Emacs 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 also provides a Hydra that manipulates Emacs windows and buffers. See the ∑ **Windows** table for its description.		
Open local copy of this PDF file. See also: ∑ **Help/Info**	`<f11> b <f1>`	(**pel-help-pdf**)	Open the PEL PDF file(s) for the current context: buffer support. It opens the local copy of this file.
Customize PEL Buffer Support See also:∑ **Customize**	`<f11> b <f2>`	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL Bookmark support: open PEL buffer support specific group. • If OTHER-WINDOW is non-nil (use **`C-u`**), display in other window.
Customize Emacs & external package buffer support See also:∑ **Customize**	`<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, nhexl. When a prefix argument (like **`C-u`**) opens the buffer inside another window. • Group belonging to files that have not yet been loaded are normally not accessible in Emacs and via the customize-group command. PEL, however, attempts to locate the file that defines a non-loaded customization group and will prompt you for loading the file if it finds it.
Manage Buffers	The following commands support buffer management.		
Open Buffer Menu	**`<C-f10>`**	(**buffer-menu-open**)	Start key navigation of the buffer menu. This is the keyboard interface to <C-down-mouse-1>
Toggle read-only status of buffer	• **`C-x C-q`** • `<f11> b r`	(**read-only-mode** &optional ARG)	When the buffer is in read-only mode the mode line shows '%%' on the left side, in the 'ch' area of *"cs:ch-fr buf pos line (major minor)"*. The manual states: *"For a read-only buffer, it shows '%*' if the buffer is modified, and '%%' otherwise."* ☛ **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	• **`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	• **`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.
Switch to buffer	**`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). ☝ 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.
List all buffers	**`C-x C-b`**	• (**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 the 'ibuffer' function instead, which provides more functionality, working like dired.
Clone buffer	`<f11> b c`	(**clone-buffer** &optional NEWNAME DISPLAY-FLAG)	Create and return a twin copy of the current buffer. • 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 '*<number>*' to the end. • 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**	• `<f11> b k` • `⌘-k` • `⌘-&`	(**kill-current-buffer**)	Kill (close) the current buffer. Does not prompt if there is no change in the buffer. • 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. • Always prompt to identify a buffer, current is identified. Press enter to kill the buffer.
Kill current buffer and close window See also: ∑ **Windows**	• **`C-x 4 0`** • `<f7> k`	(**kill-buffer-and-window**)	Kill the current buffer and delete the selected window. • 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. • 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. • 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. • 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. • 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. • This does character code conversion and applies annotations like 'write-region' does.

Operation	Keystroke	Function	Note
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. • 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. • 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	• `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 helx	Emacs provides the built-in **helx** mode to edit files in hexadecimal mode.
	• To use it you must:
	  • use the hexl-find-file to open the file in binary mode, or
	  • use the heel-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 See also: **∑ File-mngt**	`<f11> f M-x`	(**hexl-find-file** FILENAME)	Edit file FILENAME as a binary file in hex dump format. • Switch to a buffer visiting file FILENAME, creating one if none exists, and edit the file in 'hexl-mode'.
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. • 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. • 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. • 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. • With arg, don't unhexlify buffer.
Hexadecimal Editing with nhexl	📦 The **nhexl-mode** external package use 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**. • 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: • 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. • 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
View Mode commands	H, h, ? This message. Digits provide prefix arguments. - negative prefix argument. < move to the beginning of buffer. > move to the end of buffer. o scroll so that buffer end is at last line of window. SPC scroll forward "page size" lines. With prefix scroll forward prefix lines. DEL, S-SPC scroll backward "page size" lines. With prefix scroll backward prefix lines. z like SPC but with prefix sets "page size" to prefix. w like DEL but with prefix sets "page size" to prefix. d scroll forward "half page size" lines. With prefix, sets "half page size" to prefix lines and scrolls forward that much. u scroll backward "half page size" lines. With prefix, sets "half page size" to prefix lines and scrolls backward that much. RET, LFD scroll forward one line. With prefix scroll forward prefix line(s). y scroll backward one line. With prefix scroll backward prefix line(s). F revert-buffer if necessary and scroll forward. Use this to view a changing file. = prints the current line number. % goes prefix argument (default 100) percent into buffer. g goes to line given by prefix argument (default first line). . set the mark. x exchanges point and mark. @ return to mark and pops mark ring. Mark ring is pushed at start of every successful search and when jump to line occurs. The mark is set on jump to buffer start or end. m save current position in character register. ' go to position saved in character register. s do forward incremental search. r do reverse incremental search. / searches forward for regular expression, starting after current page. ! and @ have a special meaning at the beginning of the regexp. ! means search for a line with no match for regexp. @ means start search at beginning (end for backward search) of buffer. \ searches backward for regular expression, starting before current page. n searches forward for last regular expression. p searches backward for last regular expression. q quit View mode, restoring this window and buffer to previous state. q is the normal way to leave view mode. e exit View mode but stay in current buffer. Use this if you started viewing a buffer (file) and find out you want to edit it. This command restores the previous read-only status of the buffer. E exit View mode, and make the current buffer editable even if it was not editable before entry to View mode. Q quit View mode, restoring all windows to previous state. c quit View mode and maybe switch buffers, but don't kill this buffer. C quit View mode, kill current buffer and go back to other buffer. The effect of c, q and C depends on how view-mode was entered. If it was entered by **view-file**, **view-file-other-window**, **view-file-other-frame,** or M-x **dired-view-file** (M-x view-file, M-x view-file-other-window, M-x view-file-other-frame, or the Dired mode v command), then q will try to kill the current buffer. If view-mode was entered from another buffer, by **`<f11> b v`**, M-x view-buffer-other-window, M-x view-buffer-other frame, M-x view-file, M-x view-file-other-window, or M-x view-file-other-frame, then c, q and C will return to that buffer.		
Buffer Menu Mode	The list of buffers is shown inside its own buffer, *Buffer List* when (list-buffer) is executed. This buffer support the following commands. ☛The full list of key bindings is available via the **`<f1> m`** key. ☛⚠️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.		
Buffer Menu Mode keys 🚧 **Complete the list**	• **`?`** : 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** • **`<DEL>`** : Move to previous line, remove all marks on buffer : immediately if just after marking • **`M-<DEL>`** : 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 command (1)	'S' - Save the marked buffers. 'A' - View the marked buffers in the selected frame. 'H' - View the marked buffers in another frame. 'V' - Revert the marked buffers. 'T' - Toggle read-only state of marked buffers. 'L' - Toggle lock state of marked buffers. 'D' - Kill the marked buffers. 'M-s a C-s' - Do incremental search in the marked buffers. 'M-s a C-M-s' - Isearch for regexp in the marked buffers. 'r' - Replace by regexp in each of the marked buffers. 'Q' - Query replace in each of the marked buffers. 'I' - As above, with a regular expression. 'P' - Print the marked buffers. 'O' - List lines in all marked buffers which match a given regexp (like the function 'occur'). 'X' - Pipe the contents of the marked buffers to a shell command. 'N' - Replace the contents of the marked buffers with the output of a shell command. '!' - Run a shell command with the buffer's file as an argument. 'E' - Evaluate a form in each of the marked buffers. This is a very flexible command. For example, if you want to make all of the marked buffers read-only, try using (read-only-mode 1) as the input form. 'W' - As above, but view each buffer while the form is evaluated. 'k' - Remove the marked lines from the *Ibuffer* buffer, but don't kill the associated buffer. 'x' - Kill all buffers marked for deletion.		

Operation	Keystroke	Function	Note
IBuffer Mode command (2)	**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 command (3)	**Filtering commands**: ‘M-x ibuffer-filter-chosen-by-completion’ - Select and apply filter chosen by completion. ‘/ 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. ‘M-x ibuffer-filter-by-directory’ - Add a filter by directory name. ‘/ 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 command (4)	**Filter group 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 command (5)	**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 command (6)	**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. ‘`’ - Change the current display format. ‘SPC’ - Move point to the next line. ‘C-p’ - Move point to the previous line. ‘h’ - 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.		