Cut & Paste — Copy/Delete/Kill/Yank

Operation	Keystroke	Function	Note
OS Clipboard Commands	• When Emacs runs in graphical mode, the following commands can be used to copy and paste from the OS (system) clipboard. • On macOS this can also be done using the standard This can also be done with the standard **⌘-c**, **⌘-v** and **⌘-x** keystrokes. • On macOS, with emacs running under Terminal.app, you can use **⌘-v** to paste from the OS-clipboard. And when xterm-mouse-mode is off you can select text by marking it with the mouse, then use **⌘-c** to copy to the OS clipboard. **⌘-x** does not work in terminal mode. • The PEL package binds **<f11> <f12>** to xterm-mouse-mode when Emacs runs in terminal mode. Use this to change the way text selection works with the mouse.		
Copy text to clipboard	• **<f11> C c** • **⌘-c**	• (**clipboard-kill-ring-save** BEG END &optional REGION) • (**ns-copy-including-secondary**)	Copy region to kill ring, and save in the OS clipboard. In terminal mode, when the xtem-mouse-mode is off, the **⌘-c** key copies text, but copies the **terminal** text, so if you want to copy multiple lines, ensure there is only one Emacs window horizontally. In graphics mode **⌘-c** copies the text via the Emacs application and invokes the (ns-copy-including-secondary) function.
Paste text from clipboard	• **<f11> C v** • **⌘-v**	(**clipboard-yank**)	Insert the OS clipboard contents, or the last stretch of killed text. In graphics mode **⌘-v** executes the standard (yank &optional ARG) which supports the clipboard.
Cut region & place both in kill ring and on system clipboard	• **<f11> C x** • **⌘-x**	(**clipboard-kill-region** BEG END &optional REGION)	Kill the region, and save it in the OS clipboard.
Copy Commands	Emacs copy commands copy text into the "*kill ring*" . Other commands are used to take text from the kill ing and insert it in the buffer. ☛ By default, Emacs does not support the CUA compliant **C-c** for copy. To support that key you must enable the cua-mode.		
Copy character at point	**<f11> = c**	(**pel-copy-char-at-point** &optional N)	Copy single character at point. With argument N, copy N consecutive characters; a negative N copies the character backwards (before point).
Copy whitespaces at point	**<f11> = SPC**	(**pel-copy-whitespace-at-point**)	Kill all whitespace characters at/ around point on current line.
Copy complete word at point	• **<f11> = w** • **<C-kp-add>**	(**pel-copy-word-at-point**)	Copy word at point. See changing the word mode to include or exclude some characters as word delimiters: • subword-mode . To toggle that mode: **<f11> t w b** • superword-mode . To toggle that mode: **<f11> t w s**
Copy complete symbol at point	• **<f11> = .** • **M-+** • **<M-kp-add>**	(**pel-copy-symbol-at-point**)	Copy symbol at point. Syntax depends on the syntax table for the buffer. ☝ See table ∑ Text Modes for information on text modes that affects this. The **<f11> t m ?** command displays the mode and the **<f11> t m** prefix allows modifications of the mode.
Copy filename at point	**<f11> = F**	(**pel-copy-filename-at-point**)	Copy filename at point
Copy URL at point	**<f11> = u**	(**pel-copy-url-at-point**)	Copy URL at point
Copy line beginning	**<f11> = a**	(**pel-copy-line-start**)	Copy text from the beginning of the current line up to point

Copy region or line at point
★ **PEL Enhanced Key** ★
	• **M-w**
	• **<f11> = l**
	• **<f11> +**
		(**pel-copy-marked-or-whole-line**)	Flexible copy to kill ring.: copy visible region if any, otherwise copy current line to kill ring.
			The copy operation is controlled by the (optional) argument:
			• If N = 0: copy region (regardless of whether it is visible or not.
			• If a region is active/visible: copy the region's text.
			• if no region is active/visible copy N lines:
			  • If no argument, (N=1) copy current line.
			  • If N > 0: copy current line and N-1 following lines.
			  • If I < 0: copy current line and N-1 previous lines.
			All copied lines are complete.
			The copied text is saved in the kill-ring.
			All copy operations are performed by `**kill-ring-save**' (the original binding for that key).
			☛ Replaces standard binding to kill-ring-save which only copies region.
			☛ In graphics mode: text is also copied to the OS clipboard.

Copy line end	**<f11> = e**	(**pel-copy-line-end**)	Copy text from point up to the end of the line
Copy complete sentence at point	**<f11> = s**	(**pel-copy-sentence-at-point**)	Copy entire sentence at point. Toggle the minimum number of spaces that end a sentence with: pel-toggle-sentence-end : **<f11> t m s**
Copy paragraph beginning	**<f11> = b**	(**pel-copy-paragraph-start**)	beginning of paragraph to point
Copy paragraph	**<f11> = H**	(**pel-copy-paragraph-at-point**)	Copy entire paragraph at point.
Copy paragraph end	**<f11> = h**	(**pel-copy-paragraph-end**)	Copy from point to end of paragraph
Copy list at point	**<f11> = (**	(**pel-copy-list-at-point**)	Copy complete Lisp-syntax list at point.
Copy S-expression at point	**<f11> = x**	(**pel-copy-sexp-at-point**)	Copy complete Lisp S-expression at point.
Copy function at point	**<f11> = f**	(**pel-copy-function-at-point**)	Copy complete body of function at point.
Deleting Text	Emacs supports "deleting" and "killing" text. Deleted text is not retained. Killed text is retained in the "kill ring".		
Delete 1 Character	**Delete Keys:** Emacs recognizes 2 delete keys: 1) a **delete forward** and 2) a **delete backward** (backspace). Some keyboards have both, others have only one of them (e.g. macOS laptop keyboards). On those the forward delete key is composed with the **Fn** key and the backspace key. ☛The behaviour of the delete keys is controlled by the normal-erase-is-backspace variable, which can be customized and controlled by executing the command normal-erase-is-backspace-mode. See: Emacs Manual -- If <DEL> Fails to Delete. ◉ This table uses the ⌦ and ⌫ symbols to represent these 2 keys: 1. ⌦ := "*forward delete*" := **<deletechar>** := **Fn** ⌫ 2. ⌫ := "*backward delete*" := **<backspace>** Often labelled "delete" on keyboards. ⚠ **C-**⌦ and **C-**⌫ are not accessible in terminal mode.		
character - forward	**C-d**	(**delete-char** N &optional KILLFLAG)	Delete following N characters (previous if N is negative). N defaults to 1. ☛When region is marked: region is deleted if delete-selection-mode is on, otherwise the region is not deleted.
character - forward	⌦	(**delete-forward-char** N &optional KILLFLAG)	Delete following N characters (previous if N is negative). N defaults to 1. ☛When region is marked: region is deleted, regardless of argument and state of delete-selection-mode.
Delete character - backward	⌫	(**backward-delete-char-untabify** ARG &optional KILLP)	Deletes character before cursor (deletes backward), replaces hard tab with spaces as required. With arguments: • positive numeric argument: kill that many characters backward • negative numeric argument: kill that many characters forward When region is marked: the region is deleted, regardless of argument.

Operation	Keystroke	Function	Note
Delete whitespace	The following Emacs commands delete whitespaces. The deleted characters are not copied in the kill ring. These commands are also described in the **Text Whitespace** table.		
Delete all spaces between 2 words	`M-\`	(**delete-horizontal-space** &optional BACKWARD-ONLY)	Delete all spaces and tabs around point. Only works when cursor is on the spaces between the words or on the first character of the second word.
Delete all spaces but one beween words	`M-SPC`	(**just-one-space** &optional N)	Delete all spaces and tabs around point, leaving one space (or N spaces). If N is negative, delete newlines as well, leaving -N spaces. • This command ensures that words are separated by just one space character. • The cursor may be between the words but can also be on the fist character of the word. • At the end of the word it inserts a space.
Delete all contiguous blank lines after point	`C-x C-o`	(**delete-blank-lines**)	• On blank line, delete all surrounding blank lines, leaving just one. • On isolated blank line, delete that one. • On nonblank line, delete any immediately following blank lines.
Delete Indentation, join this line to the previous one	`M-^`	(**delete-indentation** &optional ARG)	Adjoint the current line with the line above. It joins 2 lines by deleting the intervening newline, along with any indentation following it.
Cycle spacing around point	`<f11> t SPC`	(**cycle-spacing** &optional N PRESERVE-NL-BACK MODE)	Manipulate whitespace around point in a smart way. • The first call in a sequence acts like 'just-one-space'. It deletes all spaces and tabs around point, leaving one space (or N spaces). N is the prefix argument. If N is negative, it deletes newlines as well, leaving -N spaces. (If PRESERVE-NL-BACK is non-nil, it does not delete newlines before point.) • The second call in a sequence deletes all spaces. • The third call in a sequence restores the original whitespace (and point). The easiest way to use this command for the second or third call (or further) is to issue it once and then use the repeat command (`C-x z` or `<f5>`).
Delete all trailing whitespaces	`<f11> t w t`	(**delete-trailing-whitespace** &optional START END)	Delete trailing whitespace in the entire (or narrowed part of the) buffer or in the marked region. • This command deletes whitespace characters after the last non-whitespace character in each line between START and END. It does not consider formfeed characters to be whitespace. • If this command acts on the entire buffer, it also deletes all trailing lines at the end of the buffer if the variable 'delete-trailing-lines' is non-nil.
Remove non required whitespaces	`<f11> t w c`	(**whitespace-cleanup**)	Cleanup some blank problems (non-required whitespace) in all buffer or at region. • It usually applies to the whole buffer, but in transient mark mode when the mark is active, it applies to the region. It also applies to the region when it is not in transient mark mode, the mark is active and **C-u** was pressed just before calling 'whitespace-cleanup' interactively.
Delete all spaces between point and next non-white	`C-⌦`	(**pel-delete-to-next-visible**)	Delete all whitespace between point and next non-whitespace character. on macOS laptop, use: `Fn C-delete`
Delete duplicate lines	`<f11> - *`	(**delete-duplicate-lines** BEG END &optional REVERSE ADJACENT KEEP-BLANKS INTERACTIVE)	Delete all but one copy of any identical lines in the region (or entire buffer if nothing marked). • If REVERSE is non-nil (interactively, with a C-u prefix), it searches backwards and keeps the last instance of each repeated line. • Identical lines need not be adjacent, unless the argument ADJACENT is non-nil (interactively, with a C-u C-u prefix). This is a more efficient mode of operation, and may be useful on large regions that have already been sorted. • If the argument KEEP-BLANKS is non-nil (interactively, with a C-u C-u C-u prefix), it retains repeated blank lines. • Prints a message describing the number of deletions.
Kill Commands	Emacs kill commands erase text and copy it into the kill ring.		
Kill character at point	`<f11> - c`	(**pel-kill-char-at-point** &optional N)	Kill single character at point. With argument N, kill N consecutive characters; a negative N kills characters backwards.
Kill trough next occurrence of char	`M-z char`	(**zap-to-char** ARG CHAR)	Kill up to and including ARGth occurrence of CHAR. Case is ignored if 'case-fold-search' is non-nil in the current buffer. Goes backward if ARG is negative; error if CHAR not found.
Kill text between point and mark	`S-⌦`	(**kill-region** BEG END &optional REGION)	Kill text between mark and point, even if region is not marked. See also: `C-w`.
Kill whitespace at point	`<f11> - SPC`	(**pel-kill-whitespace-at-point**)	Kill all whitespace characters at/around point on current line. Copy them to kill ring.
Kill word backward	• `M-⌫` • `C-⌫`	(**backward-kill-word** ARG)	Kill characters backward until beginning of word.
Kill word (forward)	• `C-S-⌦` • `M-d`	(**kill-word** ARG)	By default kill forward from point up to the end of the current word. Numeric argument specify number of consecutive words. Negative argument reverses the direction.
Kill word at point	• `<f11> - w` • `<C-kp-subtract>`	(**pel-kill-word-at-point**)	Kill the complete word at point, regardless of point's position inside the word.
Kill symbol at point	• `<f11> - .` • `<M-kp-subtract>`	(**pel-kill-symbol-at-point**)	Kill the complete word at point as identified by word and symbol syntactic unit, regardless of point's position inside the word. This is useful in source code files when the subword-mode and superword-mode are not activated; it kills all consecutive characters that include symbol characters such as '-'.
Kill beginning of line	• `M-0 C-k` • `C-\` • `<f11> - a`	(**pel-kill-from-beginning-of-line**)	Kills the beginning of the line up to the cursor. In terminal the M binding ⌫ does not work properly, and they do different things! • <M-⌫> binds to <C-backspace> executing backward-kill-word. • <M-S-⌫> binds to (mark-defun &optional ARG) instead (which is bound to C-M-h). The binding works properly in graphics mode. This used to be mapped to (backward-kill-word ARG) to implement delete word backward. But remapped it.
Kill to end of line	• `M-⌦` • `C-k` • `<f11> - e`	(**kill-line** &optional ARG)	Kills from current position to end of line. If no visible characters on it kill through newline. M-⌦ is bound to (insert-parentheses &optional ARG) as in M-(in terminal mode. The M-⌦ binding works properly in graphics mode.
Kill whole line	`C-S-⌫`	(**kill-whole-line** &optional ARG)	Deletes current line (in graphics mode). Use `C-w`: it's more flexible.

Operation	Keystroke	Function	Note

Kill/Delete marked region/line(s)
 	
★ PEL Enhanced Key ★
	• **C-w**
	• **<f11> - l**
	• ⌘-x
		(**pel-kill-or-delete-marked-or-whole-line** &optional N)	Flexible region/whole-line kill/delete.
			• N=0 := kill region (active/visible or not)
			• Sign of N selects operation:
			  • positive := kill (default)
			  • negative := delete
			• Select text to delete/kill based on presence of region:
			  • if a region is marked: kill/delete region's text,
			  • if no region: kill/delete abs(N) lines, start at point.
			• If operation is to kill 1 line and the line is empty, then delete line instead of killing it.
			• **Scenarios:**
			  • With no arg:
			  • with no active/visible region: kill current line, but if line is empty delete it.
			  • with an active/visible region: kill region's text.
			  • With arg **0**: (**M-0 C-w**) : kill region's text, whether region is active/visible or not.
			  • With a non zero arg:
			  • With no region active/visible:
			  • With arg **-** : (**M- - C-w**) or (**C- - C-w**) : delete current line
			  • With arg **- 1** : (**M- - 1 C-w**) or (**C- - 1 C-w**) : delete current line
			  • With arg **4**: (**M - 4 C-w**) : kill 4 lines including current one.
			  • With arg **-3**: (**M- - 3 C-w**) : delete 3 lines including current one.
			  • With a region active/visible:
			  • With any negative mark argument: delete the region's text.
			  • With no argument or any positive argument: kill the region's text.
			☛ This replaces the standard Emacs binding to kill-region which always kill text between mark and point, even when the region is not marked. When text is killed it is killed with kill-region, so it retains the filtering and kill ring text appending capabilities.
			☛ In graphics mode this also copies text to the OS clipboard.
			 On macOS in graphics mode only: PEL rebinds ⌘-x from (kill-region) to this command, making this easy to use key able to perform more.

Kill sentence - backward	**C-x** ⌫	(**backward-kill-sentence** &optional ARG)	Kill back from point to start of sentence. With arg, repeat, or kill forward to Nth end of sentence if negative arg -N.
Kill sentence - forward	**M-k**	(**kill-sentence** &optional ARG)	Kill from point to end of sentence. With arg, repeat; negative arg -N means kill back to Nth start of sentence.
Kill sentence at point	**<f11> - s**	(**pel-kill-sentence-at-point**)	Kill complete sentence at point.
Kill forward to end of paragraph	**<f11> - h**	(**kill-paragraph** ARG)	Kill forward to end of paragraph. With arg N, kill forward to Nth end of paragraph; negative arg -N means kill backward to Nth start of paragraph.
Kill back to start of paragraph	**<f11> - b**	(**backward-kill-paragraph** ARG)	Kill back to start of paragraph. With arg N, kill back to Nth start of paragraph; negative arg -N means kill forward to Nth end of paragraph.
Kill complete paragraph at point	**<f11> - H**	(**pel-kill-paragraph-at-point** &optional N)	Kill complete paragraph at point. With argument N, kill N consecutive paragraphs; a negative N kills the current one and N-1 previous paragraphs.
Kill next Lisp S-expression	• **C-M-k** • **<f11> -]**	(**kill-sexp** &optional ARG)	• No argument: kill the next sexp (or the current from the point forward). • With negative sign: kill the previous sexp (the sexp backward). • For example: **M- - C-M-k** kills the sexp backward. • With numeric argument: kill that many sexp in the direction identified by the sign of the argument.
Kill previous Lisp S-expression	• **C-M-**⌦ • **<f11> - [**	(**backward-kill-sexp** &optional ARG)	Kill the sexp (balanced expression) preceding point. • With ARG, kill that many sexps before point. • Negative arg -N means kill N sexps after point. • This command assumes point is not in a string or comment. ⚠ Note: In some text (like The Common Lisp Cookbook - Using Emacs as a Lisp IDE) , the **C-M-<backspace>** keystroke is being described to kill the previous sexp. This key does not seem to be used anymore. This key chord is normally not accessible in terminal mode as it would map to **C-M-h** instead. The **C-M-**⌦ binding only works in terminal mode. Since this key-chord is not the best match for the operation, use **M- - C-M-k** instead or use the PEL **<f11> - [**
Kill Lisp S-Expression at point	**<f11> - x**	(**pel-kill-sexp-at-point**)	Kill the S-Expression at point. The point must be at the opening parenthesis or just after the closing parenthesis.
Kill Lisp list at point	**<f11> - (**	(**pel-kill-list-at-point**)	Kill the balanced expression at point: a block of text between parentheses, braces, squared or angled bracket, single or double quotes. Point must be located at the opening block character.
Kill function at point	**<f11> - f**	(**pel-kill-function-at-point**)	Kill the function at point. Point can be anywhere, or just past the function code.
Kill filename at point	**<f11> - F**	(**pel-kill-filename-at-point**)	Kill the filename at point. Point can be located anywhere inside the file name or right after.
Kill URL at point	**<f11> - u**	(**pel-kill-url-at-point**)	Kill the URL at point. Point can be located anywhere inside the file name or right after.
Yank	Emacs calls "yanking" the action of inserting previously killed or copied text, retrieved it from the "*kill ring*". Other editors call this "*pasting text*".		
Yank last killed into buffer	• **C-y** • ⌘-v	(**yank** &optional ARG)	Reinsert ("paste") the last stretch of killed text. • More precisely, reinsert the most recent kill, which is the stretch of killed text most recently killed OR yanked. Put point at the end, and set mark at the beginning without activating it. With just C-u as argument, put point at beginning, and mark at end. With argument N, reinsert the Nth most recent kill. ☛ In graphical mode: supports OS clipboard.
Paste from OS clipboard	⌘-y	(**ns-paste-secondary**)	On macOS in graphics mode only: paste from OS clipboard (not from kill ring).
Replace last yank with previous kill	**M-y** **<f11> y**	• (**yank-pop** &optional ARG) • (**popup-kill-ring**)	Replace just-yanked stretch of killed text with a different stretch. • This command is allowed only immediately after a 'yank' or a 'yank-pop'. At such a time, the region contains a stretch of reinserted previously-killed text. 'yank-pop' deletes that text and inserts in its place a different stretch of killed text. • With no argument, the previous kill is inserted. With argument N, insert the Nth previous kill. If N is negative, this is a more recent kill. • The sequence of kills wraps around, so that after the oldest one comes the newest one. ☞ Also referred to as: "yank next". ☛ In **graphics mode** when popup-kill-ring package and its pre-requisites pos-tip and popup are installed, the **M-y** keystroke pops up a menu of kill-ring entries to allow selection of the entry to yank: you don't need to use **C-y** to first yank text. In this environment, if you still want to to use **C-y** and then want to perform yank-pop manually, use the PEL **<f11> y** keystroke, which is always available and always bound to yank-pop.

Operation	Keystroke	Function	Note
Append to Kill Ring	**C-M-w**	(**append-next-kill** &optional INTERACTIVE)	Preparation command. Next kill command issued after this will add to the top of the kill ring item (the previous kill): • If the next command kills forward from point, the kill is appended to the previous killed text. • If the command kills backward, the kill is prepended. If the next command is not a kill command, this has no effect.
Manage Kill Ring	The following are examples of commands that can be used to show the kill ring and the various variables that control it. The kill-ring is an Emacs variable. It can be manipulated by Emacs Lisp code and its content can be shown using the help variable command. The maximum number of elements inside the kill ring is also controllable.		
Display content of kill ring	**<f1> v kill-ring <RET>**		Display the content of the kill ring in the *Help* buffer
Display kill ring size	**<f1> v kill-ring-max <RET>**		Display the maximum number of kill ring entries in the *Help* buffer.
Set kill ring size	M-x set-variable <RET> kill-ring-max <RET>		The variable **kill-ring-max** is the number of entries in the kill ring. Defaults to 60.
Select text stored in kill ring	**<F10>** → Edit → Select and Paste		Use the Select and Paste menu entry to list each entry of the kill ring and insert it at point.

Cut & Paste — References

Topic & Link	Notes
GNU Emacs Manual: Killing and Moving Text	
GNU Emacs Manual: Killing - Yanking	
Copy & Paste	
Emacs Wiki - Copy and Paste	
simpleclip	
Emacs Wiki - Deleting Whitespace	
Delete without storing to Kill Ring	
Emacs: how to delete text without kill ring? @ StackOverflow	
Emacs: Deleting a line without sending it to the kill ring @ StackExchange	
Backspace without adding to kill ring @ Stack Exchange	
Kill or copy current line with minimal keystrokes	
show-marks.el @ Emacs Wiki	