




















Indenting & Tab

Description	Keystroke	Function	Note
<div><div>Indentation under Emacs</div><div><ul style="list-style-type: none">Use hard tabs or spaces for indentationSet hard-tab visual widthAdjust tab stopShow tab/indent settingsAlign on returnInsert hard-tabTo next tab stoptabify & untabifyBehaviour of tab keyInsert newline, split lineIndent regionDelete indentationMove to 1st non-blankIndenting /un-indenting rigidlyText alignment on newlineIndent-toolsIndent-barsSmart-shiftSmart-tabs</div></div>	<div>Emacs controls indentation and behaviour of the tab key according to various rules controlled by the buffer's major mode.</div> <div><ul style="list-style-type: none">The modes default behaviour may be modified by the use of minor modes.<ul style="list-style-type: none">Several major modes identify one or several variable that control indentation levels and behaviour. Some use default indentation variables.Some programming languages (such as Go) impose hard-tab for indentation, using tab for indentation and space for alignment. Works very nicely.There are several indentation schemes: hard-tab only for indentation, spaces only for indentation, hard-tabs for indentation space for alignment, free mix of hard-tabs and spaces for indentation and whitespace.Emacs can support all those schemes. It can tabify or untabify source code.Emacs controls the <i>display rendering</i> of hard tabs via the tab-width customizable user-option variable.The indentation width is often independent from the tab width but not always. Again it depends on the major mode used.</div> <div>PEL supports various indentation mechanisms and also provides some of its own extensions. It also provides easy access to external packages that implement other behaviours and features.</div> <div><ul style="list-style-type: none">The packages are activated by turning on the corresponding pel-use customizable user-option)setting it to t):<div><div><div> The indent-tools external package</div><div> pel-use-indent-tools</div><div>Provides commands to alter indentation of code.</div></div><div><div> The indent-bars external package</div><div> pel-use-indent-bars</div><div>Provides ability to show indent bar at each indentation level.</div></div><div><div> The smart-shift external package</div><div> pel-use-smart-shift</div><div>Provides ability to shift text areas left/right by indentation levels.</div></div><div><div> The smart-tabs external package</div><div> pel-use-smart-tabs</div><div>Use it when using tabs for indentation and spaces for alignment.</div></div></div></div> <div>Information related to indentation is described in the pages related to programming major modes. The information in this page is generic and complements the mode specific information.</div>		
Last updated on:	2025-10-31	 See Showing Information About Indentation and Hard Tab Control under PEL	
<div>Open this PDF file.</div> <div>See also: 🔗 Help/Info</div>	<f11> <tab> <f1>	(pel-help-pdf & optional OPEN-WEB-PAGE)	Open the 🔗 Indentation local PDF. 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.
🔗 Customize PEL highlighting control	<f11> <tab> <f2>	(pel-customize-pel & optional OTHER-WINDOW)	Customize PEL support for indentation management <ul style="list-style-type: none">If OTHER-WINDOW is non-nil (use C-u), display in other window.
🔗 Customize Emacs indentation control	<f11> <tab> <f3>	(pel-customize-library & optional OTHER-WINDOW)	Customize Emacs indentation control groups: indent, Indent-bars indent-tools , smart-shift . <ul style="list-style-type: none">If OTHER-WINDOW is non-nil (use C-u), display in another window.
Use 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 . <ul style="list-style-type: none">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 See also: 🔗 Whitespace		
Toggle use of hard tabs for indentation in the current buffer	<ul style="list-style-type: none"><f11> <tab> m<f11> t w I	(pel-toggle-indent-tabs-mode & optional ARG)	Toggle whether indentation can insert hard tab characters in the current buffer. <ul style="list-style-type: none">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.This uses Emacs indent-tabs-mode function and provides more feedback.
Hard-tab “width”	The current-buffer value of tab-width affects the <i>visual rendering</i> of the hard-tab character. It does not affect the content of the buffer or file.		
<div>Set visual rendering of hard tabs for the current buffer</div> <div><ul style="list-style-type: none">Does not change buffer/file content</div>	<f11> <tab> w	(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. <ul style="list-style-type: none">This modifies a buffer local value of the the tab-width user-option. The change is temporary and affects the current buffer only.PEL provides a specialized user-option to set the default value of tab-width for several major modes. For example, to change the tab width used for all Go source code files, change the 'pel-go-tab-width' user-option variable instead. See the documentation of each major mode for more information.
Tab stops	Emacs keeps track of a set of “ <i>tab-stop</i> ” columns that can be used as reference points to align text. Something similar to typewriter tab-stop .		
<div>Change the tab stops</div> <div><ul style="list-style-type: none">used by M-i</div>	<f11> <tab> e	(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. <ul style="list-style-type: none">The tab stop take effect at the top of the buffer, as used by M-i
<div>Show Indentation settings</div> <div>For several major modes, a PEL mode-specific user-option controls the the value of the tab-width variable for the mode. For example, pel-c-tab-width is used for c-mode buffers.</div> <div>Note however, that indentation for C buffer is controlled by the c-basic-offset user-option.</div> <div>Other major modes may provide mode specific control variables and the printed information will differ.</div>	<ul style="list-style-type: none"><f11> <tab> ?<f11> ? <tab> <div>The command opens a buffer showing the indentation control for the current buffer.<ul style="list-style-type: none">The first line shows the buffer name and date.The first group of user options are buffer specific and take precedence over the global user-options.The next group are global user-options that are used when they are not overridden by major-mode specific ones.<ul style="list-style-type: none">inside c-mode buffers, PEL has user-variables that take over.All user-option variables are help buttons; click on them (in graphics mode) or tab and return to open help on the</div>	(pel-show-indent & optional APPEND)	<div>Print info about indentation control in a *pel-indent-info* help-mode buffer.</div> <div><ul style="list-style-type: none">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.</div> <div><pre> ----Indentation Control from alloc.c --- Wednesday, April 30, 2025 @ 08:21:59 ---- - pel-c-indent-width : 4 - pel-c-tab-width : 4 - pel-c-use-tabs : nil ----- The above major-mode specific user options take precedence over the following global ones (unless they are set by file variables): - c-basic-offset : 4 - tab-width : 4 -> Use pel-set-tab-width to change locally and have tabs rendered with a different width. - indent-tabs-mode : nil - standard-indent : 4 - tab-always-indent : t - tab-stop-list : nil -UUU:%*- F1 *pel-indent-info* All (1,0) (Help WK Anzu) 08:22 2.34 ----- f11 TAB ? </pre><div>These are help buttons! Click on them in graphic mode or text with mouse enabled. In text-mode without mouse support, <tab> to navigate and <ret> to open help.</div></div>
<div>Toggle text alignment on pel-newline-and-indent-below</div> <div>See also: 🔗 Align</div>	<f11> M-RET	(pel-toggle-newline-indent-align)	Toggle variable <i>pel-newline-does-align</i> for the local buffer.
	It affects the way function ' pel-newline-and-indent-below ' operates. <ul style="list-style-type: none">If <i>pel-newline-does-align</i> is t, it aligns several syntactic element in the current block: the comments, the assignments. set modes that automatically activates <i>pel-newline-does-align</i> by adding the major mode to 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)See the list with <f11> t a ?		
<div>Show state of text modes</div> <div><ul style="list-style-type: none">whether hard tabs are used for indentation, tab-widthwhether newline aligns textelectric-quote-modedelete-selection-modeenriched-modeoverwrite-modecase foldingsubword, superword, glass modesvisible-mode, smart-dash-modeparagraph definition</div>	<f11> t m ?	(pel-show-text-modes)	Display the state of the various text modes in the mini buffer.
Output example ➡	 Quickly show several settings inside the mode line: the tab settings, text alignment on newline, whitespace mode, etc... <ul style="list-style-type: none">When indent-tabs-mode is active, Emacs inserts a number of hard tabs and spaces.<ul style="list-style-type: none">The number of hard tabs instead depends on the amount of characters required for indentation and the tab-width.If indent-tabs-mode is not active, then Emacs inserts only space characters. PEL provides user-options of the form pel-<mode>-use-tabs which is used to initialize the indent-tabs-mode supported major modes buffers. <div>Text Modes Status:<ul style="list-style-type: none">- Local indent-tabs-mode : off: use spaces, Tab width = 8- Local newline does align : off . Automatically activated by modes (<f11> t a <f2>): (c-mode c++-mode sh-mode)- Local electric-quote-mode: off , electric-quote-local-mode: not loaded.- whitespace-mode : not loaded, show-trailing-whitespace : off , indicate-empty-lines: off.- enriched-mode : not loaded.- overwrite mode : off , delete-selection-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 : not loaded.- Sentences end with 2 space characters.- paragraph-start : "^\[\t]*\$" ,- paragraph-separate: "[^\[\t]*\$" ,</div>		

Description	Keystroke	Function	Note																							
Indenting and un-indenting rigidly	The following commands provide non-semantic indentation of the current line or marked region. <ul style="list-style-type: none">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	<ul style="list-style-type: none">C-x <tab><f11> <tab> <tab><tab>q	(pel-indent-rigidly &optional N) <div>-----✂ PEL uses the above instead of the standard: (indent-rigidly START END ARG &optional INTERACTIVE)</div>	Enter a mode to indent rigidly the marked region or current line N times. <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 lineN > 1 : operate on the current line and N-1 lines below.N < -1 : operate on the current line and (abs N) -1 lines above. Once the command is issued use the keys listed below to indent or un-indent by indent-width step or by single column. <div>-----</div> Indent all lines starting in the region. <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>.																							
See also: 🔗 Key-Chords	Both of these commands activate a transient mode where Emacs prompts for extra keys to control how to indent or un-indent. The capabilities are controlled by the variable <i>indent-rigidly-map</i> with by default provides: <table><tr><td>S-<right></td><td>indent-rigidly-right-to-tab-stop</td><td>S-<left></td><td>indent-rigidly-left-to-tab-stop</td></tr><tr><td><right></td><td>indent-rigidly-right</td><td><left></td><td>indent-rigidly-left</td></tr></table> Typing any other key deactivates the transient mode. <ul style="list-style-type: none">The S-<right> and S-<left> keys indent/de-indent to the next tab-stop position, which is controlled by the tab-width user option.<ul style="list-style-type: none">With PEL, for several major modes, the indentation is controlled by a mode-specific user option variable . For example, for buffers in c-mode, the value of pel-c-tab-width is automatically stored into tab-width when the buffer is opened. ⚠ If you use the cua-mode: the 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>).			S-<right>	indent-rigidly-right-to-tab-stop	S-<left>	indent-rigidly-left-to-tab-stop	<right>	indent-rigidly-right	<left>	indent-rigidly-left															
S-<right>	indent-rigidly-right-to-tab-stop	S-<left>	indent-rigidly-left-to-tab-stop																							
<right>	indent-rigidly-right	<left>	indent-rigidly-left																							
See also: <ul style="list-style-type: none">🔗 I - C🔗 I - C++🔗 I - D🔗 M reStructuredText	📦🔗 With PEL, the <tab>q key-chord is available when pel-use-key-chord is non-nil. See 🔗 Key-Chords . 🗨 Command numeric prefix is available with the key-chord binding.																									
Indent line(s) rigidly	<ul style="list-style-type: none"><f6> <tab><f11> <tab> c	(pel-indent-lines &optional N)	Indent current or marked lines by N indentation levels																							
Un-indent line(s) rigidly	<ul style="list-style-type: none"><backtab><f6> <backtab><f11> <tab> C	(pel-unindent-lines &optional N)	<ul style="list-style-type: none">Un-indent current line or marked lines by N indentation levels.																							
	<ul style="list-style-type: none">Works with point anywhere on the line.All lines touched by the region are indented.A special argument N can specify more than one indentation level. It defaults to 1.If a negative number is specified, 'pel-unindent-lines' is used.If a region is marked, the function does not deactivate it to allow repeated execution of the command. It also modifies the region to include all characters in all affected lines.Use C-g to de-activate the region.Handles presence of hard tabs:<ul style="list-style-type: none">If indent-tabs-mode is non-nil the indentation is created with a mix of hard-tabs and space characters.If indent-tabs-mode is nil, any hard tab in the indentation of the marked lines is replaced by the proper number of spaces. Hard tabs after first non-whitespace character on the line are left.																									
Indent-tools	The indent-tools external package provides several commands to indent, un-indent and navigate across indented text levels. <ul style="list-style-type: none">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). <ul style="list-style-type: none">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: <ul style="list-style-type: none">pel-indent-tools-key-bound : activates the C-c > key binding either globally or for python-mode only.																									
Open the indent-tools hydra	<ul style="list-style-type: none"><f11> <tab> <f7><f7> <tab>C-c >	(indent-tools-hydra/body)	Activate the "indent-tools-hydra" hydra.																							
See also: 🔗 I - Python	🐍 With PEL, this key binding is only available when: <ul style="list-style-type: none">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 user-option, the default is C-c >																									
See also: 🔗 Hide/Show	The heads for the associated hydra are: <pre>>: 'indent-tools-indent', <: 'indent-tools-demote', E: 'indent-tools-indent-end-of-defun', c: 'indent-tools-comment', U: 'indent-tools-uncomment', P: 'indent-tools-indent-paragraph', l: '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-select-end-of-tree', n: 'indent-tools-goto-next-sibling', p: 'indent-tools-goto-previous-sibling', 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</pre>																									
	<table><tr><th>Indent</th><th>Navigation</th><th>Actions</th></tr><tr><td>> indent</td><td>j v</td><td>K kill</td></tr><tr><td>< de-indent</td><td>k A</td><td>i imenu</td></tr><tr><td>l end of level</td><td>n next sibling</td><td>C Copy...</td></tr><tr><td>E end of fn</td><td>p previous sibling</td><td>c comment</td></tr><tr><td>P paragraph</td><td>u up parent</td><td>U uncomment (paragraph)</td></tr><tr><td>SPC space</td><td>d down child</td><td>f fold</td></tr><tr><td>_ undo</td><td>e end of tree</td><td>q quit</td></tr></table> 🗨 The f key toggles the element folding. Press once to hide the sub-tree, press-again to display it back.			Indent	Navigation	Actions	> indent	j v	K kill	< de-indent	k A	i imenu	l end of level	n next sibling	C Copy...	E end of fn	p previous sibling	c comment	P paragraph	u up parent	U uncomment (paragraph)	SPC space	d down child	f fold	_ undo	e end of tree
Indent	Navigation	Actions																								
> indent	j v	K kill																								
< de-indent	k A	i imenu																								
l end of level	n next sibling	C Copy...																								
E end of fn	p previous sibling	c comment																								
P paragraph	u up parent	U uncomment (paragraph)																								
SPC space	d down child	f fold																								
_ undo	e end of tree	q quit																								

Description	Keystroke	Function	Note
indent-bars	<ul style="list-style-type: none"> A minor-mode that displays vertical bars over each indentation level. The style of the bars is customizable. <div>  The indent-bars external package  PEL activates it when the pel-use-indent-bars user-option is turned on (set to <code>t</code>). </div>		
Toggle indent-bars	<code><f11> <tab> b b</code>	<code>(indent-bars-mode &optional ARG)</code>	Toggle showing bars that indicate indentation. A minor mode.
Reset indent bar config	<code><f11> <tab> b r</code>	<code>(indent-bars-reset &rest R)</code>	Reset indent-bars config.
Reset style of indent bars	<code><f11> <tab> b s</code>	<code>(indent-bars-reset-styles &rest R)</code>	Reset all styles' colors and faces. <ul style="list-style-type: none"> Useful for calling after theme changes.
Smart-shift	<p>The smart-shift external package simplifies shifting a complete line or region of lines right or left but also up or down.</p> <ul style="list-style-type: none"> It is implemented as a minor or global minor mode that must be enabled first. <ul style="list-style-type: none"> Automatically activate the smart-shift-mode in specified major mode by customizing the <code>pel-<mode>-activates-minor-modes</code> user-options. 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: <div>  The smart-shift external package  PEL activates it when the pel-use-smart-shift user-option is turned on (set to <code>t</code>).</div> PEL also provides the <code>pel-smart-shift-keybinding</code> user-option that allows you to select whether the shift keys used by smart-shift mode is the default provided keys only or whether you also want to activate another set. <ul style="list-style-type: none"> The default are always available when smart-shift mode is active: <code>C-c <right></code> , <code>C-c <left></code> , <code>C-c <up></code> and <code>C-c <down></code>. PEL can also activate one of the following extra key binding sets: <ul style="list-style-type: none"> Using the control cursor key : <code>C-c C-<right></code> , <code>C-c C-<left></code> , <code>C-c C-<up></code> and <code>C-c C-<down></code>. Using the <code><f9></code> key as prefix: <code><f9> <right></code> , <code><f9> <left></code> , <code><f9> <up></code> and <code><f9> <down></code> 		
Customize with: <code><f11> <tab> s <f2></code>			
Toggle smart-shift mode in current buffer	<code><f11> <tab> s</code>	<code>(smart-shift-mode &optional ARG)</code>	Activate/de-activate the smart-shift mode in the current buffer. <ul style="list-style-type: none"> Activate the line-shift key bindings listed below, in the current buffer. <ul style="list-style-type: none"> With PEL, the actual key binding selected for the line shift commands depend on the value of the <code>pel-smart-shift-keybinding</code> user-option.
Toggle smart-shift mode globally	<code><f11> <tab> S</code>	<code>(global-smart-shift-mode &optional ARG)</code>	<ul style="list-style-type: none"> 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:	<div>  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. </div> 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	<ul style="list-style-type: none"> <code>C-c <right></code> <code>C-c C-<right></code> <code><f9> <right></code> 	<code>(smart-shift-right &optional ARG)</code>	Shift the line or region to the ARG times to the right.
Shift line or region left	<ul style="list-style-type: none"> <code>C-c <left></code> <code>C-c C-<left></code> <code><f9> <left></code> 	<code>(smart-shift-left &optional ARG)</code>	Shift the line or region to the ARG times to the left.
Shift line or region up	<ul style="list-style-type: none"> <code>C-c <up></code> <code>C-c C-<up></code> <code><f9> <up></code> 	<code>(smart-shift-up &optional ARG)</code>	Shift the line or region to the ARG times to the upwards.
Shift line or region down	<ul style="list-style-type: none"> <code>C-c <down></code> <code>C-c C-<down></code> <code><f9> <down></code> 	<code>(smart-shift-down &optional ARG)</code>	Shift the line or region to the ARG times to the downwards
smart-tabs	<div>  The smart-tabs external package  PEL activates it when the pel-use-smart-tabs user-option is turned on (set to <code>t</code>). </div>		
Toggle smart-tabs mode	<code><f11> <tab> M-s</code>	<code>(smart-tabs-mode &optional ARG)</code>	Toggle smart-tabs minor mode. <ul style="list-style-type: none"> Intelligently indent with tabs, align with spaces!

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?? There are several views on the use of hard-tab and space characters for indenting source code. They are: <ol style="list-style-type: none"> 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. Use hard tabs for indentation, and space character for alignment. <ul style="list-style-type: none"> 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 <i>size</i> 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. <ul style="list-style-type: none"> To use this method in other programming languages, you can use the smart-tabs-mode explained in the Smart-Tabs Emacs Wiki page. <p>Emacs support all modes. It has 2 different buffer local variables that are important and control the rendering of hard-tabs and the indentation:</p> <ul style="list-style-type: none"> 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. <p>PEL provides access to the smarttabs package but it's not yet fully configured for programming modes nor tested. 🐛 For CC modes it provides PEL user-options that control the indentation using method 2.</p> <p>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 correctly.</p>	
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?	

Title & URL	Description
<u>Emacs Wiki - Indenting C</u>	
<u>Indent preprocessor directives as C code in emacs</u>	Does not fully address the way I want to have multi-indentations for pre-processor
<u>elisp code - ppindent.el</u>	Implements pre-processor indentation with the # always in the first column. Not yet exactly what I want.
<u>Demystify C++ Metaprograms using Emacs</u>	
<u>Programming in C++, Rules and Recommendations</u>	ellemtel style