↖ Topic Index

Undo/Redo/Repeat/Command Prefix Arguments

Operation	Keystroke	Function	Note
Undo, Redo, Repeat • Disable undo • Undo changes • Classic undo-redo ○ Show undo-tree • undo-propose • goto-last change • Repeat last command • Command arguments ○ Undo Reference Last updated on:			Emacs **standard undo** mechanism is powerful but unusual for users of other editors because: • 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. Fortunately for people that find it difficult to handle Emacs default undo, there are external packages that help for Emacs prior to 28. 📦 The external **undo-tree** package helps, but unfortunately it might lead to corrupted or lost data during complex undo/redo sequences. It 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. 📦 The external **undo-propose** package is not as powerful but also helps showing the history of complex undo/redo sessions. 📦 The external **vundo** package provides a visual undo tree. Emacs 28 introduces the the undo-redo that allows using a simpler undo/redo similar to other systems. **pel-use-undo-tree**: activate use of the external **undo-tree** package. ⚠️ **No longer recommended due to potential data corruption**. **pel-use-undo-propose**: activate use of the external **undo-propose** package. ☛ Use it only when using Classic Emacs undo. It can help learn it too! **pel-use-simple-undo**: activate use of undo-only and undo-redo instead of classic undo. Emacs >= 28 only. **pel-use-vundo**: activate use of the external **vundo** package. Emacs >= 28 only. **Restrict undo to a region:** ☝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. ☝✂ `C--` is normally bound to (undo) but the PEL setup sets it to (**negative-argument**) to help editing motion flow. `M-_` and `C-_` are also bound to (**negative-argument**) for the same reasons, not to the undo or redo commands. 2025-08-25
Open this PDF Σ Undo/Redo/Repeat/Arg file	`<f11> u <f1>`	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the Σ **Undo/Redo/Repeat/Arg** local PDF. If the prefix argument (like `C-u` or `M--`) is used, then it opens the remote GitHub hosted raw PDF instead. If the **pel-flip-help-pdf-arg** user-option is set it's the other way around.
Σ **Customize PEL undo control**	`<f11> u <f2>`	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL undo support: undo, undo-propose, undo-tree and vundo. • If OTHER-WINDOW is non-nil (use `C-u )`, display in other window.
Σ **Customize Emacs undo control**	`<f11> u <f3>`	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs undo support: undo, undo-propose, undo-tree, vundo (for Emacs >= 28).
Show active undo status	`<f11> u ?`	(**pel-undo-info** &optional APEND)	Print values of undo//redo control variable and user-options in a *pel-undo-info* **help-mode** buffer. • Clear previous buffer content by default. Use prefix arg to append to buffer instead. ☝Use this to see the values of relevant user-options and buttons to gain access to their customization.
Disabling Undo			Undo tracking is activated by default but can be disabled and re-enabled. See Σ **File-mngt** for buffer reverting command to restore file's content in buffer.
Disable undo in buffer	`M-x buffer-disable-undo`	(**buffer-disable-undo** &optional BUFFER)	Disable undo in current buffer. Deletes all previous undo information for that buffer if it previously existed. • No effect if undo was previously disabled. • Interactively it's not possible to pass an argument.
Enable undo in buffer	`M-x buffer-enable-undo`	(**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). • Interactively it's not possible to pass an argument.
Undoing Changes			PEL provides support for several external package and control that affect the behaviour of Emacs undo operation. See the options above. PEL adjust key bindings according to the package selected by customization. Use `<f11> u <f2>` to select the one you prefer.
Simple undo/redo			Since Emacs 28, undo-redo is available and complements the undo-only that was available since Emacs 22, allowing the use of undo/redo system more familiar to most people not already familiar with Emacs.
Simple undo : pel-use-undo-tree = nil : pel-use-simple-undo = t : Emacs >= 28	• `C-/` • `C-x u` • `M-u` • `C-z` • `s-z` • `⌘-z`	(**undo-only** &optional ARG)	Undo some previous changes. • Repeat this command to undo more changes. • A numeric ARG serves as a repeat count. • Contrary to 'undo', this will not redo a previous undo.
Simple redo : pel-use-undo-tree = nil : pel-use-simple-undo = t ⚠️ For Emacs >= 28	• `C-?` • `C-M-_` • `M-U` • `<f11> u r`	(**undo-redo** &optional ARG)	Undo the last ARG undos, i.e., redo the last ARG changes. • Interactively, ARG is the prefix numeric argument and defaults to 1. • It undoes previous undo commands, but doesn't record itself as an undoable command, as opposed to the original **undo** (shown above).
Classic Emacs Undo/Redo			Original/standard Emacs undo/redo system. As explained in Emacs Manual Undo section: "*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.*" • 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.
Undo : pel-use-undo-tree = nil : pel-use-simple-undo = nil : pel-use-vundo = nil|t	• `C-/` • `C-x u` • `M-u` • `C-z` • `s-z` • `⌘-z`	(**undo** &optional ARG)	Undo last changes **using standard Emacs** undo. Also used to undo an undo, causing a redo! • A numeric ARG serves as a repeat count. PEL uses it when the **pel-use-undo-tree** user option is nil (the default). ☝If you are not familiar with standard Emacs undo, please first read about it before using it. • It might seems strange at first to use the same key to undo and redo.
Undo 📦 : pel-use-undo-tree = t : pel-use-simple-undo = nil : pel-use-vundo = nil	• `<f11> u u`	(**pel-undo** &optional ARG) • (**undo-tree-undo** &optional ARG) • (**undo** &optional ARG)	Undo changes. Does not redo. • A numeric ARG serves as a repeat count. • In Transient Mark mode when the mark is active, only undo changes within the current region. Similarly, when not in Transient Mark mode, just **C-u** as an argument limits undo to changes within the current region. • `C-/` only works in graphics mode • `s-z` and `⌘-z` only work in macOS graphic mode. Note: with PEL, `⌘-z` is `s-z`.
			☝⚠️ With PEL, when **pel-use-undo-tree** is **t**, this key is bound to **pel-undo** which uses **undo-tree-undo** by default. • You can, however toggle the local or global **undo-tree-mode** by issuing the `M-x global-undo-tree-mode` or `M-x undo-tree-mode`. If the undo-tree-mode is not set in the buffer, PEL will use the Emacs standard **undo** command until the undo-tree-mode is re-enabled.
Redo 📦 : pel-use-undo-tree = t : pel-use-simple-undo = nil : pel-use-vundo = nil	• `M-U` • `<f11> u r` • `s-Z` • `⌘-Z`	(**pel-redo** &optional ARG) • (**undo-tree-redo** &optional ARG) • (**undo** &optional ARG)	Redo changes. A numeric ARG serves as a repeat count. • In Transient Mark mode when the mark is active, only redo changes within the current region. Similarly, when not in Transient Mark mode, just **C-u** as an argument limits redo to changes within the current region. • `s-Z` and `⌘-Z` only works in graphics mode • Note: with PEL, `⌘-Z` is `s-Z`.
			☝⚠️ With PEL, when **pel-use-undo-tree** is **t**, this key is bound to **pel-redo** which uses **undo-tree-redo** by default. • You can, however toggle the local or global undo-tree-mode by issuing the `M-x global-undo-tree-mode` or `M-x undo-tree-mode`. If the undo-tree-mode is not set in the buffer, PEL will use the Emacs standard **undo** command until the undo-tree-mode is re-enabled.

Operation	Keystroke	Function	Note
Show undo tree 📦 🔄 : pel-use-undo-tree = t : pel-use-vundo = nil	**<f11> u v**	(**undo-tree-visualize**)	Show undo tree of current buffer. The *undo tree* keys are: • **<up>/<down>** : move up/down the undo tree nodes • **<right>/<left>** : changes branch when at a branch root • **s** : 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. • **d** : shows diff between buffer and currently selected undo node!! • **t** : toggles showing relative timestamp on undo nodes
	☝⚠️ With PEL, this is available when **pel-use-undo-tree** is **t** 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.		
📦 🔄 : pel-use-vundo = t ⚠️ For Emacs >= 28	When **pel-use-vundo** is active (set to t), then the undo command is bound to **<f11> u v**		
	<f11> u v	(**vundo**)	Display visual undo for the current buffer. • For Emacs running in graphics mode this is inside a pop-up window. For Emacs running in terminal mode, this is show at the top or bottom (the default) of the Emacs window. It is customizable. • While displaying the undo tree, you can use the cursor keys to navigate through the undo tree nodes and the original buffer is updated accordingly. • Type **RET** to accept and stop showing the undo tree. Also: • **f** to go forward • **b** to go backward • **n** to go to the node below when you at a branching point • **p** to go to the node above • **a** to go back to the last branching point • **e** to go forward to the end/tip of the branch • **q** to quit, you can also type **C-g**
Switch branch of undo tree 📦 🔄 : pel-use-undo-tree = t : pel-use-vundo = nil	**<f11> u x**	(**undo-tree-switch-branch** BRANCH)	Switch to a different BRANCH of the undo tree. • This will affect which branch to descend when *redoing* changes using 'undo-tree-redo'.
	☝⚠️ With PEL, this is available when **pel-use-undo-tree** is **t** 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.		
undo-propose	📦 **undo-propose** creates an undo history buffer where only undo key bindings to undo/redo are allowed. Use it to replay your undo and restore the buffer to a state that holds the text want to retrieve. You can then copy it back to the original (or another) buffer. Requires 🔄 **pel-use-undo-propose** to activate the external **undo-propose** package.		
Open a undo-propose temporary buffer - undo if inside Undo-Propose buffer	**C-c u**	(**undo-propose**)	Open a temporary *Undo Propose* buffer copy of the current buffer to navigate its undo history. • Copies 'current-buffer' and 'buffer-undo-list' to a new temporary buffer, which is read-only except for undo commands. After finished undoing, type: • **C-c C-c** to accept the chain of undos, or • **C-c C-s** to copy the buffer but squash the undo's into a single edit event event. • To cancel, type **C-c C-k** • To view an ediff type **C-c C-d** If already inside an 'undo-propose' buffer, this will simply call 'undo'.
Goto Last Change	📦 This requires the **goto-last-change.el** package. 🔄 Under PEL set the **pel-use-goto-last-change** user option to activate this.		
Goto last change	• **<f11> u ** • **C-x C-** • **<f6> **	(**goto-last-change** &optional MARK-POINT MINIMAL-LINE-DISTANCE)	Set point to the position of the last change in the current buffer. • Consecutive calls set point to the position of the previous change. • With a prefix arg (optional arg MARK-POINT non-nil), set mark so **C-x C-x** will return point to the current position.
Repeat Command	The following commands can repeat the last command and show		
Repeat last operation	• **C-x z** • **<f5>**	(**repeat** REPEAT-ARG)	Repeat most recently executed command. • 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. • When using **C-x z** to perform repeat, once one **C-x z** has been typed for the first repeat, type **z** again to again repeat. Typing **z** continuously continue to repeat last command (any command, even undo). • PEL provides the **<f5>** key to perform repeat.
Redo/edit last complex command executed	• **C-x Esc Esc** • **C-x M-Esc** • **C-x M-:**	(**repeat-complex-command** ARG)	Edit and re-evaluate last complex command, or ARGth from last. • 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. • If the command has been changed or is not the most recent previous command it is added to the front of the command history. • You can use the minibuffer history commands **M-n** and **M-p** to get different commands to edit and resubmit.
List command history See also: **Help**	**<f11> ? d H**	(**list-command-history**)	List history of commands that used the minibuffer. • Show list of commands in the *Command History* buffer as a list of Emacs Lisp forms.
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 **before** 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 ***Keystroke*** operation N times. • N may be larger than 9: to pass a numeric prefix of 23 Hold the Meta key then type 2 followed by 3.
Repeat prefix	**C-u** <*number*> ***Keystroke***	(**universal-argument**)	**C-u** N, where N is a typed number, tells emacs to repeat the next operation N times. • Pressing **C-u** before executing a command is a way to pass extra information to that command. • For example, **C-u 5 C-b** means move the cursor left 5 characters. Sometimes the extra information is just the fact that **C-u** was pressed.
Prefix repeat 4 times	**C-u**	(**universal-argument**)	**C-u** alone stands for a flag, or a repeat factor of 4.
Prefix repeat 16 times	**C-u C-u**	(**universal-argument**)	**C-u C-u** means 4 * 4 = 16
Prefix repeat 64 times	**C-u C-u C-u**	(**universal-argument**)	**C-u C-u C-u** means 4*4*4 = 64 repeat.
Negative argument	• **C--** • **M--** • **C-M--** • **C-_** • **M-_**	(**negative-argument**)	Begin a negative numeric argument for the next command. • If needed, **C-u** following digits or minus sign ends the argument. • 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.

Undo — Reference

GNU EMacs Lisp Manual — Command Overview	Describes that prior to executing a command Emacs runs **undo-boundary** to create undo boundary.
GNU Emacs Lisp Manual — Maintaining Undo Lists	Describes the standard Emacs undo mechanism.
Emacs undo-tree package	Author's we site, describes undo-tree.