

# File Management

Operation	Keystroke	Function	Note
<u>Opening file</u>	<div>The following commands are available to open/visit files in Emacs buffers.</div> <div><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></ul></div>		
<b>Open (visit) a file/directory</b>  (See also:  Mode - Dired)	<b>C-x C-f</b>	<div><ul style="list-style-type: none"><li>(find-file FILENAME &amp;optional WILDCARDS)</li></ul></div> <div>-----</div> <div><ul style="list-style-type: none"><li>(ido-find-file)</li></ul></div>	<div>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.</div> <div>When <b>ido</b> mode is used, you can also:</div> <div><ul style="list-style-type: none"><li>Type <b>C-x f</b> to change to original find-file</li><li>Type <b>C-j</b> to accept the file/directory name verbatim without replacement or suggestion.</li></ul></div> <div>Note: it is also possible to change the read-only state of a buffer with <b>C-x C-q</b>. So you can open a file with <b>C-x C-f</b> and then change the buffer to read-only mode.</div> <div> PEL activates ido when the <b>pel-use-ido-mode</b> customize variable is set to <b>t</b>.</div>
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.
<u>Open another file in buffer</u>	<b>C-x C-v</b>	<div><ul style="list-style-type: none"><li>(find-alternate-file FILENAME &amp;optional WILDCARDS)</li><li>(ido-find-alternate-file)</li></ul></div>	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.
<u>Open file in other window</u>	<div><ul style="list-style-type: none"><li><b>C-x 4 f</b></li><li><b>&lt;f11&gt; f o</b></li></ul></div>	<div><ul style="list-style-type: none"><li>(find-file-other-window FILENAME &amp;optional WILDCARDS)</li><li>(ido-find-file-other-window)</li></ul></div>	<div>Edit file FILENAME, in another window.</div> <div>Like <b>C-x C-f</b>, but creates a new window or reuses an existing one.</div>
<u>Open file in other frame</u>	<b>C-x 5 f</b>	<div><ul style="list-style-type: none"><li>(find-file-other-frame FILENAME &amp;optional WILDCARDS)</li><li>(ido-find-file-other-frame)</li></ul></div>	<div>Edit file FILENAME, in another frame.</div> <div>Like <b>C-x C-f</b>, but creates a new frame or reuses an existing one.</div>
<u>Open a file in read-only mode</u>	<b>C-x C-r</b>	<div><ul style="list-style-type: none"><li>(find-file-read-only FILENAME &amp;optional WILDCARDS)</li><li>(ido-find-file-read-only)</li></ul></div>	<div>Edit file FILENAME but don't allow changes.</div> <div>Like <b>C-x C-f</b>, but marks buffer as read-only. Use <b>C-x C-q</b> to permit editing.</div>
Open file in other window in read-only mode	<div><ul style="list-style-type: none"><li><b>C-x 4 r</b></li><li><b>&lt;f11&gt; f r</b></li></ul></div>	<div><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></div>	<div>(find-file-read-only-other-window FILENAME &amp;optional WILDCARDS)</div> <div>Edit file FILENAME in another window but don't allow changes.</div> <div>Like <b>C-x 4 C-f</b>, but marks buffer as read-only. Use <b>C-x C-q</b> to permit editing.</div>
<u>Insert other file into the current buffer after cursor</u>	<b>C-x i</b>	<div><ul style="list-style-type: none"><li>(insert-file FILENAME)</li><li>(ido-insert-file)</li></ul></div>	Inserts the text of another file after point. Set mark after the inserted text.
<b>Open file or web-page whose name is at point</b>  ★★	<b>C-^</b>	(pel-find-file-at-point-in-window &optional N)	<div>Open the file, library or the URL, named at point.</div> <div><ul style="list-style-type: none"><li>If the string identifies a URL, the function opens the page in the default browser.</li><li>If the string identifies a file name, the file is opened in Emacs in the window identified by the N argument. 8: up, 2: down, 4:left, 5:current, 6:right, 0: other, negative: new. Selecting Minibuffer is not allowed.</li><li>If the file is not found, the function prompts. If the name corresponds to an Emacs library file, you can type <b>1</b> to open the library. You can also edit the file name collected before attempting to open it again. Or quit.</li><li>If the file name is followed by line and column numbers the point is moved to that position.</li></ul></div> <div>More information available in the command's help docstring.</div>
<b>Run grep via find</b>  (See also:  grep)	<div><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></div>	(find-grep COMMAND-ARGS)	<div>Run grep via find, with user-specified args COMMAND-ARGS.</div> <div><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></div>
<u>Reverting Files</u>	<div>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.</div> <div><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 <b>revert-without-query</b> 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></div>		
<b>Revert a buffer</b>  (See also:  Diff & Merge)	<div><ul style="list-style-type: none"><li><b>&lt;f11&gt; f R</b></li><li><b> -u</b></li></ul></div>	(revert-buffer &optional IGNORE-AUTO NOCONFIRM PRESERVE-MODES)	<div>Replace current buffer text with the text of the visited file on disk.</div> <div><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></div> <div> You can use <b>ediff-current-file</b> to see the difference between the buffer and its disk file. PEL binding for this is <b>&lt;f11&gt; e b f</b>.</div>
<u>Toggle auto-revert mode</u>	<b>&lt;f11&gt; f A</b>	(auto-revert-mode &optional ARG)	<div>Toggle reverting buffer when the file changes (Auto-Revert Mode).</div> <div>With a prefix argument ARG, enable Auto-Revert Mode if ARG is positive, and disable it otherwise.</div> <div><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></div>
<u>Toggle auto-revert tail mode</u>	<b>&lt;f11&gt; f T</b>	(auto-revert-tail-mode &optional ARG)	<div>Toggle reverting tail of buffer when the file grows.</div> <div><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></div>
<u>Cancel/restart auto-revert timer</u>	<b>&lt;f11&gt; f SPC</b>	(pel-auto-revert-set-timer)	<div>Restart or cancel the timer used by Auto-Revert Mode.</div> <div><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 '<b>auto-revert-interval</b>' (which is normally 5 seconds by default).</li></ul></div> <div> :</div> <div><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 <b>auto-revert-interval</b> when <b>auto-revert-set-timer</b> is executed.</div>

Operation	Keystroke	Function	Note
<b>Saving Files</b>	Use the following commands to save the content of a buffer to a filesystem file.		
<b>Save file to disk</b>	<ul style="list-style-type: none"> <li><b>C-x C-s</b></li> <li><b>⌘-s</b></li> </ul>	( <b>save-buffer</b> &optional ARG)	Save current buffer to associated file. By default, it makes the previous version into a <b>backup file</b> 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> <li>🍏 On macOS in graphics mode only: <b>⌘-s</b> brings a OS file-save dialog.</li> </ul>
<b>Save all/some files</b>	<b>C-x s</b>	( <b>save-some-buffers</b> &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>
<b>Write buffer to specified file</b>	<b>C-x C-w</b>	<ul style="list-style-type: none"> <li>(<b>write-file</b> FILENAME &amp;optional CONFIRM)</li> <li>(<b>ido-write-file</b>)</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> <li>💡 Use that command to rename the file.</li> </ul>
<b>Changed current buffer changed state</b>	<b>M--</b>	( <b>not-modified</b> &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>
<b>Open Dired (Directory Editor)</b>	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.		
<b>Open a directory editor</b>  (See also: ⌘ Mode - Dired)	<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>(<b>dired</b> DIRNAME &amp;optional SWITCHES)</li> <li>-----</li> <li>(<b>ido-dired</b>)</li> </ul>	Opens a Dired-mode buffer on the specified directory. Prompt for the directory name. <ul style="list-style-type: none"> <li>🔗 PEL activates ido when the pel-use-ido-mode customize variable is set to t.</li> </ul>
<b>Run Dired in other (next) window</b>	<b>C-x 4 d</b>	( <b>dired-other-window</b> )	Opens a Dired-mode buffer on the specified directory inside another window. Prompt for the directory name.
<b>List Directory</b>	<b>C-x C-d</b>	( <b>list-directory</b> 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’.
<b>Search for files with ‘find’ and open Dired buffer</b>	<b>&lt;f11&gt; f d</b>	( <b>find-dired</b> DIR ARGS)	Prompts for the root to search from, and a <b>find</b> command to search for files with the Unix find. Opens a Dired-mode buffer and show the files found in there.
<b>Search directory for files and open Dired buffer for those</b>	<b>&lt;f11&gt; f n</b>	( <b>find-name-dired</b> 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: <pre>find . -name 'PATTERN' -ls</pre>
<b>Find files in a directory and open Dired output</b>	<b>&lt;f11&gt; f h</b>	( <b>find-grep-dired</b> DIR REGEXP)	Find files in DIR that contain matches for REGEXP and start Dired on output. The command run (after changing into DIR) is: <pre>find . \( -type f -exec 'grep-program' 'find-grep-options' -e REGEXP {} \; \) -ls</pre> 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.
<b>Find Emacs Lisp files in directory tree</b>	<b>&lt;f11&gt; f l</b>	( <b>find-lisp-find-dired</b> DIR REGEXP)	Find Emacs Lisp files in DIR, matching REGEXP. Open *Find Lisp Dired* buffer on output.
<b>Automatic File Time Stamp</b>	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: <pre>(add-hook 'before-save-hook 'time-stamp)</pre> 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> 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> These variables can be set in your init file or via the Emacs customization system. <ul style="list-style-type: none"> <li>⚠ To be automatically updated, the time-stamp string must be placed within the first 8 lines of the file.</li> <li>🔗 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.</li> </ul>		
<b>Update file time stamp</b>  (See Also: ⌘ Inserting Text)	<b>&lt;f11&gt; f t</b>	( <b>time-stamp</b> )	Force update the time stamp string(s) in the current buffer. The time stamp is updated if the one of the following strings is found in the first 8 lines of the file: <ul style="list-style-type: none"> <li>Time-stamp: &lt;&gt;</li> <li>Time-stamp: “ ”</li> </ul> 🍌 If you want time stamps updated automatically, write the following inside your init.el file: <pre>(add-hook 'before-save-hook 'time-stamp)</pre>
<b>Toggle time stamp automatic update</b>		( <b>time-stamp-toggle-active</b> &optional ARG)	Toggle ‘time-stamp-active’, setting whether <b>&lt;f11&gt; f t</b> updates a buffer. <ul style="list-style-type: none"> <li>With ARG, turn time stamping on if and only if arg is positive.</li> </ul>
<b>Inserting &amp; Automatically Updating Copyrights</b>	Emacs has built-in support for insertion and update of copyright notices inside files. <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:               <pre>(add-hook 'before-save-hook 'copyright-update)</pre> </li> </ul> ⚠ 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.		
<b>Insert copyright notice at point</b>  (See Also: ⌘ Inserting Text)	<b>&lt;f11&gt; i C</b>	( <b>copyright</b> &optional STR ARG)	Insert a copyright by \$ORGANIZATION notice at cursor. <ul style="list-style-type: none"> <li>If the ORGANIZATION environment variable is not available, Emacs prompts for it.</li> </ul>

Operation	Keystroke	Function	Note
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: (add-hook 'before-save-hook 'copyright-update)</p>
<a href="#">File Lock Protection</a>	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 steal the lock (with 's'), proceed ('p') to edit the file anyway or quit ('q').		

### 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>	