Windows - Managing and Moving To Other Windows

Operation	Keystroke	Function	Note

Window Operations

See also:
- ∑ Customize
- ∑ Key-Chords
- ∑ Frames
- ∑ Speedbar

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, but not using the standard wind move key bindings (which use Shift with cursors) 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.
- **ace-window**, an external package, which extends the **C-x o** command by displaying **Ace target in the windows' upper left corner** to quickly select the target window to move to and possibly operate on.
- **key-chord**, an external package, that PEL use (when **pel-use-key-chord** user option is **t**) to activate dual-key chords to move across windows.
- 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.

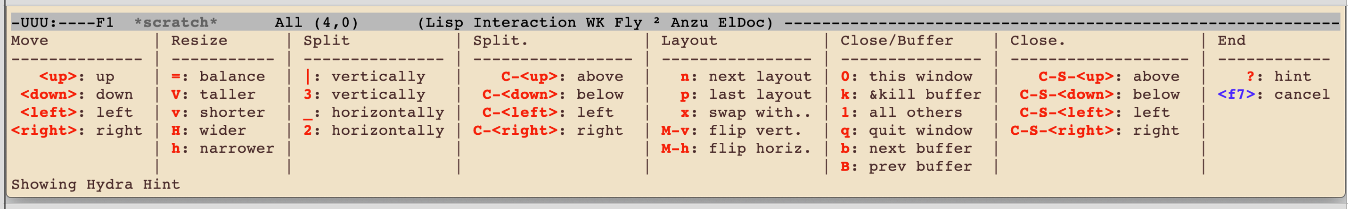
☝ Operations on windows can be applied to windows in other frames, whether Emacs is running in graphics mode or in terminal mode.
In terminal mode only one frame is visible at a time though.

Operation	Keystroke	Function	Note
Open this PDF file. See also: **∑ Help/ Info**	**<f11> w <f1>**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the local copy of the **∑ Windows** PDF file unless a command prefix (like **C-u**) was used. In that case it opens the Github-hosted file instead.
∑ 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.

PEL Window Management Hydra

- Requires the **hydra** external package. With PEL user option **pel-use-hydra** set to **t**, PEL activates the **hydra external package** and also creates a Hydra set of keys to help speed up navigation and management of windows. These keys are identified in the table below.
- To start this Hydra, hit the **<f7>** key, then hit one of the following keys once or several times.
- The keys that are in the PEL window hydra are all identified below with a **<f7>** prefix, but after typing **<f7>** once, you can hit several other window hydra keys without typing the **<f7>** prefix again.
- While active the Hydra Hint is normally shown in the minibuffer (like the one shown below). While the Hydra is active use the **?** key to toggle it 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 **?**.
- To cancel the Hydra hit the **<f7>** key again or use a command not available in the windows management hydra.
- 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 that shows up in the minibuffer area as soon as one of its keys is pressed is shown below.
☝ 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 visible in the current window.



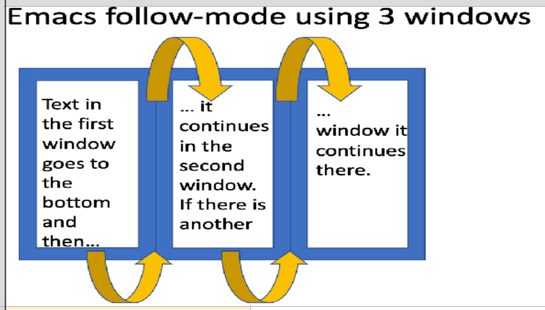
```
-UUU:----F1  *scratch*      All (4,0)      (Lisp Interaction WK Fly ² Anzu ElDoc) ------------------------------------------------------------
Move           | Resize      | Split           | Split.             | Layout            | Close/Buffer     | Close.               | End
-------------- | ----------- | --------------- | ------------------ | ----------------- | ---------------- | -------------------- | ------------
   <up>: up    | =: balance  | |: vertically   |    C-<up>: above   |    n: next layout | 0: this window   |    C-S-<up>: above   |    ?: hint
 <down>: down  | V: taller   | 3: vertically   |  C-<down>: below   |    p: last layout | k: &kill buffer  |  C-S-<down>: below   | <f7>: cancel
 <left>: left  | v: shorter  | _: horizontally |  C-<left>: left    |    x: swap with.. | 1: all others    |  C-S-<left>: left    |
<right>: right | H: wider    | 2: horizontally | C-<right>: right   |  M-v: flip vert.  | q: quit window   | C-S-<right>: right   |
               | h: narrower |                 |                    |  M-h: flip horiz. | b: next buffer   |                      |
               |             |                 |                    |                   | B: prev buffer   |                      |
Showing Hydra Hint
```

☝ The ace-window bound to **C-x o** key provides a partially overlapping feature set but they differ in their key assignments. See below.

Operation	Keystroke	Function	Note
Move point to other window **- C-u: swap** **- C-u C-u: delete**	**C-x o**	(**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		(**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**. Demo: **C'est la Z, video 5**
			• With only 2 windows, move to the other window. • With 3 windows or more: display an **Ace target in the windows' upper left corner** : it identifies the window target. • Type this number to move to the window, or • Type one of the following letters, followed by the target 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.
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.

<u>Operation</u>	<u>Keystroke</u>	Function	<u>Note</u>
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.
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. **pel-windmove-on-f1-cursor** controls the <f1> binding, also on by default.		
<u>Move to window above</u>	• `<f11> <up>` • `<f1> <up>` • `<Esc> <up>` • `⌘-<up>` • `❖-<up>` • `<f7><up>` • <u>`yu`</u>	(**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 signaled. 📦☑ With PEL, the <u>`yu`</u> key-chord is also available when key-chord is available and active. See **<u>∑ Key-Chords</u>**.
<u>Move to window below</u>	• `<f11> <down>` • `<f1> <down>` • `<Esc> <down>` • `⌘-<down>` • `❖-<down>` • `<f7> <down>` • <u>`bn`</u>	(**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 signaled. 📦☑ With PEL, the <u>`bn`</u> key-chord is also available when key-chord is available and active. See **<u>∑ Key-Chords</u>**.
<u>Move to window at left</u>	• `<f11> <left>` • `<f1> <down>` • `<Esc> <left>` • `⌘-<left>` • `❖-<left>` • `<f7> <left>` • <u>`gf`</u>	(**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 signaled. 📦☑ With PEL, the <u>`gf`</u> key-chord is also available when key-chord is available and active. See **<u>∑ Key-Chords</u>**.
<u>Move to window at right</u>	• `<f11> <right>` • `<f1> <right>` • `<Esc> <right>` • `⌘-<right>` • `❖-<right>` • `<f7> <right>` • <u>`jk`</u>	(**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 signaled. 📦☑ With PEL, the <u>`jk`</u> key-chord is also available when key-chord is available and active. See **<u>∑ Key-Chords</u>**.
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 **<u>ace-window</u>** external package. ☑ PEL downloads, install and activates it when the **pel-use-ace-window** user options is set to **t**.
<u>Close</u>/Create Windows	The following commands are used to create and remove windows. The last 2 rows correspond to two sets of four PEL commands bound to cursor keys.		
<u>Close this windows</u>	• `C-x 0` • `<f7> 0` • `<f7> d`	(**delete-window** &optional WINDOW)	This just closes the window and moves the cursor to the next window.
<u>Kill current buffer and close window</u> See also: **<u>∑ Buffers</u>**	• `C-x 4 0` • `<f7> k`	(**kill-buffer-and-window**)	Kill the current buffer and delete the selected window.
Close a 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. 📦 Requires the **<u>ace-window</u>** external package. ☑ PEL downloads, installs and activates it when the **pel-use-ace-window** user options is set to **t**.
<u>Close all other windows</u>	• `C-x 1` • `<f7> 1` • `<f7> .`	(**delete-other-windows** &optional WINDOW)	Make current window fill its frame.
Maximize one window, identified by number	`<f11> w m`	(**ace-maximize-window**) — — — — — — — — — — — — (**ace-delete-other-windows**)	Maximize a window. Close all windows except the window selected by number, a number shown in the top-left corner of the window. 📦 Requires the **<u>ace-window</u>** 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**.
Create new window below	• `C-x 2` • `<f7> 2` • `<f7> -`	(**split-window-below** &optional SIZE)	Split the selected window into two windows, one above the other. • The selected window is above. The newly split-off window is below and displays the same buffer. ☛ Note that Emacs default behaviour attempts to maximize the view into the current buffer when splitting the buffer into 2 windows. This means that the cursor will not be located in the same position in the new window. To change this behaviour and keep the same point in both windows, execute *(setq split-window-keep-point nil)*. The PEL packages does that.
Create new window at right	• `C-x 3` • `<f7> 3` • `<f7> |`	(**split-window-right** &optional SIZE)	Split the selected window into two side-by-side windows. • The selected window is on the left. The newly split-off window is on the right and displays the same buffer.
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**) • (**pel-create-window-left**) • (**pel-create-window-down**) • (**pel-create-window-up**)	Create a window at the location pointed by the cursor’s direction, and move point inside the new window. • 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.

Operation	Keystroke	Function	Note
Close a window at 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.
Resize Window	The following commands are used to change the current window size. Except for 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`**'. • The PEL package 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, 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.		
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.
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. They depend on 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.
Open Buffer in another window	With the following commands you can show a different buffer inside another window. One command select that other window (move point to that window) and the other does not. Under PEL both commands are bound to the IDO version of the command when the pel-use-ido customization variable is set to **t**, otherwise they retain the Emacs default binding. The IDO binding provides more information at the prompt.		
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.
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. When *pel-use-ido* is customized to **t**, (ido-display-buffer) is used, which prompts and provides easy to select list of available buffer names. Otherwise the standard Emacs (display-buffer) is used prompting without showing the available buffers.

Operation	Keystroke	Function	Note
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`	(**pel-toggle-window-dedicated**)	Toggle the dedicated status of the current window. Use with care.
Follow Mode	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.		
See also: **∑ Scrolling**	 Emacs follow-mode using 3 windows Text in the first window goes to the bottom and then… … it continues in the second window. If there is another … window it continues there.		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.
Scrolling Window	☛ For all other commands to scroll the window text, see the **∑ Scrolling** page.		

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.