File Management

Operation	Keystroke	Function	Note
Opening file	The following commands are available to open/visit files in Emacs buffers. For some of them the corresponding ido mode function is also shown. Note: Emacs uses the word "visiting" instead of "opening" files.		
Open (visit) a file/directory (See also: ⧖ Mode - Dired)	`C-x C-f`	• (**find-file** FILENAME &optional WILDCARDS) – – – – – – – – – – – – – – – – – – • (**ido-find-file**)	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: • Type **C-x f** to change to original find-file • Type **C-j** to accept the file/directory name verbatim without replacement or suggestion. 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.
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 another file in buffer	`C-x C-v`	• (**find-alternate-file** FILENAME &optional WILDCARDS) • (**ido-find-alternate-file**)	Kills buffer and open the newly specified file in a new buffer same window. When ido-mode is used, the ido-find-alternate-file is used instead. Useful when just selected an empty file just selected by mistake.
Open file in other window	• `C-x 4 f` • `<f11> f o`	• (**find-file-other-window** FILENAME &optional WILDCARDS) • (**ido-find-file-other-window**)	Edit file FILENAME, in another window. Like **C-x C-f**, but creates a new window or reuses an existing one.
Open file in other frame	`C-x 5 f`	• (**find-file-other-frame** FILENAME &optional WILDCARDS) • (**ido-find-file-other-frame**)	Edit file FILENAME, in another frame. Like **C-x C-f**, but creates a new frame or reuses an existing one.
Open a file in read-only mode	`C-x C-r`	• (**find-file-read-only** FILENAME &optional WILDCARDS) • (**ido-find-file-read-only**)	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	• `C-x 4 r` • `<f11> f r`	• (**find-file-read-only-other-window** FILENAME &optional WILDCARDS) • (**ido-find-file-read-only-other-window**)	(find-file-read-only-other-window FILENAME &optional WILDCARDS) Edit file FILENAME in another window but don't allow changes. Like **C-x 4 C-f**, but marks buffer as read-only. Use **C-x C-q** to permit editing.
Insert other file into the current buffer after cursor	`C-x i`	• (**insert-file** FILENAME) • (**ido-insert-file**)	Inserts the text of another file after point. Set mark after the inserted text.
Open file or web-page whose name is at point ★★	`C-^`	(**pel-find-file-at-point-in-window** &optional N)	Open the file, library or the URL, named at point. • If the string identifies a URL, the function opens the page in the default browser. • 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. • If the file is not found, the function prompts. If the name corresponds to an Emacs library file, you can type **1** to open the library. You can also edit the file name collected before attempting to open it again. Or quit. • If the file name is followed by line and column numbers the point is moved to that position. More information available in the command's help docstring.
Reverting Files	If the file's content changed on the disk and you want to refresh the Emacs buffer visiting that file, you need to "revert" the file. If you want to use Emacs to monitor the content of a file that is continuously modified by an external process (like a log file) set the ***revert-without-query*** variable to a list of regular expressions describing the field it'll apply to. You can also activate the auto-revert mode for the current buffer or globally and restart its timer.		
Revert a buffer	• `<f11> f R` • `⌘-u`	(**revert-buffer** &optional IGNORE-AUTO NOCONFIRM PRESERVE-MODES)	Replace current buffer text with the text of the visited file on disk. This undoes all changes since the file was visited or saved. With a prefix argument, offer to revert from latest auto-save file, if that is more recent than the visited file. This is also the command to use to reload a file that was modified on the file system.
Toggle auto-revert mode	`<f11> f A`	(**auto-revert-mode** &optional ARG)	Toggle reverting buffer when the file changes (Auto-Revert Mode). With a prefix argument ARG, enable Auto-Revert Mode if ARG is positive, and disable it otherwise. • Auto-Revert Mode is a minor mode that affects only the current buffer. When enabled, it reverts the buffer when the file on disk changes. • When a buffer is reverted, a message is generated. This can be suppressed by setting 'auto-revert-verbose' to nil.
Toggle auto-revert tail mode	`<f11> f T`	(**auto-revert-tail-mode** &optional ARG)	Toggle reverting tail of buffer when the file grows. • With a prefix argument ARG, enable Auto-Revert Tail Mode if ARG is positive, and disable it otherwise. • When Auto-Revert Tail Mode is enabled, the tail of the file is constantly followed, as with the shell command 'tail -f'. This means that whenever the file grows on disk (presumably because some background process is appending to it from time to time), this is reflected in the current buffer. • You can edit the buffer and turn this mode off and on again as you please. But make sure the background process has stopped writing before you save the file!
Cancel/restart auto-revert timer	`<f11> f SPC`	(**pel-auto-revert-set-timer**)	Restart or cancel the timer used by Auto-Revert Mode. • If such a timer is active, cancel it. • Start a new timer if Global Auto-Revert Mode is active or if Auto-Revert Mode is active in some buffer. • Restarting the timer ensures that Auto-Revert Mode will use an up-to-date value of '***auto-revert-interval***'(which is normally 5 seconds by default). 💻 : **pel-auto-revert-set-timer** is a thin wrapper over **auto-revert-set-timer** that displays a warning if executed when the buffer is not already in auto-revert-mode. It also displays the value of *auto-revert-interval* when **auto-revert-set-timer** is executed.
Saving Files	Use the following commands to save the content of a buffer to a filesystem file.		
Save file to disk	• `C-x C-s` • `⌘-s`	(**save-buffer** &optional ARG)	Save current buffer to associated file. By default, it makes the previous version into a backup file if previously requested or if this is the first save. • With **C-u**: marks this version to become a backup when the next save is done • With **C-u C-u**: makes the previous version into a backup file • With **C-u C-u C-u**: marks this version to become a backup when the next save is done, and makes the previous version into a backup file. • With prefix 0: never make the previous version into a backup file. On macOS in graphics mode only: `⌘-s` brings a OS file-save dialog.

Operation	Keystroke	Function	Note
Save all/some files	`C-x s`	(**save-some-buffers** &optional ARG PRED)	Prompt for files that are modified. Options: • y : save • n : don't save • C-r : look at the buffer in question • d : view differences with diff-buffer-with-file
Write buffer to specified file	`C-x C-w`	• (**write-file** FILENAME &optional CONFIRM) • (**ido-write-file**)	Similar to "Save-As": prompt for the filename. • Can also be yanked in the mini buffer, use `M-n` to edit it. 💡 Use that command to rename the file.
Changed current buffer changed state	`M-~`	(**not-modified** &optional ARG)	Mark current buffer as unmodified, not needing to be saved. • With `C-u` prefix ARG, mark buffer as modified, so `C-x C-s` will save.
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: ∑ Mode - Dired)	• `C-x d` • `⌘-D`	• (**dired** DIRNAME &optional SWITCHES) – – – – – – – – – – – – – – – – – – • (**ido-dired**)	Opens a Dired-mode buffer on the specified directory. Prompt for the directory name. • The PEL package use IDO; the keys are bound to the (ido-dired) command.
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'.
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. Opens a Dired-mode buffer and show the files found in there.
Search directory for files and open Dired buffer for those	`<f11> f n`	(**find-name-dired** DIR PATTERN)	Search DIR recursively for files matching the globbing pattern PATTERN, and run Dired on those files. PATTERN is a shell wildcard (not an Emacs regexp) and need not be quoted. The default command run (after changing into DIR) is: find . -name 'PATTERN' -ls
Find files in a directory and open Dired output	`<f11> f h`	(**find-grep-dired** DIR REGEXP)	Find files in DIR that contain matches for REGEXP and start Dired on output. The command run (after changing into DIR) is: find . \(-type f -exec 'grep-program' 'find-grep-options' -e REGEXP {} \; \) -ls where the first string in the value of the variable 'find-ls-option' specifies what to use in place of "-ls" as the final argument.
Find Emacs Lisp files in directory tree	`<f11> f l`	(**find-lisp-find-dired** DIR REGEXP)	Find Emacs Lisp files in DIR, matching REGEXP. Open *Find Lisp Dired* buffer on output.

Automatic File Time Stamp	Emacs has a built-in automatic time-stamping of files. It must be activated by adding the **time-stamp** function to the **before-save-hook** variable. This can either be done via Emacs customization system or explicitly inside your init file with the following code:
	`(add-hook 'before-save-hook 'time-stamp)`
	
	The time stamp will be added to files that contain, inside their first 8 lines, a line that looks like one of the following:
	• `Time-stamp: <>`
	• `Time-stamp: " "`
	
	The format of the time stamp is controlled by several variables:
	• **time-stamp-format** specifies the format of the time stamp. Something like `"%:y-%02m-%02d %02H:%02M:%02S %u"` to specify the date and time in ISO format, with the user login's name.
	• **time-stamp-time-zone** specifies the time zone selection:
	  • nil: Emacs local time
	  • t: Universal time
	  • wall : system wall clock time
	  • TZ : controlled by a TZ environment variable
	
	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.		

Update file time stamp (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: • Time-stamp: <> • Time-stamp: " " ☝ If you want time stamps updated automatically, write the following inside your init.el file: `(add-hook 'before-save-hook 'time-stamp)`
Toggle time stamp automatic update		(**time-stamp-toggle-active** &optional ARG)	Toggle 'time-stamp-active', setting whether `<f11> f t` updates a buffer. • With ARG, turn time stamping on if and only if arg is positive.
Inserting & Automatically Updating Copyrights	Emacs has built-in support for insertion and update of copyright notices inside files. • Two commands, shown below, are provided to manually insert or update the file's copyright notice. • The copyright notice can be automatically updated by adding the **copyright-update** function to the list of **before-save-hook** variable with the following code: `(add-hook 'before-save-hook 'copyright-update)` ⚠️ To be automatically updated, the copyright notice must be placed within an area at the beginning of the file specified by the value of the **copyright-limit** variable, normally defined as the first 2000 characters. This variable is customizable.		
Insert copyright notice at point (See Also: ∑ Inserting Text)	`<f11> i C`	(**copyright** &optional STR ARG)	Insert a copyright by $ORGANIZATION notice at cursor. • If the ORGANIZATION environment variable is not available, Emacs prompts for it.

Operation	Keystroke	Function	Note
Update file's copyright notice		(**copyright-update** &optional ARG INTERACTIVEP)	Update copyright notice to indicate the current year. • With prefix ARG, replace the years in the notice rather than adding the current year after them. If necessary, and 'copyright-current-gpl-version' is set, any copying permissions following the copyright are updated as well. • If non-nil, INTERACTIVEP tells the function to behave as when it's called interactively. ⚠️ Even when used interactively copyright-update does not warn if there is no copyright in the current buffer to update. It does not create a missing notice. ☝ If you want to be prompted automatically to update an existing but out-of-date copyright notice, write the following inside your init.el file: `(add-hook 'before-save-hook 'copyright-update)`
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 steal the lock (with 's'), proceed ('p') to edit the file anyway or quit ('q').		

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?	