Indenting & Tab

Description	Keystroke	Function	Note
Indentation under Emacs	🚧 Emacs controls indentation via major modes and the tab key may take different and surprising behaviour for people just starting with Emacs. This table is in early stage of development, more to come with documentation of the behaviour for different modes.		
Insert Literal Tab	`C-q <tab>`	(**quoted-insert** ARG) <tab>	Inserts a hard tab inside the file but moves the cursor to the column that represents the next multiple of *tab-width*.
Behaviour of Tab	In Emacs the behaviour of the <tab> key depends on the major mode of the current buffer. This key is rebound by several major mode. By default, in text modes, tabs are set to 8 spaces, inserting hard tabs. However, if there is text in the above lines, the tab moves to the spot under the word above. Note that if a line is full of text (without any space), then the tab stops controlled by the ruler take effect again.		
Indent current line (or region)	`<tab>`	(**indent-for-tab-command** &optional ARG)	Indent the current line or region, or insert a tab, as appropriate. • This function either inserts a tab, or indents the current line, or performs symbol completion, depending on 'tab-always-indent'. The function called to actually indent the line or insert a tab is given by the variable '**indent-line-function**'. • If a prefix argument is given, after this function indents the current line or inserts a tab, it also rigidly indents the entire balanced expression which starts at the beginning of the current line, to reflect the current line's indentation. • In most major modes, if point was in the current line's indentation, it is moved to the first non-whitespace character after indenting; otherwise it stays at the same position relative to the text. • If 'transient-mark-mode' is turned on and the region is active, this function instead calls 'indent-region'. In this case, any prefix argument is ignored. ⚠️🚧 The behaviour of the tab key vastly differ between major modes. This ranges from not moving the cursor at all if the indentation is identified as correct for the current context, to cycling through various potential positions to just what someone new to Emacs would expect. Much more has to be documented on the behaviour of that key and how it can be controlled and customized. It's quite possible that the best way to document its behaviour would be to place a description inside the table of each major mode.
		indent-for-tab-command &optional ARG)	In **Lisp** related modes. • indent-line-function = indent-relative. • tab-always-indent = t 🚧The above values that I got in Emacs via inspection do not explain tab behaviour in the Emacs Lisp mode (which is to indent the code according to Emacs Lisp semantics, a **very** useful feature when writing Lisp code). In this mode tab corrects the indentation of the code at the current line, which may be to indent, de-indent or do nothing.
		(**c-indent-line-or-region** &optional ARG REGION)	In C related modes. Indent active region, current line, or block starting on this line. • In Transient Mark mode, when the region is active, reindent the region. • Otherwise, with a prefix argument, rigidly reindent the expression starting on the current line. • Otherwise reindent just the current line.
Indent lines of list after point **(CLBC s3.lisp)**	`C-M-q`	(**indent-sexp** &optional ENDPOS)	Indent each line of the list starting just after point. • If optional arg ENDPOS is given, indent each line, stopping when ENDPOS is encountered.
Insert spaces or tabs to next defined tab-stop column	`M-i`	(**tab-to-tab-stop**)	Works in following modes: • *.c: C-mode : every 4, inserts spaces • C++ mode: every 4 • Erlang mode: every 4 • *.py: Python mode: every 8, inserts spaces • Emacs-Lisp: every 4: (4 8 12 16) • Haskell: every 8: (8 16 24 32) • text: every 4: (4 8 12 16) • reStructuredText: every 4 • org mode: every 4 • Fundamental mode: every 4 • makefile, *.mk: BSDmakefile mode: shows every 4, inserts tabs. <tab> inserts tabs and moves cursor.
Insert an indented line below current line	• M-<ret> • <f11> <tab> <ret>	(**pel-newline-and-indent-below**)	Insert an indented line just below current line.
Change the tab stops	`M-x edit-tab-stops`		Opens a ***Tab Stops* buffer**. Identify the tab stops in the first line with colons. Use C-c C-c to activate and exit the buffer. Again, the tab stop take effect at the top of the buffer,
Change the tab width	`M-: (setq tab-width N)`		The variable *tab-width* normally defaults to 8 in emacs. It can be set locally inside a buffer with (setq tab-width N) or globally with (setq-default tab-width N). The M-: keystroke allows evaluating a lisp expression interactively (as the above two). Note that any literal tab in the buffer impact the location of the column where the next character shows. When changing the tab-width, the layout shown in the window that contains literal tabs will be modified according to the new tab-width value.
Make <tab> insert space/tab	`M-: (setq indent-tabs-mode nil/t)`		By default, pressing <tab> insert literal (hard) tabs inside the file. The indent-tabs-mode variable controls that: set to t it inserts tabs, set to nil it inserts spaces.
Next line, indented	`C-j`	(**electric-newline-and-maybe-indent**)	Add new line and indent next line. Indentation is controlled by the variable **left-margin**. Pressing **Tab** anywhere on the line also indents the line properly.

Indent Region	`C-M-\`	(**indent-region** START END &optional COLUMN)	Indent each nonblank line in the region.
			• A numeric prefix argument specifies a column: indent each line to that column.
			• With no prefix argument, the command chooses one of these methods and indents all the lines with it:
			  1. If 'fill-prefix' is non-nil, insert 'fill-prefix' at the beginning of each line in the region that does not already begin with it.
			  2. If 'indent-region-function' is non-nil, call that function to indent the region.
			  3. Indent each line via 'indent-according-to-mode'.

Move to fist nonbank character on the line	`M-m`	(**back-to-indentation**)	Move point to the first non-whitespace character on this line.
Split current line & indent	`C-M-o`	(**split-line** &optional ARG)	Split current line, moving portion beyond point vertically down. If the current line starts with 'fill-prefix', insert it on the new line as well. With prefix ARG, don't insert 'fill-prefix' on new line.

Description	Keystroke	Function	Note
Delete Indentation, join this line to the previous one (See also ∑ Cut & Paste Whitespace)	`M-^`	(**delete-indentation** &optional ARG)	Join this line to previous and fix up whitespace at join. • If there is a fill prefix, delete it from the beginning of this line. • With argument, join this line to following line.

Indent relative to line above	`<f11> <tab> r`	(**indent-relative** &optional FIRST-ONLY UNINDENTED-OK)	Space out to under next indent point in previous nonblank line. An indent point is a non-whitespace character following whitespace.
			
			`The following line shows the indentation points in this line.`
			` ^      ^  ^  ^  ^     ^      ^   ^  ^  ^`
			• If FIRST-ONLY is non-nil (ie. using `C-u` prefix) then only the first indent point is considered.
			• If the previous nonblank line has no indent points beyond the column point starts at, then 'tab-to-tab-stop' is done, if both FIRST-ONLY and UNINDENTED-OK are nil, otherwise nothing is done.
			• If there isn't a previous nonblank line and UNINDENTED-OK is nil, call 'tab-to-tab-stop'.
			
			Essentially, this command inserts whitespace at point, until point is aligned with the first non-whitespace character on the previous line (actually, the last non-blank line). If point is already farther right than that, run tab-to-tab-stop instead—unless called with a numeric argument, in which case do nothing.

Indenting and un-indenting rigidly		The following commands provide non-semantic indentation of the current line or marked region. • The first command allows you to use further keystrokes to fine-tune the indentation back and forth using cursor keys. That's probably all you ever need to use. • Currently, PEL also provides the last 2 commands that indent or un-indent the current line or marked region. Once used, the region remains marked to allow further use of the command.	

Indent/Unindent rigidly	• `C-x <tab>`
	• `<f11> <tab> <tab>`
		(**pel-indent-rigidly &optional N**)
		
		——————————
		✂ PEL uses the above instead of the standard:
		
		(**indent-rigidly** START END ARG &optional INTERACTIVE)
			Indent rigidly the marked region or current line N times.
			• **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):
			  • 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.
			
			✂ PEL rebinds this key, but it extends the functionality: pel-indent-rigidly uses indent-rigidly, described below the dashed line.
			———————————————————————
			Indent all lines starting in the region.
			• 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>`.
			———————————————————————
			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 *indent-rigidly-map* with by default provides:
			  • `S-<right>` indent-rigidly-right-to-tab-stop
			  • `S-<left>` indent-rigidly-left-to-tab-stop
			  • `<right>` indent-rigidly-right
			  • `<left>` indent-rigidly-left
			Typing any other key deactivates the transient mode.
			
			⚠ Since cua-mode uses `C-x`, to invoke this command when cua-mode is active, type it really fast or type `C-x C-x <tab>` (or use the PEL binding `<f11> <tab> <tab>`).

Indent rigidly C-mode style	• `<f6> <tab>` • `<f11> <tab> c`	(**pel-insert-c-indent** &optional N)	Insert as many spaces as identified by **c-basic-offset** variable on the current line or all marked lines. • If a region was marked before the command it remains marked, allow further use of the same or other command to control the region. Use `C-g` to de-activate the region.
Un-indent rigidly C-mode style	• `<backtab>` • `<f6> <backtab>` • `<f11> <tab> C`	(**pel-unindent** &optional N)	Un-indent current line or marked lines by N times **c-basic-offset** spaces. • Works for point is anywhere on the line. • If a region was marked before the command it remains marked, allow further use of the same or other command to control the region. Use `C-g` to de-activate the region. • 🚧 Limitation: does not handle hard tabs properly.
Controlling use of hard tabs or spaces for indentation		The use of hard tabs or spaces for indentation is controlled by the Emacs (customizable) variable **indent-tabs-mode**. Like several Emacs variable this variable has global impact, but this can be overridden by directory local value, file local value and buffer local value allowing fine control over set of files and buffers. PEL provides the following related commands.	
Toggle use of hard tabs and only spaces for indentation in the current buffer (See also: ∑ Whitespace)	`<f11> t w I`	(**pel-toggle-indent-tabs-mode** &optional ARG)	Toggle use of hard tabs or spaces for indentation in current buffer. • Beep on each change to warn user of the change and display new value. • If ARG is positive set to use hard tabs, otherwise force use of spaces only.
Replacing Tabs with spaces or spaces with tabs		The following two commands can be used to replace hard tabs in a file with the corresponding number of space characters while retaining the same indentation and vice-versa.	
Replace tabs with spaces in a region (See also: ∑ Whitespace)	`<f11> t w SPC`	(**untabify** START END &optional ARG)	Convert all tabs in region to multiple spaces, preserving columns. • If called interactively with prefix ARG, convert for the entire buffer. • First select a region (Use C-x h for selecting the whole file). Then use the *untabify* function to replace all tabs by spaces in that region.
Replace multiple spaces with tabs in a region (See also: ∑ Whitespace)	`<f11> t w <tab>`	(**tabify** START END &optional ARG)	Convert multiple spaces in region to tabs when possible. • A group of spaces is partially replaced by tabs when this can be done without changing the column they end at. • If called interactively with prefix ARG, convert for the entire buffer.

Indentation — References

Title & URL	Description
Understanding GNU Emacs and Tabs	Overview description of how Emacs handle the Tab key, often used for strict indentation in many editors. In Emacs it can do much more.
GNU Emacs Manual - Indentation	
GNU Emacs Manual - Indentation for Programs	
Indentation Styles	
Indentation Styles @ Wikipedia	
StackOverflow - Emacs BSD/Allman Style with 4 Space Tabs?	
GNU Emacs Manual - Styles	
Emacs BSD/Allman Style with 4 Space Tabs?	
Emacs: Linux Kernel Style but with Allman/BSD Style Braces?	
Emacs Wiki - Indenting C	
Indent preprocessor directives as C code in emacs	Does not fully address the way I want to have multi-indentations for pre-processor
elisp code - ppindent.el	Implements pre-processor indentation with the # always in the first column. Not yet exactly what I want.
Demystify C++ Metaprograms using Emacs	
Programming in C++, Rules and Recommendations	ellemtel style