












# Diff & Merge

Operation	Keystroke	Function	Note
<b>Diff, Merge &amp; Patch Files within Emacs</b> <ul style="list-style-type: none"> <li>Help &amp; Customization</li> <li>Diff files, diff-mode <ul style="list-style-type: none"> <li>move in hunks/files</li> <li>apply/rev changes</li> </ul> </li> <li>highlight changes</li> <li>manipulate hunks</li> <li>Ediff</li> <li>Side-by-Side Diff</li> <li>Compare Directories</li> <li>Ediff Merge</li> <li>Specialized Ediff</li> <li>Smerge</li> <li>Patch files</li> </ul> See <a href="#">🔗 VCS-Git</a> <a href="#">xMagit</a> <a href="#">🔗 VCS-Mercurial</a>	Emacs has complete support to perform <b>text file diff</b> , 2-way merge and <b>3-way merge</b> operations. It nicely compares to <a href="#">what's available outside Emacs</a> . <ul style="list-style-type: none"> <li>Emacs has two built-in packages that support comparing files: <b>diff</b> and <b>ediff</b>. <ul style="list-style-type: none"> <li>diff can be used to show a diff file, navigate through it and apply the diff hunk to a file in any direction (apply changes or revert them)</li> <li>ediff is more powerful than diff and more visual, using one buffer per file. ediff supports <b>3-way merge</b> and supports diff of directory trees.</li> </ul> </li> <li>For <b>3-way merge</b> operations Emacs also provides smerge. <ul style="list-style-type: none"> <li>The smerge system is quite useful: if activated by <b>pel-use-smerge</b> set to <i>auto</i>, Emacs will automatically launch smerge when it detects a (D)VCS merge markup inside a file, similar to what <b>Git</b> or <b>Mercurial</b> will do when they cannot automatically complete a merge operation themselves.</li> <li>smerge provides all the commands to perform the merge. If you prefer to use 3 buffers you can invoke ediff directly from smerge.</li> <li>smerge inherit from the older but still supported <b>Emerge</b> built-in package.</li> <li>PEL does not provide key bindings for Emerge as ediff commands are easier to use and more powerful.</li> </ul> </li> </ul> PEL also provides support for the following extra external packages: <ul style="list-style-type: none"> <li> The <b>diffview-mode</b> package provides side-by-side diff format.  PEL activates it when <b>pel-use-diffview</b> is set to <b>t</b>.</li> <li> The <b>ztree external package</b> provides a tree-view directory diff.  PEL activates it when <b>pel-use-ztree</b> is set to <b>t</b>.</li> </ul>		
Last updated on:	2025-04-23		
<b>Open this PDF file.</b> See also: <a href="#">🔗 Help/Info</a>	<div>&lt;f11&gt; d &lt;f1&gt;</div> <div>&lt;f12&gt; &lt;f1&gt;</div>	<div>(pel-help-pdf &amp;optional OPEN-WEB-PAGE)</div>	Open the <a href="#">🔗 Diff &amp; Merge</a> local PDF. If the prefix argument (like <b>C-u</b> or <b>M--</b> ) is used, then it opens the remote GitHub hosted raw PDF instead. If the <b>pel-flip-help-pdf-arg</b> user-option is set it's the other way around.
<b>Customize PEL support for diff</b> See also: <a href="#">🔗 Customize</a>	<div>&lt;f11&gt; d &lt;f2&gt;</div> <div>&lt;f12&gt; &lt;f2&gt;</div>	<div>(pel-customize-pel &amp;optional OTHER-WINDOW)</div>	Customize PEL support for diff: smerge, ztree <ul style="list-style-type: none"> <li>If OTHER-WINDOW is non-nil (use <b>C-u</b>), display in other window.</li> </ul>
<b>Customize Emacs for specify diff mode</b> <ul style="list-style-type: none"> <li>Adjusted for the current diff mode being used</li> </ul>	<div>&lt;f12&gt; &lt;f3&gt;</div> <div>&lt;f11&gt; d &lt;f3&gt;</div> <div>&lt;f11&gt; d e &lt;f3&gt;</div> <div>&lt;f11&gt; d s &lt;f3&gt;</div>	<div>(pel-customize-library &amp;optional OTHER-WINDOW)</div>	Customize Emacs support for currently active diff mode. For this & following: If OTHER-WINDOW non-nil ( <b>C-u</b> ), display in other window.
See also: <a href="#">🔗 Customize</a>			Customize Emacs support for diff, ediff, emerge, smerge, ztree.
			Customize Emacs ediff.
			Customize Emacs smerge
<a href="#">Diff</a>	The <b>diff-mode</b> provides a set of commands to compare files and buffers. This mode is used when the buffer holds a patch file created by various tools. Use the following commands to compare files and buffers and enter the diff-mode. See the commands provided by the diff-mode below.		
<b>Compare 2 files</b>	<div>&lt;f11&gt; d f</div>	<div>(diff OLD NEW &amp;optional SWITCHES NO-ASYNC)</div>	Find and display the differences between OLD and NEW files. <ul style="list-style-type: none"> <li>Prompt for NEW, then OLD files.</li> </ul>
<b>Compare file with its backup</b>	<div>&lt;f11&gt; d k</div>	<div>(diff-backup FILE &amp;optional SWITCHES)</div>	Diff this file with its backup file or vice versa. <ul style="list-style-type: none"> <li>Uses the latest backup, if there are several numerical backups.</li> <li>If this file is a backup, diff it with its original.</li> <li>The backup file is the first file given to 'diff'.</li> <li>With prefix arg, prompt for diff switches.</li> </ul>
<b>Compare buffer and associated file</b>	<div>• &lt;f11&gt; d b</div> <div>• &lt;f11&gt; b =</div>	<div>(diff-buffer-with-file &amp;optional BUFFER)</div>	View the differences between BUFFER and its associated file. See also <a href="#">🔗 Buffers</a>
<b>Compare current and other window</b>	<div>&lt;f11&gt; d w</div>	<div>(compare-windows IGNORE-WHITESPACE)</div>	Compare text in current window with text in another window.
	<ul style="list-style-type: none"> <li>The <b>compare-windows-get-window-function</b> user option defines how to get another window. Set to <i>compare-windows-get-recent-window</i> by default.</li> <li>Compares the text starting at point in each window, moving over text in each one as far as they match.</li> <li>This command pushes the mark in each window at the prior location of point in that window.</li> <li>If both windows display the same buffer, the mark is pushed twice in that buffer: first in the other window, then in the selected window.</li> </ul> 🍌 Use this for simple comparison between 2 windows just to see if they have the same content. <ul style="list-style-type: none"> <li>Split your frame in 2 windows, loaded with the buffer of the files you want to compare.</li> <li>Place point at the top of each buffer and issue the command.</li> <li>Point will be moved to the first difference in both window. If there is no difference point will be moved at the end of each window.</li> </ul>		
<a href="#">diff-mode</a> commands	Use the following commands in a buffer using the <a href="#">diff-mode</a> major mode to manipulate the patch diff file and the <a href="#">diff hunks</a> . 🍌 The following commands are also available from the Diff menu on the menu-bar accessible via the <b>&lt;f10&gt;</b> key. See <a href="#">🔗 Menus</a>		
<b>diff-mode setup</b>	<div>&lt;f12&gt; &lt;f4&gt; ?</div>	<div>(pel-diff-show-status)</div>	Display diff-mode status info in mini-buffer: whether it jumps to new or old source.
<b>Toggle Next-Error-Follow minor mode</b>	<div>C-c C-f</div>	<div>(next-error-follow-minor-mode &amp;optional ARG)</div>	Minor mode for compilation, occur and diff modes. <ul style="list-style-type: none"> <li>With a prefix argument ARG, enable mode if ARG is positive, and disable it otherwise. If called from Lisp, enable mode if ARG is omitted or nil.</li> <li>When turned on, cursor motion in the compilation, grep, occur or diff buffer causes automatic display of the corresponding source code location.</li> </ul>
<b>Jump to source file</b> <ul style="list-style-type: none"> <li>open source of current hunk in a new window</li> </ul>	<div>C-c C-c</div>	<div>(diff-goto-source &amp;optional OTHER-FILE EVENT)</div>	Jump to the corresponding source line. <ul style="list-style-type: none"> <li>'diff-jump-to-old-file' (or its opposite if the OTHER-FILE prefix arg is given) determines whether to jump to the <b>old</b> or the <b>new</b> file.</li> <li>If the prefix arg is bigger than 8 (for example with <b>C-u C-u</b>) then 'diff-jump-to-old-file' is also set, for the next invocations. Use <b>&lt;f12&gt; &lt;f4&gt; ?</b> to see status.</li> </ul>
<b>To next hunk</b>	<div>• M-n</div> <div>• C-M-i</div>	<div>(diff-hunk-next &amp;optional COUNT)</div>	Go to the next COUNT'th hunk.
<b>To previous hunk</b>	<div>• M-p</div> <div>• &lt;Esc&gt; &lt;backtab&gt;</div>	<div>(diff-hunk-prev &amp;optional COUNT)</div>	Go to the previous COUNT'th hunk
<b>To next file</b>	<div>• M-}</div> <div>• M-N</div> <div>• &lt;f6&gt; &lt;down&gt;</div>	<div>(diff-file-next &amp;optional COUNT)</div>	Go to the next COUNT'th file.
<b>To previous file</b>	<div>• M-{</div> <div>• M-P</div> <div>• &lt;f6&gt; &lt;up&gt;</div>	<div>(diff-file-prev &amp;optional COUNT)</div>	Go to the previous COUNT'th file
<b>Show all files present in diff inside an <u>occur</u> buffer</b>	<div>&lt;f6&gt; o</div>	<div>(pel-diff-hunk-files-occur &amp;optional NLINES)</div>	Show hunk files of current path patch inside an occur buffer. <ul style="list-style-type: none"> <li>Each line shown with NLINES before &amp; after, or -NLINES before if NLINES &lt; 0.</li> <li>NLINES defaults to 0, overriding list-matching-lines-default-context-lines.</li> <li>If a region is defined the search is restricted to the region. See <a href="#">occur search</a>.</li> </ul>
<b>Restrict view to current hunk or file</b> See <a href="#">🔗 Narrowing</a>	<div>C-c C-n</div>	<div>(diff-restrict-view &amp;optional ARG)</div>	Restrict the view to the current hunk. This is a <a href="#">buffer narrowing type</a> of command. <ul style="list-style-type: none"> <li>If the prefix ARG is given, restrict the view to the current file instead.</li> <li>Use <b>C-x n w</b> to widen the buffer back.</li> </ul>
<b>Apply current hunk to source file</b> ★★ • <a href="#">Reverse a change</a>	<div>C-c C-a</div>	<div>(diff-apply-hunk &amp;optional REVERSE)</div>	Apply <b>current hunk</b> ( <i>not all hunks for the file!</i> ) to the source file and go to the next. <ul style="list-style-type: none"> <li>By default, the new source file is patched, but if the variable 'diff-jump-to-old-file' is non-nil, then the old source file is patched instead.</li> <li>🍌 With a <b>C-u</b> prefix argument, REVERSE the hunk, to discard a change!</li> </ul>
<b>Apply diff with Ediff</b> ★★ 🍌 Apply VCS commit set to another (set of) file(s)	<div>C-c C-e</div>	<div>(diff-ediff-patch)</div>	Call 'ediff-patch-file' on the current buffer: Query for a file name, and then run Ediff by patching that file. If several files are in the diff, the Ediff buffer creates one session for each. <ul style="list-style-type: none"> <li>If optional PATCH-BUF is given, use the patch in that buffer &amp; don't ask user.</li> <li>If prefix argument ARG, then: if even argument, assume that the patch is in a buffer. If odd -- assume it is in a file.</li> </ul>

Operation	Keystroke	Function	Note
Highlight changes in hunk	C-c C-b	(diff-refine-hunk)	Highlight changes of hunk at point at a finer granularity.
Test if hunk can be applied	C-c C-t	(diff-test-hunk &optional REVERSE)	See whether it's possible to apply the current hunk. With a prefix argument, try to REVERSE the hunk.
Convert unified diff to context diff	C-c C-d	(diff-unified->context START END)	Convert unified diffs to context diffs. <ul style="list-style-type: none"> <li>START and END are either taken from the region (if a prefix arg is given) or else cover the whole buffer.</li> </ul>
Reverse direction of diff	C-c C-r	(diff-reverse-direction START END)	Reverse the direction of the diffs. <ul style="list-style-type: none"> <li>START and END are either taken from the region (if a prefix arg is given) or else cover the whole buffer.</li> </ul>
Split hunk	C-c C-s	(diff-split-hunk)	Split the current (unified diff) hunk at point into two hunks. <div>👉 Useful when you want to apply or revert only one part of it.</div>
Convert context diff to unified diff	C-c C-u	(diff-context->unified START END &optional TO-CONTEXT)	Convert context diffs to unified diffs. <ul style="list-style-type: none"> <li>START and END are either taken from the region (when it is highlighted) or else cover the whole buffer.</li> <li>With a prefix argument, convert unified format to context format.</li> </ul>
Re-diff current hunk, ignore whitespace	C-c C-w	(diff-ignore-whitespace-hunk)	Re-diff the current hunk, ignoring whitespace differences.
Re-diff all hunks, ignore whitespace	<f6> w	(pel-diff-ignore-whitespace-in-hunks)	Re-diff <b>all</b> hunks in buffer, ignoring whitespace differences.
<div>Ediff</div> <div>See also:</div> <ul style="list-style-type: none"> <li>🔗 Scrolling</li> <li>🔗 Help/Info</li> </ul>	<div>Ediff sessions have several commands, shown in the Ediff Quick Help buffer.</div> <ul style="list-style-type: none"> <li>Type ? to toggle Ediff Quick Help from 1 line to multiple that shows all available commands.</li> <li>For more info about a command, place point on the command character and type RET: Emacs will open the Ediff Quick Help Commands Info buffer.</li> </ul> <div>👉 While showing 2 or 3 buffer/files in windows Ediff provides the <b>⌵/⌴</b> keys for scrolling up/down.</div> <div>PEL scroll sync commands (&lt;f11&gt;  ) can also be used to provide single line scroll between synced windows.</div> <div>Ediff is a major mode. As for all major modes, you can display help for the mode with describe-mode bound to C-h m and &lt;f1&gt; m</div>		
Display Ediff Manual	<f11> d e ?	(ediff-documentation &optional NODE)	Display Ediff's manual. <ul style="list-style-type: none"> <li>With optional NODE, goes to that node.</li> </ul>
Ediff 2 files  ★★	<f11> d 2	(pel-ediff-2files &optional N)	Run ediff-files on the files of current and the other window. <ul style="list-style-type: none"> <li>Select the current file and the other file without prompting.</li> <li>With numeric argument if N is in [2,8] range, select other window identified by the direction corresponding to the cursor in a numeric keypad: <div>8 := 'up</div> <div>4 := 'left   5 := 'current   6 := 'right</div> <div>2 := 'down</div> </li> </ul>
Display registry of active Ediff sessions	<f11> d e R	<ul style="list-style-type: none"> <li>(eregistry)</li> <li>(ediff-show-registry)</li> </ul>	Display registry of all active Ediff sessions.
Ediff file against previous revision	<f11> d r	(pel-ediff-revisions)	Run ediff-revision on the file in the current window. <ul style="list-style-type: none"> <li>Prompts for revisions, default to current copy and last commit.</li> </ul>
Compare buffer with its file on disk	<ul style="list-style-type: none"> <li>&lt;f11&gt; d e b f</li> <li>&lt;f11&gt; b M-=</li> </ul>	(ediff-current-file)	Compare the buffer with its file on disk. This function can be used as a safe version of revert-buffer. See also 🔗 Buffers
Compare 2 buffers	<f11> d e b b	(ediff-buffers BUFFER-A BUFFER-B &optional STARTUP-HOOKS JOB-NAME)	Compare 2 buffers. <ul style="list-style-type: none"> <li>Prompts for buffer A and buffer B</li> </ul>
Compare 3 buffers	<f11> d e b 3	(ediff-buffers3 BUFFER-A BUFFER-B BUFFER-C &optional STARTUP-HOOKS JOB-NAME)	Compare 3 buffers. <ul style="list-style-type: none"> <li>Prompts for buffer A, buffer B and buffer C.</li> </ul>
Compare file with its backup 🔗 Autosave/backup	<f11> d e f k	(ediff-backup FILE)	Compare a file with its backup. If there are several numerical backups, use the latest. <ul style="list-style-type: none"> <li>If the file is itself a backup, then compare it with its original.</li> </ul>
Compare 2 files	<f11> d e f f	<ul style="list-style-type: none"> <li>(ediff FILE-A FILE-B &amp;optional STARTUP-HOOKS)</li> <li>(ediff-files FILE-A FILE-B &amp;optional STARTUP-HOOKS)</li> </ul>	Compare 2 files. Uses either diff or ediff-files. <ul style="list-style-type: none"> <li>Prompts for file A and B.</li> <li>PEL provide a shortcut function <b>pel-ediff-2files</b> mapped to &lt;f11&gt; d 2 which does not prompt. See above.</li> </ul>
Compare 3 files	<f11> d e f 3	<ul style="list-style-type: none"> <li>(ediff3 FILE-A FILE-B FILE-C &amp;optional STARTUP-HOOKS)</li> <li>(ediff-files3 FILE-A FILE-B FILE-C &amp;optional STARTUP-HOOKS)</li> </ul>	Compare 3 files. <ul style="list-style-type: none"> <li>Prompts for file A, B and C.</li> </ul>
Compare revision of buffer with file revision	<f11> d e f r	(ediff-revision &optional FILE STARTUP-HOOKS)	Compare versions of the current buffer, if the buffer is visiting a file under VCS. <ul style="list-style-type: none"> <li>Prompts for the file name and each of its revisions.</li> <li>PEL provide a shortcut function <b>pel-ediff-revision</b> mapped to &lt;f11&gt; d r.</li> </ul>
Compare versions of files in a given directory	<f11> d e d r	<ul style="list-style-type: none"> <li>(edir-revisions DIR1 REGEXP)</li> <li>(ediff-directory-revisions DIR1 REGEXP)</li> </ul>	Compare versions of files in a given directory. <ul style="list-style-type: none"> <li>Ediff selects only the files that are under version control.</li> <li>Prompts for directory and regexp to identify files: if empty: selects all files.</li> </ul>
Compare text in 2 windows word-by-word	<f11> d e w w	(ediff-windows-wordwise DUMB-MODE &optional WIND-A WIND-B STARTUP-HOOKS)	Compare text visible in 2 windows word-by-word. <ul style="list-style-type: none"> <li>Uses current and other (next) window.</li> </ul>
Compare text in 2 windows line-by-line	<f11> d e w l	(ediff-windows-linewise DUMB-MODE &optional WIND-A WIND-B STARTUP-HOOKS)	Compare text visible in 2 windows line-by-line. <ul style="list-style-type: none"> <li>Uses current and other (next) window.</li> </ul>
Compare 2 regions word-by-word	<f11> d e r w	(ediff-regions-wordwise BUFFER-A BUFFER-B &optional STARTUP-HOOKS)	Compare text visible in 2 regions word-by-word. <ul style="list-style-type: none"> <li>Prompts for the 2 buffers and regions.</li> </ul>
Compare 2 regions line-by-line	<f11> d e r l	(ediff-regions-linewise BUFFER-A BUFFER-B &optional STARTUP-HOOKS)	Compare text visible in 2 regions line-by-line. <ul style="list-style-type: none"> <li>Prompts for the 2 buffers and regions.</li> </ul>
Side-by-Side Diff	<div>Using 📦 <a href="#">diffview-mode</a> package 🐙 PEL activates it when <b>pel-use-diffview-mode</b> is set to t.</div> <div>During that mode:</div> <ul style="list-style-type: none"> <li>} : Next file</li> <li>{ : Previous file</li> <li>1 : Align windows</li> <li>q : Quit</li> <li>With PEL, use <b>pel-toggle-scroll-sync</b>, mapped to &lt;f11&gt;    to scroll both windows. 🐛🚨 Use with care: scrolling seems to stick. See 🔗 Scrolling</li> <li>To return to the original window layout you can use <b>winner-undo</b> &lt;f11&gt; w p, with PEL. <ul style="list-style-type: none"> <li>See 🔗 Windows (end of page 4)</li> </ul> </li> </ul>		
current buffer	<f11> d	(diffview-current)	Show current diff buffer in a side-by-side view.
current region	<f11> d M-	(diffview-region)	Show current diff region in a side-by-side view.

Operation	Keystroke	Function	Note
Compare Directories	The built-in Ediff can compare 2 or 3 directories.  The <b>ztree external package</b> provides a tree-view directory diff.  PEL activates it when <b>pel-use-ztree</b> is set to <b>t</b> .		
Compare common files in 2 directories	<f11> d e d d	<ul style="list-style-type: none"> <li>(edirs DIR1 DIR2 REGEXP)</li> <li>(ediff-directories DIR1 DIR2 REGEXP)</li> </ul>	Compare files common to two directories. <ul style="list-style-type: none"> <li>Prompts for directory A &amp; B and a regexp to identify files. If empty: select all files.</li> </ul>
Compare common files in 3 directories	<f11> d e d 3	<ul style="list-style-type: none"> <li>(edirs3 DIR1 DIR2 DIR3 REGEXP)</li> <li>(ediff-directories3 DIR1 DIR2 DIR3 REGEXP)</li> </ul>	Compare files common to three directories. <ul style="list-style-type: none"> <li>Prompts for directory A, B and C and a regexp to identify files: if empty: selects all files.</li> </ul>
Compare 2 directories with ztree-diff	<f11> d z	(ztree-diff DIR1 DIR2)	Open an interactive buffer with the directory tree of the path given, and highlight of file differences between the directories. DIR1 := left directory. DIR2:=right directory <ul style="list-style-type: none"> <li>Interactively: prompts for the 2 directories.</li> </ul>  Requires the <b>ztree external package</b> .  PEL activates it if <b>pel-use-ztree</b> is <b>t</b> .
Ztree Diff buffer keys: <ul style="list-style-type: none"> <li>Ediff 2 files ▾</li> </ul>	<ul style="list-style-type: none"> <li>&lt;SPACE&gt; : Open/close directory</li> <li>&lt;TAB&gt; : switch between panels</li> <li>&lt;RET&gt; : Open/close directory   <b>Ediff 2 files</b> /open orphan file</li> <li>x : 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 may take a long time. I have see Emacs hang when typing <b>x</b> again during that time.</li> </ul> </li> <li>&lt;f5&gt; : force full rescan, re-write buffer (useful when changing window size).</li> </ul>		<ul style="list-style-type: none"> <li>C : copy current file/directory to the directory shown in the other side (prompts)</li> <li>D : delete current file/directory (prompts)</li> <li>h : toggle display of identical files/directories.</li> <li>H : toggle display of filtered files.</li> <li>r : rescan/refresh current file/directory</li> <li>v : visit current file in <u>view-mode</u> . In view-mode the following keys are available (see <u>Buffers</u> for a complete list):               <ul style="list-style-type: none"> <li>q: quit view-mode and return to Ztree Diff</li> <li>e: leave view-mode, edit/visit the file normally.</li> </ul> </li> </ul>
Ediff Merge	Use the following commands to perform <b>3-way merges</b> within Emacs using the merge capability of the ediff built-in package.		
Merge 2 files	<f11> d e m f	<ul style="list-style-type: none"> <li>(ediff-merge FILE-A FILE-B &amp;optional STARTUP-HOOKS MERGE-BUFFER-FILE)</li> <li>(ediff-merge-files FILE-A FILE-B &amp;optional STARTUP-HOOKS MERGE-BUFFER-FILE)</li> </ul>	Merge two files without ancestor. <ul style="list-style-type: none"> <li>Prompt for FILE-A and FILE-B, the names of the files to be merged.</li> <li>The result is stored into an *ediff-merge* buffer, not a file.               <ul style="list-style-type: none"> <li>Save it into a file with <b>C-x C-s</b>. See <b>screenshot example in PEL manual</b>.</li> </ul> </li> </ul>
Merge 2 files with ancestor	<f11> d e m F	<ul style="list-style-type: none"> <li>(ediff-merge-with-ancestor FILE-A FILE-B FILE-ANCESTOR &amp;optional STARTUP-HOOKS MERGE-BUFFER-FILE)</li> <li>(ediff-merge-files-with-ancestor FILE-A FILE-B FILE-ANCESTOR &amp;optional STARTUP-HOOKS MERGE-BUFFER-FILE)</li> </ul>	Merge two files with ancestor. <ul style="list-style-type: none"> <li>Prompt for FILE-A and FILE-B, the names of the files to be merged, and FILE-ANCESTOR, the name of the ancestor file.</li> <li>The result is stored into an *ediff-merge* buffer, not a file.               <ul style="list-style-type: none"> <li>Save it into a file with <b>C-x C-s</b>. See <b>screenshot example in PEL manual</b>.</li> </ul> </li> </ul>
Merge 2 buffers	<f11> d e m b	(ediff-merge-buffers BUFFER-A BUFFER-B &optional STARTUP-HOOKS JOB-NAME MERGE-BUFFER-FILE)	Merge buffers without ancestor. <ul style="list-style-type: none"> <li>Prompt for BUFFER-A and BUFFER-B, the buffers to be merged.</li> </ul>
Merge 2 buffers with ancestor	<f11> d e m B	(ediff-merge-buffers-with-ancestor BUFFER-A BUFFER-B BUFFER-ANCESTOR &optional STARTUP-HOOKS JOB-NAME MERGE-BUFFER-FILE)	Merge buffers with ancestor. <ul style="list-style-type: none"> <li>Prompts for BUFFER-A and BUFFER-B, the buffers to be merged, and BUFFER-ANCESTOR, their ancestor.</li> </ul>
Merge versions of files in a directory	<f11> d e m d	<ul style="list-style-type: none"> <li>(edir-merge-revisions DIR1 REGEXP &amp;optional MERGE-AUTOSTORE-DIR)</li> <li>(ediff-merge-directory-revisions DIR1 REGEXP &amp;optional MERGE-AUTOSTORE-DIR)</li> </ul>	Merge versions of files in a given directory. <ul style="list-style-type: none"> <li>Ediff selects only the files that are under version control.</li> <li>Prompts for directory and regexp to identify files: if empty: selects all files.</li> </ul>
Merge versions of files in a directory using other versions as their ancestors	<f11> d e m D	<ul style="list-style-type: none"> <li>(edir-merge-revisions-with-ancestor DIR1 REGEXP &amp;optional MERGE-AUTOSTORE-DIR)</li> <li>(ediff-merge-directory-revisions-with-ancestor DIR1 REGEXP &amp;optional MERGE-AUTOSTORE-DIR)</li> </ul>	Merge versions of files in a given directory using other versions as ancestors. <ul style="list-style-type: none"> <li>Ediff selects only the files that are under version control.</li> <li>Prompts for directory and regexp to identify files: if empty: selects all files.</li> </ul>
Merge file commons to 2 directories	<f11> d e m c	<ul style="list-style-type: none"> <li>(edirs-merge DIR1 DIR2 REGEXP &amp;optional MERGE-AUTOSTORE-DIR)</li> <li>(ediff-merge-directories DIR1 DIR2 REGEXP &amp;optional MERGE-AUTOSTORE-DIR)</li> </ul>	Merge files common to two directories. <ul style="list-style-type: none"> <li>Run Ediff on a pair of directories, DIR1 and DIR2, merging files that have the same name in both.</li> <li>The third argument, REGEXP, is nil or a regular expression; only file names that match the regexp are considered.</li> <li>MERGE-AUTOSTORE-DIR is the directory in which to store merged files.</li> </ul>
Merge file commons to 2 directories with ancestors	<f11> d e m C	<ul style="list-style-type: none"> <li>(edirs-merge-with-ancestor DIR1 DIR2 ANCESTOR-DIR REGEXP &amp;optional MERGE-AUTOSTORE-DIR)</li> <li>(ediff-merge-directories-with-ancestor DIR1 DIR2 ANCESTOR-DIR REGEXP &amp;optional MERGE-AUTOSTORE-DIR)</li> </ul>	Merge files in directories DIR1 and DIR2 using files in ANCESTOR-DIR as ancestors. <ul style="list-style-type: none"> <li>Ediff merges files that have identical names in DIR1, DIR2.</li> <li>If a pair of files in DIR1 and DIR2 doesn't have an ancestor in ANCESTOR-DIR, Ediff will merge without ancestor.</li> <li>The fourth argument, REGEXP, is nil or a regular expression; only file names that match the regexp are considered.</li> </ul>
Merge 2 versions of visited file	<f11> d e m r	(ediff-merge-revisions &optional FILE STARTUP-HOOKS MERGE-BUFFER-FILE)	Merge two versions of the file visited by the current buffer.
Merge 2 versions of visited file with ancestor	<f11> d e m R	(ediff-merge-revisions-with-ancestor &optional FILE STARTUP-HOOKS MERGE-BUFFER-FILE)	Merge two versions of the file visited by the current buffer with ancestor.
Specialized Ediff	Some packages provide specialization of Ediff-based comparisons.		
ParInfer EDiff Diff current code before/after ParInfer modifications 🔗 - Emacs Lisp	<ul style="list-style-type: none"> <li>&lt;f12&gt; a D</li> <li>&lt;f11&gt; SPC l a D</li> </ul>	(parinfer-diff)	Diff current code and the code after applying Indent Mode in Ediff. Use this to browse and apply the changes. <ul style="list-style-type: none"> <li>  Requires the <b>parinfer package</b>. ⚠ This is an <b>obsolete package</b>.</li> <li> PEL activates this when the <b>pel-use-parinfer</b> user option is set to <b>t</b>.</li> </ul>
Smerge	Use the built-in smerge package to edit files that contain 3-way merge conflict annotations placed by a 3-way merge operation that requires your input. Start the merge session by executing the smerge-start-session or have it start <ul style="list-style-type: none"> <li>If you want to automatically launch a smerge session on files that contain diff conflict annotations, set <b>pel-use-smerge</b> to auto.               <ul style="list-style-type: none"> <li>Conflict annotation string is a string like “&lt;&lt;&lt;&lt;&lt;&lt;&lt;&lt;” that starts at the beginning of a line.</li> </ul> </li> </ul> When a merge session is active: <ul style="list-style-type: none"> <li>the merge menu is available. See <u>Menus</u></li> <li>with PEL, the <b>&lt;f6&gt; s</b> key acts as a prefix to the smerge commands.</li> </ul>		
Start a smerge session	<f11> d s	(smerge-start-session)	Turn on ‘smerge-mode’ and move point to first conflict marker. If no conflict maker is found, turn off ‘smerge-mode’.
Move to next conflict	<ul style="list-style-type: none"> <li>C-c ^ n</li> <li>&lt;f6&gt; s n</li> </ul>	(smerge-next &optional COUNT)	Go to the next COUNT’t’h conflict.
Move to previous conflict	<ul style="list-style-type: none"> <li>C-c ^ p</li> <li>&lt;f6&gt; s p</li> </ul>	(smerge-prev &optional COUNT)	Go to the previous COUNT’t’h conflict
Keep all	<ul style="list-style-type: none"> <li>C-c ^ a</li> <li>&lt;f6&gt; s a</li> </ul>	(smerge-keep-all)	Concatenate all versions.
Revert to base	<ul style="list-style-type: none"> <li>C-c ^ b</li> <li>&lt;f6&gt; s b</li> </ul>	(smerge-keep-base)	Revert to the base version.
Keep current	<ul style="list-style-type: none"> <li>C-c ^ RET</li> <li>&lt;f6&gt; s RET</li> </ul>	(smerge-keep-current)	Use the current (under the cursor) version.

