









File Management

Operation	Keystroke	Function	Note
<b>File Management</b> See also: <ul style="list-style-type: none"><li>⌘M <b>Dired</b></li><li>⌘ <b>Customize</b></li></ul>	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. <ul style="list-style-type: none"><li>The directory editing (dired) commands are mainly listed in the ⌘M <b>Dired</b> table.</li><li>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 &lt;f11&gt; &lt;1&gt; 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.</li></ul>		
Open this PDF file. See also: ⌘ <b>Help/Info</b>	<f11> f <f1> 1	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the local copy of the ⌘ <b>File-mngt</b> 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:⌘ <b>Customize</b>	<f11> f <f2> 1	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL support for file management. <ul style="list-style-type: none"><li>If OTHER-WINDOW is non-nil (use C-u) , display in other window.</li></ul>
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).
<b>Opening file</b>	The following commands are available to open/visit files in Emacs buffers. <ul style="list-style-type: none"><li>For some of them the corresponding ido mode function is also shown.</li><li><b>Note:</b> Emacs uses the word “visiting” instead of “opening” files.</li><li>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.</li><li>👉 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.</li><li>See the ⌘ <b>Customize</b> table for more info.</li></ul>		
<b>Open (visit) a file/directory</b>  See also: <ul style="list-style-type: none"><li>⌘M <b>Dired</b></li><li>⌘ <b>Customize</b></li></ul>	C-x C-f	<ul style="list-style-type: none"><li>(find-file FILENAME &amp;optional WILDCARDS)</li><li>-----</li><li>(ido-find-file)</li></ul>	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. This can be replaced by the ido-mode by the ido-find-file: it provides suggestions.  When ido mode is used, you can also: <ul style="list-style-type: none"><li>Type C-x f to change to original find-file</li><li>Type C-j to accept the file/directory name verbatim without replacement or suggestion.</li></ul> Note: it is also possible to change the read-only state of a buffer with C-x C-q. So you can open a file with C-x C-f and then change the buffer to read-only mode.  🔑📦📖 PEL supports several completion modes can all be activated and dynamically selected. See the ⌘ <b>Customize</b> table for more info.
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 currently file visited in current buffer with the default OS application. See also: <ul style="list-style-type: none"><li>⌘M <b>Dired</b></li></ul>	<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"><li>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.</li></ul>
Open another file in buffer	C-x C-v	<ul style="list-style-type: none"><li>(find-alternate-file FILENAME &amp;optional WILDCARDS)</li><li>(ido-find-alternate-file)</li></ul>	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"><li>C-x 4 f</li><li>&lt;f11&gt; f o</li></ul>	<ul style="list-style-type: none"><li>(find-file-other-window FILENAME &amp;optional WILDCARDS)</li><li>(ido-find-file-other-window)</li></ul>	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	<ul style="list-style-type: none"><li>(find-file-other-frame FILENAME &amp;optional WILDCARDS)</li><li>(ido-find-file-other-frame)</li></ul>	Edit file FILENAME, in another frame.  Like C-x C-f, but creates a new frame or reuses an existing one.
Open a file in read-only mode	C-x C-r	<ul style="list-style-type: none"><li>(find-file-read-only FILENAME &amp;optional WILDCARDS)</li><li>(ido-find-file-read-only)</li></ul>	Edit file 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.
Open file in other window in read-only mode	<ul style="list-style-type: none"><li>C-x 4 r</li><li>&lt;f11&gt; f o</li></ul>	<ul style="list-style-type: none"><li>(find-file-read-only-other-window FILENAME &amp;optional WILDCARDS)</li><li>(ido-find-file-read-only-other-window)</li></ul>	(find-file-read-only-other-window FILENAME &optional WILDCARDS) Edit file FILENAME in another window but don’t allow changes. Like C-x 4 C-f, but marks buffer as read-only. Use C-x C-q to permit editing.
Open a file in hexl-mode  See also: ⌘ <b>Buffers</b>	<f11> f M-x	(hexl-find-file FILENAME)	Edit file FILENAME as a binary file in hex dump format. <ul style="list-style-type: none"><li>Switch to a buffer visiting file FILENAME, creating one if none exists, and edit the file in ‘hexl-mode’.</li></ul>
Open filename at point in a browser See also: <ul style="list-style-type: none"><li>⌘ <b>Key-Chords</b></li><li>⌘ <b>Web</b></li></ul>	<ul style="list-style-type: none"><li>&lt;f11&gt; f /</li><li>6u</li></ul>	(pel-browse-filename-at-point)	Open the file name at point inside the system’s browser. <ul style="list-style-type: none"><li>If point is at a directory name, open the systems application that browses directories (eg. macOS Finder, Windows Explorer).</li></ul> 👉 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: <ul style="list-style-type: none"><li>⌘ <b>Key-Chords</b></li><li>⌘ <b>Web</b></li><li>⌘ <b>Customize</b></li></ul>	<ul style="list-style-type: none"><li>&lt;f11&gt; f M-/</li><li>7u</li></ul>	(browse-url-at-point &optional ARG)	Ask a WWW browser to load the URL at or before point. <ul style="list-style-type: none"><li>Variable ‘browse-url-browser-function’ says which browser to use.</li><li>Optional prefix argument ARG non-nil inverts the value of the option ‘browse-url-new-window-flag’.</li></ul> 👉 PEL provides the <f11> <f2> M-g u key sequence to open the browse-url group that contains relevant the user options.



Operation	Keystroke	Function	Note
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. <ul style="list-style-type: none"> <li>Then set mark, jump to choice, and try to fetch it. The menu is cached in ‘ffap-menu-alist’, and rebuilt by ‘ffap-menu-rescan’.</li> <li>The optional RESCAN argument (a prefix, interactively) forces a rebuild. Searches with ‘ffap-menu-regexp’.</li> </ul>
File Lock Protection	Emacs protects against multiple process modifying the same file with a lock. If you attempt to edit the buffer of a locked file, Emacs will prompt. You can: <ul style="list-style-type: none"> <li>steal the lock (with ‘s’),</li> <li>proceed (‘p’) to edit the file anyway or</li> <li>quit (‘q’).</li> </ul>		
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"> <li>C-x i</li> <li>&lt;f11&gt; f i</li> </ul>	<ul style="list-style-type: none"> <li>(insert-file FILENAME)</li> <li>(ido-insert-file)</li> </ul>	Insert contents of file FILENAME into buffer after point. <ul style="list-style-type: none"> <li>Set mark after the inserted text.</li> </ul>
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"> <li>Set mark after the inserted text.</li> </ul>
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 MUSTBENEW)	Write current region into specified file. <ul style="list-style-type: none"> <li>Prompts for the specified file.</li> </ul>
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"> <li>Prompts for the specified file.</li> </ul>
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"> <li>Only the 12 low bits of MODE are used.</li> <li>Prompts for file name and then for chmod-like file mode value.</li> </ul>
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"> <li>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.</li> <li>You can also activate the auto-revert mode for the current buffer or globally and restart its timer.</li> </ul>		
Revert a buffer  See also:  Diff & Merge	<ul style="list-style-type: none"> <li>&lt;f11&gt; f r f</li> <li>⌘-u</li> </ul>	(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"> <li>This undoes all changes since the file was visited or saved.</li> <li>With a prefix argument, offer to revert from latest auto-save file, if that is more recent than the visited file.</li> <li>This is also the command to use to reload a file that was modified on the file system.</li> </ul> You can use <b>ediff-current-file</b> 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"> <li>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.</li> <li>When a buffer is reverted, a message is generated. This can be suppressed by setting ‘auto-revert-verbose’ to nil.</li> </ul>
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"> <li>With a prefix argument ARG, enable Auto-Revert Tail Mode if ARG is positive, and disable it otherwise.</li> <li>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.</li> <li>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!</li> </ul>
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"> <li>If such a timer is active, cancel it.</li> <li>Start a new timer if Global Auto-Revert Mode is active or if Auto-Revert Mode is active in some buffer.</li> <li>Restarting the timer ensures that Auto-Revert Mode will use an up-to-date value of ‘<i>auto-revert-interval</i>’ (which is normally 5 seconds by default).</li> </ul> : <b>pel-auto-revert-set-timer</b> is a thin wrapper over <b>auto-revert-set-timer</b> 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 <b>auto-revert-set-timer</b> is executed.
Saving Files	Use the following commands to save the content of a buffer to a filesystem file.		
Save file to disk	<ul style="list-style-type: none"> <li>C-x C-s</li> <li>⌘-s</li> </ul>	(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"> <li>With <b>C-u</b>: marks this version to become a backup when the next save is done</li> <li>With <b>C-u C-u</b>: makes the previous version into a backup file</li> <li>With <b>C-u C-u C-u</b>: marks this version to become a backup when the next save is done, and makes the previous version into a backup file.</li> <li>With prefix 0: never make the previous version into a backup file.</li> </ul> On macOS in graphics mode only: <b>⌘-s</b> brings a OS file-save dialog.
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"> <li>y : save</li> <li>n : don’t save</li> <li>C-r : look at the buffer in question</li> <li>d : view differences with diff-buffer-with-file</li> </ul>
Write buffer to specified file	C-x C-w	<ul style="list-style-type: none"> <li>(write-file FILENAME &amp;optional CONFIRM)</li> <li>(ido-write-file)</li> </ul>	Similar to “Save-As”: prompt for the filename. <ul style="list-style-type: none"> <li>Can also be yanked in the mini buffer, use <b>M-n</b> to edit it.</li> </ul> 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"> <li>With <b>C-u</b> prefix ARG, mark buffer as modified, so <b>C-x C-s</b> will save.</li> </ul>

Operation	Keystroke	Function	Note
Automatic File Time Stamp	<p>Emacs has a built-in automatic time-stamping of files. It must be activated by adding the <b>time-stamp</b> function to the <b>before-save-hook</b> variable. This can either be done via Emacs customization system or explicitly inside your init file with the following code:</p> <pre>(add-hook 'before-save-hook 'time-stamp)</pre> <ul style="list-style-type: none"> <li>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"> <li>Time-stamp: &lt;&gt;</li> <li>Time-stamp: " "</li> </ul> </li> </ul> <div>  The format of the time stamp is controlled by several variables: <ul style="list-style-type: none"> <li><b>time-stamp-format</b> 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.</li> <li><b>time-stamp-time-zone</b> specifies the time zone selection: <ul style="list-style-type: none"> <li>nil: Emacs local time</li> <li>t: Universal time</li> <li>wall : system wall clock time</li> <li>TZ : controlled by a TZ environment variable</li> </ul> </li> </ul> <ul style="list-style-type: none"> <li>These variables can be set in your init file or via the Emacs customization system.</li> </ul> <p>⚠ To be automatically updated, the time-stamp string must be placed within the first 8 lines of the file.</p> <p>➡ 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.</p> </div>		
Update file time stamp  See also: <a href="#">⌘ Inserting Text</a>	<f11> f t	(time-stamp)	<p>Force update the time stamp string(s) in the current buffer.</p> <p>The time stamp is updated if the one of the following strings is found in the first 8 lines of the file:</p> <ul style="list-style-type: none"> <li>Time-stamp: &lt;&gt;</li> <li>Time-stamp: " "</li> </ul> <p>👉 If you want time stamps updated automatically, write the following inside your init.el file:</p> <pre>(add-hook 'before-save-hook 'time-stamp)</pre>
Toggle time stamp automatic update		(time-stamp-toggle-active &optional ARG)	<p>Toggle 'time-stamp-active', setting whether &lt;f11&gt; f t updates a buffer.</p> <ul style="list-style-type: none"> <li>With ARG, turn time stamping on if and only if arg is positive.</li> </ul>
Inserting & Automatically Updating Copyrights	<p>Emacs has built-in support for insertion and update of copyright notices inside files.</p> <ul style="list-style-type: none"> <li>Two commands, shown below, are provided to manually insert or update the file's copyright notice.</li> <li>The copyright notice can be automatically updated by adding the <b>copyright-update</b> function to the list of <b>before-save-hook</b> variable with the following code:</li> </ul> <pre>(add-hook 'before-save-hook 'copyright-update)</pre> <p>⚠ 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 <b>copyright-limit</b> variable, normally defined as the first 2000 characters. This variable is customizable.</p>		
Insert copyright notice at point See also: <a href="#">⌘ Inserting Text</a>	<f11> i C	(copyright &optional STR ARG)	<p>Insert a copyright by \$ORGANIZATION notice at cursor.</p> <ul style="list-style-type: none"> <li>If the ORGANIZATION environment variable is not available, Emacs prompts for it.</li> </ul>
Update file's copyright notice		(copyright-update &optional ARG INTERACTIVEP)	<p>Update copyright notice to indicate the current year.</p> <ul style="list-style-type: none"> <li>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.</li> <li>If non-nil, INTERACTIVEP tells the function to behave as when it's called interactively.</li> </ul> <p>⚠ 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.</p> <p>👉 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:</p> <pre>(add-hook 'before-save-hook 'copyright-update)</pre>
Directory Tree Browsers	<p>Emacs supports mechanisms to browse file directories. This includes:</p> <ul style="list-style-type: none"> <li>Emacs built-in <a href="#">⌘ M Dired</a> directory editor, along with several extensions. You can have several different Dired buffers in an Emacs session.</li> <li>The Emacs built-in <a href="#">⌘ Speedbar</a> and its extensions. There can only be one instance of a Speedbar buffer and that can be inside another frame.</li> <li>Several other external packages: <ul style="list-style-type: none"> <li></li> </ul> </li> </ul>		
View Directory Tree with NeoTree	<div>  The <a href="#">NeoTree external package</a> provides a Vim-NerdTree like tree-view of a directory with expansion/collapse. </div> <div>  PEL activates it when <b>pel-use-neotree</b> is set to t. </div> <div>  <ul style="list-style-type: none"> <li>&lt;f11&gt; B &lt;f2&gt; opens the PEL customization group to set <b>pel-use-neotree</b>.</li> <li>&lt;f11&gt; B &lt;f3&gt; prompts, select neotree to open the neotree customization group.</li> </ul> </div>		
View directory tree with NeoTree	<f11> B N	(neotree-toggle)	<p>Toggle show the NeoTree window.</p> <p>➡ There is only <b>one</b> NeoTree window. It is a dedicated window.</p> <p>In the NeoTree buffer the following keys are available:</p> <ul style="list-style-type: none"> <li><b>n</b> next line</li> <li><b>p</b> previous line.</li> <li><b>SPC</b> or <b>RET</b> or <b>TAB</b> : Open current item if it is a file. <ul style="list-style-type: none"> <li>Fold/Unfold current item if it is a directory.</li> </ul> </li> <li><b>U</b> Go up a directory</li> <li><b>g</b> Refresh</li> <li><b>A</b> Maximize/Minimize the NeoTree Window</li> <li><b>H</b> Toggle display hidden files</li> <li><b>O</b> Recursively open a directory</li> <li><b>C-c C-n</b> Create a file or create a directory if filename ends with a '/'</li> <li><b>C-c C-d</b> Delete a file or a directory.</li> <li><b>C-c C-r</b> Rename a file or a directory.</li> <li><b>C-c C-c</b> Change the root directory.</li> <li><b>C-c C-p</b> Copy a file or a directory.</li> </ul>
View Directory Tree with ZTree	<div>  The <a href="#">ztree external package</a> provides a text-based tree-view of a directory with expansion/collapse. </div> <div>  PEL ztree customization: <ul style="list-style-type: none"> <li>&lt;f11&gt; B &lt;f2&gt; opens the PEL customization group (select the tree subgroup) . See also:<a href="#">⌘ Customize</a>. <ul style="list-style-type: none"> <li> PEL activates it when <b>pel-use-ztree</b> is set to t.</li> </ul> </li> <li>Modify one of the following PEL provided customization user options: <ul style="list-style-type: none"> <li><b>pel-ztree-dir-move-focus</b> : set to t to move focus to new entry when &lt;RET&gt; is typed.</li> <li><b>pel-ztree-dir-filter-list</b> : add a list of regexp to ignore more file. Do not enter quote for string. For example, to ignore the .pyc files, enter <b>^.*pyc</b> on a line.</li> <li><b>pel-ztree-show-filtered-files</b> : set to t to display filtered files until <b>H</b> is typed. Normally they are not shown until <b>H</b> is typed.</li> </ul> </li> <li>&lt;f11&gt; B &lt;f3&gt; prompts, select ztree to open the ztree customization group itself.</li> </ul> <p>1. Execute <b>M-x pel-init</b> after settling and applying new values to activate the new values.</p> </div>		
View directory as tree with ztree-dir	<f11> B Z	(ztree-dir PATH)	<p>Open an interactive buffer with the directory tree of the PATH given.</p> <p>➡ Opens the tree buffer in the current window.</p> <p>➡ There can be several buffers with different ztree-dir trees.</p>

Operation	Keystroke	Function	Note
		In the Ztree Dir buffer the following keys are available: <ul style="list-style-type: none"><li>• <b>H</b> : toggle display of filtered files.</li><li>• <b>&gt;</b> : narrow/display directory on current line</li><li>• <b>&lt;</b> : widen/display parent directory</li><li>• <b>d</b> : Open Dired at point.</li><li>• <b>x</b> : Toggle expand/collapse of all nodes of the subtree.<ul style="list-style-type: none"><li>• ⚠ Use <b>x</b> with care! On large directory trees it takes a long time. I have see Emacs hang when typing <b>x</b> again during that time. 🐛 Investigate.</li></ul></li></ul>	
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.		
Open a directory editor  See also: <a href="#">M Dired</a>	<ul style="list-style-type: none"><li>• <b>C-x d</b></li><li>• <b>%-D</b></li></ul>	<ul style="list-style-type: none"><li>• (dired DIRNAME &amp;optional SWITCHES)</li><li>_____</li><li>• (ido-dired)</li></ul>	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.
Run Dired in other (next) window	<b>C-x 4 d</b>	(dired-other-window)	Opens a Dired-mode buffer on the specified directory inside another window. Prompt for the directory name.
List Directory	<b>C-x C-d</b>	(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 ( <b>C-u</b> ) means supply -l switch to ‘ls’.
Searching/Finding Files	The following commands can be used to search for file by name or content. Ref: <a href="#">Video: .Emacs #6 : searching and finding files.</a>		
Run grep via find  See also: <a href="#">Grep</a>	<ul style="list-style-type: none"><li>• <b>&lt;f11&gt; f g</b></li><li>• <b>&lt;f11&gt; g f</b></li></ul>	(find-grep COMMAND-ARGS)	Run grep via find, with user-specified args COMMAND-ARGS. <ul style="list-style-type: none"><li>• Collect output in a buffer.</li><li>• While find runs asynchronously, you can use the <b>C-x `</b> command to find the text that grep hits refer to.</li><li>• This command uses a special history list for its arguments, so you can easily repeat a find command.</li></ul>
Search for file with locate	<b>&lt;f11&gt; f L</b>	(locate SEARCH-STRING &optional FILTER ARG)	Prompt for a search pattern and search for filenames using the system <b>locate</b> 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"><li>• The search result is shown in a “Locate” buffer.</li><li>• 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.</li></ul> ➡ Use man to get more information on locate ( <b>&lt;f11&gt; x m locate</b> )
Search for files with ‘find’ and open Dired buffer	<b>&lt;f11&gt; f d</b>	(find-dired DIR ARGS)	Prompts for the root to search from, and a <b>find</b> command to search for files with the Unix find. <ul style="list-style-type: none"><li>• Specify the arguments for the <a href="#">find command</a>. For example, to perform a case insensitive search for all .h files: -iname “*.h” for all .h files.</li><li>• Opens a Dired-mode buffer and show the files found in there.</li></ul>
Search directory for files and open Dired buffer for those	<b>&lt;f11&gt; f n</b>	(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	<b>&lt;f11&gt; f h</b>	(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	<b>&lt;f11&gt; f l</b>	(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
<a href="#">Emacs Display - Mode Line</a>	Read first. Describes what the Emacs mode line displays.
<a href="#">GNU Emacs Manual - File Handling</a>	Describes how to open and deal with files and directories in Emacs.
<a href="#">GNU EMACS Manual - Interactive Do</a>	Describes the ido-mode, a nice addition that helps with completing file names at prompts.
<a href="#">Display path of file in status bar</a>	In graphics mode, display the buffer name and the full path file in parenthesis inside the frame title bar.
<a href="#">How do I rename an open file in Emacs?</a>	