el* and perl-tidy-ediff.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. • Integrated buffer-only perltidy commands from **HaraldJoerg/cperl-mode** perl-tidy-ediff with improved command interface (2 commands do everything instead of 3). • **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 **𝔓𝔩 - 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.

Open Perl 5 PDF	`<f11> p perl5`	(**pel-help-pdf-select** &optional OPEN-GITHUB-PAGE-P)	Open the Perl 5 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.

Show perl-mode status	`<f12> <f4> ?`	(**pel-perl-show-status** &optional APPEND)	Show current buffer 'cperl-mode' status in specialized *pel-perl-info* buffer. • Clear previous buffer content (and move point to top of buffer) unless optional APPEND argument is non-nil,in which case it appends to the previous report and move point to the end of buffer..
PCRE support	See **PCRE support** in **Search/Replace** for commands to activate Perl Compatible regular expression search and operations.		
Perl Doc The Perl Documentation web page provides the complete information on-line:	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. • Overview • Tutorials • Reference Manual • Operators • Functions • Variables Modules Utilities • History • Internals and C Language Interface • Language-specific Miscellaneous Platform-specific		
Show perldoc info • **-f** switch to get info on a specific built-in function, as in: `-f split` • **-l** (ell) switch to print the file location, as in: `-l Pod::Simple` • **-m** switch to see the raw source, as in: `-m Pod::Simple` • **-q** switch to get FAQ info on a topic: as in `-q random`	• `C-c C-h F` • `C-c C-h f` • `C-c C-h p`	(**cperl-perldoc** WORD)	Prompt and show documentation about Perl item: run perldoc on WORD. • You can specify perldoc options if needed (see in title column). • Specify any module, as in Pod::Simple, or documents, as in: **perldoc**, **perltoc**, **perlsyn**, **perlfunc**
	`C-c C-h F`	(**cperl-info-on-command** COMMAND)	Show documentation for Perl command COMMAND in other window. Obsolete since Emacs 30.1 • If perl-info buffer shown in some frame, uses this frame. Perl info pages are no longer distributed. • Customized by setting variables '**cperl-shrink-wrap-info-frame**', '**cperl-max-help-size**'.
	`C-c C-h f`	(**cperl-info-on-current-command**)	Show documentation for Perl command at point in other window. Obsolete since Emacs 30.1
	`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'. Obsolete since Emacs 30.1
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.
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.
Open file at point (for Perl) **File mngt** **Tramp** aware	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-.**		
Show searched directory trees	`<f12> <f4> .`	(**pel-perl-show-source-directories**)	Display the list of source directories searched by pel-open-at-point. The list is controlled by Perl's @INC array and the extra list provided by **pel-perl-source-directories** user-option.
Open file or web-page whose name is at point ★★ Command is also specialized for: • **reStructuredText** • **𝔓𝔩 - C**, **𝔓𝔩 - C++** • **𝔓𝔩 - Emacs Lisp** • **𝔓𝔩 - Erlang** • **𝔓𝔩 - UNIX Shell**	• `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**.
Generic Delimiting characters **Select target window** ☛ **N>20 : open the directory** ☛ **See function docstring for more info.**	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, 7or 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 #. See **Comments** for the generic commands that manage and control comments		
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**.		

Description	Keystroke	Function	Note
Navigation Commands	Some navigation commands have special behaviour for Perl; they are shown below. They are available in **perl-mode** and **cperl-mode**. • There's also several generic navigation commands that work in Perl buffer too. See ⅀ **Navigation** for those.		
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.
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.
Marking Commands	The following marking commands are specialized for Perl. See ⅀ **Marking** for the generic ones that can also be used in Perl buffers.		
Mark function	`C-M-h`	(**perl-mark-function**)	Put mark at end of Perl function, point at beginning
Indentation Control	The following indentation control commands are specialized for Perl. See ⅀ **Indentation** for the generic ones that can also be used in Perl. • The indentation behaviour of the `<tab>` key is controlled by the **perl-mode** and **cperl-mode** customization user-options. Use `<f12> <f3>` to open customization.		
Show currently used indentation style	`<f12> <f4> s`	(**pel-perl-show-style**)	Show the name of the currently used indentation style.
Select new/restore old indentation style	`<f12> <f4> <TAB>`	(**pel-perl-set-style**)	Set Perl indentation style to named style. Prompt for indentation style name and apply it. ⚠️ This change does **not** persist. Manually access the customized buffer and save it.
Indent • **perl-mode** ➨	`<tab>`	(**perl-indent-command** &optional ARG)	Indent Perl code in the active region or current line. In Transient Mark mode, when the region is active, reindent the region. • Otherwise with a prefix argument, reindent the current line unconditionally. • Otherwise if '**perl-tab-always-indent**' is nil and point is not in the indentation area at the beginning of the line, insert a tab. • Otherwise, indent the current line. If point was within the indentation area, it is moved to the end of the indentation area. If the line was already indented properly and point was not within the indentation area, and if 'perl-tab-to-comment' is non-nil (the default), then do the first possible action from the following list: 1) delete an empty comment 2) move forward to start of comment, indenting if necessary 3) move forward to end of line 4) create an empty comment 5) move backward to start of comment, indenting if necessary.
• **cperl-mode** ➨		(**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`	(**perl-indent-exp**)	Indent each line of the Perl grouping following point.
• **cperl-mode** ➨		(**cperl-indent-exp**)	Simple variant of indentation of continued-sexp. • Won't 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)	Indents all the lines whose first character is between START and END inclusive. • Won't 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.
Newline and indent • **cperl-mode** ➨	`C-j`	(**newline-and-indent**)	Insert a newline at point, then indent the newly created line. Use it to split a line. • Indentation is done using the value of '**indent-line-function**' which is set to **cperl-indent-line**.
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.		
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**'.
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.
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. • The following cperl commands toggle some of its electric behaviour.		
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**'. ⚠️By default it is nil, which (like null) prevents activation of the auto-help. To activate it, set **cperl-lazy-help-time** to an integer value.
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.
here-doc support	The following **cperl-mode** commands operate on **Perl here documents**.		
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.

Description	Keystroke	Function	Note
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.
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.
Verify & refactor Perl code	The following commands provide Perl verification and refactoring facilities.		
• **perltidy** ⚠️ currently requires a lperltidy to be present on local host event when processing aremote file.	The following 2 commands run **perltidy** between areas of the Perl code in the current buffer and the tidied version of that code. All work is done inside Emacs buffers, no file is affected. After generating the tidied code the functions open an ediff session to compare your code and the tidied code, allowing you to decide what to use. Requires 📦 **HaraldJoerg/cperl-mode** activated by the **pel-use-perl** user-option. 🎛 The perl-tidy-ediff commands execute the perltidy identified by **perl-tidy-command** user-option with options identified by the **perl-tidy-ediff-args** user-option. • See **Perl::Tidy @meta::cpan** for the perltidy list of options, also Online PerlTidy , which provides help on the various options categorized by feature set.		
Run perltidy on **current buffer or region**. Ediff changes.	`<f12> T`	(**pel-perl-tidy-ediff**)	Run perltidy on current buffer, than start an ediff session, comparing the original source with perltidy output. Error messages are saved in the *perl-tidy-errors* buffer. • If an area is marked, run perltidy on the marked area only.
Run perltidy on **current subroutine**. Ediff changes. See ⅀ **Narrowing**	`<f12> t`	(**perl-tidy-ediff-sub**)	Run the perltidy program on the subroutine that stats before point and start en ediff session to compare the original code with the tidied version. • Error messages are saved in the *perl-tidy-errors* buffer. ☝ The buffer is automatically narrowed to the current function. • When quitting the ediff session it remains narrowed. Use `C-x n w` to widen the buffer back.
• **perlcritic**	Perform static code analysis with **perlcritic** (which must be installed separately).		
Check code with perlcritic ⅀ **Tramp**-aware : run remote host's **perlcritic** on remote file.	`<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.
• **Other refactoring**			
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.
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 Damian Conway's Perl Best Practice book, Vertical Alignment section of chapter 2. 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.
Testing Perl code	The following command provides a way to test Perl code locally. See **Perl 5 Syntax** for web-sites that provide a Perl interpreter.		

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.
• **perltidy**	• PerlTidy @ Wikipedia , PerlTidy Home Pages: @SourceForge @GitHub. Perl::Tidy GitHub repo, **Perl::Tidy @meta::cpan** • **Online PerlTidy** , which provides help on the various options categorized by feature set.

Document	Notes
perlsec - Perl security	
Perl Community	Perl has along history and is quite vast. Here’s a collection of links to various web sites that can provide information about it. It is far from complete and collected without much background on what happened in lots of cases and no opinion yet taken on most of this. • strictures vs Schmorp common::sense, Marc A.Lehman common::sense package.