Update file time stamp

See also: <u>**∑** File mngt</u>

Toggle time stamp automatic update

<f11> f t

<f11> f M-t

(time-stamp)

(time-stamp-toggle-active &optional ARG)

## **Inserting Text**

inserting lext				
<u>Description</u>	<u>Keystroke</u>	Function	<u>Note</u>	
Inserting Text	Customization to co Simple command PEL specialized  PEL specialized  The lice external	ontrol automatic insertion of tim I based text insertions: commands to insert formatted rnal package activated by	zed text at point (cursor) location. This includes: ne stamp, update of copyright notice.  text like time stamps, file path, file name, copyright notice, Greek letters, commented lines, etc  pel-use-lice user option, used to insert open source licence text.	
See <u>∑</u>	<ul> <li>The <u>smart-dash</u> external package activated by <u>pel-use-smart-dash</u>, used to automatically convert dash into underscore when typing.</li> <li>The <u>smartparens</u> external package activated by <u>pel-use-smartparens</u> to provide automatic insertion of balanced block pairs for code.</li> <li>Specialized template-based text insertion:         <ul> <li>PEL <u>tempo skeletons</u> based templates for generic &amp; specialized boilerplate file sections: file, class, function header, document section header.</li> <li>The <u>yasnippet</u> external package activated by <u>pel-use-yasnippet</u> to insert code from predefined snippets.</li> </ul> </li> <li>The <u>yasnippet-snippets</u> external package activated by <u>pel-use-yasnippet-snippets</u> which provides a large amount of snippets.</li> </ul>			
Open this PDF file. See also: <u>▼ Help/Info</u>	• <f11> i <f1> • <f11> y <f1> • <f11> _ <f1></f1></f11></f1></f11></f1></f11>	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the <u>Normal Text</u> local PDF. If the prefix argument (like <b>C-u</b> or <b>M</b> ) is used, then it opens the remote GitHub hosted raw PDF instead. If the <b>pel-flip-help-pdf-arg</b> user-option is set it's the other way around.	
<u>∑ Customize</u> PEL Text Insertions control	<f11> i <f2></f2></f11>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL text insertion support: lice, smart-dash, smartparens, tempo, time-stamp, yasnippet. Also <b>pel-activate-f9-for-greek</b> (see below).  • If OTHER-WINDOW is non-nil (use <b>C-u</b> ), display in other window.	
<u>&gt; Customize</u> Emacs Text Insertions control	<f11> i <f3></f3></f11>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs text insertion support: lice, smart-dash, tempo, time-stamp, yasnippet	
Insert Greek Letter See also: <u>▼ Input Method</u>	<ul> <li><f9> (a)</f9></li> <li><f6> g (b)</f6></li> <li>Insert a greek letter: type <f9> followed by a key in [a-zA-Z] range inserts the Unicode character for the equivalent Greek letter Examples: <f9> a inserts α <f9> b inserts β <f9> a inserts A <f9> b inserts β</f9></f9></f9></f9></f9></li> <li>The insertions work everywhere insert is allowed, including in response to prompts.</li> <li>Use <f9> C-h or which-key mode and type <f9> to see all keys.</f9></f9></li> <li>The <f9> key binding is only available when the pel-activate-f9-for-greek user-option is turned on. The <f6> g binding is always available.</f6></f9></li> <li>This is not a command bound to a key: it's an additional set of bindings added to Emacs key-translation-map.</li> </ul>			
Insert time & file info	The following commar	nds insert time stamps of speci	ific formats and name of the current file.	
Insert current date	• <f11> i d • <f6> d</f6></f11>	(pel-insert-current-date &optional UTC)	Insert current date (only, no time) at point.  • Local by default, UTC if C-u prefix used.	
Insert current date & time	• <f11> i D • <f6> D</f6></f11>	(pel-insert-current-date- time &optional UTC)	Insert current date and time at point.  • Local by default, UTC if C-u prefix used.	
Insert current filename	• <f11> i f • <f6> f</f6></f11>	(pel-insert-filename &optional N)	Insert the file name of the currently edited file at point.  By default: insert filename of current buffer with complete absolute path.  With a numeric argument you can select the file name of the current buffer or the buffers in the 4 surrounding windows. 8: up, 2: down, 4: left, 6: right. Any other number identifies the current window.  When the numeric argument is positive the file with complete absolute path is inserted,  With negative numeric argument the path is omitted.	
Insert time stamp	• <f11> i t • <f6> t</f6></f11>	(pel-insert-iso8601- timestamp &optional UTC)	Insert ISO 8601 conforming abbreviated YYYY-MM-DD hh:mm:ss format timestamp.  • Local by default, UTC if C-u prefix is used.	
Insert software license text	• <f11> i L • <f6> L</f6></f11>	(lice NAME)	Insert license and headers at point.  Prompts for license NAME, which is a license template name like "mit", "gpl-3.0", etc The list is available with TAB completion: hit TAB on prompt to get the complete list of templates.  PEL activates it if pel-use-lice user option is t.	
Automatic File Time Stamp on file save  References: TimeStamps @ EmacsWiki	This can either be dor (add-hook	ne via Emacs customization sys'before-save-hook 'time be added to files that contain,	les. It must be activated by adding the time-stamp function to the before-save-hook variable. stem or explicitly inside your init file with the following code: -stamp) inside their first 8 lines, a line that looks like one of the following:	
Change time stamp format in:     markdown file				
reStucturedText file		time-stamp-pattern consists of 4 parts, each one controlled by a variable:     time-stamp-line-limit: identifies where in the file the time stamp can be located. Defaults to 8: the first 8 lines.		
See also: <u>▼ File mngt</u>	<ul> <li>time-stamp-start: identifies the text pattern that precedes the time stamp.</li> <li>time-stamp-end: identifies the end of the time stamp.</li> <li>time-stamp-format specifies the format of the time stamp.</li> <li>Something like "%:y-%02m-%02d %02H:%02M:%02S %u" to specify the date and time in ISO format, with the user login's name.</li> <li>time-stamp-time-zone specifies the time zone selection:         <ul> <li>nil : Emacs local time</li> <li>t : Universal time</li> <li>wall : system wall clock time</li> </ul> </li> </ul>			
	The time-star		me-zone variables can be set in your init file or via the Emacs customization system.	
	1 · · · · · · · · · · · · · · · · · · ·	ed in the <b>time-stamp</b> customize the format or the pattern prece	zation group.  eding or after the automatically updated time stamp, it is best to use file local variables: this will	
	allow automat  By default, the time	ic time stamp updates in files whe-stamp string must be placed	with various formats. As an example, see the top and end of the PEL manual raw format file.  d within the first 8 lines of the file, otherwise it will not be updated automatically.	
	Carrier Co.		e set the <b>time-stamp-line-limit</b> file local variable. automatic generation of time-stamps:	
	pel-update-time     corresponding to	e-stamp user-option controls we Emacs settings as described a	whether time-stamps are automatically update time stamps in all files where a valid time-stamp above. Set it to <b>t</b> (the default) to allow automatic time stamp updates. Set it to nil to prevent them. litting session by using the <f11> <b>f</b> M-t key sequence.</f11>	
	_		ackage provides a set of text insert commands which include inserting a time stamp.	

Force update the time stamp string(s) in the current buffer.

• Updates a time stamp of format recognized by *Emacs current settings* even when automatic

time-stamp update is off.

• More information about the "Emacs current settings" in the description block above.

Toggle 'time-stamp-active', setting whether <f11> f t updates a buffer.

• With ARG, turn time stamping on if and only if arg is positive.

To insert a non-updatable time stamp, the PEL package provides a set of text insert commands which include inserting a time stamp .

Description	<u>Keystroke</u>	Function	<u>Note</u>
Inserting & Automatically Updating Copyrights	Two commands, sh	own below, are provided to ma	of copyright notices inside files.  anually insert or update the file's copyright notice.  d by adding the copyright-update function to the list of before-save-hook variable with the
	(add-hook 'before-save-hook 'copyright-update)  1 To be automatically updated, the copyright notice must be placed within an area at the beginning of the file specified by the value of the copyright-limit variable, normally defined as the first 2000 characters. This variable is customizable.		
Insert copyright notice See also: File mngt	<f11> i c</f11>	(copyright &optional STR ARG)	Insert a copyright by \$ORGANIZATION notice at cursor.  • If the ORGANIZATION environment variable is not available, Emacs prompts for it.
Update file's copyright notice	<f11> i M-c</f11>	(copyright-update &optional ARG INTERACTIVEP)	Update copyright notice to indicate the current year.  • With prefix ARG, replace the years in the notice rather than adding the current year after them. If necessary, and 'copyright-current-gpl-version' is set, any copying permissions following the copyright are updated as well.
Only update exiting notice.  Does not create one if it's missing.	copyright-update does not warn if there is no copyright in the current buffer to update. It does not create a missing notice.  copyright-update does not warn if there is no copyright in the current buffer to update. It does not create a missing notice.  for a copyright user option to the copyright user op		
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 <b>Comments</b> table.		
Insert commented line See also: <u>▼ Comments</u>	• <f11> i 1 • <f6> 1</f6></f11>	(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.
Comment-underline current line with level adornment 1-9	<f11> _ *</f11>	(pel-commented-adorn-1)	Insert a commented level-x reST line adornment at point.  • 🛎 := 1 to 9 for levels 1 to 9
Comment-underline current line with level 10 adornment	<f11> _ 0</f11>	(pel-commented-adorn-10)	Insert a commented level-10 reST line adornment at point.
Smart Dash Mode	<ul> <li>Anyone that has been writing Lisp code for a while knows that using dash as word separator instead of underscore is more natural and faster to type. Unfortunately most programming languages (all non-Lisp?) have restrictions on the characters available in identifiers and underscore is often used. Typing underscore requires hitting the Shift key and it annoys some people that enjoyed writing Lisp code. This is where the smart-dash-mode helps. You can insert underscore in text by typing the dash key without hitting the Shift key! A very useful mode.</li> <li>More information is available in the author's page.</li> <li>Requires the smart-dash external package.</li> <li>PEL activates it when pel-use-smart-dash is set to t.</li> <li>To activate smart-dash-mode automatically:</li> <li>for major modes supported by PEL, add smart-dash-mode to the pel-<mode>activates-minor-mode user-option for the specific mode.</mode></li> <li>for other modes, add the mode name to the pel-modes-activating-smart-dash-mode user-option.</li> </ul>		
Toggle smart-dash mode	<f11> i -</f11>	(smart-dash-mode &optional ARG)	Toggle the smart-dash-mode on/off.
See also:  Numkeypad  Text Modes  Mode Line	<ul> <li>When smart-dash-mode is active, it redefines the dash key to insert an underscore within C-style identifiers and a dash otherwise. This allows you to type all_lowercase_c_identifiers as comfortably as you would lisp-style-identifiers.</li> <li>While Smart-Dash mode is active, you can type C-q - or use the minus key on the numeric keypad to override it and insert a dash after a C-style identifier character. You might need to do this if you want to type a cramped-looking expression like x-5.</li> <li>If Smart-Dash mode is activated while in a C-like mode (c-mode, c++-mode, and objc-mode by default, customizable with 'smart-dash-c-modes') it will also activate Smart-Dash-C mode, which translates "_&gt;" into "-&gt;" and "" into "" automatically so that struct pointer member access and postfix-decrement aren't made more difficult by Smart-Dash mode's tendency to insert underscores at the tail ends of identifiers whether you want it to or not. Note that this will necessitate that you type literal underscores if you want more than one underscore in a row.</li> <li>Normally when smart-dash-mode is active the numeric dash key (<kp-subtract>) acts as a smart-dash only.</kp-subtract></li> <li>However, with PEL, the behaviour of the keypad '-' is only partly affected when the smart-dash-mode is active and it depends on the Numlock</li> </ul>		
	<ul> <li>with an area marked:</li> <li>with area marked with er/expand-region:</li> <li>kill marked area</li> <li>In Numlock ON:</li> </ul>		ill marked area assert an underscore after letter, number or underscore, dash otherwise nore the marked area; insert a dash at point educes the marked area semantically as controlled by er/expand-region and key support, see      Mumkeypad   Mumke
Smartparens Mode  • Smartparens manual	Simplify insertion of m	natching pairs with the smartpa	hort lighter of a green dash is showing in the mode line when smart-dash-mode is active.  arens minor mode. PEL binds a set of keys, described below, to toggle activation of that mode.  PEL activates it when pel-use-smartparens is set to t.
See also: <u>▼</u> <b>X</b> Smartparens	•	enhances the behaviour of certer: smartparens-mode: SP	tain keys, namely those that are part of any pair or tag. smartparens-strict-mode: SP/s
Help on smartparens	<f11> ( ?</f11>	(sp-cheat-sheet &optional ARG)	Generate a cheat sheet of all the smartparens interactive functions. Shows inside Emacs buffer.  • Without a prefix argument, print only the short documentation and examples.  • With non-nil prefix argument ARG, show the full documentation for each function.  • You can follow the links to the function or variable help page.  • To get back to the full list, use M-x help-go-back.  • You can use 'beginning-of-defun' and 'end-of-defun' to jump to the previous/next entry.  • Examples are fontified using the 'font-lock-string-face' for better orientation.
Describe user system	<f11> ( M-?</f11>	(sp-describe-system STARTERKIT)	Describe user's system. Prompt for starter kit: Evil, Spacemac, Vanilla.  • The output of this function can be used in bug reports.
Toggle smartparens mode	<f11> ( (</f11>	(smartparens-mode &optional ARG)	Toggle smartparens mode.
Toggle smartparens-strict mode	<f11> ( )</f11>	(smartparens-strict-mode &optional ARG)	<ul> <li>Toggle the strict smartparens mode.</li> <li>When strict mode is active, 'delete-char', 'kill-word' and their backward variants will skip over the pair delimiters in order to keep the structure always valid (the same way as 'paredit-mode' does). This is accomplished by remapping them to 'sp-delete-char' and 'sp-kill-word'. There is also function 'sp-kill-symbol' that deletes symbols instead of words, otherwise working exactly the same (it is not bound to any key by default).</li> <li>When strict mode is active, this is indicated with "/s" after the smartparens indicator in the mode list</li> </ul>
Toggle smartparens mode	<f11> ( M-(</f11>	(smartparens-global-mode &optional ARG)	Toggle Smartparens mode in all buffers.  • With prefix ARG, enable Smartparens-Global mode if ARG is positive; otherwise, disable it.  • Smartparens mode is enabled in all buffers where 'turn-on-smartparens-mode' would do it.

<u>Description</u>	<u>Keystroke</u>	Function	<u>Note</u>	
Toggle smartparens-strict mode	<f11> ( M-)</f11>	(smartparens-global-strict- mode &optional ARG)	Toggle Smartparens-Strict mode in all buffers.  • With prefix ARG, enable Smartparens-Global-Strict mode if ARG is positive; otherwise, disable	
			<ul> <li>it.</li> <li>Smartparens-Strict mode is enabled in all buffers where 'turn-on-smartparens-strict-mode' would do it.</li> </ul>	
Text and code skeletons	Several mechanisms have been developed to allow easy insertion of predefined text in Emacs.  • Emacs provides the built-in skeleton mechanism and the tempo skeletons.			
tempo skeletons	PEL supports both. They are used a little bit differently.  • PEL provides key bindings to the tempo skeletons: the generic code templates, accessible via the <f6> prefix key, and the language-specific code templates, accessible via the <f12> key prefix.</f12></f6>			
Generic skeletons	PEL provides <b>generic</b> tempo skeletons as well as some specialized for specific programming languages. The generic skeletons are less powerful but often good enough for most types of files. They support all types of files recognized by Emacs as long as Emacs understands the way comments work for the file type which is normally the case. If Emacs does not know the file type the commands assume the file uses a comment start only and will prompt for that string.			
<u>&gt; Customize</u> PEL Text Insertions control	<f6> <f2></f2></f6>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL generic tempo skeleton customization groups that control the format of the various skeletons including the generic skeleton used by the <f6> h key (se below).  • If OTHER-WINDOW is non-nil (use C-u), display in other window.</f6>	
Insert generic file module header block — Language	<f6> h</f6>	(pel-generic-file-header)	Insert a file header block at the top of the file. Works only for buffer visiting a file.	
After inserting the template, navigate though areas that must be filled with:  • tempo-forward-mark: C-c.  • tempo-backward-mark: C-c, See examples in manual	Supports all text file types.  • 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.  • Specify the format of the header via the user-options in the pel-pkg-generic-code-style customization group accessible via <f6> <f2>  • The files that have no extensions are often used in Unix-like OS shell scripts. These files are also supported as Emacs can recognize them if they are stored in a bin directory. PEL also has special support for them and is controlled by the pel-sh-script-skeleton-control customization group, which is accessible as a child of the main group.  • After inserting the template you can use the tempo-forward-mark and tempo-backward-mark to move point to the beginning of each section that must be filled.  • The command key binding <f6> h is available only 1 second after Emacs has started.</f6></f2></f6></f12></f12>			
Toggle pel-tempo-mode	<f6> SPC</f6>	(pel-tempo-mode &optional	Toggle PEL tempo mode on/off.	
		ARG)	PEL tempo mode activates C-c . and C-c , as well as to C-c C and C-c C-, key bindings to navigate across tempo mark hot-spots. When pel-tempo-mode is active the pel-tempo-mode lighter (‡) is shown on the status bar. The second set of keys are only available when Emacs runs in graphics mode.  The pel-generic-file-header command inserts the text using a tempo skeleton: the PEL tempo mode is automatically activated by typing <f6> h.</f6>	
Jump to next tempo mark	• C-c M-f • C-c . • C-c C	(tempo-forward-mark)	Jump to the next mark in 'tempo-back-mark-list': the location where code must be updated inside the inserted skeleton.  • These key key bindings are only available when pel-tempo-mode is active.	
Jump to previous tempo mark	• C-c M-b • C-c , • C-c C-,	(tempo-backward-mark)	Jump to the previous mark in 'tempo-back-mark-list': the location where code must be updated inside the inserted skeleton.  • These key binding are only available when pel-tempo-mode is active.	
Store PEL code template settings in .dir-locals.el to fine-tune layout of files in a directory tree	Emacs user options by default take effect globally. But by using file and directory variables (see File/Directory Variables) they can also be used to take effect on a single file or all files inside a directory tree. So by default, the user options that control the PEL tempo template take effect globally. If you want to change the behaviour for only one file, write the user option control block at the end of that file. If you want to control the behaviour of the PEL tempo templates for all files inside a directory tree create a .dir-locals file and store the values of the relevant options variables inside that file. This allows you to control the user options affecting the format of the tempo templates precisely.			
Example:	you can keep this for of files have a 1-line copy	other files but inside a directory	odule-section-titles identifies the 3 sections "Module Description", "Dependencies" and "Code" you can force all shell-mode files to use 2 sections: "Description" and "Script" and ensure that all sel file containing the following code:	
		;;; Directory Local variables ;;; For more information see (info "(emacs) Directory Variables")		
		(pel-generic-skel-with- (pel-generic-skel-with- (pel-generic-skel-modul		
Entering Templated Text with Tempo Skeletons See also: • Major mode specific: • MI - C • MI - C++ • MI - Enacs Lisp • MI - Erlang • M reStructuredText	Emacs built-in support includes the tempo skeletons. PEL implements extension to the tempo skeleton Emacs built-in package under two prefix keys:  The commands under the <f6> prefix keys insert template text that are adapted to each major mode. They are generic in nature, and dynamically adapt to the major mode and the comment style supported by the major mode. The layout of the templates is the same for every major mode, they differ only by the comment strings.  The commands under the <f12> <f12> prefix key insert templates specialized for the programming or markup language of the major mode that support this key prefix. PEL attempts to use the same key bindings for equivalent concepts (such as file header block) inside each mode specific instance of the <f12> <f12> key maps as much as possible. The tempo skeletons provided by PEL can be quite complex and their formats are controlled by user options. PEL currently only support this key prefix with for the following major modes (more are planned):  C++, Emacs Lisp, Erlang  restructuredText</f12></f12></f12></f12></f6>			
Major-mode specific Tempo Templates Prefix	<ul> <li><f12> <f12></f12></f12></li> <li>Key prefix sequence to the list of tempo skeleton commands.         <ul> <li>This command prefix is available only for some major modes (see the list in the first column) of the section row above.</li> <li>The commands under this prefix insert text specialized for their specific major mode, as opposed to the commands bound to the <f6> prefix key. For more information see the language specialized reference table.</f6></li> </ul> </li> </ul>			
Entering Templated Test			ackage which provides another way to insert templated text, and <u>yasnippet-snippets</u> external	
with Yasnippet	package which provides a large set of code snippets for a large set of major modes.  • To use yasnippets, you must type the snippet abbreviation and then hit the TAB key to expand the text.			
• See also: <u>Customize</u>	Requires <a href="mailto:yasnippet">yasnippet</a> activated when <a href="mailto:pel-use-yasnippet">pel-use-yasnippet</a> is set to t or to use-from-start.  Requires <a href="mailto:yasnippet-snippets">yasnippet-snippets</a> is set to t.  Use the key <			

<u>Description</u>	<u>Keystroke</u>	Function	<u>Note</u>
Toggle YASnippet minor mode on/off	<f11> y y</f11>	(yas-minor-mode &optional ARG)	Toggle YaSnippet mode.
	<ul> <li>When YASnippet mode is enabled, 'yas-expand', normally bound to the TAB key, expands snippets of code depending on the major mode.</li> <li>With no argument, this command toggles the mode. Positive prefix argument turns on the mode. Negative prefix argument turns off the mode. YASnippet mode key bindings:         <ul> <li>key</li> <li>binding</li> </ul> </li> </ul>		
	C-c & C-s	yas-new-snippet yas-insert-snippet yas-visit-snippet-file	
Toggle YASnippet global mode on/off	<f11> y Y</f11>	(yas-global-mode &optional ARG)	Toggle Yas minor mode in all buffers.  • With prefix ARG, enable Yas-Global mode if ARG is positive; otherwise, disable it.
Expand snippet whose name is just before point	TAB	(yas-expand &optional FIELD)	Expand a snippet before point. If no snippet expansion is possible, do nothing.  This key binding is only active when the YASnippet mode is active. Once the snippet was expanded the TAB key normal behaviour is restored.
Write a new snippet	• <f11> y n</f11>	(yas-new-snippet &optional NO-TEMPLATE)	Pops a new buffer for writing a snippet.  • Expands a snippet-writing snippet, unless the optional prefix arg NO-TEMPLATE is non-nil.
	• C-c & C-n	NO-TEMPLATE)	
Prompt for snippet and insert it	• <f11> y s</f11>	(yas-insert-snippet &optional NO-CONDITION)	Choose a snippet to expand, pop-up a list of choices according to 'yas-prompt-functions'  • With prefix argument NO-CONDITION, bypass filtering of snippets by condition.
	• C-c & C-s		
Visit a snippet file	• <f11> y v</f11>	(yas-visit-snippet-file)	Choose a snippet to edit, selection like 'yas-insert-snippet'.
	• C-c & C-v		Only success if selected snippet was loaded from a file. Put the visited file in 'snippet-mod
Display all snippets for current major mode	<f11> y t</f11>	(yas-describe-tables &optional WITH- NONACTIVE)	Display snippets for each table.
Prints Yasnippet version info	<f11> y ?</f11>	(yas-about)	Prints version information in the mini buffer.

## **Inserting Text — References**

Topic & link	Description
GNU Emacs Manual: Time Stamps	
Smart-Dash Mode homepage	A description of this extremely useful mode and why it was created.