















# Undo/Redo/Repeat/Command Prefix Arguments

| Operation   | Keystroke  | Function   | Note   |
|---|--|--|--|
| <b>Undo</b>   | <p>Emacs <b>standard undo</b> mechanism is powerful but unusual for users of other editors because:</p> <ul style="list-style-type: none"> <li>Emacs standard undo command allows undoing an undo operation, providing redo capability. This comes with a complexity cost because the user need to think about wether it wants to undo or redo and may need to type something just to change from undo to redo or vice-versa. For a lot of people this quickly becomes difficult to manage.</li> </ul> <p>Fortunately for people that find it difficult to handle Emacs default undo, there is a nice external package: the <b>undo-tree</b> package.</p> <ul style="list-style-type: none"> <li>The undo-tree package provides 2 different command for undo and redo. The undo only undoes. And the commands maintain a tree of undo/redo operations that can be shown visually with the extra ability to select undo/redo history branches.</li> </ul> <p>PEL customization allows selection of either mechanism:</p> <ul style="list-style-type: none"> <li>If the <b>pel-use-undo-tree</b> user option is <b>t</b>, then PEL uses the undo-tree package, otherwise it uses the standard Emacs undo.</li> </ul> <ul style="list-style-type: none"> <li>The first row below shows the PEL bindings to standard undo. The rows after that describe the binding used when the undo-tree package is activated.</li> <li>Note that even when undo-tree package is activated and used, it is possible to disable the undo-tree-mode globally or locally using the global-undo-tree-mode and undo-tree-mode commands. When that's the case, the bindings use what is described on the first row. But it's also possible to switch back to the undo-tree-mode using the same commands again.</li> </ul> <p><b>Restrict undo to a region:</b></p> <p>One very interesting aspect of the Emacs undo system is that we can restrict it to an area of the buffer by first selecting a region and then performing the undo operation while the region is active/visible. Nothing outside the region will be affected by the undo commands. The undo actions outside of the marked region are not lost; once the there is no marked region further undo actions will undo changes in the text.</p> <p>👉🔪 <b>C--</b> is normally bound to (undo) but the PEL setup sets it to <b>(negative-argument)</b> to help editing motion flow. <b>M-_</b> and <b>C-_</b> are also bound to <b>(negative-argument)</b> for the same reasons, not to the undo or redo commands.</p> |  |  |
| <b>Open this PDF file.</b><br>See also: <a href="#">🔗 Help/Info</a>   | <b>&lt;f11&gt; u &lt;f1&gt;</b>  | <b>(pel-help-pdf &amp;optional OPEN-WEB-PAGE)</b>  | Open the <a href="#">🔗 Undo/Redo/Repeat/Arg</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><a href="#">🔗 Customize</a> PEL undo control</b>   | <b>&lt;f11&gt; u &lt;f2&gt;</b>  | <b>(pel-customize-pel &amp;optional OTHER-WINDOW)</b>  | Customize PEL undo support. <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><a href="#">🔗 Customize</a> Emacs undo control</b>   | <b>&lt;f11&gt; u &lt;f3&gt;</b>  | <b>(pel-customize-library &amp;optional OTHER-WINDOW)</b>  | Customize Emacs undo support: undo, undo-tree.   |
| <b>Undoing Changes</b>  | <p>The following commands are available to undo changes to a buffer.</p> <ul style="list-style-type: none"> <li>It's also possible to <b>undo all changes to a buffer associated with a file</b> by reverting the content of the buffer to the content of the file.                             <ul style="list-style-type: none"> <li>See the File Management section: <a href="#">🔗 File-mngt</a></li> </ul> </li> </ul> <p>📦 PEL provides the powerful <b>undo-tree</b> external package instead of the default undo  when the <b>pel-use-undo-tree</b> user option is set to <b>t</b>.</p> <p>⚠ Note, however, that undo-tree may break during intensive redo operations. Use with care.</p> <p>🔧 With Emacs 28 better alternatives exist and will be incorporated by PEL soon. You may want to use simple-undo with Emacs 28. See below.</p>   |  |  |
| <b>Simple undo/redo</b>   | Since Emacs 28, undo-redo is available and complements the <b>undo-only</b> that was available since Emacs 22, allowing the use of undo/redo system more familiar to most people not already familiar with Emacs.  |  |  |
| <b>Simple undo</b><br>: pel-use-undo-tree = nil<br>: pel-use-simple-undo = t<br>: Emacs >= 28   | <ul style="list-style-type: none"> <li><b>C- /</b></li> <li><b>C-x u</b></li> <li><b>M-u</b></li> <li><b>C-z</b></li> <li><b>s-z</b></li> <li><b>⌘-z</b></li> </ul>  | <b>(undo-only &amp;optional ARG)</b>   | Undo some previous changes. <ul style="list-style-type: none"> <li>Repeat this command to undo more changes.</li> <li>A numeric ARG serves as a repeat count.</li> <li>Contrary to 'undo', this will not redo a previous undo.</li> </ul>  |
| <b>Simple redo</b><br>: pel-use-undo-tree = nil<br>: pel-use-simple-undo = t<br>⚠ For Emacs >= 28   | <ul style="list-style-type: none"> <li><b>C- ?</b></li> <li><b>C-M-_</b></li> <li><b>M-U</b></li> <li><b>&lt;f11&gt; u r</b></li> </ul>  | <b>(undo-redo &amp;optional ARG)</b>   | Undo the last ARG undos, i.e., redo the last ARG changes. <ul style="list-style-type: none"> <li>Interactively, ARG is the prefix numeric argument and defaults to 1.</li> <li>It undoes previous undo commands, but doesn't record itself as an undoable command, as opposed to the original <b>undo</b> (shown above).</li> </ul>  |
| <b>Classic Emacs Undo/Redo</b>  | Original/standard Emacs undo/redo system. As explained in Emacs Manual Undo section: <i>"Any command other than an undo command breaks the sequence of undo commands. Starting from that moment, the entire sequence of undo commands that you have just performed are themselves placed into the undo record."</i>  |  |  |
| <b>Undo</b><br><br>: pel-use-undo-tree = nil<br>: pel-use-simple-undo = nil   | <ul style="list-style-type: none"> <li><b>C- /</b></li> <li><b>C-x u</b></li> <li><b>M-u</b></li> <li><b>C-z</b></li> <li><b>s-z</b></li> <li><b>⌘-z</b></li> </ul>  | <b>(undo &amp;optional ARG)</b>  | Undo last changes <b>using standard Emacs</b> undo. Also used to undo an undo, causing a redo! <ul style="list-style-type: none"> <li>A numeric ARG serves as a repeat count.</li> </ul>  PEL uses it when the <b>pel-use-undo-tree</b> user option is nil (the default). <p>👉 If you are not familiar with standard Emacs undo, please first read about it before using it.</p> <ul style="list-style-type: none"> <li>It might seems strange at first to use the same key to undo and redo.</li> </ul>  |
| <b>Undo</b><br><br> <br>: pel-use-undo-tree = t<br>: pel-use-simple-undo = nil | <ul style="list-style-type: none"> <li><b>&lt;f11&gt; u u</b></li> </ul>   | <b>(pel-undo &amp;optional ARG)</b> <ul style="list-style-type: none"> <li><b>(undo-tree-undo &amp;optional ARG)</b></li> <li><b>(undo &amp;optional ARG)</b></li> </ul> | Undo changes. Does not redo. <ul style="list-style-type: none"> <li>A numeric ARG serves as a repeat count.</li> <li>In Transient Mark mode when the mark is active, only undo changes within the current region. Similarly, when not in Transient Mark mode, just <b>C-u</b> as an argument limits undo to changes within the current region.</li> <li><b>C- /</b> only works in graphics mode</li> <li><b>s-z</b> and <b>⌘-z</b> only work in macOS graphic mode. Note: with PEL, <b>⌘-z</b> is <b>s-z</b>.</li> </ul> <p>👉 ⚠ With PEL, when <b>pel-use-undo-tree</b> is <b>t</b>, this key is bound to <b>pel-undo</b> which uses <b>undo-tree-undo</b> by default.</p> <ul style="list-style-type: none"> <li>You can, however toggle the local or global <b>undo-tree-mode</b> by issuing the <b>M-x global-undo-tree-mode</b> or <b>M-x undo-tree-mode</b>. If the undo-tree-mode is not set in the buffer, PEL will use the Emacs standard <b>undo</b> command until the undo-tree-mode is re-enabled.</li> </ul> |
| <b>Redo</b><br><br> <br>: pel-use-undo-tree = t<br>: pel-use-simple-undo = nil | <ul style="list-style-type: none"> <li><b>M-U</b></li> <li><b>&lt;f11&gt; u r</b></li> <li><b>s-Z</b></li> <li><b>⌘-Z</b></li> </ul>   | <b>(pel-redo &amp;optional ARG)</b> <ul style="list-style-type: none"> <li><b>(undo-tree-redo &amp;optional ARG)</b></li> <li><b>(undo &amp;optional ARG)</b></li> </ul> | Redo changes. A numeric ARG serves as a repeat count. <ul style="list-style-type: none"> <li>In Transient Mark mode when the mark is active, only redo changes within the current region. Similarly, when not in Transient Mark mode, just <b>C-u</b> as an argument limits redo to changes within the current region.</li> <li><b>s-z</b> and <b>⌘-z</b> only works in graphics mode</li> <li>Note: with PEL, <b>⌘-Z</b> is <b>s-z</b>.</li> </ul> <p>👉 ⚠ With PEL, when <b>pel-use-undo-tree</b> is <b>t</b>, this key is bound to <b>pel-redo</b> which uses <b>undo-tree-redo</b> by default.</p> <ul style="list-style-type: none"> <li>You can, however toggle the local or global undo-tree-mode by issuing the <b>M-x global-undo-tree-mode</b> or <b>M-x undo-tree-mode</b>. If the undo-tree-mode is not set in the buffer, PEL will use the Emacs standard <b>undo</b> command until the undo-tree-mode is re-enabled.</li> </ul>   |
| <b>Show undo tree</b><br><br> <br>: pel-use-undo-tree = t                      | <b>&lt;f11&gt; u v</b>   | <b>(undo-tree-visualize)</b>   | Show undo tree of current buffer.<br>The "undo tree" keys are: <ul style="list-style-type: none"> <li><b>&lt;up&gt;/&lt;down&gt;</b> : move up/down the undo tree nodes</li> <li><b>&lt;right&gt;/&lt;left&gt;</b> : changes branch when at a branch root</li> <li><b>s</b> : toggle selection mode: normally moving restores right away, this other mode allows you to move in the tree without changing the controlled buffer until RET is typed.</li> <li><b>d</b> : shows diff between buffer and currently selected undo node!!</li> <li><b>t</b> : toggles showing relative timestamp on undo nodes</li> </ul> <p>👉 ⚠ With PEL, this is available when <b>pel-use-undo-tree</b> is <b>t</b> but also while the global or local undo-tree-mode is active, which it should be unless you explicitly disabled one of these via the global-undo-tree-mode or undo-tree-mode commands. If that is the case, re-enable the undo-tree-mode and you will be able to use the command.</p>                                   |

| Operation   | Keystroke  | Function  | Note   |
|---|--|---|--|
| <b>Switch branch of undo tree</b><br><br>  : pel-use-undo-tree = t | <f11> u x  | (undo-tree-switch-branch BRANCH)                              | Switch to a different BRANCH of the undo tree. <ul style="list-style-type: none"> <li>This will affect which branch to descend when *redoing* changes using ‘undo-tree-redo’.</li> </ul>   |
|   |   With PEL, this is available when <b>pel-use-undo-tree</b> is <b>t</b> but also while the global or local undo-tree-mode is active, which it should be unless you explicitly disabled one of these via the global-undo-tree-mode or undo-tree-mode commands. If that is the case, re-enable the undo-tree-mode and you will be able to use the command. |   |  |
| Goto last change  | <f11> u \  | (goto-last-change &optional MARK-POINT MINIMAL-LINE-DISTANCE) | Set point to the position of the last change. <ul style="list-style-type: none"> <li>Consecutive calls set point to the position of the previous change.</li> <li>With a prefix arg (optional arg MARK-POINT non-nil), set mark so <b>C-x C-x</b> will return point to the current position.</li> </ul>  This requires the <a href="#">goto-last-change.el</a> package.<br> Under PEL set the <a href="#">pel-use-goto-last-change</a> user option to activate this. |
| Enable undo in buffer   |  | (buffer-enable-undo &optional BUFFER)                         | Enable undo recording in the current buffer. No effect if the undo was already recorded as its the case for all buffers except some (like the buffers that have a name that starts with a space). <ul style="list-style-type: none"> <li>Interactively it’s not possible to pass an argument.</li> </ul>   |
| Disable undo in buffer  |  | (buffer-disable-undo &optional BUFFER)                        | Disable undo in current buffer. Deletes all previous undo information for that buffer if it previously existed. <ul style="list-style-type: none"> <li>No effect if undo was previously disabled.</li> <li>Interactively it’s not possible to pass an argument.</li> </ul>   |
| Repeat  | The following commands can repeat the last command and show  |   |  |
| Repeat last operation   | <ul style="list-style-type: none"> <li><b>C-x z</b></li> <li>&lt;f5&gt;</li> </ul>   | (repeat REPEAT-ARG)   | Repeat most recently executed command. <ul style="list-style-type: none"> <li>With a prefix argument, supply a prefix argument to that command. Otherwise, give the command the same prefix argument it was given before, if any.</li> <li>When using <b>C-x z</b> to perform repeat, once one <b>C-x z</b> has been typed for the first repeat, type <b>z</b> again to again repeat. Typing <b>z</b> continuously continue to repeat last command (any command, even undo).</li> <li>PEL provides the &lt;f5&gt; key to perform repeat.</li> </ul>  |
| Redo/edit last complex command executed   | <ul style="list-style-type: none"> <li><b>C-x Esc Esc</b></li> <li><b>C-x M-Esc</b></li> <li><b>C-x M-:</b></li> </ul>   | (repeat-complex-command ARG)                                  | Edit and re-evaluate last complex command, or ARGth from last. <ul style="list-style-type: none"> <li>A complex command is one which used the minibuffer. The command is placed in the minibuffer as a Lisp form for editing. The result is executed, repeating the command as changed.</li> <li>If the command has been changed or is not the most recent previous command it is added to the front of the command history.</li> <li>You can use the minibuffer history commands <b>M-n</b> and <b>M-p</b> to get different commands to edit and resubmit.</li> </ul>   |
| List command history<br>See also: <a href="#">⌘ Help</a>  | <f11> ? d H  | (list-command-history)  | List history of commands that used the minibuffer. <ul style="list-style-type: none"> <li>Show list of commands in the “Command History” buffer as a list of Emacs Lisp forms.</li> </ul>  |
| Command Arguments   | All Emacs keystrokes are bindings to an Emacs command, a function that supports interactive use. A lot of these commands have arguments. The user can provide these arguments by using the following key strokes <b>before</b> typing the needed command using the keys listed below.  |   |  |
| Prefix repeat N time  | M- <number> Keystroke  |   | Meta N, where N is a typed number, tells Emacs to repeat the next <i>Keystroke</i> operation N times. <ul style="list-style-type: none"> <li>N may be larger than 9: to pass a numeric prefix of 23 Hold the Meta key then type 2 followed by 3.</li> </ul>  |
| Repeat prefix   | C-u <number> Keystroke   | (universal-argument)  | <b>C-u</b> N, where N is a typed number, tells emacs to repeat the next operation N times. <ul style="list-style-type: none"> <li>Pressing <b>C-u</b> before executing a command is a way to pass extra information to that command.</li> <li>For example, <b>C-u 5 C-b</b> means move the cursor left 5 characters. Sometimes the extra information is just the fact that <b>C-u</b> was pressed.</li> </ul>  |
| Prefix repeat 4 times   | C-u  | (universal-argument)  | <b>C-u</b> alone stands for a flag, or a repeat factor of 4.   |
| Prefix repeat 16 times  | C-u C-u  | (universal-argument)  | <b>C-u C-u</b> means 4 * 4 = 16  |
| Prefix repeat 64 times  | C-u C-u C-u  | (universal-argument)  | <b>C-u C-u C-u</b> means 4*4*4 = 64 repeat.  |
| Negative argument   | <ul style="list-style-type: none"> <li><b>C--</b></li> <li><b>M--</b></li> <li><b>C-M--</b></li> <li><b>C-_</b></li> <li><b>M-_</b></li> </ul>   | (negative-argument)   | Begin a negative numeric argument for the next command. <ul style="list-style-type: none"> <li>If needed, <b>C-u</b> following digits or minus sign ends the argument.</li> <li>The PEL package also binds the Control and Meta underscore keys to the negative-argument function to help improve typing speed when entering negative arguments.</li> </ul>  |

## Undo — Reference

|  |  |
|--|--|
|  |  |
| <a href="#">GNU EMacs Lisp Manual — Command Overview</a>       | Describes that prior to executing a command Emacs runs <b>undo-boundary</b> to create undo boundary. |
| <a href="#">GNU Emacs Lisp Manual — Maintaining Undo Lists</a> | Describes the standard Emacs undo mechanism.   |
| <a href="#">Emacs undo-tree package</a>                        | Author’s we site, describes undo-tree.   |