















File Management

Operation	Keystroke	Function	Note
<a href="#">Opening file</a>	<div>The following commands are available to open/visit files in Emacs buffers.</div> <ul style="list-style-type: none"><li>For some of them the corresponding <code>ido</code> 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, <code>find-file</code> 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 <code>pel-init</code>. See the <a href="#">Σ Customize</a> table for more info.</li></ul>		
<b>Open (visit) a file/directory</b>  (See also: <a href="#">Σ Mode - Dired</a> )	<b>C-x C-f</b>	<ul style="list-style-type: none"><li>(<b>find-file</b> FILENAME &amp;optional WILDCARDS) -----</li><li>(<b>ido-find-file</b>)</li></ul>	<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 <code>ido-find-file</code>: it provides suggestions.</div> <div>When <code>ido</code> mode is used, you can also:</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>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 supports several completion modes can all be activated and dynamically selected. See the <a href="#">Σ Customize</a> table for more info.</div>
<b>Open file using OS file-open dialog</b>	<b>⌘-o</b>	( <b>ns-open-file-using-panel</b> )	🍏 On macOS in graphics mode only: open a file, select the file name via an OS File dialog.
<b>Open another file in buffer</b>	<b>C-x C-v</b>	<ul style="list-style-type: none"><li>(<b>find-alternate-file</b> FILENAME &amp;optional WILDCARDS)</li><li>(<b>ido-find-alternate-file</b>)</li></ul>	Kills buffer and open the newly specified file in a new buffer same window. When ido-mode is used, the <code>ido-find-alternate-file</code> is used instead. Useful when just selected an empty file just selected by mistake.
<b>Open file in other window</b>	<ul style="list-style-type: none"><li><b>C-x 4 f</b></li><li><b>&lt;f11&gt; f o</b></li></ul>	<ul style="list-style-type: none"><li>(<b>find-file-other-window</b> FILENAME &amp;optional WILDCARDS)</li><li>(<b>ido-find-file-other-window</b>)</li></ul>	<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>
<b>Open file in other frame</b>	<b>C-x 5 f</b>	<ul style="list-style-type: none"><li>(<b>find-file-other-frame</b> FILENAME &amp;optional WILDCARDS)</li><li>(<b>ido-find-file-other-frame</b>)</li></ul>	<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>
<b>Open a file in read-only mode</b>	<b>C-x C-r</b>	<ul style="list-style-type: none"><li>(<b>find-file-read-only</b> FILENAME &amp;optional WILDCARDS)</li><li>(<b>ido-find-file-read-only</b>)</li></ul>	<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>
<b>Open file in other window in read-only mode</b>	<ul style="list-style-type: none"><li><b>C-x 4 r</b></li><li><b>&lt;f11&gt; f o</b></li></ul>	<ul style="list-style-type: none"><li>(<b>find-file-read-only-other-window</b> FILENAME &amp;optional WILDCARDS)</li><li>(<b>ido-find-file-read-only-other-window</b>)</li></ul>	<div>(<code>find-file-read-only-other-window</code> 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>
<b>Open file or web-page whose name is at point</b>  ★★	<ul style="list-style-type: none"><li><b>C-^</b></li><li><b>&lt;f11&gt; f .</b></li><li><b>6y</b></li></ul>	( <b>pel-find-file-at-point-in-window</b> &optional N)	<div>Open the file, library or the URL, named at point.</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, inexistent or dedicated window 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>More information available in the command’s help docstring.</div> <div>📁🔗 With PEL, the <b>6y</b> key-chord is also available when key-chord is available and active. See <a href="#">Σ Key-Chords</a>.</div>
<b>Run grep via find</b>  (See also <a href="#">Σ 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>	( <b>find-grep</b> COMMAND-ARGS)	<div>Run grep via find, with user-specified args COMMAND-ARGS.</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>
<b><a href="#">Insert text of another file at point</a></b>	<div>The following commands can be used to insert text from other files at point in the current buffer.</div>		
<b>Insert file at point</b>	<ul style="list-style-type: none"><li><b>C-x i</b></li><li><b>&lt;f11&gt; f i</b></li></ul>	<ul style="list-style-type: none"><li>(<b>insert-file</b> FILENAME)</li><li>(<b>ido-insert-file</b>)</li></ul>	<div>Insert contents of file FILENAME into buffer after point.</div> <ul style="list-style-type: none"><li>Set mark after the inserted text.</li></ul>
<b>Insert file literally at point</b>	<b>&lt;f11&gt; f I</b>	( <b>insert-file-literally</b> FILENAME)	<div>Insert contents of file FILENAME into buffer after point with no conversion.</div> <ul style="list-style-type: none"><li>Set mark after the inserted text.</li></ul>
<b><a href="#">Write text into specified file</a></b>	<div>The following commands can be used to write text selected from current buffer into specified file.</div>		
<b>Write region text to file</b>	<b>&lt;f11&gt; f w</b>	( <b>write-region</b> START END FILENAME &optional APPEND VISIT LOCKNAME MUSTBENEW)	<div>Write current region into specified file.</div> <ul style="list-style-type: none"><li>Prompts for the specified file.</li></ul>
<b>Append region text to file</b>	<b>&lt;f11&gt; f W</b>	( <b>append-to-file</b> START END FILENAME)	<div>Append the contents of the region to the end of file FILENAME.</div> <ul style="list-style-type: none"><li>Prompts for the specified file.</li></ul>
<b>Set file mode</b>	<b>&lt;f11&gt; f m</b>	( <b>set-file-modes</b> FILENAME MODE)	<div>Set mode bits of file named FILENAME to MODE (an integer).</div> <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 <code>chmod</code>-like file mode value.</li></ul>
<b><a href="#">Reverting Files</a></b>	<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> <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>		
<b>Revert a buffer</b>  (See also: <a href="#">Σ Diff &amp; Merge</a> )	<ul style="list-style-type: none"><li><b>&lt;f11&gt; f r f</b></li><li><b>⌘-u</b></li></ul>	( <b>revert-buffer</b> &optional IGNORE-AUTO NOCONFIRM PRESERVE-MODES)	<div>Replace current buffer text with the text of the visited file on disk.</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>👉 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>

Operation	Keystroke	Function	Note
<u>Toggle auto-revert mode</u>	<b>&lt;f11&gt; f r a</b>	( <b>auto-revert-mode</b> &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>
<u>Toggle auto-revert tail mode</u>	<b>&lt;f11&gt; f r t</b>	( <b>auto-revert-tail-mode</b> &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>
<u>Cancel/restart auto-revert timer</u>	<b>&lt;f11&gt; f r SPC</b>	( <b>pel-auto-revert-set-timer</b> )	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 '<b>auto-revert-interval</b>' (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.
<b>Saving Files</b>	Use the following commands to save the content of a buffer to a filesystem file.		
<u>Save file to disk</u>	<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 backup file 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.
<u>Save all/some files</u>	<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><b>y</b> : save</li> <li><b>n</b> : don't save</li> <li><b>C-r</b> : look at the buffer in question</li> <li><b>d</b> : view differences with diff-buffer-with-file</li> </ul>
<u>Write buffer to specified file</u>	<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> </ul>  Use that command to rename the file.
<u>Changed current buffer changed state</u>	<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>View Directory Tree with ZTree</b>	 The <a href="#">ztree external package</a> provides a text-based tree-view of a directory with expansion/collapse.  PEL activates it when <b>pel-use-ztree</b> is set to <b>t</b> .  PEL driven customization: <ol style="list-style-type: none"> <li>Use <b>&lt;f11&gt; &lt;f1&gt; f</b> to quickly access the customization group.</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 <b>t</b> to move focus to new entry when <b>&lt;RET&gt;</b> 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 <b>t</b> 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>Execute <b>M-x pel-init</b> after settling and applying new values to activate the new values.</li> </ol>		
<u>View directory as tree with ztree-dir</u>	<b>&lt;f11&gt; z</b>	( <b>ztree-dir</b> PATH)	Open an interactive buffer with the directory tree of the PATH given.  This requires the <a href="#">ztree external package</a> .  PEL activates it when <b>pel-use-ztree</b> is set to <b>t</b> . 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>
<b>View Directory Tree with NeoTree</b>	 The <a href="#">NeoTree external package</a> provides a Vim-NerdTree like tree-view of a directory with expansion/collapse.  PEL activates it when <b>pel-use-neotree</b> is set to <b>t</b> .		
<u>View directory tree with NeoTree</u>	<b>&lt;f11&gt; N</b>	( <b>neotree-toggle</b> )	Toggle show the NeoTree window.  This requires the <a href="#">NeoTree external package</a> .  PEL activates it when <b>pel-use-neotree</b> is set to <b>t</b> . In the NeoTree buffer the following keys are available: <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>

Operation	Keystroke	Function	Note
<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 C-x 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:  Mode - Dired)	<ul style="list-style-type: none"> <li>C-x d</li> <li> -D</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 <b>pel-use-ido-mode</b> customize variable is set to <b>t</b> .
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	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’.
<b>Searching/Finding Files</b>	Ref: <a href="#">Video: ..Emacs #6 : searching and finding files.</a>		
Search for file with locate		(locate SEARCH-STRING &optional FILTER ARG)	
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 <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	<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: <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. 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.
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.
<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.            To be automatically updated, the time-stamp string must be placed within the first 8 lines of the file. 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.		
<b>Update file time stamp</b>  (See Also:  Inserting Text)	<f11> f t	(time-stamp)	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>
Toggle time stamp automatic update		(time-stamp-toggle-active &optional ARG)	Toggle ‘time-stamp-active’, setting whether <f11> f t 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>  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.</li> </ul>		
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"> <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)	Update copyright notice to indicate the current year. <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> 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: <pre>(add-hook 'before-save-hook 'copyright-update)</pre>
<b>File Lock Protection</b>	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
<b><u>Emacs Display - Mode Line</u></b>	Read first. Describes what the Emacs mode line displays.
<b><u>GNU Emacs Manual - File Handling</u></b>	Describes how to open and deal with files and directories in Emacs.
<b><u>GNU EMACS Manual - Interactive Do</u></b>	Describes the ido-mode, a nice addition that helps with completing file names at prompts.
<b><u>Display path of file in status bar</u></b>	In graphics mode, display the buffer name and the full path file in parenthesis inside the frame title bar.
<b><u>How do I rename an open file in Emacs?</u></b>	