















Emacs Buffers

Operation	Keystroke	Function	Note
Emacs Buffers	<p>Emacs information and edited files are all held inside Emacs buffers. This table lists the commands you can use to list and manage buffers.</p> <p> PEL provides the pel-pkg-for-buffer customization group to control some aspect of Emacs buffers. The user options are:</p> <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: <code>fname dir</code>•  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. <p>PEL also provides a Hydra that manipulates Emacs windows and buffers. See the § Windows table for its description.</p>		
Open this PDF file. See also: § Help/Info	<f11> b <f1>	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the local copy of the § Buffers PDF file unless a command prefix (like C-u) was used. In that case it opens the Github-hosted file instead.
§ 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.
Manage Buffers	The following commands support buffer management.		
Open Buffer Menu	<C-f10>	(buffer-menu-open)	Start key navigation of the buffer menu. <ul style="list-style-type: none">• Lists the buffers by major-mode when several buffers of the same major-mode are opened.• This is the keyboard interface to <C-down-mouse-1>
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 prompt). <ul style="list-style-type: none">👉 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. <ul style="list-style-type: none">➡ 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.
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).
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 <code>mode</code> line shows ‘%%’ on the left side, in the ‘ch’ area of “ <code>cs:ch-fr buf pos line (major minor)</code> ”. The manual states: “ <i>For a read-only buffer, it shows ‘%%’ if the buffer is modified, and ‘% %’ otherwise.</i> ” <ul style="list-style-type: none">➡ See also: the View Mode activating commands toward the end of this table.• 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).
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.
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.
Show name of previous buffer in window	<f11> b P	(pel-show-window-previous-buffer)	Show the name of previous buffer used in the current window.
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.
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.• 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 ' <i><number></i> ' 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 See also: § Windows	<ul style="list-style-type: none">• <f11> b k• ☞-k• ☞-&	(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 § Windows for more info.
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 See also: § Windows	<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 § Windows for more info.
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.

Operation	Keystroke	Function	Note
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 section also states: “One way to utilize indirect buffers is to display multiple views of an outline” (such as Org-Mode files). The following commands are available to manage indirect buffers.		
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 hexl mode to edit files in hexadecimal mode. <ul style="list-style-type: none"> 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: File-mngt			
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. <ul style="list-style-type: none"> This is not an ordinary major mode; it alters some aspects of the current mode's behavior, but not all; also, you can exit Hexl mode and return to the previous mode using 'hexl-mode-exit'. This function automatically converts a buffer into the hexl format using the function 'hexlify-buffer'. Each line in the buffer has an "address" (displayed in hexadecimal) representing the offset into the file that the characters on this line are at and 16 characters from the file (displayed as hexadecimal values grouped every 'hexl-bits' bits, and as their ASCII values). If any of the characters (displayed as ASCII characters) are unprintable (control or meta characters) they will be replaced by periods.
Insert a byte in decimal	C-M-d	(hexl-insert-decimal-char ARG)	Insert a character given by its decimal code ARG times at point.
Insert a byte in octal	C-M-o	(hexl-insert-octal-char ARG)	Insert a character given by its octal code ARG times at point.
Insert a byte in hex	C-M-x	(hexl-insert-hex-char ARG)	Insert a character given by its hexadecimal code ARG times at point.
Goto 512-byte page start	C-M-a	(hexl-beginning-of-512b-page)	Go to beginning of 512 byte boundary.
Goto to 512-byte page end	C-M-e	(hexl-end-of-512b-page)	Go to end of 512 byte boundary.
Goto 1K end	C-x]	(hexl-end-of-1k-page)	Go to end of 1KB boundary.
Goto 1K beginning	C-x [(hexl-beginning-of-1k-page)	Go to beginning of 1KB boundary.
Goto address entered in hexadecimal	M-g	(hexl-goto-hex-address HEX-ADDRESS)	Go to Hexl mode address (hex string) HEX-ADDRESS. <ul style="list-style-type: none"> Signal error if HEX-ADDRESS is out of range.
Goto to address entered in decimal	M-j	(hexl-goto-address ADDRESS)	Go to hexl-mode (decimal) address ADDRESS. <ul style="list-style-type: none"> Signal error if ADDRESS is out of range.
Exit hexl mode	C-c C-c	(hexl-mode-exit &optional ARG)	Exit Hexl mode, returning to previous mode. <ul style="list-style-type: none"> With arg, don't unhexlify buffer.
Hexadecimal Editing with nhexl	 The nhexl-mode external package used to display and manipulate the content of the current buffer in hexadecimal and manipulate hex dump files.  PEL downloads installs and activates this package when the pel-use-nhexl user option is set to t . <ul style="list-style-type: none"> Use the <f11> b <f2> key sequence to open the PEL buffer customization buffer to access this user option. Once the hexadecimal mode is on, turn it off by executing the nhexl-mode command again.  Good nhexl-mode features: <ul style="list-style-type: none"> The nhexl-mode keeps the undo history when you toggle the nhexl mode. Something that the helx mode does not do. You can use all of the normal navigation commands. You don't need to use specialized commands. PEL home and end commands work. 		
Toggle buffer between normal and hex display	<f11> b x	(nhexl-mode &optional ARG)	Toggle minor mode to edit files via hex-dump format.  Requires the nhexl-mode package  activated when pel-use-nhexl user option is t .
Activate Hex nibble editing mode	<f11> b X	(nhexl-nibble-edit-mode &optional ARG)	Minor mode to edit the hex nibbles in 'nhexl-mode'.  Note: only works after nhexl-mode has been activated once.  Requires the nhexl-mode package  activated when pel-use-nhexl user option is t .
Buffer View Mode	Several commands (view-buffer, etc..., see at top of this table) activate the View Mode for a buffer where the buffer is essentially read-only and special commands are available.		
View buffer - no modification allowed	<f11> b v	(view-buffer BUFFER &optional EXIT-ACTION)	View BUFFER in View mode, returning to previous buffer when done. <ul style="list-style-type: none"> Emacs commands editing the buffer contents are not available; instead, a special set of commands (mostly letters and punctuation) are defined for moving around in the buffer. Space scrolls forward, Delete scrolls backward. For a list of all View commands, type H or h while viewing. See the View Mode command list below.

Operation	Keystroke	Function	Note
IBuffer Mode Marking commands	m	: Mark the buffer at point.	
	t	: Unmark all currently marked buffers, and mark all unmarked buffers.	
	* c	: Change the mark used on marked buffers.	
	u	: Unmark the buffer at point.	
	DEL	: Unmark the previous buffer.	
	M-DEL	: Unmark buffers marked with MARK.	
	U	: Unmark all marked buffers.	
	* M	: Mark buffers by major mode.	
	* u	: Mark all "unsaved" buffers. This means that the buffer is modified, and has an associated file.	
	* m	: Mark all modified buffers, regardless of whether they have an associated file.	
	* s	: Mark all buffers whose name begins and ends with ‘’.	
	* e	: Mark all buffers which have an associated file, but that file doesn’t currently exist.	
	* r	: Mark all read-only buffers.	
	* /	: Mark buffers in ‘dired-mode’.	
	* h	: Mark buffers in ‘help-mode’, ‘apropos-mode’, etc.	
	.	: Mark buffers older than ‘ibuffer-old-time’.	
	d	: Mark the buffer at point for deletion.	
	% n	: Mark buffers by their name, using a regexp.	
	% m	: Mark buffers by their major mode, using a regexp.	
	% f	: Mark buffers by their filename, using a regexp.	
	% g	: Mark buffers by their content, using a regexp.	
	% L	: Mark all locked buffers.	
IBuffer Mode Filtering commands		(ibuffer-filter-chosen-by-completion)	Select and apply filter chosen by completion against available filters. <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)	Limit current view to buffers with directory matching QUALIFIER. <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.
	/ 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.		
	, : 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.		
	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. 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.		