Undo/Redo/Repeat/Command Prefix Arguments

Operation	Keystroke	Function	Note
Undo			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 is a nice external package: the **undo-tree** package. • 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. PEL customization allows selection of either mechanism: • If the **pel-use-undo-tree** user option is **t**, then PEL uses the undo-tree package, otherwise it uses the standard Emacs undo. • 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. • 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. **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. 📦 PEL uses the **undo-tree package** instead of the default undo. Under PEL activate the undo-tree package by setting the **pel-use-undo-tree** user option to **t**.
Open this PDF file. See also: **∑ Help/Info**	**<f11> u <f1>**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the local copy of the **∑ Undo/Redo/Repeat/Arg** PDF file unless a command prefix (like **C-u**) was used. In that case it opens the Github-hosted file instead.
∑ Customize PEL undo control	**<f11> u <f2>**	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL undo support. • If OTHER-WINDOW is non-nil (use **C-u**), display in other window.
∑ Customize Emacs undo control	**<f11> u <f3>**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs undo support: undo, undo-tree.
Undo : pel-use-undo-tree = nil	• **C-/** • **C-x u** • **M-u** • **C-z** • **s-z** • **⌘-z** • **<f11> u u**	(**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-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**. PEL uses this when the **pel-use-undo-tree** user option is t. 📦 PEL uses the **undo-tree package** instead of the default undo. ☝⚠ 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	• **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**. 📦 PEL uses the **undo-tree package** instead of the default undo. Under PEL activate the undo-tree package by setting the **pel-use-undo-tree** user option to **t**. ☝⚠ 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.
Show undo tree : pel-use-undo-tree = t	**<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 📦 PEL uses the **undo-tree package** instead of the default undo. Under PEL activate the undo-tree package by setting the **pel-use-undo-tree** user option to **t**. ☝⚠ 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.
Switch branch of undo tree : pel-use-undo-tree = t	**<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'. 📦 PEL uses the **undo-tree package** instead of the default undo. Under PEL activate the undo-tree package by setting the **pel-use-undo-tree** user option to **t**. ☝⚠ 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.

Operation	Keystroke	Function	Note
Goto last change	**<f11> u **	(**goto-last-change** &optional MARK-POINT MINIMAL-LINE-DISTANCE)	Set point to the position of the last change. • 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. 📦 This requires the **goto-last-change.el** package. Under PEL set the **pel-use-goto-last-change** 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). • Interactively it's not possible to pass an argument.
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. • No effect if undo was previously disabled. • Interactively it's not possible to pass an argument.
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.
Undo all changes made since last saving the file	☛ It's also possible to undo all changes to a buffer associated with a file by reverting the content of the buffer to the content of the file. See the File Management section.		
Repeat			
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).
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.
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-_** • **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.