












Key-Chords & Key-Seq

Action	Keystroke	Function	Note
Two Characters <ul style="list-style-type: none">key-chord, orkey-seq	<div>  With PEL user option pel-use-key-chord set to t or to use-from-start,<ul style="list-style-type: none">PEL activates the key-chord external package.PEL activates the global and mode-specific key chord bindings identified in the pel-key-chords user option.If pel-use-key-chord is set to use-from-start it activates the key-chords when Emacs starts, otherwise you must first activate the key-chord mode with the key-chord-mode command which PEL maps to the keystroke <f11> M-K.If pel-use-key-seq user option is set to t, the key-seq external package is also loaded and you can identify your key-chord sequences as key-seq sequences. The key-seq sequences impose a key order for detection which might help fast typists: if you define “4r” as you key-seq sequence it will only trigger the action if you type ‘4’ then ‘r’ quickly. Typing the ‘r’ then the ‘4’ quickly will not trigger the action. Note, however that key sequences defined with key-seq must only use ASCII characters in the decimal range of [32,126]. This means you cannot use control characters in key-seq sequences.For key-chord sequences you can use ASCII control characters; to include them in the 2 character sequence when editing you key-chord in the pel-key-chords, type C-q followed by the control key. For example type C-q C-i to insert a tab.</div> <div> PEL provides a set of pre-defined key-chords in the pel-key-chords user option and maps the the <f11> <f1> K to quickly access the PEL key-code customize buffer. and edit these values. You can add, delete or edit any of the provided key-chords, which provide examples of the ways to define your own key-chords. The list of key-chords PEL pre-defines and provides as default are show in the rows below.</div> <div><ul style="list-style-type: none">A key chord is a group of 2 normal, non-modifier keys that must be typed simultaneously to activate the action identified in the key chord definition.<ul style="list-style-type: none">Here, we are not talking of something like the normal Emacs key bindings like C-s, where the Control key and the s key are type together to do a CONTROL-S or where M-b represents using the Meta key and the b key together. The key-chords discussed here allow you to define actions when you type, for example, the key ‘j’ and the key ‘k’ together, or when you type the ‘.’ key twice quickly. <i>When the key-chord-mode is active</i> these special key-chord events are triggering the action you key-chord definition identifies. If the key-chord-mode is off, you get the normal Emacs behaviour of inserting the two keys inside the current buffer at point location.</div> <div> PLE also provides the following control user options for key-chords and key-seq:<ul style="list-style-type: none">pel-key-chord-two-keys-delay:<div>Max time delay between two key press to be considered a key chord.</div>pel-key-chord-one-key-delay:<div>Max time delay between 2 press of the same key to be considered a key chord. This should normally be a little longer than `key-chord-two-keys-delay`.</div>pel-key-chord-in-macros:<div>If nil, don't expand key chords when executing keyboard macros. If non-nil, expand chord sequences in macros, but only if a similar chord was entered during the last interactive macro recording. (This carries a bit of guesswork. We can't know for sure when executing whether two keys were typed quickly or slowly when recorded.)</div></div> <div> An earlier version of PEL (published June 14, 2020) did not support key-seq and only supported key-chord. If you used that version and saved a modified version of pel-key-chords, you will have to either delete it from your customization file or edit the elisp S-exp manually and add ‘key-chord’ (without the quotes) as the 3rd element of each list.</div>		
Toggle key-chord mode	<f11> M-K	(key-chord-mode ARG)	Toggle key chord mode. <ul style="list-style-type: none">With positive ARG enable the mode. With zero or negative arg disable the mode.A key chord is two keys that are pressed simultaneously, or one key quickly pressed twice. <div> Requires the key-chord external package.  PEL activates it when pel-use-key-chord user option is t.</div>
PEL Key-chords	The following rows describe the key-chords PEL defines by default in the pel-key-chords user option. <ul style="list-style-type: none">You can use them when the key-chord-mode is active.You can also decide to change them if they do not suit you, delete or add new ones by customizing the pel-key-chords user option. PEL provides a key binding to quickly access the customize buffer for key-chord control: <f11> <f1> KThe pel-key-chords user option has complete docstring that describes how to add news values.		
Key Chords (See also:  Customize)	<f11> <f1> K	(pel-customize-key-chords &optional OTHER-WINDOW)	Customize PEL Key Chord support. <ul style="list-style-type: none">If OTHER-WINDOW is non-nil (use C-u), display in another window.
PEL Pre-defined key-chords	<ul style="list-style-type: none">PEL default for pel-key-chords are identified in the tables of this document with the characters underlined.In some cases the key-chord is a simple binding to execute a command or an Emacs Lisp lambda form. In that case the 2 key-chord keys are shown in the keystroke column alone, simply underlined.In other cases, the key-chord inserts characters and execute commands. In such as case, the 2 key-chord keys are also shown in the keystroke column alone, but instead of describing the function in the function column, the cell shows the key-chord string which represent both the character inserted and the key code for the command.<ul style="list-style-type: none">For example, the key-chord that consist of typing the < key and the > key together is represented as the <> key-chord and the expansion is show as “<>\C-b” . The effect is to insert both angle brackets and put point in between, since C-b is bound to to command backward-char.The color of the key-chord corresponds to the availability of the commands used, if any. A key-chord that depends only on Emacs standard commands or simple characters is therefore shown in black. <div> PEL pre-defined key-chords are key-chords, not key-seq. Note that key-seq cannot use tab the way it is used for pel-indent-rigidly below.</div> <div> With key-chord or key-seq defined as lambdas, you can pass arguments to the called command, just as any other key binding. You can control whether a key-chord is allowed in a read-only buffer for example, and/or pass numeric arguments. PEL uses this ability in the definitions of the key-chords using pen commands but it could be applied to anything defied with a lambda.</div>		
Insert <> and place point between them	<>	<>\C-b	Global: available in all modes.
Insert [] and place point between them	[]	[]\C-b	Global: available in all modes.
Insert {} and place cursor between	{ }	{\n\n}\C-p\C-p	Available in c-mode and c++-mode
Move to window above	<u>y</u><u>u</u>	(windmove-up &optional ARG)	Select the window above the current one. <ul style="list-style-type: none">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. Global: available in all modes.
Move to window below	<u>b</u><u>n</u>	(windmove-down &optional ARG)	Select the window below the current one. <ul style="list-style-type: none">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. Global: available in all modes.
Move to window at left	<u>f</u><u>g</u>	(windmove-right &optional ARG)	Select the window to the right of the current one. <ul style="list-style-type: none">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. Global: available in all modes.
Move to window at right	<u>j</u><u>k</u>	(windmove-left &optional ARG)	Select the window to the left of the current one. <ul style="list-style-type: none">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. Global: available in all modes.

Action	Keystroke	Function	Note
Indent rigidly	<u><tab>q</u>	(pel-indent-rigidly &optional N)	<p>Indent rigidly the marked region or current line N times.</p> <ul style="list-style-type: none"> • If a region is marked, it uses 'indent-rigidly' and provides the same prompts to control indentation changes. • If no region is marked, it operates on current line(s) identified by the numeric argument N (or if not specified N=1): <ul style="list-style-type: none"> • N = [-1, 0, 1] : operate on current line • N > 1 : operate on the current line and N-1 lines below. • N < -1 : operate on the current line and (abs N) -1 lines above. <p>👉 Command numeric prefix is available with the key-chord binding.</p> <p>Indent all lines starting in the region.</p> <ul style="list-style-type: none"> • If called interactively with no prefix argument, activate a transient mode in which the indentation can be adjusted interactively by typing <left>, <right>, <S-left>, or <S-right>. <p>-----</p> <p>These commands activate a transient mode where Emacs prompts for extra keys to control how to indent. Indenting and un-indenting is possible. The capabilities are controlled by the variable <i>indent-rigidly-map</i> with by default provides:</p> <ul style="list-style-type: none"> • S-<right> indent-rigidly-right-to-tab-stop • S-<left> indent-rigidly-left-to-tab-stop • <right> indent-rigidly-right • <left> indent-rigidly-left <p>Typing any other key deactivates the transient mode.</p>
Correct mode at point	<u>4r</u>	(flyspell-correct-word-before-point &optional EVENT OPOINT)	<p>Pop up a menu of possible corrections for misspelled word before point.</p> <p>Available when current buffer has flyspell-mode or flyspell-prog-mode enabled.</p>
Find file at point (See also <u>Σ</u> File-mngt, <u>Σ</u> reStructuredText)	<u>6y</u>	(pel-open-at-point &optional N)	<p>Open the file, library or the URL, named at point, with potential line & column #s.</p> <ul style="list-style-type: none"> • If point is on a reStructuredText link in a rst-mode buffer, open the link target (that might be a local file or a URL on remote web site. In the latter case the page is opened in the systems' browser). • If embedded space(s) are allowed in the filename, then point must be located at the first of the 2 delimiter characters. These delimiter character can be any of the following: ``~' () [] {} <> ' ' " " 「 」 □ ◇ 《 》 [] [] « » , , <> () . . • In the above list, the first 12 characters are ASCII characters, the remainders are Unicode characters: ``' ' " " 「 」 □ ◇ 《 》 [] [] « » , , <> () . . • Tab and newline are also delimiter characters. • If embedded space in the file name is not allowed, then the file name must also be enclosed in the above delimiters, the space acts as an extra delimiter, and point can be positioned anywhere between the delimiters. • If the string identifies a URL, the function opens the page in the default browser. • Prompts for incomplete file names, allowing editing the find file (with completion), search for libraries files (type 1) according to current file type. <p>🔧 Currently only supports Emacs Lisp files. Planning to support other programming languages with and without project management packages.</p> <ul style="list-style-type: none"> • Without argument: <ul style="list-style-type: none"> • If file is already opened in a window, move point to that window and to the line column coordinates if specified following the file name at point. • If no window holds that file, select the target window based on the number of editable windows in frame: if 1, split that window and use the new window, if 2: use the other window, if 3 or more, use the current window. • With numeric argument N: <ul style="list-style-type: none"> • N < 0 : create a new window and use that • N = 0: use the 'other' (the next) window • N = 1,3,7or above (excluding 9): select the target window based on the number of editable windows in frame: if 1, split that window and use the new window, if 2: use the other window, if 3 or more, use the current window. • N is: 8: up, 2: down, 4:left, 5:current, 6:right. • N is 9: open the file in the system's browser, and for a directory name at point open the application associated with directory browsing (eg. macOS Finder, Windows Explorer). • Selecting Minibuffer, inexistent or dedicated window is not allowed. • If the file name is followed by line and column numbers the point is moved to that position. <p>More information available in the command's help docstring.</p> <p>👉 Command numeric prefix is available with the key-chord binding.</p> <p>Global: available in all modes.</p>
Open filename at point in a browser (See also <u>Σ</u> File-mngt)	<u>6u</u>	(pel-browse-filename-at-point)	<p>Open the file name at point inside the system's browser.</p> <ul style="list-style-type: none"> • If point is at a directory name, open the systems application that browses directories (eg. macOS Finder, Windows Explorer). <p>👉 This is the same as using pel-open-at-point with the argument N set to 9. It is easier to type and PEL assigns its own key-chord for it.</p>
Search word at point from top of current buffer	<u>.-i</u>	(pel-search-word-from-top &optional N)	<p>Search word at point from top/bottom of buffer in window identified by N.</p> <ul style="list-style-type: none"> • Search direction: <ul style="list-style-type: none"> • If N is nil, 0 or larger, perform a search-forward from the top of the buffer in window identified by N. • If N is negative: perform a isearch-backward from the bottom of the buffer in the window selected by the absolute value of N. • Window selection: <ul style="list-style-type: none"> • If N is not specified, nil, 1, 3, 7 or 9 and larger: search in current window. • If N is 0: : search in other window • If N in [2,8] range, search in window identified by the direction corresponding to the cursor in a numeric keypad: <pre> 8 := 'up 4 := 'left 5 := 'current 6 := 'right 2 := 'down </pre> • Temporary word mode toggle: detecting a 'word' is affected by the subword-mode and superword-mode. When searching in current buffer, the following values of N temporary toggle the mode when grabbing the word: <ul style="list-style-type: none"> • If N is 7: temporary toggle subword-mode to grab the word. • If N is 9: temporary toggle superword-mode to grab the word. • Explicitly selecting the minibuffer window, or a non-existing window is not allowed, and search is done in current window. • Searched word is remembered and can be used again to repeat an interactive search with C-s or C-r. • Position before searched word is pushed on the mark ring. <p>👉 Using superword-mode allows you to search for function names in buffer for programming languages. If you do not want to change the mode but want to search for the word as interpreted by the other state of the mode type the command with N equal to 9: M-9 <f11> s .</p> <p>👉 Command numeric prefix is available with the key-chord binding.</p> <p>Global: available in all modes.</p>

Key-Chords — References

Topic & Link	Description
Emacs normal key sequences	Emacs supports binding commands to key sequences of your choice. In the sequence you can have keys that are typed along with one or several key modifiers (Control, Meta, Super, Hyper, Alt) and you can use several keys: the first set(s) being used as key prefixes.
Key Sequences @ EmacsWiki	Describes what a normal/standard Emacs key sequence is.
Key-Chord extension package	The key-chord package provides the ability to unbind commands to an event consisting of typing 2 keys simultaneously, those keys not using key modifiers. For example you could bind a command to pressing the '4' and the 'r' key simultaneously.
key-chord @ MELPA	This page shows the doc coming from the key-chord.el. It's where PEL gets the file from.
Key Chords @ Emacs Wiki	Some interesting discussion about key-chord.
key-chord.el @ Emacs Wiki	
Key-Set extension package	<p>The key-seq package builds on key-chord package. It changes the way simultaneous keys are detected and imposes an order to these keys. So for a key-seq of "jk" it only accepts the keys if 'j' is type before and 'k' is typed quickly following it. Depending of your typing skills this may help reduce the unwanted triggers of key chords.</p> <p>PEL supports both key-chord and key-seq to the level where you can define a mix of bindings: they can be key-chord or key-seq bindings.</p>
key-seq @ MELPA	
key-seq @ GitHub	