Indenting & Tab

		indenting &	lab
<u>Description</u>	<u>Keystroke</u>	Function	<u>Note</u>
Indentation under Emacs Indentation with PEL Behaviour of tab key Text alignment on newline Vertical alignment Indent region	 Emacs controls indentation according to various rules controlled by the buffer major mode. Furthermore the behaviour of the tab key is also controlled by the major mode; it may have surprising behaviour for people learning Emacs. The standard behaviour may be modified by the use of major and minor modes. Several major modes implement special indentation schemes, such as Lisp where indentation is inferred by the code itself as opposed to Pythor that uses indentation for defining scopes. Several major modes identify a variable that sets the indentation level. Refer to the information on the programming language major mode. Some programming languages (such as Go) impose hard-tab for indentation, using tab for indentation and space for alignment (works very nic Most languages never identified any rule, which led in some case to all sorts of conventions: use of both tabs and spaces, spaces only, with var number of positions for the indentation level. 		
Delete indentation Indenting /un-indenting rigidly Controlling use of tabs and spaces for indentation Replacing tab with spaces or vice-versa Indent-tools Smart-shift	Emacs can support anything Emacs controls the display of the go-mode, for exampentified the entirely controlled by hard. The indentation width is often PEL supports various indentation implement other behaviours, supports the indent-tools externated by The indent-tools externated by The indent-shift externated by The indent-sh	g. It can tabify or untabify source coderendering of hard tabs by the tab-widele, will move the first non-whitespace of tabs. It does not change the content of independent from the tab width but mechanisms and also provides some porting various major modes. This includes the provided of the provided o	character location inside the buffer as you modify the tab-width as indentation is of the file, just the way the file looks on the screen. not always. Again it depends on the major mode used. of its own extensions. It also provides easy access to external packages that
Open this PDF file. See also: <u>∑ Help/Info</u>	<f11> <tab> <f1></f1></tab></f11>	(pel-help-pdf &optional OPEN- WEB-PAGE)	Open the <u>National local PDF.</u> If the prefix argument (like C-u or M) is used, then it opens the remote GitHub hosted raw PDF instead. If the pel-flip-help-pdf-arg user-option is set it's the other way around.
<u>∑ Customize</u> PEL highlighting control	<f11> <tab> <f2></f2></tab></f11>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL support for indentation management • If OTHER-WINDOW is non-nil (use C-u), display in other window.
<u>> Customize</u> Emacs indentation control	<f11> <tab> <f3></f3></tab></f11>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs indentation control groups: indent, indent-tools, smart-shift. • If OTHER-WINDOW is non-nil (use C - u), display in another window.
Indentation with PEL	For several major modes, the value of the tab-width variable is controlled by a mode specific user options variable, like pel-c-tab-width for buffers in c-mode. In those buffers the value of tab-width will be set by PEL to the mode specific value when the buffer is opened. See with <f11> <tab>?</tab></f11>		
Show Indentation settings	<f11> <tab> ?</tab></f11>	(pel-show-indent &optional APPEND)	Print info about indentation control in a *pel-indent-info* help-mode buffer. Buffer-specific values of relevant user-options as buttons to use to get more info and change their customized values. Includes major-mode specific ones. Clear previous buffer content. Use prefix arg (like C-u) to append instead.
Set visual rendering of hard tabs for the current	<f11> M-t</f11>	(pel-set-tab-width N)	Change the tab width of the current buffer, only affecting the display rendering of hard tabs inserted in the buffer text. Prompts for a new value in the [2, 8] range.
buffer	PEL provides a specialized use	he ['] pel-go-tab-width ' user-option var	The change is temporary and affects the current buffer only. p-width for several major modes. For example, to change the tab width used for all riable instead.
Show state of text modes whether hard tabs are used for indentation, tabwidth electric-quote-mode delete-selection-mode enriched-mode overwrite-mode case folding subword, superword, glass modes visible-mode, smartdash-mode paragraph definition	inserted for indentation: • When indent-tabs-mode is act tabs and spaces. • The number of hard tabs ins characters required for indertination of the characters. • FEL provides user-options	(pel-show-text-modes) If ul to see if hard-tabs or spaces are stive, Emacs inserts a number of hard stead depends on the amount of thation and the tab-width. If up the inserts only space of the form pel- <mode>-use-tabs adent-tabs-mode for a buffer in the</mode>	Display the state of the various text modes in the mini buffer. -UU-:F1 keymaps.c Top (1,0) Git-master (C/*la WK Anzu Modes status: - Local indent-tabs-mode : off: use spaces. Tab width = 8 - Local electric-quote-mode: off. - delete-selection-mode : off. - enriched-mode : not loaded. - overwrite mode : off. - case-fold-search: on, sort-fold-case: not loaded. - subword mode: off, superword mode: on, glass-mode: not loaded. - visible-mode: off, smart-dash-mode: on. Sentences end with 2 space characters. - paragraph-start : "[]*\\('/+**\\)[]*\$\\\^^\L" - paragraph-separate: "[]*\\('/+**\\)[]*\$\\\^^\L"
Insert Literal Tab	C-q <tab></tab>	(quoted-insert ARG) <tab></tab>	Inserts a hard tab inside buffer. Render text according to value of tab-width.
Behaviour of Tab Key	By default, in text modes, tabs	are set to 8 spaces, inserting hard tak	of the current buffer. This key is rebound by several major mode. by. However, if there is text in the above lines, the tab moves to the spot under 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.
(0)	 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.		
	<tab></tab>	indent-for-tab-command &optional ARG)	In Lisp related modes. indent-line-function = indent-relative. • tab-always-indent = t
	Several major modes adjust the	ne behaviour of the tab key to perform	semantically aware indentation, such as what is being done in Lisp.
	<tab></tab>	(c-indent-line-or-region &optional ARG REGION)	In C related modes: Indent active region, current line, or block starting on the line. In Transient Mark mode, when the region is active, reindent the region. With prefix argument, rigidly reindent the expression starting on current line. Otherwise reindent just the current line.
Indent lines of list after point Example: <u>CLBC s3.lisp</u>	С-М-Ф	(indent-sexp & optional ENDPOS) (c-indent-exp & optional SHUTUP-P)	Indent each line of the list starting just after point. • The command used depends on the major mode of the current buffer.
Insert spaces or tabs to next defined tab-stop column $\underbrace{\mathfrak{P}\mathfrak{l}-C}_{},\underbrace{\mathfrak{P}\mathfrak{l}-C++}_{},\underbrace{\mathfrak{P}\mathfrak{l}-D}_{}$	 M-i (tab-to-tab-stop) Insert spaces or tabs to next defined tab-stop column. The exact location of the next tab stop is identified by the value of the tab-stop-list and tab-width for the current buffer. With PEL, for several major modes, the value of the tab-width variable is controlled by a mode specific user options variable, like pel-c-tab-width for buffers in c-mode. In those buffers the value of tab-width is set by PEL to the mode specific value when opening the buffer. See with <f11> <tab></tab> tab>?</f11> 		
Insert an indented line below current line See also: ∑ Align	• M-RET • <f11> <tab> RET</tab></f11>	(pel-newline-and-indent-below)	Insert an indented line just below current line. • The command can also align text vertically if this special mode was activated for the buffer with the <f11> M-RET. • To see the current behaviour use <f11> t a ?: it displays whether the M-</f11></f11>
			RET command aligns text or not.

<u>Description</u>	<u>Keystroke</u>	Function	<u>Note</u>
Toggle text alignment on	<f11> M-RET</f11>	(pel-toggle-newline-indent-align)	Toggle variable <i>pel-newline-does-align</i> for the local buffer.
pel-newline-and-indent- below See also: <u>Newligh</u>	This toggles the way function 'pel-newline-and-indent-below' operates. • If pel-newline-does-align is t, it aligns several syntactic element in the current block: the comments, the assignments. • If pel-newline-does-align is t, it aligns several syntactic element in the current block: the comments, the assignments. • If pel-newline-does-align is automatically activated (set to t) by adding the major mode to the list in the pel-modes-activating-align-on-return user option. • This affects the behaviour of the following commands:		
	 pel-cc-newline (assigned to RET in CC modes like c-mode, c++-mode and d-mode). pel-newline-and-indent-below (assigned the M-RET) 		
Display current buffer's vertical alignment behaviour See also: ∑Indentation	<f11> t a ?</f11>	(pel-align-info &optional APPEND)	Print information about text aligning info in a *pel-align-info* help-mode buffer. Prints current state and values of relevant user-options as buttons you can use to get more info and change their customized values: behaviour of M-RET in the current buffer: show if that command aligns text or not. whether the modes are activating vertical alignment on return. Clear previous buffer content. Use prefix arg (like C-u) to append instead.
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)	globally with (setq-default tab-width above two). Note that any literal tab in the buffichanging the tab-width, the layout new tab-width value.	ults to 8 in emacs. It can be set locally inside a buffer with (setq tab-width N) or N). The M-: keystroke allows evaluating a lisp expression interactively (as the er impact the location of the column where the next character shows. When shown in the window that contains literal tabs will be modified according to the local value of tab-width is controlled by PEL and set to the value of a mode specific
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>
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	м-т	(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 line to the previous one See also: ∑ Cut & Paste ∑ Whitespace	• M-^ • <f11></f11>	(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 The following line shows the</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.
	If the previous nonblank line ha UNINDENTED-OK are nil, other If there isn't a previous nonblar	rwise nothing is done. nk line and UNINDENTED-OK is nil, ca	point starts at, then 'tab-to-tab-stop' is done, if both FIRST-ONLY and II '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 (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 can be a superior of the previous line (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 can be a superior of the previous line (non-blank line).		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 See also: <u>∑ Key-Chords</u>	• C-x <tab> • <f11> <tab> <tab> • <tab>q</tab></tab></tab></f11></tab>	(pel-indent-rigidly &optional N)	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. With PEL, the <tab>q key-chord is available when pel-use-key-chord is non-nil. See ★ Key-Chords.</tab>
See also: • <u>\$\mathbf{Y}\tilde{\text{\colored}}\text{\colored}}\$ • <u>\$\mathbf{Y}\tilde{\text{\colored}}\text{\colored}\$ • <u>\$\mathbf{Y}\tilde{\text{\colored}}\text{\colored}\$ • <u>\mathbf{Y}\tilde{\text{\colored}}\text{\colored}\$</u></u></u></u>		➤ PEL uses the above instead of the standard: (indent-rigidly START END ARG	 Scommand numeric prefix is available with the key-chord binding. ⇒ 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 <1eft>.
	The capabilities are controlled by • S- <right> indent-rigit • S-<left> indent-rigit • <right> indent-rigit • <left> indent-rigit Typing any other key deactivates • The S-<right> and S-<left a<="" for="" invalue="" is="" major="" of="" pel,="" pel-c-tab-width="" several="" td="" with="" •=""><td>&optional INTERACTIVE) a a transient mode where Emacs prome the variable indent-rigidly-map with be dily-right-to-tab-stop dily-right dily-left the transient mode. Et> keys indent/de-indent to the next modes, the indentation is controlled by automatically stored into tab-width who cua-mode uses C-x, to invoke this co</td><td><pre><right>, <s-left>, or <s-right>. pts for extra keys to control how to indent. Indenting and un-indenting is possible. y default provides: t tab-stop position, which is controlled by the tab-width user option. y a mode-specific user option variable . For example, for buffers in c-mode, the</s-right></s-left></right></pre></td></left></right></left></right></left></right>	&optional INTERACTIVE) a a transient mode where Emacs prome the variable indent-rigidly-map with be dily-right-to-tab-stop dily-right dily-left the transient mode. Et> keys indent/de-indent to the next modes, the indentation is controlled by automatically stored into tab-width who cua-mode uses C-x, to invoke this co	<pre><right>, <s-left>, or <s-right>. pts for extra keys to control how to indent. Indenting and un-indenting is possible. y default provides: t tab-stop position, which is controlled by the tab-width user option. y a mode-specific user option variable . For example, for buffers in c-mode, the</s-right></s-left></right></pre>

<u>Description</u>	<u>Keystroke</u>	Function	<u>Note</u>
Indent line(s) rigidly	• <f6> <tab> • <f11> <tab> c</tab></f11></tab></f6>	(pel-indent-lines &optional N)	Indent current or marked lines by N indentation levels
	Works with point anywhere on All lines touched by the region A special argument N can spec If a negative number is specifie If a region is marked, the funct characters in all affected lines. Use C-g to de-activate the reg Handles presence of hard tabs If indent-tabs-mode is non-region.	are indented. cify more than one indentation level. It d, 'pel-unindent-lines' is used. ion does not deactivate it to allow report gion. the indentation is created with a mix ny hard tab in the indentation of the m	eated execution of the command. It also modifies the region to include all
Un-indent line(s) rigidly	<pre>• <backtab> • <f6> <backtab> • <f11> <tab> C</tab></f11></backtab></f6></backtab></pre>	(pel-unindent-lines &optional N)	Un-indent current line or marked lines by N indentation levels.
	If a region was marked, the funcharacters in all affected lines Use C-g to de-activate the reg Handles presence of hard tabs If indent-tabs-mode is non-regions.	are un-indented. tion does not deactivate it to allow rep action does not deactivate it to allow re gion. : iii the indentation is created with a mix by hard tab in the indentation of the m	epeated execution of the command. It also modifies the region to include all
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 <u>global impact</u> , but this can be overridden by <u>directory local</u> value, <u>file local</u> value and <u>buffer local</u> value allowing fine control over set of files and buffers. PEL provides the following related commands. See also: <u>Whitespace</u>		
Toggle use of hard tabs and only spaces for indentation in the current buffer	<f11> t w I</f11>	(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</f11>	(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></tab></f11>	(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.
Indent-tools	The indent-tools external package provides several commands to indent, un-indent and navigate across indented text levels. It provides a minor mode and a key hydra that provides all of these commands. The indent-tools external package PEL activates it when the pel-use-indent-tools user-option is turned on (set to t). This also automatically activates the hydra external package. PEL provide a global key binding to its key hydra and provides the ability to activate the proposed key binding globally and for python mode: pel-indent-tools-key-bound: activates the C-c > key binding either globally or for python-mode only.		
Open the indent-tools	<f11> <tab> <f7></f7></tab></f11>	(indent-tools-hydra/body)	Activate the "indent-tools-hydra" hydra.
Nydra See also: 191 - Python	C-c >		with PEL, this key binding is only available when: globally, when pel-indent-tools-key-bound is set to globally, in python-mode only when pel-indent-tools-key-bound is set to python. The actual key is selected by indent-tools indent-tools-keymap-prefix useroption, the default is C−c >
See also: » Hide/Show	The heads for the associated hydra are: >: 'indent-tools-indent', <: 'indent-tools-demote', E: 'indent-tools-comment', U: 'indent-tools-comment', P: 'indent-tools-indent-paragraph', 1: 'indent-tools-indent-end-of-level', K: 'indent-tools-kill-tree', C: 'indent-tools-copy-hydra/body', s: 'indent-tools-select', e: 'indent-tools-goto-end-of-tree', u: 'indent-tools-goto-parent', d: 'indent-tools-goto-child', S: 'indent-tools-goto-next-sibling', p: 'indent-tools-goto-parent', i: 'helm-imenu', j: 'forward-line', k: 'previous-line', SPC: 'indent-tools-indent-space', _: 'undo-tree-undo', L: 'recenter-top-bottom', f: 'yafolding-toggle-element', q: exit		-UUU:F1 somedata.yml All (1,0) (YAML WK Fly Anzu) Indent Navigation Actions

Smart-shift	The <u>smart-shift</u> external package simplifies shifting a complete line or region of lines right or left but also up or down. • It is implemented as a minor or global minor mode that must be enabled first.		
∑ Customize with: <f11> <tab> s <f2></f2></tab></f11>	 Automatically activate the smart-shift-mode in specified major mode by customizing the pel-<mode>-activates-minor-modes user-options.</mode> You can also use the commands manually or through the key bindings provided by PEL to activate the smart-shift-mode in the current buffer or globally for all buffers. PEL controls it through customization user-options: The <u>smart-shift</u> external package		
Toggle smart-shift mode in current buffer	<f11> <tab> s</tab></f11>	(smart-shift-mode &optional ARG)	Activate/de-activate the smart-shift mode in the current buffer. Activate the line-shift key bindings listed below, in the current buffer. With PEL, the actual key binding selected for the line shift commands depend on the value of the pel-smart-shift-keybinding user-option.
Toggle smart-shift mode globally	<f11> <tab> S</tab></f11>	(global-smart-shift-mode &optional ARG)	Toggle Smart-Shift mode in all buffers. With prefix ARG, enable Global Smart-Shift mode if ARG is positive; otherwise, disable it. Smart-Shift mode is enabled in all buffers where 'smart-shift-mode-on' would do it.
When smart-shift mode is active:	As described above, with PEL only one of the extra key bindings provided by PEL can be enabled via the pel-smart-shift-keybinding user-option. So unlike other key binding description cells in this and other tables, only one of the last 2 key bindings is available in the smart-shift minor mode.		
Shift line or region right	• C-c <right> • C-c <c-right> • <f9> <right></right></f9></c-right></right>	(smart-shift-right &optional ARG)	Shift the line or region to the ARG times to the right.
Shift line or region left	• C-c <left> • C-c C-<left> • <f9> <left></left></f9></left></left>	(smart-shift-left &optional ARG)	Shift the line or region to the ARG times to the left.
Shift line or region up	• C-c <up> • C-c C-<up> • <f9> <up></up></f9></up></up>	(smart-shift-up &optional ARG)	Shift the line or region to the ARG times to the upwards.
Shift line or region down	• C-c <down> • C-c C-<down> • <f9> <down></down></f9></down></down>	(smart-shift-down &optional ARG)	Shift the line or region to the ARG times to the downwards

Function

Note

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 Basic Concepts Tutorial @ XEmacs	A tutorial on indentation written by KaiGrossjohann

Tabs or space for indentation??

Description

Keystroke

There are several views on the use of hard-tab and space characters for indenting source code. They are:

- Use only hard-tab for indentation. Uncontrolled use of tabs or spaces for alignment.
 Use only space characters for indentation. Popular in C like languages. Also popular in Python.
- 3. Use hard tabs for indentation, and space character for alignment.
- Method 1 was popular originally since it reduces file size when hard tab size was always the same. But soon it became possible to identify a different number of character positions to render a hard tab. And then it became impossible to guarantee the rendering of code indentation and alignment when the number of hard-tabs did not match the indentation level of a line of source code.
- A reaction to this problem is to use Method 2 where hard-tabs are banned. The rendering is therefore always the same no matter what the *size* of a hard tab is since you don't use any. This however increases the size of files. Not a problem for storage today you'd say, but perhaps a problem for data transfer and/or power consumption. Method 3 is used by some programming environments. The Go programming language imposes the use of hard-tabs for indentation. And if you want to align text at the right of the
- indentation level, you use spaces.

 To use this method in other programming languages, you can use the smart-tabs-mode explained in the **Smart-Tabs Emacs Wiki page**.

Emacs support all modes. It has 2 different buffer local variables that are important and control the rendering of hard-tabs and the indentation:

- tab-width: How many columns a hard-tab occupies, the distance between tab-stops. indentation offset variable: a variable for each major mode, like **c-basic-offset** for CC modes (C, C++, Java, etc...), that identifies the number of columns per indentation level.

PEL does not yet integrate the smarttabs package. 25 For CC modes it provides PEL user-options that control the indentation using method 2.

Using method 3 requires a better understanding from all developers working on the source code with all their editors being able to handle the mix of hard tab and space characters

Smarttabs @ GitHub	Starttabs source code repository.
Indentation Styles for Curly Bracket Languages	
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