↖ Topic Index

Comments

Action	Key binding	Command	Note
Comment Commands ○ Configure Commenting • comment/uncomment code • align on new line ○ Kill comments • Hide/show comments • Insert commented line	Emacs provides generic support to handle source code **comments**. Some external packages also handle comments generically. • Since the concept of comment depends on the programming language, Emacs major mode syntax logic is used to identify and render comments. Some major modes have extra commands for handling comments. These other commands are described in the pages for the markup and programming languages. • See also the **Hide/Show** page; it describes a Hydra that provides quick access to selectively hide parts of text and code. • The behaviour of the comment commands depend on the value of several comment user-options. List them all with **`<f11> ; ?`**		
Last updated on:	2025-10-30	See also: **Hide/Show**	
Open this PDF file. See also: **Help/Info**	**`<f11> ; <f1>`**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the **Comments** local PDF. If the prefix argument (like **`C-u`** or **`M--`**) is used, then it 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 Comment Support See also: **Customize**	**`<f11> ; <f2>`**	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL Bookmark support: open PEL programming language group (which holds generic comment user options). • If OTHER-WINDOW is non-nil (use **`C-u`**), display in other window.
Customize Emacs & external package comment support See also: **Customize**	**`<f11> ; <f3>`**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs and external packages related to buffer. This includes the following groups: comment and hideshow. With prefix argument (like **`C-u`**) opens the buffer inside another window.
	☝Group belonging to files that have not yet been loaded are normally not accessible in Emacs via the customize-group command. But PEL attempts to locate the file that defines a non-loaded customization group and will prompt for loading the file if it finds it.		
Configure Comments	Use the following commands to see how comments are controlled in the current buffer and to change it.		
Show values of Emacs comment-control variables **Provide access to customization of each user-option**	**`<f11> ; ?`**	(**pel-comment-show-variables** &optional ONLY-USER-OPTIONS)	Display the names and values of the comment related variables. • Open a *comment-vars* buffer and insert the information at the end of the buffer. If the ONLY-USER-OPTIONS argument is non-nil, include only user option variables.
	🛠This helps understand the impact of variables and user options controlling the behaviour of commands that controls the comments generation.		
Set comment column to current column	**`C-x ;`**	(**comment-set-column** ARG)	Set the *comment-column* variable (shown in the ruler as '#'). • With no argument: set to the current column. • With any positive argument set comment-column to the same column as the previous comment and align/create a comment on this line at that column. • With - as arg, kill any comment on current line.
Set comment column to previous comment and adjust/insert a comment	**`C-u C-x ;`**	(**comment-set-column** 1)	
Set comment start string	**`<f11> ; b`**	(**pel-comment-start**)	Show and set comment start string for current mode. Controlled by variable: ***comment-start***
Set comment end string	**`<f11> ; e`**	(**pel-comment-end**)	Show and set comment end string for current mode. Controlled by variable: ***comment-end*** • By default it's "". It must be "" if the comment ends at the end of the line.
Set comment continue (middle) string	**`<f11> ; m`**	(**pel-comment-middle**)	Show/set comment continue/middle string for current mode. Controlled by variable: ***comment-continue.*** By default it is *nil*. Can be set to any string. This is used for multi-line comments
Change comment style for buffer	**`<f11> ; s`**	(**pel-comment-style** &optional CUSTOMIZE)	Select a comment style for the buffer: prompts with the list of available styles, showing the currently used one. Apply the choice to the current buffer. • With **`C-u`** prefix, open the customize buffer to control selection of the default comment style for all buffers (the **comment-style** user option).
As of Emacs 30, Emacs supports 8 different comment styles, listed here: ➡➡	Emacs supports several comment styles, as specified by the **comment-styles** user-option (which can be modified). Some of these styles only take effect when a region of several lines is comments. By changing the style you can create the boxed comments, for instance and also uncomment the box comment with comment-swim (bound to **`M-;`**) and then change for another comment style in the same buffer. • The style selected by the command only affects the current buffer. It is not persistent. The persistent setting is the **comment-style** user option. • 0 = plain: Start in column 0 (do not indent), as in Emacs-20 • 1 = indent-or-triple: Start in column 0, but only for single-char starters • 2 = indent: Full comment per line, ends not aligned • 3 = aligned: Full comment per line, ends aligned • 4 = box: Full comment per line, ends aligned, + top and bottom • 5 = extra-line: One comment for all lines, end on a line by itself • 6 = multi-line: One comment for all lines, end on last commented line • 7 = box-multi: One comment for all lines, + top and bottom		
Set Fill Column • See: **Filling/Justification**	• **`C-x f`** • **`<f11> t f c`**	(**set-fill-column** ARG)	When no prefix value: prompts for column. • If with **`C-u`** prefix: use current column. • If with **`C-u`** prefix followed by a numeric value: use that value as the new column.
Toggle auto fill inside comments only • See: **Filling/Justification**	**`<f11> t f ;`**	(**pel-auto-fill-only-comments**)	Toggle the ***comment-auto-fill-only-comments*** variable to control whether filling is done everywhere or only inside the comments.
See all comment control variables	• **`C-h v comment- <tab><tab>`** • **`<f1> v comment- <tab><tab>`**		Lists all variables that have a name that starts with the word "comment-". ☝See the Help table for more information on getting help from within Emacs.
(un-)comment code	Use the following commands to comment or un-comment code using the comment style corresponding to the buffer's major-mode.		
Comment/uncomment current line	• **`C-x C-;`** • **`<f11> ; l`** • **`<f6> ;`**	(**comment-line** N)	Comment/uncomment current line (the comment is at the beginning of the line). • Always start the comment at the beginning of the line. • With numerical argument N, comment N lines, starting at current one.
Insert, realign, comment/ uncomment region **With PEL**: Comment the current line with **`M-0 M-;`**	**`M-;`**	(**comment-dwim** ARG)	Insert or realign comment on current line (or region if a region is active). • On a single line, the comment is placed *after* the code. • **`C-u M-;`** executes comment-kill
		(**pel-comment-dwim** ARG)	Same as **comment-dwim** but comments the current line with a numeric ARG or 0.
Comment/Uncomment each line in the region. **Only available for some programming languages, including:** • **C, C++, D, Erlang**	**`C-c C-c`**	(**comment-region** BEG END &optional ARG)	Comment or uncomment each line in the region. • With just **`C-u`** prefix arg, uncomment each line in region BEG .. END. • Numeric prefix ARG means use ARG comment characters. • If ARG is negative, delete that many comment characters instead.
	• The strings used as comment starts are built from '**comment-start**' and '**comment-padding**'; the strings used as comment ends are built from '**comment-end**' and 'comment-padding'. • By default, the '**comment-start**' markers are inserted at the current indentation of the region, and comments are terminated on each line (even for syntaxes in which newline does not end the comment and blank lines do not get comments). This can be changed with '**comment-style**'. ☝If you try this when no region is marked and the **`/* */`** style comments is active, the comment ends on the next space, which is probably not what you want. The command comment-dwim works better.		
Uncomment a region	**`<f11> ; u`**	(**uncomment-region** BEG END &optional ARG)	Uncomment each line in the BEG .. END region. The numeric prefix ARG can specify a number of chars to remove from the comment delimiter.
	☝Sometimes the comment-dwim, shown above, does not uncomment a region, it adds another layer of comment. In that case try this command instead.		
Comment the region as a box	**`<f11> ; B`**	(**comment-box** BEG END &optional ARG)	Comment the marked region in a comment box.
New line & aligned comment	• **`M-j`** • **`C-M-j`**	Mapped to: • C, C++ modes: (**c-indent-new-comment-line** &optional SOFT ALLOW-AUTO-FILL) • Buffer Menu, text mode, python mode: (**comment-dwim** ARG)	When **`M-j`** is typed, on a line that contains a comment, a new line is entered and a comment is placed in the same column, with point placed after the same number of spaces.

Action	Key binding	Command	Note
Kill comments			
Kill comment on current line	`<f11> ; k`	(**comment-kill** ARG)	Kill the complete comment text on the line (whether it is at the beginning of the line or after some code) and remember in kill ring. • With numeric argument, kill comment on as many lines staring with the current one. • For one line this can also be executed with `C-u M-;`
Delete all comments in buffer or marked region See also: • **Σ Cut & Paste**	`<f11> ; ⌫`	(**pel-delete-all-comments**)	Delete all comments in current (possibly narrowed) buffer or marked region. ☝ To delete all comments inside a region mark the region first. You can also narrow a region and then use this command to remove all comments from that narrowed region, without affecting anything else. See the **Σ Narrowing** table for information on narrowing.
Kill all comments in buffer or marked region (& retain them in kill ring) See also: • **Σ Cut & Paste**	`<f11> ; -` `<f11> - ;`	(**pel-kill-all-comments**)	Kill all comments in current (possibly narrowed) buffer or marked region and retain them in kill ring. ☝ To kill all comments inside a region mark the region first. You can also narrow a region and then use this command to remove all comments from that narrowed region, without affecting anything else. See the **Σ Narrowing** table for information on narrowing.
Hide/Show Comments See also: **Σ Hide/Show**	The hide-cmnt file, written by Drew Adams, provides the following commands to quickly hide/show the comments in a buffer. • PEL provides binding for these 2 commands, but the file must be installed manually copied in a directory the is in your Emacs load path. • You can download the file from the hide-comnt.el EmacsWiki link or from the hide-count EmacsMirror. They should contain the exact same code. ☝ Very useful to see a list of methods without all comments when you also use the Hide/Show Mode commands (see **Σ Hide/Show** Code table). 📦 Both of these commands require the **hide-comnt.el** package (see above). PEL activates it when the **pel-use-hide-comnt** user option is **t**.		
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.

Show (or hide) comments in buffer or marked region	`<f11> ; :`	(**hide/show-comments** &optional HIDE/SHOW START END)	Hide or show comments in buffer or active region.
			• Hide if no argument. To show, use any prefix argument (any of the `C-u`, `M- -`, `M-0` to `M-9` will do).
			• If a region is active the command applies to the active region, otherwise it applies to the entire or narrowed buffer.
			• Uses 'save-excursion', restoring point.
			• Option 'show-invisible-comments-shows-all':
			  • If non-nil then using this command to show invisible text shows *ALL* such text, regardless of how it was hidden. IOW, it does not just show invisible text that you previously hid using this command.
			  • If nil (the default value) then using this command to show invisible text makes visible only such text that was previously hidden by this command. (More precisely, it makes visible only text whose 'invisible' property has value 'hide-comment'.)

Insert Commented Lines	The following commands help insert commented lines or just underlines the current line of text using the character corresponding to one of the adornment level used for reStructuredText sections. The strings are commented according to the major mode of the current buffer. If the buffer has no identified comment strings, the command prompts for them the first time it is used in that type of buffer. The following commands are also listed in the **Σ Inserting Text** table.		
Insert generic file module header block — Language agnostic	`<f6> h`	(**pel-generic-file-header**)	Insert a file header block at the top of the file. • Works only for buffer visiting a file. • Supports all programming and markup language files that have a dedicated major mode. It is also available in buffers for major modes explicitly supported by the `<f12> <f12>` key prefix. This way, those modes can use two different commands to insert file header blocks, each having its own different format. • It supports several programming and markup language and uses the comment style identified by the file extension. If the comment style is unknown the command prompts for one. • The layout of the entered text is controlled by user options. It is possible to create a user-specified skeleton this command will used instead of the one provided by PEL. ⚠ This command is available only 2 seconds after Emacs has started.
Insert commented line See also: **Σ Inserting Text**	• `<f11> i l` • `<f6> l`	(**pel-insert-line** &optional LINELEN)	Insert a (commented) line before/at current line. • If point is at the beginning of the line insert it there. • If point is in the middle of a line, move point at beginning of line before inserting it. • The number of dash characters of the line is specified by LINELEN: • If LINELEN is not specified the buffer's fill-column value is used. • It supports several programming and markup language and uses the comment style identified by the file extension. If the comment style is unknown the command prompts for one. ☛ fill-column is customizable and can be used as a file or directory variable.
Comment-underline current line with level 1 adornment	`<f11> _ 1`	(**pel-commented-adorn-1**)	Insert a commented level-1 reST line adornment at point.
Comment-underline current line with level 2 adornment	`<f11> _ 2`	(**pel-commented-adorn-2**)	Insert a commented level-2 reST line adornment at point.
Comment-underline current line with level 3 adornment	`<f11> _ 3`	(**pel-commented-adorn-3**)	Insert a commented level-3 reST line adornment at point.
Comment-underline current line with level 4 adornment	`<f11> _ 4`	(**pel-commented-adorn-4**)	Insert a commented level-4 reST line adornment at point.
Comment-underline current line with level 5 adornment	`<f11> _ 5`	(**pel-commented-adorn-5**)	Insert a commented level-5 reST line adornment at point.
Comment-underline current line with level 6 adornment	`<f11> _ 6`	(**pel-commented-adorn-6**)	Insert a commented level-6 reST line adornment at point.
Comment-underline current line with level 7 adornment	`<f11> _ 7`	(**pel-commented-adorn-7**)	Insert a commented level-7 reST line adornment at point.
Comment-underline current line with level 8 adornment	`<f11> _ 8`	(**pel-commented-adorn-8**)	Insert a commented level-8 reST line adornment at point.
Comment-underline current line with level 9 adornment	`<f11> _ 9`	(**pel-commented-adorn-9**)	Insert a commented level-9 reST line adornment at point.
Comment-underline current line with level 10 adornment	`<f11> _ 0`	(**pel-commented-adorn-10**)	Insert a commented level-10 reST line adornment at point.

Comments — References

GNU Emacs Manual - Manipulating Comments	
GNU Emacs Lisp — Tips on Writing Comments	Emacs Lisp comments conventions/guidelines.
Drew Adams hide-cmnt @ EmacsWiki	Drew Adams wrote this nice utility to quickly hide all comments in buffer that is under a mode that understands comments. En passant, merci Drew pour ce code (entre autres librairies) et la page en français. Il faudrait bien que je m'y mette et traduise tout mon document un jour…