



Buffers

Operation	Keystroke	Function	Notes
Manage Buffers	The following commands support buffer management. The <code><f11> <f1> b</code> key opens the buffer management customization (see Σ Customize).		
Open Buffer Menu	<code><C-f10></code>	(buffer-menu-open)	Start key navigation of the buffer menu. This is the keyboard interface to <code><C-down-mouse-1></code>
Toggle read-only status of buffer	<ul style="list-style-type: none"><code>C-x C-q</code><code><f11> b r</code>	(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 <code>manual</code> states: “ <i>For a read-only buffer, it shows ‘%’ if the buffer is modified, and ‘% %’ otherwise.</i> ” ➡ See also: the View Mode activating commands toward the end of this table. <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).
Switch to next buffer	<ul style="list-style-type: none"><code>C-x <right></code><code>C-x C-<right></code><code><f11> b n</code>	(next-buffer)	Switch to the next buffer displayed in the current window.
Switch to previous buffer	<ul style="list-style-type: none"><code>C-x <left></code><code>C-x C-<left></code><code><f11> b p</code>	(previous-buffer)	Switch to the previous buffer displayed in the current window.
Show name of previous buffer in window	<code><f11> b P</code>	(pel-show-window-previous-buffer)	Show the name of previous buffer used in the current window.
Switch to previous buffer in window	<code><f11> b l</code>	(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	<code>C-x b</code>	(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	<code>C-x C-b</code>	<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 the <code>ibuffer</code> function instead, which provides more functionality, working like <code>dired</code> .
Clone buffer	<code><f11> b c</code>	(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 <code><N></code> at the end as necessary to create a unique buffer name. For example if buffer <code>*Help*</code> is opened it opens another one named <code>*Help*<2></code> (or <code>*Help*<3></code> if <code>*Help*<2></code> already exists, etc…)
Toggle buffer between normal and hex display	<code><f11> b x</code>	(nhexl-mode &optional ARG)	Toggle minor mode to edit files via hex-dump format. 📦 Requires the <code>nhexl-mode</code> package  activated when <code>pel-use-nhexl</code> user option is <code>t</code> .
Activate Hex nibble editing mode	<code><f11> b X</code>	(nhexl-nibble-edit-mode &optional ARG)	Minor mode to edit the hex nibbles in ‘nhexl-mode’. ⚠️ Note: only works after <code>nhexl-mode</code> has been activated once. 📦 Requires the <code>nhexl-mode</code> package  activated when <code>pel-use-nhexl</code> user option is <code>t</code> .
Rename a buffer	<code><f11> b R</code>	(rename-buffer NEWNAME &optional UNIQUE)	If <code>UNIQUE</code> argument is non-nil via <code>C-u M-x rename-buffer</code> , the name is auto generated to be unique.
Rename buffer - use unique name	<code><f11> b U</code>	(rename-uniquely)	Rename the current buffer by adding ‘<number>’ to the end. Use this if you want multiple <code>*Buffer*</code> or <code>*Info*</code> buffers for example. Example: StackExchange: How can I have multiple help buffer with different content
Kill current buffer	<ul style="list-style-type: none"><code><f11> b k</code><code>⌘-k</code><code>⌘-g</code>	(kill-current-buffer)	Kill (close) the current buffer. Does not prompt if there is no change in the buffer.
Kill buffer	<code>C-x k</code>	(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 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 <code>BUFFER-OR-NAME</code> , 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	<code><f11> b M-a</code>	(append-to-buffer BUFFER START END)	Append to specified <code>BUFFER</code> the text of the region. <ul style="list-style-type: none">The text is inserted into that buffer before its point.<code>BUFFER</code> can be a buffer or the name of a buffer; this function will create <code>BUFFER</code> if it doesn’t already exist.
Prepend region to specified buffer	<code><f11> b M-p</code>	(prepend-to-buffer BUFFER START END)	Prepend to specified <code>BUFFER</code> the text of the region. <ul style="list-style-type: none">The text is inserted into that buffer after its point.<code>BUFFER</code> can be a buffer or the name of a buffer; this function will create <code>BUFFER</code> if it doesn’t already exist.
Copy region to specified buffer (replacing old content)	<code><f11> b C-c</code>	(copy-to-buffer BUFFER START END)	Copy to specified <code>BUFFER</code> the text of the region. <ul style="list-style-type: none">The text is inserted into that buffer, replacing existing text there.<code>BUFFER</code> can be a buffer or the name of a buffer; this function will create <code>BUFFER</code> if it doesn’t already exist.
Insert content of specified buffer at point	<code><f11> b i</code>	(insert-buffer BUFFER)	Insert after point the contents of <code>BUFFER</code> . <ul style="list-style-type: none">Puts mark after the inserted text.<code>BUFFER</code> may be a buffer or a buffer name.
Append region’s text to specified file	<code><f11> b f</code>	(append-to-file START END FILENAME)	Append the contents of the region to the end of file <code>FILENAME</code> . <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 <code>Org-Mode</code> files). The following commands are available to manage indirect buffers.		
Create indirect buffer explicitly	<code><f11> b I m</code>	(make-indirect-buffer BASE-BUFFER NAME &optional CLONE)	Create and return an indirect buffer for buffer <code>BASE-BUFFER</code> , named <code>NAME</code> . <ul style="list-style-type: none"><code>BASE-BUFFER</code> should be a live buffer, or the name of an existing buffer.<code>NAME</code> should be a string which is not the name of an existing buffer.Optional argument <code>CLONE</code> non-nil means preserve <code>BASE-BUFFER</code>’s state, such as major and minor modes, in the indirect buffer.<code>CLONE</code> nil means the indirect buffer’s state is reset to default values.

Operation	Keystroke	Function	Notes
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.
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.
View Mode commands	<p>H, h, ? This message.</p> <p>Digits provide prefix arguments.</p> <p>- negative prefix argument.</p> <p>< move to the beginning of buffer.</p> <p>> move to the end of buffer.</p> <p>o scroll so that buffer end is at last line of window.</p> <p>SPC scroll forward "page size" lines.</p> <p>With prefix scroll forward prefix lines.</p> <p>DEL, S-SPC scroll backward "page size" lines.</p> <p>With prefix scroll backward prefix lines.</p> <p>z like SPC but with prefix sets "page size" to prefix.</p> <p>w like DEL but with prefix sets "page size" to prefix.</p> <p>d scroll forward "half page size" lines. With prefix, sets "half page size" to prefix lines and scrolls forward that much.</p> <p>u scroll backward "half page size" lines. With prefix, sets "half page size" to prefix lines and scrolls backward that much.</p> <p>RET, LFD scroll forward one line. With prefix scroll forward prefix line(s).</p> <p>y scroll backward one line. With prefix scroll backward prefix line(s).</p> <p>F revert-buffer if necessary and scroll forward.</p> <p>Use this to view a changing file.</p> <p>= prints the current line number.</p> <p>% goes prefix argument (default 100) percent into buffer.</p> <p>g goes to line given by prefix argument (default first line).</p> <p>. set the mark.</p> <p>x exchanges point and mark.</p> <p>@ return to mark and pops mark ring.</p> <p>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.</p> <p>m save current position in character register.</p> <p>' go to position saved in character register.</p> <p>s do forward incremental search.</p> <p>r do reverse incremental search.</p> <p>/ searches forward for regular expression, starting after current page.</p> <p>! and @ have a special meaning at the beginning of the regexp.</p> <p>! means search for a line with no match for regexp. @ means start search at beginning (end for backward search) of buffer.</p> <p>\ searches backward for regular expression, starting before current page.</p> <p>n searches forward for last regular expression.</p> <p>p searches backward for last regular expression.</p> <p>q quit View mode, restoring this window and buffer to previous state.</p> <p>q is the normal way to leave view mode.</p> <p>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.</p> <p>This command restores the previous read-only status of the buffer.</p> <p>E exit View mode, and make the current buffer editable even if it was not editable before entry to View mode.</p> <p>Q quit View mode, restoring all windows to previous state.</p> <p>c quit View mode and maybe switch buffers, but don't kill this buffer.</p> <p>C quit View mode, kill current buffer and go back to other buffer.</p> <p>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.</p> <p>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.</p>		
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.		
	<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.</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 		
🔑🔑🔑 Complete the list			

Operation	Keystroke	Function	Notes
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' - lsearch 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.	
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.	