












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 See also: 🔗 Key-Chords	<ul style="list-style-type: none">C-x <tab><f11> <tab> <tab><tab>q	(pel-indent-rigidly &optional N) -----  PEL uses the above instead of the standard: (indent-rigidly START END ARG &optional INTERACTIVE)	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. ----- 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>.																									
 PEL rebinds this key, but extends the functionality: pel-indent-rigidly uses indent-rigidly, described below the dashed line. See also: <ul style="list-style-type: none">🔗 I - C🔗 I - C++🔗 I - D🔗 reStructuredText	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><ul style="list-style-type: none">S-<right> indent-rigidly-right-to-tab-stop<right> indent-rigidly-right</td><td>S-<left> indent-rigidly-left-to-tab-stop</td><td><left> 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>).   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.			<ul style="list-style-type: none">S-<right> indent-rigidly-right-to-tab-stop<right> indent-rigidly-right	S-<left> indent-rigidly-left-to-tab-stop	<left> indent-rigidly-left																						
<ul style="list-style-type: none">S-<right> indent-rigidly-right-to-tab-stop<right> indent-rigidly-right	S-<left> indent-rigidly-left-to-tab-stop	<left> indent-rigidly-left																										
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 un-indented.If region was marked, the function does not deactivate it to allow repeated execution of the command.If a region was 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 linesUse 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 See also: 🔗 I - Python	<ul style="list-style-type: none"><f11> <tab> <f7><f7> <tab>C-c >	(indent-tools-hydra/body)	Activate the "indent-tools-hydra" hydra.  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>	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
	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																										
			 The f key toggles the element folding. Press once to hide the sub-tree, press-again to display it back.																									

Description	Keystroke	Function	Note
Smart-shift ⓘ Customize with: <f11> <tab> s <f2>	The smart-shift external package simplifies shifting a complete line or region of lines right or left but also up or down. <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 pel-<mode>-activates-minor-modes 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: <ul style="list-style-type: none"> The smart-shift external package PEL activates it when the pel-use-smart-shift user-option is turned on (set to t). PEL also provides the pel-smart-shift-keybinding 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. The default are always available when smart-shift mode is active: C-c <right>, C-c <left>, C-c <up> and C-c <down>. PEL can also activate one of the following extra key binding sets: <ul style="list-style-type: none"> Using the control cursor key : C-c C-<right>, C-c C-<left>, C-c C-<up> and C-c C-<down>. Using the <f9> key as prefix: <f9> <right> , <f9> <left> , <f9> <up> and <f9> <down> 		
Toggle smart-shift mode in current buffer	<f11> <tab> s	(smart-shift-mode &optional ARG)	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 pel-smart-shift-keybinding user-option.
Toggle smart-shift mode globally	<f11> <tab> S	(global-smart-shift-mode &optional ARG)	<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:	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	<ul style="list-style-type: none"> C-c <right> C-c C-<right> <f9> <right> 	(smart-shift-right &optional ARG)	Shift the line or region to the ARG times to the right.
Shift line or region left	<ul style="list-style-type: none"> C-c <left> C-c C-<left> <f9> <left> 	(smart-shift-left &optional ARG)	Shift the line or region to the ARG times to the left.
Shift line or region up	<ul style="list-style-type: none"> C-c <up> C-c C-<up> <f9> <up> 	(smart-shift-up &optional ARG)	Shift the line or region to the ARG times to the upwards.
Shift line or region down	<ul style="list-style-type: none"> C-c <down> C-c C-<down> <f9> <down> 	(smart-shift-down &optional ARG)	Shift the line or region to the ARG times to the downwards

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. Emacs support all modes. It has 2 different buffer local variables that are important and control the rendering of hard-tabs and the indentation: <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. PEL does not yet integrate the smarttabs package. 🚧 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 correctly.	
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