⬉ Topic Index

Windows — Managing Emacs Windows

Operation	Keystroke	Function	Note

Window Operations
 	
See also:
• ⅀ **Customize**
• ⅀ **Key-Chords**
• ⅀ **Frames**
• ⅀ **Speedbar**
• **Emacs Lisp Windows section.**
 	
• Follow Mode
 	
☝ All window scrolling commands are described in the ⅀ **Scrolling** page.
	Emacs basic window management commands are bound to **C-x o**, **C-x 0**, **C-x 1**, **C-x 2** and **C-x 3** with some derivatives and support for multiple frames. These basic facilities can be extended by several built-in and external packages:
	• **windmove**, built-in, activated by PEL, with different key bindings to preserve ability to shift-mark when moving across text with cursor.
	• **winner**, also built-in, which provides the ability to restore previous window pane layouts. PEL activates it when **pel-use-winner** user option is t.
	• **layout-restore** 📦 PEL activates it with **pel-use-restore-layout** user-option set to **t**. This associates layouts to buffers. ⚠️🚧 conflicts with some modes.
	• **ace-window**, 📦 extends the **C-x o** command by displaying **Ace target in the windows' upper left corner** for quick navigation and access to buttons. PEL activates it when **pel-use-ace-window** user option is t.
	• **key-chord**, 📦 to activate dual-key chords to move across windows. PEL activates it when **pel-use-key-chord** user option is **t.**
	• Windows can be **dedicated** to specific buffers, for example by **Speedbar** (see ⅀ **Speedbar**).
	• Several windows with the same buffers can operate as a single flow with **follow mode.**
	PEL provides extra commands and key bindings:
	  • It adds several key bindings under the <f11> key prefix. These are available in both graphics and terminal modes.
	  • On macOS, in graphics mode only, the ⌘ key is mapped to the super prefix key (s-).
	  • ❖ On Windows, the **Menu key** is mapped to the hyper key. Below the ❖ icon is used to represent the Menu key under Windows.
	  • ☛ In graphics mode, mouse operations are available.
	    • They can also be enabled in terminal mode, with the xterm-mouse-mode enabled. With PEL, use **<f11><f12>** to toggle the xterm-mouse-mode.
	☝ Operations on windows can be applied to windows in other frames, whether Emacs is running in graphics mode or in terminal mode.
	In terminal mode only one frame is visible at a time though.		

Open this PDF file. See also: ⅀ **Help/Info**	**<f11> w <f1>**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the ⅀ **Windows** 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.
⅀ **Customize PEL window control**	**<f11> w <f2>**	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL Window support. • If OTHER-WINDOW is non-nil (use **C-u**), display in other window.
⅀ **Customize Emacs window control**	**<f11> w <f3>**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs Window support groups: windows, ace-window, ace-window-display, winner, windmove and windresize. ⚠️ windresize does not uses its own group. It places its customization inside the Emacs convenience group instead. PEL opens that group for it: look for Windresize user options there.
ace-window # on ⅀ **Mode Line**	With **ace-window-display-mode** user-option on, the window number is shown on the left of the mode-line. • Type **<f11> <f2> o ace-window-display-mode** to open the customize buffer to change it. ⚠️ Activating this will increase your Emacs init time. Instead, use **ace-window-display-mode**, **<f11> w #** , to activate it manually.		
Toggle showing ace-window # on window mode line	• **<f11> w #** • **<f11> M-l #**	(**ace-window-display-mode** &optional ARG)	Toggle the ace-window-display-mode, a minor mode that displays the ace window number of each window inside the left hand side of its mode line. 📦 Requires the **ace-window** external package. PEL use **pel-use-ace-window** .
PEL Window Hydra Quickly: • Navigate through windows • Swap windows • Close window [Kill buffer] • Create/Split normal/side/root windows • Resize window • Fit size to buffer content • Flip vertical/ horizontal layout • Change to previous/next layout • Display different buffer in window • Change window dedication settings • Change buffer in window • Recenter buffer • Dedicated window	📦 Needs **hydra** external package. PEL user option **pel-use-hydra** set to **t** activate it & create a Hydra to speed up navigation and management of windows. **To start this hydra**, hit the **<f7>** key, then hit one of the listed hydra keys once or several times. • Then follow by typing the PEL Window Hydra keys, shown below. You can hit several different in succession without having to type the **<f7>** prefix again. • While active the Hydra Hint is shown in the minibuffer (as shown below). Type the **?** key to toggle the hint info off or back on. • To have the Hydra hint off when the Hydra activates set the **hydra-is-helpful** user option to nil (but then you can still toggle it on/off with **?**. ☝ You can use several other commands key sequences while the hydra is active. **To cancel the Hydra** hit the **<f7>** key again. ☝ Use the **q** key to quit from buffers that can be dismissed like the *Help* buffer. It also changes the buffer visible in the normal windows. You can also use **b** and **B** to change the buffer currently visible in the current window. ☝ The **windresize** command (describe below) provides an alternative for most of the commands (not all) available in this Hydra. ☝ The **ace-window** command bound to **C-x o** key provides a partially overlapping feature set but has a different key assignment than the Hydra **#** key. • The name of the PEL window hydra commands are not listed below. They all have a name that begins with **pel-∑wnd/** and ends with the same name as the command function listed in the Function column. For example, **pel-∑wnd/windmove-up** is bound to **<f7> <up>**. A snapshot of the window management hydra hint menu shows up in the minibuffer area as soon as one of its keys is pressed:		

Create	Split	Layout	Move	Resize	Close	Buffer	Other
M-8: side⬆	2: -	n: next layout	<up>: ⬆	=: balance	0: this	K: kill buf/win	<M-up>: scroll down
M-2: side⬇	3: |	p: last layout	<down>: ⬇	V: taller	o: other	k: kill buffer	<M-down>: scroll up
M-4: side⬅	C-<up>: ⬆	x: swap with.#	<left>: ⬅	v: shorter	1: others	b: next buffer	d: un/dedicate
M-6: side➡	C-<down>: ⬇	M-v: flip vert.	<right>: ➡	H: wider	C-S-<up>: above	B: prev buffer	M-?: hint cfg
M-r: root⬇	C-<left>: ⬅	M-h: flip horiz.	#: to #	h: narrower	C-S-<down>: below	5: recenter	?: hint
M-R: root➡	C-<right>: ➡			.: fit2buf	C-S-<left>: left		q: quit
					C-S-<right>: right		<f7>: cancel

Switch to the **pel-∑buffer Hydra** by typing **<f7><f7><f9>.** See ⅀ **Buffers**

Operation	Keystroke	Function	Note
Move point to other window **- C-u: swap** **- C-u C-u: delete**	• **C-x o** ✱ **<f7> #**	(**other-window** COUNT &optional ALL-FRAMES)	Select (move point) to other window. Select another window in cyclic ordering of windows. • With prefix argument consider all frames. • This is Emacs default behaviour for this key. And PEL's default: **pel-use-ace-window** = **nil.** Change it to activate the functionality described in next row.
• **Move to other window** • **Move to specified window Ace target** • **Operate on specified window** See also: ⅀ **Customize** Demo: **C'est la Z, video 5**		(**ace-window** ARG)	Move to (and possibly operate on) window selected by an Ace target code. 📦 Requires the **ace-window** external package. PEL downloads, installs and activates it when the ***pel-use-ace-window*** user option is set to **t**.

	• With only 2 windows in the current frame, move to the other window.
	• With 3 windows or more: display an **Ace target in the windows' upper left corner** that identifies the window target:
	  • Type the displayed window number to move to that window (which is all that's available with <f7> #) , or (with **C-x o**)
	    • type one of the following letters, followed by the target window number to move to the target window and operate on it:
	      • **x** - delete window
	      • **m** - swap windows
	      • **M** - move window
	      • **c** - copy window
	      • **j** - select buffer
	      • **n** - select the previous window
	      • **u** - select buffer in the other window
	      • **c** - split window fairly, either vertically or horizontally
	      • **v** - split window vertically
	      • **b** - split window horizontally
	      • **o** - maximize current window
	      • **?** - show these command bindings
	☝ This supports selecting windows in other frames (both in graphics and terminal mode)
	  • In graphics mode the other Emacs frames are in other OS *window*.
	  • In text terminal mode, other Emacs frames are hidden (as they occupy the exact same OS window): just one Emacs frame is displayed.
	☝ An argument can be used to perform more operations:
	  • To force a window number prompt, use any negative prefix (including just typing **C- -** alone). Useful with several frames when current frame has 1 or 2 windows active.
	  • Prefixed with one **C-u**, does a **swap** between the selected window and the current window, so that the selected buffer moves to current window (and current buffer moves to selected window). The PEL **<f11> w x** key does the same (but does not prompt when there are only 2 windows.)
	  • Prefixed with two **C-u**'s, **deletes** the window identified by the window number.		

Operation	Keystroke	Function	Note
Move point to next window • **can specify all frames**	`<f11> w o`	(**pel-other-window** &optional ALL-FRAMES)	Move to other window, like the original other-window. • With any prefix argument consider all frames. Without argument move only within current frame. • Useful when '**other-window**' has been remapped to something like 'ace-window' and want to see where the *next* window is.
Move point to previous window • **can specify all frames**	`<f11> w O`	(**pel-other-window-backward** &optional N)	Select Nth previous window. • n defaults to 1 : meaning direct previous window. • with negative n: move as (abs n) but consider all frames. If n is positive consider only current frame. • This is the inverse of what does the standard (other-window). • This command might be useful when ace-window is not used.
Move point to identified window • **Esc-cursor keys for windmove**	Along with several other key bindings, PEL creates the `<Esc>`-cursor key bindings described below. In some circumstances, these key bindings can conflict with some other bindings, for example in Org-mode these keys can be translated to Meta-cursor keys that are bound to Org-mode operations. PEL provides the following user options to control the key bindings: • **pel-windmove-on-esc-cursor** controls the `<Esc>` bindings, it is on by default on macOS and Windows, but off on Linux. • This affects the behaviour of the <Esc> cursor key bindings in org buffer as well to ensure a regular navigation across all buffers. • ⚠️ Several Linux distros map `C-M-` bindings such as `C-M-<right>` and `C-M-<left>` If this is not the case for your Linux system, you can activate this, otherwise don't because it will prevent you from using the `Esc C-` bindings in replacement for the `C-M-` bindings you need to access several Emacs commands. • **pel-windmove-on-f1-cursor** controls the `<f1>` binding, also on by default.		
Move to window above	• `<f11> <up>` • `<f1> <up>` • `<Esc> <up>` • `⌘-<up>` • `❖-<up>` ✱ `<f7> <up>` • `yu`	(**windmove-up** &optional ARG)	Select the window above the current one. • With no prefix argument, or with prefix argument equal to zero, "up" is relative to the position of point in the window; otherwise it is relative to the left edge (for positive ARG) or the right edge (for negative ARG) of the current window. • If no window is at the desired location, an error is signalled. With PEL, the `yu` key-chord is also available when key-chord is available and active. See Σ **Key-Chords**.
Move to window below	• `<f11> <down>` • `<f1> <down>` • `<Esc> <down>` • `⌘-<down>` • `❖-<down>` ✱ `<f7> <down>` • `bn`	(**windmove-down** &optional ARG)	Select the window below the current one. • With no prefix argument, or with prefix argument equal to zero, "down" is relative to the position of point in the window; otherwise it is relative to the left edge (for positive ARG) or the right edge (for negative ARG) of the current window. • If no window is at the desired location, an error is signalled. With PEL, the `bn` key-chord is also available when key-chord is available and active. See Σ **Key-Chords**.
Move to window at left	• `<f11> <left>` • `<f1> <down>` • `<Esc> <left>` • `⌘-<left>` • `❖-<left>` ✱ `<f7> <left>` • `gf`	(**windmove-left** &optional ARG)	Select the window to the left of the current one. • With no prefix argument, or with prefix argument equal to zero, "left" is relative to the position of point in the window; otherwise it is relative to the top edge (for positive ARG) or the bottom edge (for negative ARG) of the current window. • If no window is at the desired location, an error is signalled. With PEL, the `gf` key-chord is also available when key-chord is available and active. See Σ **Key-Chords**.
Move to window at right	• `<f11> <right>` • `<f1> <right>` • `<Esc> <right>` • `⌘-<right>` • `❖-<right>` ✱ `<f7> <right>` • `jk`	(**windmove-right** &optional ARG)	Select the window to the right of the current one. • With no prefix argument, or with prefix argument equal to zero, "right" is relative to the position of point in the window; otherwise it is relative to the top edge (for positive ARG) or the bottom edge (for negative ARG) of the current window. • If no window is at the desired location, an error is signalled. With PEL, the `jk` key-chord is also available when key-chord is available and active. See Σ **Key-Chords**.
Swap (eXchange) windows	• `<f11> w x` ✱ `<f7> x`	(**ace-swap-windows**)	Swap buffers of the current window with another. If 3 windows or more, a single digit shows up in the top-left corner identifying the number to type to swap to this window. Requires the **ace-window** external package. PEL downloads, install and activates it when the **pel-use-ace-window** user options is set to **t**.
Close Windows	The following commands are used to remove (close) windows. The last row correspond to a set of four PEL commands bound to cursor keys.		
Close this windows	• `C-x 0` ✱ `<f7> 0`	(**delete-window** &optional WINDOW)	This just closes the window and moves the cursor to the next window.
Close other (next) window	• `<f11> w w` ✱ `<f7> o`	(**pel-close-other-window**)	Close the other window. Hide its buffer, does not kill it. • Useful to close temporary window, like the help window, without having to move into it.
Close all other windows	• `C-x 1` ✱ `<f7> 1`	(**delete-other-windows** &optional WINDOW)	Maximize current window: make current window fill its frame. Close all other windows.
Close window identified by number	`<f11> w k`	(**ace-delete-window**)	Delete a window selected by a number, a number shown in the top-left corner of the window. • If there's only 2 windows, kills the other window. If only 1 window is used, does not kill it. Needs **ace-window** external package. PEL downloads, installs and activates it when the **pel-use-ace-window** user options is set to **t**.
Maximize window identified by number	`<f11> w m`	(**ace-maximize-window**) — — — — — — — — — — — — (**ace-delete-other-windows**)	Maximize specified window. Close all windows except the window selected by number, a number shown in the top-left corner of the window. Needs **ace-window** external package. The old versions used ace-window-maximize, but newer versions use ace-delete-maximize-windows. PEL uses the one that is available. PEL downloads, install and activates it when the **pel-use-ace-window** user options is set to **t**.
Close a window identified by cursor direction	• `ESC C-S-<right>` • `ESC C-S-<left>` • `ESC C-S-<down>` • `ESC C-S-<up>` • `<f1> C-S-<right>` • `<f1> C-S-<left>` • `<f1> C-S-<down>` • `<f1> C-S-<up>` • `<f11> C-S-<right>` • `<f11> C-S-<left>` • `<f11> C-S-<down>` • `<f11> C-S-<up>` ✱ `<f7> C-S-<right>` ✱ `<f7> C-S-<left>` ✱ `<f7> C-S-<down>` ✱ `<f7> C-S-<up>`	• **pel-close-window-right)** • **(pel-close-window-left)** • **(pel-close-window-down)** • **(pel-close-window-up)**	Kill window pointed by the cursor's direction. • The 4 different commands and shown in the same cell for convenience, one for each of the available cursors: `<right>`, `<left>`, `<down>` and `<up>`. • There are 4 possible sets of bindings: • 3 sets of stand-alone commands: • Commands with `<f11>` prefix, always available. • Commands with `ESC` prefix, available when **pel-windmove-on-esc-cursor** user option is on (set to t). • Commands with `<f1>` prefix, available when **pel-windmove-on-f1-cursor** user option is on (set to t). • The Hydra-based commands, with the Hydra activated with any of the key sequences that use the `<f7>` prefix. Available when **pel-use-hydra** user option is set to t.
Kill current buffer and close window See also: Σ **Buffers**	• `C-x 4 0` ✱ `<f7> K`	(**kill-buffer-and-window**)	Kill the current buffer and delete the selected window.
Kill current buffer	✱ `<f7> k`	(**pel-kill-current-buffer**)	Kill current buffer and close window **without prompting** unless it is modified. Only available in the Hydra.

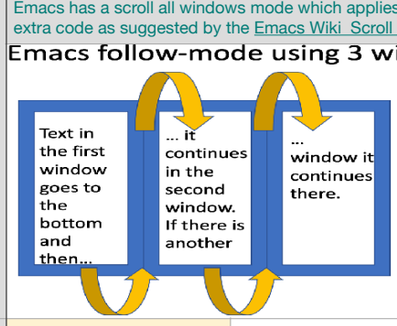
Operation	Keystroke	Function	Note
Create Window by splitting current window	The following commands create a new window by splitting the current one. The last row correspond to a set of four PEL commands bound to cursor keys. The **split-window-keep-point** user option controls whether point is kept at the same vertical position in both windows (t, the default). If nil, Emacs adjust point in the two windows to minimize redisplay. Change temporarily with: `<f11> <f4> w s`. Change permanently with: `<f11> w <f3> 1` to access the customization buffer and modify the user option.		
Toggle split window point behaviour	`<f11> w <f4> s`	(**pel-toggle-split-window-keep-point**)	Toggle the value of **split-window-keep-point** between values described above. Print description of new value. Change only affects current Emacs session, not stored.
Create new window below	• `C-x 2` ✱ `<f7> 2`	(**split-window-below** &optional SIZE)	Split current window into 2 windows. Leave point in top window. Same buffer in both. • Optional SIZE numerical argument identify line count of top window (if positive) or bottom window (if negative).
Create new window at right	• `C-x 3` ✱ `<f7> 3`	(**split-window-right** &optional SIZE)	Split current window into two side-by-side windows. Leave point in the left window. Same buffer in both. • Optional SIZE numerical argument identify column count of left-hand window (if positive) or right-hand window (if negative).

Create window at cursor direction	• `ESC C-<right>`
	• `ESC C-<left>`
	• `ESC C-<down>`
	• `ESC C-<up>`
	• `<f1> C-<right>`
	• `<f1> C-<left>`
	• `<f1> C-<down>`
	• `<f1> C-<up>`
	• `<f11> C-<right>`
	• `<f11> C-<left>`
	• `<f11> C-<down>`
	• `<f11> C-<up>`
	✱ `<f7> C-<right>`
	✱ `<f7> C-<left>`
	✱ `<f7> C-<down>`
	✱ `<f7> C-<up>`
		• (**pel-create-window-right** & optional SIZE)
		• (**pel-create-window-left** & optional SIZE)
		• (**pel-create-window-down** & optional SIZE)
		• (**pel-create-window-up** & optional SIZE)
			Create a window at the location pointed by the cursor's direction, and move point inside the new window.
			• Optional SIZE numerical argument identify either:
			  • line count of top window (if positive) or bottom window (if negative).
			  • column count of left-hand window (if positive) or right-hand window (if negative).
			• The 4 different commands and shown in the same cell for convenience, one for each of the available cursors: `<right>`, `<left>`, `<down>` and `<up>`.
			• There are 4 possible sets of bindings:
			  • 3 sets of stand-alone commands:
			  • Commands with `<f11>` prefix, always available.
			  • Commands with `ESC` prefix, available when **pel-windmove-on-esc-cursor** user option is on (set to t).
			  • Commands with `<f1>` prefix, available when **pel-windmove-on-f1-cursor** user option is on (set to t).
			  • The Hydra-based commands, with the Hydra activated with any of the key sequences that use the `<f7>` prefix. Available when **pel-use-hydra** user option is set to t.

Create Side Windows	Use the following commands to create **side windows**: special windows positioned at any of the four sides of a frame's *root* window. This allows you, for example, to create a bottom window that spans the entire frame width under several vertically split windows.		
Create new side window that holds current buffer.	• `<f11> w M-w` ✱ `<f7> M-2` ✱ `<f7> M-4` ✱ `<f7> M-6` ✱ `<f7> M-8`	(**pel-buffer-in-side-window** &optional N)	Place current buffer in a new, dedicated side window. • By default the side window is at the bottom of the current frame. • Use a numeric argument to specify a different side: For N= 2, 4, 6 or 8, select window pointed by what is pointed by cursor positioned at the layout of numeric keypad: 8 := 'top 4 := 'left 6 := 'right 2 := 'bottom
Create Frame Root Windows	⚠ Available on **Emacs 29.1 and later only.** The PEL Windows Hydra has keys that provides access to this command in all Emacs versions, but for previous versions of Emacs the Hydra uses the split-window commands (listed above) instead.		
Split root window below	`C-x w 2` ✱ `<f7> M-r`	(**split-root-window-below** &optional SIZE)	Split root window of current frame in two. • The current window configuration is retained in the top window, the lower window takes up the whole width of the frame. • Optional SIZE numerical argument identify line count of top window (if positive) or bottom window (if negative).
Split root window right	`C-x w 3` ✱ `<f7> M-R`	(**split-root-window-right** &optional SIZE)	Split root window of current frame into two side-by-side windows. • The current window configuration is retained within the left window, and a new window is created on the right, taking up the whole height of the frame. • Optional SIZE numerical argument identify column count of left-hand window (if positive) or right-hand window (if negative).
Resize Window Quickly with windresize	Resize the current window quickly using the **windresize** command (mapped to `<f11> w r` by PEL). 📦 Requires the **windresize** external package. PEL activates it when **pel-use-windresize** user-option is set to **t**. ☝ The **windresize** command ***can*** be used while the PEL Window Hydra is active, taking over Hydra keys. Complete and return to Hydra with `RET`		
Resize Window interactively	`<f11> w r`	(**windresize** &optional INCREMENT)	Resize windows interactively using the following minor mode keys. • Use `RET` to complete or `C-g` to abort. Both exit the mode.
Resize window using cursors	• `<right>` • `<left>` • `<down>` • `<up>`	• (**windresize-right** &optional N LEFT-BORDER FIXED-WIDTH) • (**windresize-left** &optional N LEFT-BORDER FIXED-WIDTH) • (**windresize-down** &optional N LEFT-BORDER FIXED-WIDTH) • (**windresize-up** &optional N LEFT-BORDER FIXED-WIDTH)	Resize the current window in the direction of the used cursor. • N is the number of lines by which moving borders.
Resize windows using direction opposite to cursor	• `C-<right>` • `C-<left>` • `C-<down>` • `C-<up>`	• (**windresize-right-minus**) • (**windresize-left-minus**) • (**windresize-down-minus**) • (**windresize-up-minus**)	Same as the above commands but use the direction opposite to the cursor.
Resize window bottom-right	`/`	(**windresize-bottom-right**)	Call 'windresize-right' and 'windresize-down' successively. • In move-borders method, move the bottom-right edge of the window outwards. • In resize-window method, enlarge the window horizontally and shrink it vertically.
Resize window top-right	`\`	(**windresize-up-right**)	Call 'windresize-right' and 'windresize-up' successively. • In move-borders method, move the upper-right edge of the window outwards. • In resize-window method, enlarge the window both horizontally and horizontally.
Resize window top-left	`M-/`	(**windresize-up-left**)	Call 'windresize-left' and 'windresize-up' successively. • In move-borders method, move the upper-left edge of the window outwards. • In resize-window method, shrink the window horizontally and enlarge it vertically.
Resize window bottom-left	`M-\`	(**windresize-bottom-left**)	Call 'windresize-left' and 'windresize-up' successively. • In move-borders method, move the bottom-left edge of the window outwards. • In resize-window method, shrink the window both horizontally and vertically.
Reposition window	• `C-M-<right>` • `C-M-<left>` • `C-M-<down>` • `C-M-<up>`	• (**windresize-right-fixed**) • (**windresize-left-fixed**) • (**windresize-down-fixed**) • (**windresize-up-fixed**)	Move the window to the direction identified by the cursor, keeping its width (or height) constant.
Set window resize/ reposition increment step	`i`	(**windresize-set-increment** &optional N)	Set the window resize increment step value to N. • Use a numeric argument prefix to set N interactively: • For example: `M-4 i` sets the increment to 4.
Increase the resize/ reposition increment step	`+`	(**windresize-increase-increment** &optional SILENT)	Increase the increment. • If SILENT is non-nil, don't output a message.

Operation	Keystroke	Function	Note
Decrease the resize/reposition increment step	**-**	(**windresize-decrease-increment** &optional SILENT)	Decrease the increment. • If SILENT is non-nil, don't output a message.
Negate resize/ reposition increment	**~**	(**windresize-negate-increment** &optional SILENT)	Negate the increment value. Changes the direction of window resize operations. • If SILENT is non-nil, don't output a message.
Balance Windows	• **=** • **C-x +**	(**windresize-balance-windows)**	Balance window sizes.
Delete current window	• **0** • **C-x 0**	(**delete-window** &optional WINDOW)	Delete current window ⚠️ During my testing **C-x 0** behaved like **windresize-other-window** instead. 🚧Should investigate. **0** works fine though.
Delete other windows	• **1** • **C-x 1**	(**windresize-delete-other-windows**)	Delete other windows.
Split window vertically	• **2** • **C-x 2**	(**windresize-split-window-vertically**)	Split window vertically. Creates 2 windows: one on top of the other.
Split window horizontally	• **3** • **C-x 3**	(**windresize-split-window-horizontally**)	Split window horizontally. Creates 2 windows side by side.
Save window configuration	**s**	(**windresize-save-window-configuration**)	Save the current window configuration in the ring.
Restore window configuration	**r**	(**windresize-restore-window-configuration**)	Restore the previous window configuration in the ring.
Move point to other adjacent window	• **M-S-<right>** • **M-S-<left>** • **M-S-<down>** • **M-S-<up>**	• (**windresize-select-right** &optional ARG) • (**windresize-select-left** &optional ARG) • (**windresize-select-down** &optional ARG) • (**windresize-select-up** &optional ARG)	Select the window identified by the cursor. • If ARG is nil or zero, select the window relatively to the point position. • If ARG is positive, select relatively to the top edge and select relatively to the bottom edge otherwise.
Move point to other window	**o**	(**windresize-other-window**)	Select other window.
Move point to previous window	**p**	(**windresize-previous-window**)	Select the previous window.
Move point to next window	**n**	(**windresize-next-window**)	Select other window.
Set window layout and exit windresize	• **x** • **RET**	(**windresize-exit**)	Keep this window configuration and exit 'windresize'.
Cancel window layout and exit windresize	• **c** • **q**	(**windresize-cancel-and-quit**)	Cancel window resizing and quit 'windresize'. • Restore window layout used before the entry into windresize mode. • The layouts, are, however still available via **winner-undo** **<f11> w p**, with PEL.
Resize Window **Using the base Emacs commands**	The following commands are used to change the current window size. Except when used inside the hydra, none of these commands are easy to re-type quickly. • The best way to use them is to type them once and then use a **repeat key**: • Emacs native repeat key is **C-x z** once and then repeat more by only typing '**z**'. PEL also binds the **<f5>** key to repeat. • PEL also provides the Window Hydra (described above) which can be started with one of the following commands using the **<f7>** prefix. Once the Hydra is entered, commands can be issued again without any prefix. Each of the first 5 commands below have 5 possible bindings: • The Emacs default key binding using the C-x prefix. • The commands with the default PEL <f11> prefix, always available. • The commands with **ESC** prefix, ☑ available when **pel-windmove-on-esc-cursor** user option is on (set to t). • The commands with **<f1>** prefix, ☑ available when **pel-windmove-on-f1-cursor** user option is on (set to t). • The Hydra-based commands, activated with any of the key sequences that use the **<f7>** prefix. ☑ Available when **pel-use-hydra** user option is set to t.		
Grow window taller	• **C-x ^** • **<f11> w s V** • **ESC M-<up>** • **<f1> M-<up>** ✱ **<f7> V**	(**enlarge-window** DELTA &optional HORIZONTAL)	Grow window taller by DELTA lines (defaults to 1), specify more with **C-u** n (or **M-** n) argument prefix. • See note above for availability of various bindings.
Shrink window smaller	• **<f11> w s v** • **ESC M-<down>** • **<f1> M-<down>** ✱ **<f7> v**	(**shrink-window** DELTA &optional HORIZONTAL)	Shrink height of window by DELTA lines (defaults to 1), specify more with **C-u** n (or **M-** n) argument prefix. • See note above for availability of various bindings.
Grow windows wider	• **C-x }** • **<f11> w s H** • **ESC M-<right>** • **<f1> M-<right>** ✱ **<f7> H**	(**enlarge-window-horizontally** DELTA)	Enlarge the current window horizontally. • See note above for availability of various bindings.
Shrink window narrower	• **C-x {** • **<f11> w s h** • **ESC M-<left>** • **<f1> M-<left>** ✱ **<f7> h**	(**shrink-window-horizontally** DELTA)	Reduce the width of the current window. • See note above for availability of various bindings.
Make all windows the same size	• **C-x +** • **<f11> w s =** • **ESC <kp-5>** • **<f1> <kp-5>** ✱ **<f7> =**	(**balance-windows** &optional WINDOW-OR-FRAME)	Balance the sizes of windows of WINDOW-OR-FRAME. • WINDOW-OR-FRAME is optional and defaults to the selected frame. • If WINDOW-OR-FRAME denotes a frame, balance the sizes of all windows of that frame. If WINDOW-OR-FRAME denotes a window, recursively balance the sizes of all child windows of that window. • See note above for availability of various bindings.
Reduce current window size if buffer is smaller than window	• **C-x -** • **<f11> w s -**	(**shrink-window-if-larger-than-buffer** &optional WINDOW)	Shrink height of current window if its buffer doesn't need so many lines. • More precisely, shrink window vertically to be as small as possible, while still showing the full contents of its buffer. • Do not shrink window to less than 'window-min-height' lines. Do nothing if the buffer contains more lines than the present window height, or if some of the window's contents are scrolled out of view, or if shrinking this window would also shrink another window, or if the window is the only window of its frame.

Operation	Keystroke	Function	Note
Fit window size to current buffer's content	• `C-x w -` ✱ `<f7> .`	(**fit-window-to-buffer** &optional WINDOW MAX-HEIGHT MIN-HEIGHT MAX-WIDTH MIN-WIDTH PRESERVE-SIZE)	Adjust size of WINDOW to display its buffer's contents exactly. • WINDOW must be a live window and defaults to the selected one. • If WINDOW is part of a vertical combination, adjust WINDOW's height. The new height is calculated from the actual height of the accessible portion of its buffer. The optional argument MAX-HEIGHT specifies a maximum height and defaults to the height of WINDOW's frame. The optional argument MIN-HEIGHT specifies a minimum height and defaults to 'window-min-height'. Both MAX-HEIGHT and MIN-HEIGHT are specified in lines and include mode and header line and a bottom divider, if any. • If WINDOW is part of a horizontal combination and the value of the option 'fit-window-to-buffer-horizontally' is non-nil, adjust WINDOW's width. The new width of WINDOW is calculated from the maximum length of its buffer's lines that follow the current start position of WINDOW. The optional argument MAX-WIDTH specifies a maximum width and defaults to the width of WINDOW's frame. The optional argument MIN-WIDTH specifies a minimum width and defaults to 'window-min-width'. Both MAX-WIDTH and MIN-WIDTH are specified in columns and include fringes, margins, a scrollbar and a vertical divider, if any.
Quick Window Layout Change	The following commands flip the layout of 2 windows: the current and *next* window between 2 horizontal windows to 2 vertical windows and vice versa.		
Flip 2 horizontal windows to 2 vertical ones	• `<f11> w v` ✱ `<f7> M-v`	(**pel-2-vertical-windows**)	Convert 2 horizontal windows into 2 vertical windows. • Flip the orientation of the current window and its next one. • The next window is placed at the right of the current window.
Flip 2 vertical windows to 2 horizontal ones	• `<f11> w h` ✱ `<f7> M-h`	(**pel-2-horizontal-windows**)	Convert 2 horizontal windows into 2 horizontal windows. • Flip the orientation of the current window and its next one. • The next window is placed below the current one.
Window Layout History	The following commands allow you to restore a previously used window layout. Two packages are available . The **winner** package, a package that is part of the standard Emacs. PEL activates them when **pel-use-winner** user option is **t**.		
Restore an earlier window configuration	• `C-c <left>` • `<f11> w p` ✱ `<f7> p`	(**winner-undo**)	Switch back to an earlier window configuration saved by Winner mode. In other words, "undo" changes in window configuration.
Restore a more recent window configuration	• `C-c <right>` • `<f11> w n` ✱ `<f7> n`	(**winner-redo**)	Restore a more recent window configuration saved by Winner mode.
Save/Restore window layout	The external **layout-restore** package. PEL activates it with **pel-use-restore-layout** user-option set to **t**. This associates layouts to buffers. ⚠️ This needs investigation work - **use caution**.		
Save Window layout	`<f11> w l s`	(**layout-save-current**)	Save the current layout, add a list of current layout to **layout-configuration-alist**.
Restore Layout	`<f11> w l r`	(**layout-restore** &optional BUFFER)	Restore the layout related to the buffer BUFFER, if there is such a layout saved in '**layout-configuration-alist**', and update the layout if necessary.
Delete Layout	`<f11> w l d`	(**layout-delete-current** &optional BUFFER)	Delete the layout information from '**layout-configuration-alist**' if there is an element list related to BUFFER.
Open Buffer in another window	With the following commands you can show a different buffer inside another window. One command select (move point to) that window. The other does not. • Under PEL, the prompt for buffer name is using the input completion method currently active (default, Ido, Helm, …) • See **Completion/Input** for more information.		
Display buffer in other window, don't select the other window.	• `C-x 4 C-o` • `<f11> w b`	(**ido-display-buffer**) – – – – – – – – – – – – – (**display-buffer** BUFFER-OR-NAME &optional ACTION FRAME)	Display a buffer in other window but don't select it.
Select buffer in other window	• `C-x 4 b` • `<f11> w B`	(**ido-switch-buffer-other-window**) – – – – – – – – – – – – – (**switch-to-buffer-other-window** BUFFER-OR-NAME &optional NORECORD)	Select buffer bufname in another window (switch-to-buffer-other-window). See Select Buffer.
Dedicated Windows	Emacs windows can be dedicated to specific buffers in such a way that future windows operations do not affect the dedicated windows. The following commands help you manage dedicated windows.		
Show dedicated status of current window	`<f11> w d ?`	(**pel-show-window-dedicated-status**)	Display the dedicated status of the current window in the echo area (the minibuffer).
Toggle dedicated status of current window	• `<f11> w d d` ✱ `<f7> d`	(**pel-toggle-window-dedicated**)	Toggle the dedicated status of the current window, changing a normal window into a dedicated one and a dedicated window into a normal one. ⚠️ Use with care after learning about dedicated windows.
Follow Mode See also: **Scrolling**	Emacs has a scroll all windows mode which applies all scroll commands to all visible windows. To support mouse wheel or scroll bar you need to implement extra code as suggested by the Emacs Wiki Scroll All Mode page.		

			When Emacs follow-mode is used on 2 or more windows, these windows show the text of the same buffer spread across these windows that act as a one continuous stream.
			
			• Follow mode is a minor mode that combines windows into one tall virtual window. This is accomplished by two main techniques:
			  • The windows always displays adjacent sections of the buffer. This means that whenever one window is moved, all the others will follow. (Hence the name Follow mode.)
			  • Should point (cursor) end up outside a window, another window displaying that point is selected, if possible. This makes it possible to walk between windows using normal cursor movement commands.
			• Follow mode comes to its prime when used on a large screen and two or more side-by-side windows are used. The user can, with the help of Follow mode, use these full-height windows as though they were one.

Toggle follow-mode See also: **Scrolling**	• `<f11> w f` • `<f11> | f`	(**follow-mode** &optional ARG)	Toggle Follow mode. With a prefix argument ARG, enable Follow mode if ARG is positive, and disable it otherwise.

Operation	Keystroke	Function	Note
recentering in current window	The following 2 command do **not** move point, but reposition the text in the current window. • These are quite useful as they can be used to refresh the view in the current window. See also: **Navigation**		
Position current line to window's Center / Bottom / Top . **Refresh screen.**	• `C-l` • `<f11> C-l` ✱ `<f7> 5`	(**recenter-top-bottom** &optional ARG)	Without argument: moves the current line to window: center -> top -> bottom. • With arg: centre first: • `C-u C-l` `C-l` `C-l` `C-l` • ➜ center ➜ bottom ➜ center ➜ top • With negative arg: bottom first: • `C-- C-l` `C-l` `C-l` • ➜ bottom ➜ center ➜ top • With arg 0: top first: • `M-0 C-l` `C-l` `C-l` • ➜ top ➜ bottom ➜ center • With numeric positive: move current line to window top position N • With negative numeric: move current line to bottom window position: -1 := last line • PEL provides the `<f11> C-l` key binding because some modes use `C-l` as a prefix key.
Reposition comment/definition in full view	• `C-M-l` • `C-[ C-l` • `Esc C-l`	(**reposition-window** &optional ARG)	Attempts to make the current comment or current definition fully visible by scrolling the lines without changing the point. • Further invocations move it to the top of the window or toggle the visibility of comments that precede it (by scrolling the lines).

Windows — Reference

Topic/URL	Comment
GNU Emacs — Displaying a Buffer in a Window	Describes the Emacs features related to displaying buffers inside windows.
GNU Emacs Lisp — Displaying Buffers — The Zen of Buffer Display	Describes the rules Emacs tries to use to control the creation of new windows when they are created dynamically from commands.