File Management

Operation	Keystroke	Function	Note
File Management See also: • ∑M Dired • ∑ Customize	Emacs provides a large set of commands to open files (Emacs documentation uses the term "finding" files for that), saving files searching for files or file content, displaying directory content, etc… These are listed in this table. • The directory editing (dired) commands are mainly listed in the ∑M **Dired** table. • There are also several Emacs internal and external packages that provide useful commands. PEL supports several of them, and activates them via customize user option variables. They are listed below. Use Emacs customize system to modify their values to activate, deactivate and modify the behaviour of these packages. PEL provides the **<f11> <1> f** key binding to quickly gain access to the appropriate customize group where you can change their values. Once you have modified the values, save them and then either execute `M-x pel-init` or restart Emacs.		
Open this PDF file. See also: ∑ Help/Info	`<f11> f <f1> 1`	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the local copy of the ∑ **File-mngt** PDF file unless a command prefix (like `C-u`) was used. In that case it opens the Github-hosted file instead.
PEL File/Directory Management See also:∑ Customize	`<f11> f <f2> 1`	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL support for file management. • If OTHER-WINDOW is non-nil (use `C-u`), display in other window.
Customize Emacs support for file management	`<f11> f <f3> 1`	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs support for file management.
Customize Emacs support for file revert	`<f11> f r <f3>`	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs support for file automatic revert management.
Customize ffap (find file at point)	`<f11> f a <f3>`	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs support for management of ffap (find file at point).
Open application	The following commands open OS level applications		
Open currently file visited in current buffer with the default OS application. See also: • ∑M Dired	`<f11> f F`	(**pel-open-in-os-app** &optional FNAME)	Open the file with the OS-registered application. 🐧🍎❖ Currently only works on Linux, macOS and Windows. • To open a file without first loading it inside an Emacs buffer, open a Dired buffer and use 'z' on the filename. You can also select several file names.
Opening file	The following commands are available to open/visit files in Emacs buffers. • For some of them the corresponding ido mode function is also shown. • **Note**: Emacs uses the word "visiting" instead of "opening" files. • The command used to 'visit' a file, find-file is Emacs default. It supports Emacs' basic tab completion. Packages that support other completion mechanisms can be installed and activated and then the command uses a different completion mechanism. • ☝PEL customization system allows you to specify whether you want to use one or several other completion mechanisms. It also has a command to change the completion mechanism dynamically. You can change it without restarting Emacs or event re-executing pel-init. • See the ∑ **Completion/Input** and ∑ **Customize** tables for more info.		
File Lock Protection	Emacs protects against multiple processes modifying the same file with a lock. If you attempt to edit the buffer of a locked file, or save a buffer of a locked file , Emacs will prompt. You can then: • steal the lock (with 's'), • proceed ('p') to edit the file anyway or • quit ('q').		
Open file using OS file-open dialog	`⌘-o`	(**ns-open-file-using-panel**)	🍎 On macOS in graphics mode only: open a file, select the file name via an OS File dialog.

Open (visit) a file/directory
 	
See also:
• ∑ Completion/Input
• ∑M Dired
• ∑ Customize
	`<f11> C-f`
	`C-x C-f`
		(**find-file** FILENAME &optional WILDCARDS)	Prompt for the file or directory name to open. Open the selected file/directory in a buffer with the appropriate mode. For directory, the buffer opens in Dired-mode.
			• With PEL, the `<f11> C-f` key binding to find-file is always available, regardless of what completion mechanism is in use. It can be used as a fallback when testing various completion packages. I have seen some of them fail and break Ido.

		(**ido-find-file**)	Same as above with Ido completion ☛See ∑ **Completion/Input** for completion modes available at the prompt. The **ido-use-filename-at-point** user-options control whether ido-find-file uses the file name at point as the basis for selecting the file name to open. PEL provides the `<f11> f M-.` key sequence to dynamically change the value. See below.
	• find-file is the original command and uses Emacs default completion. When Ido is used, the ido-find-file command is used instead. • When **ido** mode is used, you can also: • Type `C-f` or `C-x f` to change to original find-file mode and prevent Ido completion from trying to provide the name of an existing file when you want to specify the name of a file that does not exists yet. • Type **C-j** to accept the file/directory name verbatim without replacement or suggestion. This is often useful to open a directory in directory editing (dired) mode. ☝To open a file in read-only mode you can: • Use one of the commands below (`C-x C-r`, etc…) • Use `C-x C-f` then type `C-x C-q` to change the mode of the buffer to read-only mode. PEL supports dynamic selection of completion input that control the way this command operates to help you select a file name: (ido, ivy, helm).		
Open another file in buffer	`C-x C-v`	• (**find-alternate-file** FILENAME &optional WILDCARDS) • (**ido-find-alternate-file**)	Kills buffer and open the newly specified file in a new buffer same window. When ido-mode is used, the ido-find-alternate-file is used instead. Useful when just selected an empty file just selected by mistake.
Open file in other window	• `C-x 4 f` • `<f11> f o`	• (**find-file-other-window** FILENAME &optional WILDCARDS) • (**ido-find-file-other-window**)	Edit file FILENAME, in another window. Like `C-x C-f`, but creates a new window or reuses an existing one.
Open file in other frame	`C-x 5 f`	• (**find-file-other-frame** FILENAME &optional WILDCARDS) • (**ido-find-file-other-frame**)	Edit file FILENAME, in another frame. Like `C-x C-f`, but creates a new frame or reuses an existing one.
Control whether ido-find-file uses filename at point	`<f11> f M-.`	(**pel-set-ido-use-fname-at-point** &optional GLOBALLY)	Set Ido's ability to use the filename at point as a starting point in the current buffer or globally. It can set it to one of: • disabled : don't use filename at point. • guess : try to identify an exiting file name from the name at point. • literal : use name at point in the Ido search for a file name.
	• By default this commands sets ido-find-file behaviour for the current buffer only by setting a **ido-use-filename-at-point** buffer local variable. • Use any prefix argument (eg. `C-u`) to modify the behaviour globally for the current Emacs session, it does not persist across Emacs sessions. For a persistent behaviour change you must customize **ido-use-filename-at-point** user-option variable. For that, use `M-x customize-option`. • This affects the behaviour of all commands opening file using Ido completion: ido-find-file as the others.		
Open in read-only	The following commands open files in read-only mode. While in read-only mode, use Use `C-x C-q` to permit editing.		
Open a file in read-only mode	`C-x C-r`	• (**find-file-read-only** FILENAME &optional WILDCARDS) • (**ido-find-file-read-only**)	Edit FILENAME but don't allow changes. Like `C-x C-f`, but marks buffer as read-only. Use `C-x C-q` to permit editing.

Operation	Keystroke	Function	Note
Open file in other window in read-only mode	• `C-x 4 r` • `<f11> f o`	• **(find-file-read-only-other-window** FILENAME &optional WILDCARDS) • **(ido-find-file-read-only-other-window)**	(find-file-read-only-other-window FILENAME &optional WILDCARDS) Edit file FILENAME in another window but don't allow changes. Like `C-x C-f`, but marks buffer as read-only. Use `C-x C-q` to permit editing.
Open Literally	Open a file with no encoding conversion: file is opened in the Fundamental mode: the major mode normally associated with the file type is not used. ☝Note that when using Ido completion, it is possible to use a command during completion to force Ido to open the file literally. However, if you are using Emacs default completion, the following command is the only way to open a file literally.		
Visit a file literally: with no encoding support and conversion	`<f11> f M-l`	**(find-file-literally** FILENAME)	Visit file FILENAME with no conversion of any kind. • Format conversion and character code conversion are both disabled, and multibyte characters are disabled in the resulting buffer. • The major mode used is Fundamental mode regardless of the file name, and local variable specifications in the file are ignored. • Automatic uncompression and adding a newline at the end of the file due to 'require-final-newline' is also disabled. • If Emacs already has a buffer which is visiting the file, this command asks you whether to visit it literally instead.
Open binary	Open a file in hex binary mode. There are also commands to convert current buffer to hexadecimal editing, like nhexl. See **∑ Buffers**.		
Open a file in hexl-mode See also: **∑ Buffers**	`<f11> f M-x`	**(hexl-find-file** FILENAME)	Edit file FILENAME as a binary file in hex dump format, using the 'hexl-mode'. • Switch to a buffer visiting file FILENAME, creating one if none exists.
Recently opened	When the **pel-use-recentf** user option is set to **t**, Emacs remembers the list of recently opened files. • The menu bar includes a File->Open Recent menu entry. • Some other functions are activated by their respective user options.		
Open recently opened files, listed with Ido	`<f11> f f`	**(ido-recentf-open)**	Open file. Prompt suggests recently opened files wit Ido-style completion. • Type <tab> to get possible expansions listed in a separate buffer. Available only when both **pel-use-ido** and **pel-use-recentf** are set to **t**. Credits: Mickey Petersen recentf article.
Open recently opened files, listed with Counsel	`<f11> f R`	**(counsel-recentf)**	List files recently opened in a counsel buffer. • The list of recently opened files are listed in a Counsel buffer . Select one and type return to open. • Type `C-c C-o` to copy the list of files inside a special buffer. 📦 Requires **Ivy mode completion with Counsel mode** and recentf activated: set **pel-use-counsel** and **pel-use-recentf** to **t**
Edit list of recently opened files	`<f11> f M-r`	**(recentf-edit-list)**	Show a dialog to delete selected files from the recent list. Use it to remove some of the files from the list.
Open file at point	The following commands, followed by the flap commands, allow opening files from the file name taken at point (the cursor location). They work regardless of the input completion method currently used. ☝ 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 ★★ See also: • **∑ Key-Chords** • **Ϻ reStructuredText**	• `C-^` • `<f11> f .` • `6y`	**(pel-open-at-point** &optional N)	Open the file, library or the URL, named at point, with potential line & column #s. 📦 With PEL, the **`6y`** key-chord is available if pel-use-key-chord is non-nil. ☝Command prefixes are supported with the key-chord. See **∑ Key-Chords**.
	• If point is on a reStructuredText link in a rst-mode buffer, open the link target (that might be a local file or a URL on remote web site. In the latter case the page is opened in the systems' browser). • If embedded space(s) are allowed in the filename, then point must be located at the first of the 2 delimiter characters. These delimiter character can be any of the following: **" ` ' | ( ) [ ] { } < >** ‘ ’ “ ” 「」 〔〕 〈〉 《》 【】 〖〗 « » ‹ › ❮❯⦅⦆ 〘〙 • 。 • The first 12 are part of ASCII but not the remainders, which are, however part of Unicode. • Tab and newline are also delimiter characters. • If embedded space in the file name is not allowed, then the file name must also be enclosed in the above delimiters, the space acts as an extra delimiter, and point can be positioned anywhere between the delimiters. • If the string identifies a **URL**, the function opens the page in the systems' default browser. • Prompts for incomplete file names, allowing editing the find file (with completion), search for libraries files (type **1**) according to current file type. 🚧Currently only supports Emacs Lisp files. Planning to support other programming languages with and without project management packages. • Without argument: • If file 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 based on 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 numeric argument N: • N < 0 : create a new window and use that • N = 0: use the '*other*' (the next) window • N = 1,3,7or above (excluding 9): select the target window based on 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. • N is: 8: up, 2: down, 4:left, 5:current, 6:right. • N is 9: open the file in the system's browser, and for a directory name at point open the application associated with directory browsing (eg. macOS Finder, Windows Explorer). • N is 10: open the URL at point in the system's browser. • Selecting Minibuffer, inexistent or dedicated window is not allowed. • If the file name is followed by line and column numbers the point is moved to that position. • When executed from with a buffer in sh-mode, the '=' and ':' characters are used as additional delimiters. Also shell variables (such as $HOME) are expanded. More information available in the command's help docstring.		
Open filename at point in a browser See also: • **∑ Key-Chords** • **∑ Web**	• `<f11> f /` • `6u`	**(pel-browse-filename-at-point)**	Open the file name at point inside the system's browser. • If point is at a directory name, open the systems application that browses directories (eg. macOS Finder, Windows Explorer). ☝This is the same as using pel-open-at-point with the argument N set to 9. It is easier to type and PEL assigns its own key-chord for it.
Open URL at point in a browser See also: • **∑ Key-Chords** • **∑ Web** • **∑ Customize**	• `<f11> f M-/` • `7u`	**(browse-url-at-point** &optional ARG)	Ask a WWW browser to load the URL at or before point. • Variable 'browse-url-browser-function' says which browser to use. • Optional prefix argument ARG non-nil inverts the value of the option 'browse-url-new-window-flag'. ☝ PEL provides the `<f11> <f2> E u` key sequence to open the browse-url group that contains relevant the user options.
Copy URL at point in temporary file and visit the file	`<f11> f M-u`	**(pel-open-url-at-point)**	Copy the URL at point to a local temporary file and visit that file. • ⚠️The download copy of the file does not have the same name and may not open with the proper mode because it won't have an extension. The HTML formatted files will be recognized by Emacs but most of the files won't be. • Save the file somewhere else using the **C-x C-w** key sequence and identify the proper extension to activate the required major mode.
	`C-c C-f`		☝ This binding is only available when point is over the URL and the **goto-address-mode** minor mode is active. Use **<f11> f u** or **<f11> f U** to activate this mode.

Operation	Keystroke	Function	Note
ffap commands	Emacs provides the ffap (find file at point) command set. The ffap command is similar to pel-find-file-at-point-in-window but does not support line and numbers, does not support identifying a window with command arguments and is not designed to support multiple programming languages. It does however support other facilities and can be installed to replace the behaviour of standard file management command bindings such as `C-x C-f`. PEL activates the Emacs built-in ffap library when the **pel-use-ffap** user option is set to either **t** or to **ffap-bindings**. In both cases these activate the key bindings shown below. • When **pel-use-ffap** is set to ffap-bindings, then PEL also activates the standard ffap bindings which take over the behaviour of the main file finding and dired commands. This means that Ido, Ivy or Helm are no longer available for these commands. • If **pel-use-ffap** is only set to **t** then the standard ffap bindings is not activated.		
Find file/URL at point	`<f11> f a p`	(**ffap** &optional FILENAME)	Find FILENAME, guessing a default from text around point. • If 'ffap-url-regexp' is not nil, the FILENAME may also be an URL. Web URL opens in browser. • With a prefix, this command behaves exactly like 'ffap-file-finder'. • If 'ffap-require-prefix' is set, the prefix meaning is reversed. • See also the variables 'ffap-dired-wildcards', 'ffap-newfile-prompt', 'ffap-url-unwrap-local', 'ffap-url-unwrap-remote', and the functions ffap-file-at-point' and 'ffap-url-at-point'.
Find file/URL at point - read only	`<f11> f a P`	(**ffap-read-only**)	Like 'ffap', but mark buffer as read-only.
Find another file/URL at point in window	`<f11> f a v`	(**ffap-alternate-file**)	Like 'ffap' and 'find-alternate-file': kills current buffer and open new file in the same window.
Find file/URL in other window	`<f11> f a w`	(**ffap-other-window**)	Like 'ffap', but put buffer in another window.
Find file/URL in other frame	`<f11> f a f`	(**ffap-other-frame**)	Like 'ffap', but put buffer in another frame.
Find file/URL in other window - read only	`<f11> f a W`	(**ffap-read-only-other-window**)	Like 'ffap', but put buffer in another window and mark as read-only.
Find file/URL in other frame - read only	`<f11> f a F`	(**ffap-read-only-other-frame**)	Like 'ffap', but put buffer in another frame and mark as read-only.
Start Dired with file at point	`<f11> f a d`	(**dired-at-point** &optional FILENAME)	Start Dired, defaulting to file at point. See 'ffap'.
Start Dired with file at point in other window	`<f11> f a D`	(**ffap-dired-other-window**)	Like 'dired-at-point', but put buffer in another window.
Start Dired with file at point in other frame	`<f11> f a M-d`	(**ffap-dired-other-frame**)	Like 'dired-at-point', but put buffer in another frame.
List directory of file at point	`<f11> f a l`	(**ffap-list-directory**)	Like 'dired-at-point' and 'list-directory'.
Open a menu of all files, URL in current buffer.	`<f11> f a m`	(**ffap-menu** &optional RESCAN)	Put up a menu of files and URLs mentioned in this buffer. • Then set mark, jump to choice, and try to fetch it. The menu is cached in 'ffap-menu-alist', and rebuild by 'ffap-menu-rescan'. • The optional RESCAN argument (a prefix, interactively) forces a rebuild. Searches with 'ffap-menu-regexp'.
Open Dired (Directory Editor)	When "opening" (visiting) a directory Emacs opens a buffer in Dired mode, that looks like a ls -l output, which allows several operations. If you specify a directory path to Cx C-f then Dired-mode is used. You can also use the following commands to open buffer in Dired mode. ☝ It's also possible to browse a file directory tree with **file tree browsers,** like NeoTree and Ztree, described below in this table. The ∑ **Speedbar** can also be used.		
Open a directory editor See also: ∑M **Dired** ∑ **Completion/Input**	• `C-x d` • `⌘-D`	• (**dired** DIRNAME &optional SWITCHES) • (**ido-dired**)	Opens a Dired-mode buffer on the specified directory. Prompt for the directory name. PEL activates ido when the **pel-use-ido-mode** user option is set to **t**. ☛ See ∑ **Completion/Input** for completion modes available at the prompt.
Run Dired in other (next) window	`C-x 4 d`	(**dired-other-window**)	Opens a Dired-mode buffer on the specified directory inside another window. • Prompt for the directory name.
List Directory See also: ∑ **Completion/Input**	`C-x C-d`	(**list-directory** DIRNAME &optional VERBOSE)	Display a list of files in or matching DIRNAME, a la 'ls'. • DIRNAME is globbed by the shell if necessary. • Prefix arg (**C-u**) means supply -l switch to 'ls'. ☛ See ∑ **Completion/Input** for completion modes available at the prompt.
Activating URLs to browse and open files	Emacs provides the **goto-url-mode** and the **goto-url-prog-mode** that turn URLs found in the current buffer into clickable buttons. • Once the mode is active the following key sequences are available wheel point is over a URL button: • `C-c RET` or the mouse to click on the button. • If the URL is an email address a buffer to write an email to that address opens. • If the URL is a web or FTP address the system browser is invoked to open the address. • `C-c C-n` : move point to the end of the next URL in the buffer. • `C-c C-p` : move point to to the previous URL in the buffer. • `C-c C-f` : download the file identified by the URL into a local temporary file and visit the file. See (pel-open-url-at-point) above. Customization group: **goto-address** . Mostly control the regex for URL and the face used.		
Toggle goto-address-mode	`<f11> f u`	(**goto-address-mode** &optional ARG)	Minor mode to buttonize URLs and e-mail addresses in the current buffer. With a prefix argument ARG, enable the mode if ARG is positive, and disable it otherwise.
Toggle goto-addrress-prog-mode	`<f11> f U`	(**goto-address-prog-mode** &optional ARG)	Like 'goto-address-mode', but only for comments and strings.
Open the URL (email or web page)	`C-c RET`	(**goto-address-at-point** &optional EVENT)	Open the URL at point: • If URL is a web page: open it in a browser • If URL is a mail address: • Send mail to address at point: Find e-mail address around or before point. Then search backwards to beginning of line for the start of an e-mail address. • If no email address is found there, then load the URL at or before point.

Move to end of next URL in buffer
See also: ∑ **Navigation**
	`C-c C-n`
	`<f6> C-n`
		(**pel-goto-next-url**)	Move point forward to the end of the next URL located in the current buffer.
			• The global `<f6> C-n` key binding activates the goto-address-mode if it is not already active.

Move to beginning of previous URL in buffer
See also: ∑ **Navigation**
	`C-c C-p`
	`<f11> C-p`
		(**pel-goto-previous-url**)	Move point backward to the beginning of the previous URL located in the current buffer.
			• The global `<f6> C-p` key binding activates the goto-address-mode if it is not already active.

Insert text of another file at point	The following commands can be used to insert text from other files at point in the current buffer.		
Insert file at point	• `C-x i` • `<f11> f i`	• (**insert-file** FILENAME) • (**ido-insert-file**)	Insert contents of file FILENAME into buffer after point. • Set mark after the inserted text.
Insert file literally at point	`<f11> f I`	(**insert-file-literally** FILENAME)	Insert contents of file FILENAME into buffer after point with no conversion. • Set mark after the inserted text.
Write text into specified file	The following commands can be used to write text selected from current buffer into specified file.		

Operation	Keystroke	Function	Note
Write region text to file	`<f11> f w`	(**write-region** START END FILENAME &optional APPEND VISIT LOCKNAME MUSTBENEW)	Write current region into specified file. - Prompts for the specified file.
Append region text to file	`<f11> f W`	(**append-to-file** START END FILENAME)	Append the contents of the region to the end of file FILENAME. - Prompts for the specified file.
Set file mode	`<f11> f m`	(**set-file-modes** FILENAME MODE)	Set mode bits of file named FILENAME to MODE (an integer). - Only the 12 low bits of MODE are used. - Prompts for file name and then for chmod-like file mode value.
Reverting Files	If the file's content changed on the disk and you want to refresh the Emacs buffer visiting that file, you need to "revert" the file. - If you want to use Emacs to monitor the content of a file that is continuously modified by an external process (like a log file) set the ***revert-without-query*** variable to a list of regular expressions describing the field it'll apply to. - You can also activate the auto-revert mode for the current buffer or globally and restart its timer.		
Revert a buffer See also: **∑ Diff & Merge**	- `<f11> f r f` - `⌘-u`	(**revert-buffer** &optional IGNORE-AUTO NOCONFIRM PRESERVE-MODES)	Replace current buffer text with the text of the visited file on disk. - This undoes all changes since the file was visited or saved. - With a prefix argument, offer to revert from latest auto-save file, if that is more recent than the visited file. - This is also the command to use to reload a file that was modified on the file system. ☝ You can use **ediff-current-file** to see the difference between the buffer and its disk file. PEL binding for this is `<f11> e b f`.
Toggle auto-revert mode	`<f11> f r a`	(**auto-revert-mode** &optional ARG)	Toggle reverting buffer when the file changes (Auto-Revert Mode). With a prefix argument ARG, enable Auto-Revert Mode if ARG is positive, and disable it otherwise. - Auto-Revert Mode is a minor mode that affects only the current buffer. When enabled, it reverts the buffer when the file on disk changes. - When a buffer is reverted, a message is generated. This can be suppressed by setting 'auto-revert-verbose' to nil.
Toggle auto-revert tail mode	`<f11> f r t`	(**auto-revert-tail-mode** &optional ARG)	Toggle reverting tail of buffer when the file grows. - With a prefix argument ARG, enable Auto-Revert Tail Mode if ARG is positive, and disable it otherwise. - When Auto-Revert Tail Mode is enabled, the tail of the file is constantly followed, as with the shell command 'tail -f'. This means that whenever the file grows on disk (presumably because some background process is appending to it from time to time), this is reflected in the current buffer. - You can edit the buffer and turn this mode off and on again as you please. But make sure the background process has stopped writing before you save the file!
Cancel/restart auto-revert timer	`<f11> f r SPC`	(**pel-auto-revert-set-timer**)	Restart or cancel the timer used by Auto-Revert Mode. - If such a timer is active, cancel it. - Start a new timer if Global Auto-Revert Mode is active or if Auto-Revert Mode is active in some buffer. - Restarting the timer ensures that Auto-Revert Mode will use an up-to-date value of '***auto-revert-interval***'(which is normally 5 seconds by default). 💻 : **pel-auto-revert-set-timer** is a thin wrapper over **auto-revert-set-timer** that displays a warning if executed when the buffer is not already in auto-revert-mode. It also displays the value of *auto-revert-interval* when **auto-revert-set-timer** is executed.
Saving Files	Use the following commands to save the content of a buffer to a filesystem file. PEL supports the following controllable actions on file save. Each of these actions are activated via an action-specific PEL user-option, and can temporarily be disabled with a command for the file in the current buffer. The actions and their associated user-option and command are listed here: **Action** — **Activating user-option** — **Overriding command** — **Key Sequence** - Delete trailing space and lines on save — pel-delete-trailing-whitespace — pel-toggle-delete-trailing-space-on-save — `<f11> M-W` - Update time stamp on save — pel-update-time-stamp — pel-toggle-update-time-stamp-on-save — `<f11> M-T` - Update copyright notice on save — pel-update-copyright — pel-toggle-update-copyright-on-save — `<f11> M-C`		
Save file to disk	- `C-x C-s` - `⌘-s`	(**save-buffer** &optional ARG)	Save current buffer to associated file. By default, it makes the previous version into a backup file if previously requested or if this is the first save. - With `C-u`: marks this version to become a backup when the next save is done - With `C-u C-u`: makes the previous version into a backup file - With `C-u C-u C-u:` marks this version to become a backup when the next save is done, and makes the previous version into a backup file. - With prefix 0: never make the previous version into a backup file. On macOS in graphics mode only: `⌘-s` brings a OS file-save dialog. ⚠️ Save and activated on-file-save actions only occur when the buffer is in "changed" status. Use `M-~` to flip that status to force an action when it has just been activated.
Save all/some files	`C-x s`	(**save-some-buffers** &optional ARG PRED)	Prompt for files that are modified. Options: - **y** : save - **n** : don't save - **C-r** : look at the buffer in question - **d** : view differences with diff-buffer-with-file
Write buffer to specified file ☞ **Save As**	`C-x C-w`	- (**write-file** FILENAME &optional CONFIRM) - (**ido-write-file**)	Similar to "Save-As": prompt for the filename. - Can also be yanked in the mini buffer, use `M-n` to edit it. 💡 Use that command to rename the file.
Changed current buffer changed state	`M-~`	(**not-modified** &optional ARG)	Mark current buffer as unmodified, not needing to be saved. - With `C-u` prefix ARG, mark buffer as modified, so `C-x C-s` will save.
Toggle copyright update on save	`<f11> M-@`	(**pel-toggle-update-copyright-on-save** &optional GLOBALLY)	Toggle copyright update on file save and display current state. - By default change behaviour for local buffer only. - When GLOBALLY argument is non-nil, using any prefix argument, change it for all buffers for the current Emacs editing session (the change does not persist across Emacs sessions). - To modify the global state permanently modify the customized value of the 'pel-update-copyright' user option via the 'pel-pkg-for-filemng' group customize buffer with **<f11> f <f2> 1**. 🛂 This command is only available when the **pel-update-copyright** is set to **t**.
Toggle timestamp update on save	`<f11> M-T`	(**pel-toggle-update-time-stamp-on-save** &optional GLOBALLY)	Toggle time-stamp update on file save and display current state. - By default change behaviour for local buffer only. - When GLOBALLY argument is non-nil, using any prefix argument, change it for all buffers for the current Emacs editing session (the change does not persist across Emacs sessions). - To modify the global state permanently modify the customized value of the 'pel-update-time-stamp' user option via the 'pel-pkg-for-filemng' group customize buffer with **<f11> f <f2> 1**. 🛂 This command is only available when the **pel-update-time-stamp** is set to **t**.

Operation	Keystroke	Function	Note
Toggle delete trailing space on save	`<f11> M-W`	(**pel-toggle-update-time-stamp-on-save** &optional GLOBALLY)	Toggle deletion of trailing spaces on file save and display current state. • By default change behaviour for local buffer only. • When GLOBALLY argument is non-nil, using any prefix argument, change it for all buffers for the current Emacs editing session (the change does not persist across Emacs sessions). • To modify the global state permanently modify the customized value of the 'pel-delete-trailing-whitespace' user option via the 'pel-pkg-for-filemng' group customize buffer with **<f11> f <f2> 1**. 🔧 This command is only available when the **pel-delete-trailing-whitespace** user-option s set to **t**.
Inserting & Automatically Updating Copyrights	Emacs has built-in support for insertion and update of copyright notices inside files. • Two commands, shown below, are provided to manually insert or update the file's copyright notice. • The copyright notice can be automatically updated by adding the **copyright-update** function to the list of **before-save-hook** variable with the following code: `(add-hook 'before-save-hook ‘copyright-update)` ⚠️ 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 at point See also: ∑ **Inserting Text**	`<f11> i C`	(**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	`M-x copyright-update`	(**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.
	⚠️ Even when used interactively copyright-update does not warn if there is no copyright in the current buffer to update. • It does not create a missing notice. ☝ If you want to be prompted automatically to update an existing but out-of-date copyright notice, write the following inside your init.el file: `(add-hook 'before-save-hook ‘copyright-update)`		
Automatic File Time Stamp on file save References: • **TimeStamps @ EmacsWiki** • **Change time stamp format in:** • **markdown file** • **reStucturedText file** See also: ∑ **Inserting Text**	Emacs has a built-in **automatic time-stamping of files.** It must be activated by adding the **time-stamp** function to the **before-save-hook** variable. This can either be done via Emacs customization system or explicitly inside your init file with the following code: `(add-hook 'before-save-hook ‘time-stamp)` • The time stamp will be added to files that contain, inside their first 8 lines, a line that looks like one of the following: • `Time-stamp: <>` • `Time-stamp: “ “` ☝ You can, however change these defaults and get Emacs to update all sorts of time stamp formats, even inside source code statements: 🎛 Emacs controls automatic insertion of timestamp with the following variables: • **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. • **time-stamp-start:** identifies the text pattern that precedes the time stamp. • **time-stamp-end**: identifies the end of the time stamp. • **time-stamp-format** specifies the format of the time stamp. • Something like `"%:y-%02m-%02d %02H:%02M:%02S %u"` to specify the date and time in ISO format, with the user login's name. • **time-stamp-time-zone** specifies the time zone selection: • nil: Emacs local time • t: Universal time • wall : system wall clock time • TZ : controlled by a TZ environment variable The **time-stamp-format** and **time-stamp-time-zone** variables can be set in your init file or via the Emacs customization system. • They are defined in the **time-stamp** customization group. • ☝ To change the format or the pattern preceding or after the automatically updated time stamp, it is best to use file local variables: this will allow automatic time stamp updates in files with various formats. As an example, see the top and end of the PEL manual raw format file. ⚠️ By default, the time-stamp string must be placed within the **first 8 lines** of the file, otherwise it will not be updated automatically. • If you want it located somewhere else in your file set the **time-stamp-line-limit** file local variable. 🎛 PEL provides the extra user-option to control the automatic generation of time-stamps: • **pel-update-time-stamp** user-option controls whether time-stamps are automatically update time stamps in all files where a valid time-stamp corresponding to Emacs settings as described above. Set it to **t** (the default) to allow automatic time stamp updates. Set it to nil to prevent them. You can also toggle it globally for the current editing session by using the `<f11> f M-t` key sequence. ☛ To insert a non-updatable time stamp, the PEL package provides a set of text insert commands which include inserting a time stamp . • See the ∑ **Inserting Text** table for the appropriate commands.		
Update file time stamp	`<f11> f t`	(**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 automatic update	`<f11> f M-t`	(**time-stamp-toggle-active** &optional ARG)	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.
Directory Tree Browsers	Emacs supports mechanisms to browse file directories. This includes: • Emacs built-in ∑M **Dired** directory editor, along with several extensions. You can have several different Dired buffers in an Emacs session. • The Emacs built-in ∑ **Speedbar** and its extensions. There can only be one instance of a Speedbar buffer and that can be inside another frame. • Several other external packages: **Neotree**, treemacs and **Ztree**		
View Directory Tree with NeoTree	📦 The NeoTree external package provides a Vim-NerdTree like tree-view of a directory with expansion/collapse. 🔧 PEL activates it when **pel-use-neotree** is set to **t**. • `<f11> B N <f2>` opens the PEL customization group to set **pel-use-neotree**. • `<f11> B N <f3>` prompts, select neotree to open the neotree customization group. ☛ There is only **one** NeoTree window. It is a **dedicated window**. ☛ Icons used in the tree can be changed: • In text mode set pel-neotree-font-in-terminal to arrows to use arrows instead of '+'. • In graphics mode, if pel-neotree-font-in-graphics is set to icons then the icons provided by **all-the-icons package** is used. ⚠️ However, once PEL has installed the package it does not install the fonts. You must install the fonts manually by executing: `M-x all-the-icons-install-fonts`		
View directory tree with NeoTree	`<f11> B N N`	(**neotree-toggle**)	Toggle show/hide the NeoTree window.
		In the NeoTree buffer the following keys are available: • `n` next line, `p` previous line. • `>` end of buffer, `<` top buffer • `SPC` or `RET` or `TAB` : Open current item if it is a file, Fold/Unfold current item if it is a directory. • `U` Go up a directory • `g` Refresh • `A` Maximize/Minimize the NeoTree Window • `H` Toggle display hidden files. Controlled by **neo-hidden-regexp-list** user option. • `O` Recursively open a directory • `C-c C-n` Create a file or create a directory if filename ends with a '/' • `C-c C-d` Delete a file or a directory. • `C-c C-r` Rename a file or a directory. • `C-c C-c` Change the root directory. • `C-c C-p` Copy a file or a directory.	

Operation	Keystroke	Function	Note
Open NeoTree for dir of current buffer	`<f11> B N F`	(**neotree-find** &optional PATH DEFAULT-PATH)	Open a NeoTree window using the directory of the current buffer. No prompt.
Open NeoTree for specified directory	`<f11> B N D`	(**neotree-dir** PATH)	Prompt for a directory. Open a Neotree window for that directory.
Close NeoTree window	`<f11> B N H`	(**neotree-hide**)	Close the NeoTree window.
Show NeoTree window	`<f11> B N S`	(**neotree-show**)	Show the NeoTree window.
View Directory Tree with ZTree	📦 The ztree external package provides a text-based tree-view of a directory with expansion/collapse. 🎛 PEL ztree customization: • `<f11> B <f2>` opens the PEL customization group (select the tree subgroup) . See also:**∑ Customize.** • ☑ PEL activates it when **pel-use-ztree** is set to **t**. • Modify one of the following PEL provided customization user options: • **pel-ztree-dir-move-focus** : set to **t** to move focus to new entry when <RET> is typed. • **pel-ztree-dir-filter-list** : add a list of regexp to ignore more file. Do not enter quote for string. For example, to ignore the .pyc files, enter **^.*pyc** on a line. • **pel-ztree-show-filtered-files** : set to **t** to display filtered files until **H** is typed. Normally they are not shown until **H** is typed. • `<f11> B <f3>` prompts, select ztree to open the ztree customization group itself. 1. Execute **`M-x pel-init`** after settling and applying new values to activate the new values.		
View directory as tree with ztree-dir	`<f11> B Z`	(**ztree-dir** PATH)	Open an interactive buffer with the directory tree of the PATH given. ☛ Opens the tree buffer in the current window. ☛ There can be several buffers with different ztree-dir trees.
		In the Ztree Dir buffer the following keys are available: • **>** : narrow/display directory on current line **<** : widen/display parent directory • **d** : Open Dired at point. • **H** : toggle display of filtered files. Controlled by regexp in the **ztree-dir-filter-list** user option. • **x** : Toggle expand/collapse of all nodes of the subtree. • ⚠ Use **x** with care! On large directory trees it takes a long time. I have see Emacs hang when typing **x** again during that time. 🚧 Investigate.	
Searching/Finding Files	The following commands can be used to search for file by name or content. Ref: Video: .Emacs #6 : searching and finding files.		
Run grep via find See also: **∑ Grep**	• `<f11> f g` • `<f11> g f`	(**find-grep** COMMAND-ARGS)	Run grep via find, with user-specified args COMMAND-ARGS. • Collect output in a buffer. • While find runs asynchronously, you can use the **`C-x `** ` command to find the text that grep hits refer to. • This command uses a special history list for its arguments, so you can easily repeat a find command.
Search for file with locate	`<f11> f L`	(**locate** SEARCH-STRING &optional FILTER ARG)	Prompt for a search pattern and search for filenames using the system **locate** command line utility through the sell to search a database of all pathnames that match the specified search pattern. The database is recomputed periodically. • The search result is shown in a '*Locate*' buffer. • With prefix arg ARG, prompt for the exact shell command to run instead. This way you can specify options to the locate command line utility. ☛ Use man to get more information on locate (`<f11> x m` locate)
Search for files with 'find' and open Dired buffer	`<f11> f d`	(**find-dired** DIR ARGS)	Prompts for the root to search from, and a **find** command to search for files with the Unix find. • Specify the arguments for the find command. For example, to perform a case insensitive search for all .h files: -iname "*\.h" for all .h files. • Opens a Dired-mode buffer and show the files found in there.
Search directory for files and open Dired buffer for those	`<f11> f n`	(**find-name-dired** DIR PATTERN)	Search DIR recursively for files matching the globbing pattern PATTERN, and run Dired on those files. PATTERN is a shell wildcard (not an Emacs regexp) and need not be quoted. The default command run (after changing into DIR) is: find . -name 'PATTERN' -ls
Find files in a directory and open Dired output	`<f11> f h`	(**find-grep-dired** DIR REGEXP)	Find files in DIR that contain matches for REGEXP and start Dired on output. The command run (after changing into DIR) is: find . \(-type f -exec 'grep-program' 'find-grep-options' -e REGEXP {} \; \) -ls where the first string in the value of the variable 'find-ls-option' specifies what to use in place of "-ls" as the final argument.
Find Emacs Lisp files in directory tree	`<f11> f l`	(**find-lisp-find-dired** DIR REGEXP)	Find Emacs Lisp files in DIR, matching REGEXP. Open *Find Lisp Dired* buffer on output.

File Management — References

Topic & Link	Description
Emacs Display - Mode Line	Read first. Describes what the Emacs mode line displays.
GNU Emacs Manual - File Handling	Describes how to open and deal with files and directories in Emacs.
GNU EMACS Manual - Interactive Do	Describes the ido-mode, a nice addition that helps with completing file names at prompts.
Display path of file in status bar	In graphics mode, display the buffer name and the full path file in parenthesis inside the frame title bar.
How do I rename an open file in Emacs?	