




























































File Management

Operation	Keystroke	Function	Note
File Management See also: <ul style="list-style-type: none"> » Dired » Customize 	Emacs provides a large set of commands to open files (Emacs documentation uses the term “ <i>finding</i> ” files for that), saving files searching for files or file content, displaying directory content, etc... These are listed in this table. <ul style="list-style-type: none"> The directory editing (dired) commands are mainly listed in the » Dired table. There are also several Emacs internal and external packages that provide useful commands. PEL supports several of them, listed below. <ul style="list-style-type: none"> Use Emacs customize system to modify their values to activate, deactivate and modify the behaviour of these packages. PEL <f1> f key prefix followed by either <f2> to access PEL activation group and <f3> to access the external package customization groups. Once you have modified the relevant user-option values, apply or save them and then either execute M-x pel-init or restart Emacs. PEL provides integration with the following Emacs built-in libraries or functionalities: <ul style="list-style-type: none"> Library ffap  activated by pel-use-ffap to provide several commands to open file at point. Library recentf  activated by pel-use-recentf to list files recently opened. Automatic file time stamp update on file save  activated by pel-update-time-stamp. Automatic update of copyright notice year on file save  activated by pel-update-copyright. It also provides integration with the following external packages when the corresponding PEL user-option is activated: <ul style="list-style-type: none">  key-chord  activated by pel-use-key-chord, provides convenient key-chords for some commands.  rfc-mode  activated by pel-use-rfc-mode, provides ability to download and browse IETF RFC documents easily (see RFC editor).  ivy/counsel  activated by pel-use-counsel provides completion for some file commands. PEL supports more. See » Completion/Input.  NeoTree  activated by pel-use-neotree provides an alternative to » Dired to navigate a file directory.  treemacs  activated by pel-use-treemacs provides project-oriented file directory navigation.  ztree  activated by pel-use-ztree an other alternative to » Dired to navigate a file directory. 		
Open this PDF file. See also: » Help/Info	<f11> f <f1> 1	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the » File-mngt 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 File/Directory Management	<f11> f <f2> 1	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL support for file management. <ul style="list-style-type: none"> If OTHER-WINDOW is non-nil (use C-u), display in other window.
» Customize Emacs file management support	<f11> f <f3>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs support for file management. Includes the following: files, recentf, popup-switcher.
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: » 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. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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: <ol style="list-style-type: none"> steal the lock (with 's'), proceed ('p') to edit the file anyway or quit ('q'). 		
Open 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: <ul style="list-style-type: none"> » Completion/Input » Dired » Customize 	<ul style="list-style-type: none"> <f11> f f <M-f11> M-f M-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. <ul style="list-style-type: none"> With PEL, the <f11> f f and <M-f11> M-f M-f key bindings are 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.  Note that <M-f12> M-f M-f is also available in some major modes to open files in a way that takes the major mode into account, like providing a list of files in the project. Refer to major mode pages for specific information.
 <ul style="list-style-type: none"> Prevent Ido expansion with C-j  Open in read-only: 	C-x C-f		
		(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.
	<ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 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. Also useful to open a directory in dired mode.  To open a file in read-only mode you can: <ul style="list-style-type: none"> 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 file via popup menu	<f11> f M-f	(pel-psw-navigate-files)	Open file from a pop-up menu listing files in current directory. Uses (psw-navigate-files “.”) . <ul style="list-style-type: none"> Narrow menu list by typing part of the file name. You can also select directory names.  Requires popup-switcher  PEL activates when pel-use-popup-switcher is t.
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	<ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> Like C-x C-f, but creates a new frame or reuses an existing one.

Operation	Keystroke	Function	Note																
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	<ul style="list-style-type: none"> C-x i <f11> f i 	<ul style="list-style-type: none"> (insert-file FILENAME) (ido-insert-file) 	Insert contents of file FILENAME into buffer after point. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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.																		
Write region text to file	<f11> f w	(write-region START END FILENAME &optional APPEND VISIT LOCKNAME MUSTBENEV)	Write current region into specified file. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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). <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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 <i>revert-without-query</i> 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	<ul style="list-style-type: none"> <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. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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 <i>auto-revert-interval</i> 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: <table> <tr> <th>Action</th><th>Activating user-option</th><th>Overriding command</th><th>Key Sequence</th></tr> <tr> <td>• Delete trailing space and lines on save</td><td>pel-delete-trailing-whitespace</td><td>pel-toggle-delete-trailing-space-on-save</td><td><f11> M-W</td></tr> <tr> <td>• Update time stamp on save</td><td>pel-update-time-stamp</td><td>pel-toggle-update-time-stamp-on-save</td><td><f11> M-T</td></tr> <tr> <td>• Update copyright notice on save</td><td>pel-update-copyright</td><td>pel-toggle-update-copyright-on-save</td><td><f11> M-C</td></tr> </table>			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
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	<ul style="list-style-type: none"> C-x C-s ⌘-s 	(save-buffer &optional ARG)	Save current buffer to associated file. By default, it makes the previous version into a <u>backup file</u> if previously requested or if this is the first save. <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> (write-file FILENAME &optional CONFIRM) (ido-write-file) 	Similar to "Save-As": prompt for the filename. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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 .																

Operation	Keystroke	Function	Note
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. <ul style="list-style-type: none"> 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 .
Toggle delete trailing space on save See also: ⌘ Whitespace	<ul style="list-style-type: none"> <f11> M-W <f11> t w M-W 	(pel-toggle-delete-trailing-space-on-save &optional GLOBALLY)	Toggle deletion of trailing spaces on file save and display current state. <ul style="list-style-type: none"> 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).  Trailing whitespace deletion is automatically activated on file save when the pel-delete-trailing-whitespace user-option is set to t . Use this command to de-activate it or re-activate it. <ul style="list-style-type: none"> 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.
Inserting & Automatically Updating Copyrights	Emacs has built-in support for insertion and update of copyright notices inside files. <ul style="list-style-type: none"> 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: <pre>(add-hook 'before-save-hook 'copyright-update)</pre>  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. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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: <pre>(add-hook 'before-save-hook 'copyright-update)</pre>
Automatic File Time Stamp on file save References: <ul style="list-style-type: none"> TimeStamps @ EmacsWiki Change time stamp format in: <ul style="list-style-type: none"> markdown file reStructuredText 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: <pre>(add-hook 'before-save-hook 'time-stamp)</pre> <ul style="list-style-type: none"> The time stamp will be added to files that contain, inside their first 8 lines, a line that looks like one of the following: <ul style="list-style-type: none"> 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: <ul style="list-style-type: none">  Emacs controls automatic insertion of timestamp with the following variables: <ul style="list-style-type: none"> time-stamp-pattern consists of 4 parts, each one controlled by a variable: <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 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 . <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> Updates a time stamp of format recognized by <i>Emacs current settings</i> even when automatic time-stamp update is off. More information about the “<i>Emacs current settings</i>” 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. <ul style="list-style-type: none"> With ARG, turn time stamping on if and only if arg is positive.
RFC-Mode	Browsing and reading RFC Files with the following rfc-mode commands.  Requires rfc-mode  activated by pel-use-rfc-mode ,		
Read a specific RFC	<f11> B r	(rfc-mode-read NUMBER)	Read the RFC document NUMBER. Offer the number at point as default.
Browse RFCs	<f11> B R	(rfc-mode-browse)	Browse through all RFC documents referenced in the index.

Operation	Keystroke	Function	Note
Directory Tree Browsers	<p>Emacs supports mechanisms to browse file directories. This includes:</p> <ul style="list-style-type: none">Emacs built-in ⌘ 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	<p> The NeoTree external package provides a Vim-NerdTree like tree-view of a directory with expansion/collapse.</p> <p> PEL activates it when pel-use-neotree is set to t.</p> <ul style="list-style-type: none"><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. <p> There is only one NeoTree window. It is a dedicated window.</p> <p> Icons used in the tree can be changed:</p> <ul style="list-style-type: none">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. <p> 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</p>		
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:	
		<ul style="list-style-type: none">n next line, p previous line.> end of buffer, < top bufferSPC 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 directoryg RefreshA Maximize/Minimize the NeoTree WindowH Toggle display hidden files. Controlled by neo-hidden-regexp-list user option.O Recursively open a directoryC-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.	
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.
Treemacs <ul style="list-style-type: none">Manipulate directory trees associated as projects/workspacesManipulate the directories and files ★★ See: ⌘ Treemacs	<p> The treemacs external package provides a workspace/project oriented tree-based view with expansion/collapse and actions of directories and files.</p> <p> PEL activates treemacs when the pel-use-treemacs user-option is turned on (set to t).</p> <p> Treemacs has a large number of user-options in the treemacs customization group and sub-groups.</p> <p>PEL <f11> B <f3> key sequence gives access to the customization group.</p> <p>On PEL, open (or close) the treemacs buffer with the <f11> B T key sequence.</p> <ul style="list-style-type: none">In graphics mode the mouse provides access to most commands.In terminal (and graphics) mode when pain is inside the treemacs dedicated window, the treemacs major mode key-bindings, listed below, are available. <p>The treemacs-mode and extensions have an extensive command set. See ⌘ Treemacs for the complete list</p>		
Open/close treemacs	<f11> B T	(treemacs)	Initialise or toggle treemacs. See ⌘ Treemacs for treemacs-mode commands. <ul style="list-style-type: none">If the treemacs window is visible hide it.If a treemacs buffer exists, but is not visible show it.If no treemacs buffer exists for the current frame create and show it.If the workspace is empty additionally ask for the root path of the first project to add.
View Directory Tree with ZTree	<p> The ztree external package provides a text-based tree-view of a directory with expansion/collapse.</p> <p> PEL ztree customization:</p> <ul style="list-style-type: none"><f11> B <f2> opens the PEL customization group (select the tree subgroup) . See also:⌘ Customize.<ul style="list-style-type: none"> PEL activates it when pel-use-ztree is set to t.Modify one of the following PEL provided customization user options:<ul style="list-style-type: none">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. <p>1. Execute M-x pel-init after settling and applying new values to activate the new values.</p>		
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. <ul style="list-style-type: none"> 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:	
		<ul style="list-style-type: none">> : narrow/display directory on current line < : widen/display parent directoryd : 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.<ul style="list-style-type: none"> 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 See also: <ul style="list-style-type: none">⌘ Help/Info⌘ Dired	<p>The following commands can be used to search for file by name or content.</p> <ul style="list-style-type: none">See: Video: .Emacs #6 : searching and finding files. Use man to get more information,<ul style="list-style-type: none">on locate: <f11> ? m locateon find: <f11> ? m find You can manipulate the result in Dired with Dired commands. For instance type (to toggle the display of more than the file names.		
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. <ul style="list-style-type: none">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.
Run grep via find See also: ⌘ Grep	<ul style="list-style-type: none"><f11> f g<f11> g f	(find-grep COMMAND-ARGS)	Run grep via find, with user-specified args COMMAND-ARGS. <ul style="list-style-type: none">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.

Operation	Keystroke	Function	Note
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. <ul style="list-style-type: none"> Specify the arguments for the <u>find command</u>. <ul style="list-style-type: none"> For example, to perform a case insensitive search for all .h files, use: <code>-iname “*\.h”</code> 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. <ul style="list-style-type: none"> PATTERN is a shell wildcard (not an Emacs regexp) and need not be quoted. The default command run (after changing into DIR) is: <pre>find . -name 'PATTERN' -ls</pre>
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. <p>The command run (after changing into DIR) is:</p> <pre>find . \(-type f -exec 'grep-program' 'find-grep-options' -e REGEXP {} \; \) -ls</pre> <p>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.</p>
Find Emacs Lisp files in directory tree	<f11> f l	(find-lisp-find-dired DIR REGEXP)	Find Emacs Lisp files in DIR, matching REGEXP. <ul style="list-style-type: none"> 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?	