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></tab> | (quoted-insert ARG) <tab></tab> | Inserts a hard tab inside the file but moves the cursor to the column that represents the next multiple of <i>tab-width</i> . |
| Behaviour of Tab | In Emacs the behaviour of | the <tab> key depends on the major</tab> | mode of the current buffer. This key is rebound by several major mode. |
| | | • | d tabs. However, if there is text in the above lines, the tab moves to the spot under ce), then the tab stops controlled by the ruler take effect again. |
| Indent current line (or region) | <tab></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 is 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 | 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 |
| (CLBC s3.lisp) Insert spaces or tabs to next defined tab-stop column | M-i | (tab-to-tab-stop) | encountered. 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 rundamental mode: every 4 makefile, *.mk: BSDmakefile mode: shows every 4, inserts tabs. <tab> inserts tabs and moves cursor.</tab> |
| 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 <i>tab-width</i> 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</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.</tab> |

| Description | Keystroke | Function | Note |
|--|---------------------------|--|---|
| Indent/Unindent rigidly | C-x <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-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-left-to-tab-stop S-<left> indent-rigidly-left-to-tab-stop 'S-<left> indent-rigidly-left Typing any other key deactivates the transient mode.</left></left></right></s-right></s-left></right></left> |
| Burlanda kalendirik | M-x untabify | (untabify START END &optional | ⚠ 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>. First select a region (Use C-x h for selecting the whole file).</tab> |
| Replace tabs with spaces in a region | M-x uncabily | ARG) | Then use the <i>untabify</i> function to replace all tabs by spaces in that region. |
| | M-x tabify | | |
| 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) | |
| 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 | С-М-о | (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. |
| Delete Indentation, join this line to the previous one | M-^ | (delete-indentation &optional ARG) | |
| Indent rigidly C-mode style | F6 <tab></tab> | (pel-insert-c-indent &optional N) | Insert as many spaces as identified by c-basic-offset variable. TODO: add ability to detect region and apply command to all lines of the marked region. |
| Indent relative to line above | <f11> <tab> r</tab></f11> | (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 |

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 |