










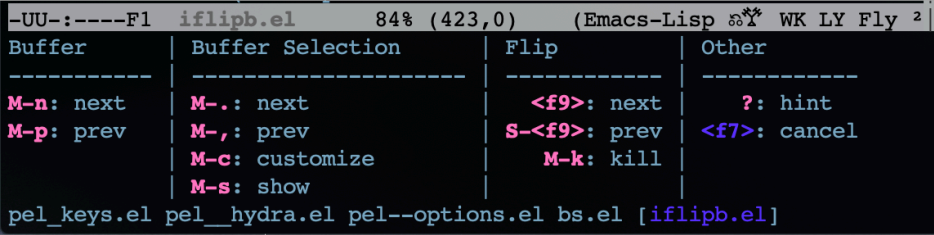


Buffers

Operation	Keystroke	Function	Note
Emacs Buffers	Emacs information and edited files are all held inside Emacs buffers. This table lists the commands you can use to list and manage buffers.  PEL provides the pel-pkg-for-buffer customization group to control some aspect of Emacs buffers. The user options are: <ul style="list-style-type: none"> pel-use-uniquify : activates uniquify to that buffer names show the distinguishing directory after the file name, like this: frame dir  pel-use-ascii-table : activates the ascii-table external package. See 🔗 Help/Info for the key binding.  pel-use-nhexl-mode : activates the nhexl-mode external package used to display and manipulate the content of the current buffer in hexadecimal.  pel-use-popup-switcher : activates the popup-switcher external package used for piping up a list of buffers. PEL also provides a Hydra that manipulates Emacs windows and buffers. See the 🔗 Windows table for its description.		
Open this PDF file. See also: 🔗 Help/Info	<f11> b <f1>	(pel-help-pdf & optional OPEN-WEB-PAGE)	Open the 🔗 Buffers 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 Buffer Support	<f11> b <f2>	(pel-customize-pel & optional OTHER-WINDOW)	Customize PEL Buffer support: open PEL buffer support specific group. <ul style="list-style-type: none"> If OTHER-WINDOW is non-nil (use C-u), display in other window.
🔗 Customize Emacs & external package buffer support	<f11> b <f3>	(pel-customize-library & optional OTHER-WINDOW)	Customize Emacs and external packages related to buffer. This includes the following customize groups: Buffer-menu, ibuffer, minibuffer, hexl, nhexl, popup-switcher. When a prefix argument (like C-u) opens the buffer inside another window. <ul style="list-style-type: none"> PEL prompts for files that may not be loaded to allow you to access all customization groups.
List Buffers & Switch to Buffer	The first 2 commands open a menu overlaid on the current buffer that you can use to switch to another buffer: <ul style="list-style-type: none"> buffer-menu-open is a drop-down hiererchical menu psw-switch-buffer is a pop-up menu. The switch-to-buffer command uses a prompt at the bottom of the frame. The list-buffers and ibuffer commands use a new buffer.		
Open buffer menu See also: 🔗 Menus	<ul style="list-style-type: none"> C-<f10> <C-down-mouse-1> 	(buffer-menu-open)	Start key navigation of the buffer menu. <ul style="list-style-type: none"> List buffers in a drop-down menu. <ul style="list-style-type: none"> Lists the buffers by major-mode when several buffers of the same major-mode are opened. In graphics mode this can also be invoked using the <C-down-mouse-1>
List open buffers in popup menu	<f11> b b	(psw-switch-buffer & optional ARG)	Show buffers list menu to switch buffer in a popup window menu. <ul style="list-style-type: none"> If ARG show only buffers with files and without * in the beginning and end of the buffer name.  Requires popup-switcher  activated by PEL when pel-use-popup-switcher user-option is turned on (t).
Switch to buffer See also: 🔗 Completion/ Input	C-x b	(switch-to-buffer BUFFER-OR-NAME & optional NORECORD FORCE-SAME-WINDOW)	Switch window to display the previous, or another buffer (entered at echo area prompt).  The invisible buffers have a name that start with a space. To see them type space and tab and a list of those buffers will appear before the list of visible buffers.  See 🔗 Completion/ Input for description of completion modes available.
List all buffers	C-x C-b	<ul style="list-style-type: none"> (list-buffers & optional ARG) (ibuffer & optional OTHER-WINDOW-P NAME QUALIFIERS NOSELECT SHRINK FILTER-GROUPS FORMATS) 	Display a list of existing buffers in a buffer named ""Buffer List"", the buffer displays information about all buffers and enters the Buffer Menu Mode . See the keystrokes for the Buffer Menu Mode below.  The PEL package uses the ' ibuffer ' function instead, which provides more functionality, working like dired, allowing to sort by name, size, mode, filtering by mode (hit return on the mode of a buffer). Type <f1> m to get the list of possible actions that can be done on the listed buffers.
Next/Previous Buffer	The following commands change current buffer to next or previous buffer, or to what was used last.		
Switch to next buffer	<ul style="list-style-type: none"> C-x <right> C-x C-<right> <f11> b n 	(next-buffer)	Switch to the next buffer displayed in the current window. <ul style="list-style-type: none"> This command is also available in the pel-🔗buffer Hydra as M-n
Switch to previous buffer	<ul style="list-style-type: none"> C-x <left> C-x C-<left> <f11> b p 	(previous-buffer)	Switch to the previous buffer displayed in the current window. <ul style="list-style-type: none"> This command is also available in the pel-🔗buffer Hydra as M-p
Switch to previous buffer in window	<f11> b l	(pel-switch-to-last-used-buffer)	Switch buffer in current window to the buffer previously seen in this window. Used twice returns to the same buffer.
To next/previous recently visited buffer	The following commands let you flip between recently visited buffers in a way that resembles what Alt-Tab and Alt-Shift-Tab does on Windows. <ul style="list-style-type: none"> A list of buffers is shown in the minibuffer at the bottom of the screen when you use the command. You can also identify buffer filtering in the iflipb customization group (use <f11> b <f3> and select iflipb to access it).  This requires the iflipb external package  PEL activates it when pel-use-iflipb user-option is turned on (set to t). This also forces activation of the hydra package because the iflipb commands are bound to the pel-🔗buffer Hydra. allowing quick single keystroke access without the use of a prefix key. <ul style="list-style-type: none"> To gain access to the keys, type <f7> <f9> key sequence to start the pel-🔗buffer Hydra. Then type <f9> or <S-f9> to flip through buffers. Stop the Hydra with <f7> 		
Activate the pel-🔗buffer Hydra	<f7> <f9>		
Flip to next buffer	<f7> <f9> ➡ <f9>	(iflipb-next-buffer ARG)	Flip to the next buffer in the buffer list. <ul style="list-style-type: none"> Consecutive invocations switch to less recent buffers in the buffer list. Buffers matching 'iflipb-always-ignore-buffers' are always ignored. Without a prefix argument, buffers matching 'iflipb-ignore-buffers' are also ignored.
Flip to previous buffer	<f7> <f9> ➡ <S-f9>	(iflipb-previous-buffer)	Flip to the previous buffer in the buffer list. Consecutive invocations switch to more recent buffers in the buffer list.
Kill buffer (but keep the flip buffer state)	<f7> <f9> ➡ M-k	(iflipb-kill-buffer)	Same as 'kill-buffer' but keep the iflipb buffer list state.
Buffer Selection	The Buffer Selection commands are also available through the pel-🔗buffer Hydra.		
Show next buffer in selection	<f7> <f9> ➡ M-.	(pel-bs-next)	Show next buffer in current window. <ul style="list-style-type: none"> Next buffer is selected by the criteria selected by bs-show and bs-configuration.
Show previous buffer in selection	<f7> <f9> ➡ M-,	(pel-bs-previous)	Show previous buffer in current window. <ul style="list-style-type: none"> Next buffer is selected by the criteria selected by bs-show and bs-configuration.
Customize buffer selection	<f7> <f9> ➡ M-c	(bs-customize)	Customization of group bs for Buffer Selection Menu.

Operation	Keystroke	Function	Note
Show Buffer Selection	<f7> <f9> ➡ M-s	(bs-show ARG)	Make a menu of buffers so you can manipulate buffers or the buffer list. <ul style="list-style-type: none"> There are many key commands similar to ‘Buffer-menu-mode’ for manipulating the buffer list and the buffers themselves. User can move with [up] or [down], select a buffer by RET or [SPC] <ul style="list-style-type: none"> Type q to leave Buffer Selection Menu without a selection. Type ? after invocation to get help on commands available. With prefix argument ARG show a different buffer list. Function ‘bs--configuration-name-for-prefix-arg’ determine accordingly name of buffer configuration.
Manage Buffers	The following commands support buffer management: display information, change read-only mode, clone buffer, rename buffer, kill buffer, etc...		
Show name of previous buffer in window	<f11> b ?	(pel-show-window-previous-buffer)	Show the name of previous buffer used in the current window.
Toggle read-only status of buffer	<ul style="list-style-type: none"> C-x C-q <f11> b r 	(read-only-mode &optional ARG)	When the buffer is in read-only mode the <u>mode line</u> shows ‘%%’ on the left side, in the ‘ch’ area of “cs:ch-fr buf pos line (major minor)”. The <u>manual</u> states: “For a read-only buffer, it shows ‘%*’ if the buffer is modified, and ‘% %’ otherwise.” <p>➡ See also: the View Mode activating commands toward the end of this table.</p> <ul style="list-style-type: none"> A buffer in View Mode cannot be modified. The View Mode may be used to ensure that no modifications are made to a buffer (visiting a file or not).
Clone buffer	<f11> b c	(clone-buffer &optional NEWNAME DISPLAY-FLAG)	Create and return a twin copy of the current buffer. <ul style="list-style-type: none"> Unlike an indirect buffer, the new buffer can be edited independently of the old one (if it is not read-only). NEWNAME is the name of the new buffer. It may be modified by adding or incrementing <N> at the end as necessary to create a unique buffer name. <ul style="list-style-type: none"> For example if buffer ‘Help’ is opened it opens another one named ‘Help’<2> (or ‘Help’<3> if ‘Help’<2> already exists, etc...)
Rename a buffer	<f11> b R	(rename-buffer NEWNAME &optional UNIQUE)	If UNIQUE argument is non-nil via C-u M-x rename-buffer, the name is auto generated to be unique.
Rename buffer - use unique name	<f11> b U	(rename-uniquely)	Rename the current buffer by adding ‘<number>’ to the end. <ul style="list-style-type: none"> Use this if you want multiple ‘Buffer’ or ‘Info’ buffers for example. Example: StackExchange: How can I have multiple help buffer with different content
Kill current buffer	<ul style="list-style-type: none"> <f11> b k ⌘-k ⌘-g 	(kill-current-buffer)	Kill (close) the current buffer. Does not prompt if there is no change in the buffer. <ul style="list-style-type: none"> PEL also provides a window management Hydra with ability to kill the current buffer. See <u>Windows</u> for more info.
See also: <u>Windows</u>			
Kill buffer	C-x k	(kill-buffer &optional BUFFER-OR-NAME)	Kill (close) the current buffer. <ul style="list-style-type: none"> Always prompt to identify a buffer, current is identified. Press enter to kill the buffer.
Kill current buffer and close window	<ul style="list-style-type: none"> C-x 4 0 <f7> k 	(kill-buffer-and-window)	Kill the current buffer and delete the selected window. <ul style="list-style-type: none"> PEL also provides a window management Hydra with ability to kill the current buffer and close windows in separate operations. See <u>Windows</u> for more info.
See also: <u>Windows</u>			
Kill some buffer		(kill-some-buffers &optional LIST)	Kill some buffers. Asks the user whether to kill each one of them.
Delete all windows of a specific buffer		(delete-windows-on &optional BUFFER-OR-NAME FRAME)	Deletes all windows showing BUFFER-OR-NAME, by calling ‘delete-window’ on those windows.
Accumulating Text	Emacs provides the following commands to insert text in buffer from various sources.		
Append region to specified buffer	<f11> b M-a	(append-to-buffer BUFFER START END)	Append to specified BUFFER the text of the region. <ul style="list-style-type: none"> The text is inserted into that buffer before its point. BUFFER can be a buffer or the name of a buffer; this function will create BUFFER if it doesn’t already exist.
Prepend region to specified buffer	<f11> b M-p	(prepend-to-buffer BUFFER START END)	Prepend to specified BUFFER the text of the region. <ul style="list-style-type: none"> The text is inserted into that buffer after its point. BUFFER can be a buffer or the name of a buffer; this function will create BUFFER if it doesn’t already exist.
Copy region to specified buffer (replacing old content)	<f11> b C-c	(copy-to-buffer BUFFER START END)	Copy to specified BUFFER the text of the region. <ul style="list-style-type: none"> The text is inserted into that buffer, replacing existing text there. BUFFER can be a buffer or the name of a buffer; this function will create BUFFER if it doesn’t already exist.
Insert content of specified buffer at point	<f11> b i	(insert-buffer BUFFER)	Insert after point the contents of BUFFER. <ul style="list-style-type: none"> Puts mark after the inserted text. BUFFER may be a buffer or a buffer name.
Append region’s text to specified file	<f11> b f	(append-to-file START END FILENAME)	Append the contents of the region to the end of file FILENAME. <ul style="list-style-type: none"> This does character code conversion and applies annotations like ‘write-region’ does.
Indirect Buffers	As described in Emacs Indirect Buffer section, “an indirect buffer shares the text of some other buffer, called the base buffer of the indirect buffer. In some ways it is a buffer analogue of a symbolic link between files. The text of the indirect buffer is always identical to the text of its base buffer; changes made by editing either one are visible immediately in the other. But in all other respects, the indirect buffer and its base buffer are completely separate. They can have different names, different values of point, different narrowing, different markers, different major modes, and different local variables.” <p>👉 Use indirect buffers to show the same file in 2 or more windows but want to narrow an area in 1 buffer while seeing the complete text in the other window.</p>		
Create indirect buffer explicitly	<f11> b I m	(make-indirect-buffer BASE-BUFFER NAME &optional CLONE)	Create and return an indirect buffer for buffer BASE-BUFFER, named NAME. <ul style="list-style-type: none"> BASE-BUFFER should be a live buffer, or the name of an existing buffer. NAME should be a string which is not the name of an existing buffer. Optional argument CLONE non-nil means preserve BASE-BUFFER’s state, such as major and minor modes, in the indirect buffer. CLONE nil means the indirect buffer’s state is reset to default values.
Create indirect buffer of current buffer	<f11> b I c	(clone-indirect-buffer NEWNAME DISPLAY-FLAG &optional NORECORD)	Create an indirect buffer that is a twin copy of the current buffer. <ul style="list-style-type: none"> Give the indirect buffer name NEWNAME. Interactively, read NEWNAME from the minibuffer when invoked with a prefix arg. If NEWNAME is nil or if not called with a prefix arg, NEWNAME defaults to the current buffer’s name. The name is modified by adding a ‘<N>’ suffix to it or by incrementing the N in an existing suffix. Trying to clone a buffer whose major mode symbol has a non-nil ‘no-clone-indirect’ property results in an error. DISPLAY-FLAG non-nil means show the new buffer with ‘pop-to-buffer’. This is always done when called interactively. Optional third arg NORECORD non-nil means do not put this buffer at the front of the list of recently selected ones.
Create indirect buffer of current buffer in another window	<ul style="list-style-type: none"> C-x 4 c <f11> b I w 	(clone-indirect-buffer-other-window NEWNAME DISPLAY-FLAG &optional NORECORD)	Like ‘clone-indirect-buffer’ but display in another window.
Edit Binary file with hexl	Emacs provides the built-in <u>hexl</u> mode to edit files in hexadecimal mode. To use it you must: <ul style="list-style-type: none"> use the hexl-find-file to open the file in binary mode, or use the hexl-mode command to convert an already opened buffer. To exit this mode and go back to the original mode type C-c C-c 		
Open a file in hexl-mode	<f11> f M-x	(hexl-find-file FILENAME)	Edit file FILENAME as a binary file in hex dump format. <ul style="list-style-type: none"> Switch to a buffer visiting file FILENAME, creating one if none exists, and edit the file in ‘hexl-mode’.
See also: <u>File-mngt</u>			
Toggle hexl mode	<f11> b M-x	(hexl-mode &optional ARG)	Toggle the hexl mode: a mode for editing binary files in hex dump format.

Operation	Keystroke	Function	Note
Buffer Menu Mode	<p>The list of buffers is shown inside its own buffer, "Buffer List" when (list-buffer) is executed. This buffer support the following commands.</p> <p>➡The full list of key bindings is available via the <f1> m key.</p> <p>➡⚠Note that PEL uses (ibuffer) for the C-x C-b key binding, so the list of commands and key bindings that are available differ. They are listed in the next section.</p>		
Buffer Menu Mode keys	<ul style="list-style-type: none"> • ? : Get help : Immediately • g : Update buffer list : immediately • C-n : next buffer in list : immediately • SPC : next buffer in list : immediately • n : next buffer in list : immediately • C-p : previous buffer in list : immediately • p : previous buffer in list : immediately • C-d : mark buffer for deletion : deleted when pressing x • d : mark buffer for deletion : deleted when pressing x • k : mark buffer for deletion : deleted when pressing x • s : save buffer : saved when pressing x • : Move to previous line, remove all marks on buffer : immediately if just after marking • M- : Remove a specific mark from all buffers : immediately if just after marking • u : unmark all marks on buffer : immediately • x : execute marked commands (delete buffers marked for deletion) : immediately • ~ : mark buffer as un-modifiable : immediately • % : toggle read-only : immediately • 1 : display emacs in full emacs screen : immediately • 2 : Display this buffer & next in horizontal window : immediately • o : replace other (next) window with this buffer : immediately • m : mark buffer to be displayed in windows : when pressing v • v : display buffers marked with in as many windows as required : immediately • q : quit buffer list : immediately 		
iBuffer Mode	<p>The commands available in the ibuffer window.</p> <p>With PEL, the C-x C-b key binding open the lbuffer window.</p>		
See also: Σ ibuffer-mode			
IBuffer Mode commands	<p>S : Save the marked buffers.</p> <p>A : View the marked buffers in the selected frame.</p> <p>H : View the marked buffers in another frame.</p> <p>V : Revert the marked buffers.</p> <p>T : Toggle read-only state of marked buffers.</p> <p>L : Toggle lock state of marked buffers.</p> <p>D : Kill the marked buffers.</p> <p>M-s a C-s : Do incremental search in the marked buffers.</p> <p>M-s a C-M-s : lsearch for regexp in the marked buffers.</p> <p>r : Replace by regexp in each of the marked buffers.</p> <p>Q : Query replace in each of the marked buffers.</p> <p>I : As above, with a regular expression.</p> <p>P : Print the marked buffers.</p> <p>O : List lines in all marked buffers which match a given regexp (like the function 'occur').</p> <p>X : Pipe the contents of the marked buffers to a shell command.</p> <p>N : Replace the contents of the marked buffers with the output of a shell command.</p> <p>! : Run a shell command with the buffer's file as an argument.</p> <p>E : Evaluate a form in each of the marked buffers. This is a very flexible command.</p> <p>For example, if you want to make all of the marked buffers read-only, try using (read-only-mode 1) as the input form.</p> <p>W : As above, but view each buffer while the form is evaluated.</p> <p>k : Remove the marked lines from the "lbuffer" buffer, but don't kill the associated buffer.</p> <p>x : Kill all buffers marked for deletion.</p>		
IBuffer Mode Marking commands	<p>m : Mark the buffer at point.</p> <p>t : Unmark all currently marked buffers, and mark all unmarked buffers.</p> <p>* c : Change the mark used on marked buffers.</p> <p>u : Unmark the buffer at point.</p> <p>DEL : Unmark the previous buffer.</p> <p>M-DEL : Unmark buffers marked with MARK.</p> <p>U : Unmark all marked buffers.</p> <p>* M : Mark buffers by major mode.</p> <p>* u : Mark all "unsaved" buffers. This means that the buffer is modified, and has an associated file.</p> <p>* m : Mark all modified buffers, regardless of whether they have an associated file.</p> <p>* s : Mark all buffers whose name begins and ends with ''.</p> <p>* e : Mark all buffers which have an associated file, but that file doesn't currently exist.</p> <p>* r : Mark all read-only buffers.</p> <p>* / : Mark buffers in 'diread-mode'.</p> <p>* h : Mark buffers in 'help-mode', 'apropos-mode', etc.</p> <p>. : Mark buffers older than 'ibuffer-old-time'.</p> <p>d : Mark the buffer at point for deletion.</p> <p>% n : Mark buffers by their name, using a regexp.</p> <p>% m : Mark buffers by their major mode, using a regexp.</p> <p>% f : Mark buffers by their filename, using a regexp.</p> <p>% g : Mark buffers by their content, using a regexp.</p> <p>% L : Mark all locked buffers.</p>		
IBuffer Mode Filtering commands		(ibuffer-filter-chosen-by-completion)	<p>Select and apply filter chosen by completion against available filters.</p> <ul style="list-style-type: none"> Indicates corresponding key sequences in echo area after filtering. The completion matches against the filter description text of ach filter in 'ibuffer-filtering-alist'.
		(ibuffer-filter-by-directory QUALIFIER)	<p>Limit current view to buffers with directory matching QUALIFIER.</p> <ul style="list-style-type: none"> For a buffer associated with file '/a/b/c.d', this matches against '/a/b'. For a buffer not associated with a file, this matches against the value of 'default-directory' in that buffer.

Operation	Keystroke	Function	Note
		/ RET : Add a filter by any major mode. / m : Add a filter by a major mode now in use. / M : Add a filter by derived mode. / n : Add a filter by buffer name. / c : Add a filter by buffer content. / b : Add a filter by basename. / f : Add a filter by filename. / . : Add a filter by file extension. / i : Add a filter by modified buffers. / e : Add a filter by an arbitrary Lisp predicate. / > : Add a filter by buffer size. / < : Add a filter by buffer size. / * : Add a filter by special buffers. / v : Add a filter by buffers visiting files. / s : Save the current filters with a name. / r : Switch to previously saved filters. / a : Add saved filters to current filters. / & : Replace the top two filters with their logical AND. / : Replace the top two filters with their logical OR. / p : Remove the top filter. / ! : Invert the logical sense of the top filter. / d : Break down the topmost filter. / / : Remove all filtering currently in effect.	
IBuffer Mode Filter commands		/ g : Create filter group from filters. / P : Remove top filter group. TAB : Move to the next filter group. M-p : Move to the previous filter group. / \ : Remove all active filter groups / S : Save the current groups with a name. / R : Restore previously saved groups. / X : Delete previously saved groups.	
IBuffer Mode Sorting commands		, : Rotate between the various sorting modes. s i : Reverse the current sorting order. s a : Sort the buffers lexicographically. s f : Sort the buffers by the file name. s v : Sort the buffers by last viewing time. s s : Sort the buffers by size. s m : Sort the buffers by major mode.	
IBuffer Mode Other commands		g : Regenerate the list of all buffers. Prefix arg means to toggle whether buffers that match ‘ibuffer-maybe-show-predicates’ should be displayed. <code>`</code> : Change the current display format . 🙌 Use this to see the complete file name when the file name is long. SPC : Move point to the next line. C-p : Move point to the previous line. h : Show this help. = : View the differences between this buffer and its associated file. RET : View the buffer on this line. o : As above, but in another window. C-o : As both above, but don’t select the new window. b : Bury (not kill!) the buffer on this line.	