↖ Topic Index

Emacs support for C3 🚧

Description	Keystroke	Function	Note
C3 Support • File associations Last updated on:	C3 is a programming language based off the C language. It is relatively new and still evolving. PEL supports the only supported mode for C3: the **tree-sitter**-based **c3-ts-mode** provided by the external **c3-ts-mode** package. 📦 Requires the **c3-ts-mode** file 🛂 PEL installs it in the utils directory when **pel-use-c3** user-option is set to **t**. • PEL associates the files with the .c3, .c3i and .c3t file extensions with **c3-ts-mode**. ⚠️ PEL support for C3 requires **Emacs >= 30.1** because tree-sitter is required by **c3-ts-mode**, and PEL only support tree-sitter for Emacs >= 30.1: • See Σ **Tree Sitter** and 🚦**Tree-sitter**. • PEL activates Σ **Speedbar** support for the C3 files when **pel-use-speedbar** user-option is on (set to **t**). • imenu support provided by **c3-ts-mode** is available. 2025-12-29		
Open this PDF file. See also: Σ **Help/Info**	`<f11> SPC M-G <f1>` `<f12> <f1>`	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the **𝔓𝔈𝔏 - C3** 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.
Σ **Customize** PEL C3 support	`<f11> SPC M-G <f2>` `<f12> <f2>`	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL C3 support. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window.
Σ **Customize** Emacs C3 support	`<f11> SPC M-G <f3>` `<f12> <f3>`	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs C3 support: c3-ts. • If OTHER-WINDOW is non-nil (use `C-u`), display in another window. ☝Several aspects of c3-ts-mode are controlled by PEL as defined in the user-options accessed by `<f12> <f2>` and override the c3-ts ones.
Show PEL setup for C3	`<f11> SPC M-G ?` `<f12> ?`	(**pel-c3-setup-info &** **optional APPEND**)	Display C3 setup information inside a *pel-c3-info* buffer with buttons providing quick access to the customization buffer of each variable shown. The information shown includes the value and interpretation of: c3-ts-indent-offset, tab-width, activated minor modes To append information in the buffer instead of clearing the previous content type any prefix argument (such as `C-u`) before the command keystroke.
Set visual rendering of hard tabs for the current buffer See Σ **Indentation** for more information and commands.	`<f11> <tab> w` `<f12> M-t`	(**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. • This modifies a buffer local value of the the **tab-width** user-option. • The change is temporary and affects the current buffer only. • To change the tab width used for all C3 source code files, change the '**pel-c3-tab-width**' user-option variable instead.
Comments	See also: Σ **Comments**		
Insert, realign, comment/uncomment region **With PEL**: Comment the current line with `M-0 M-;`	`M-;`	(**comment-dwim** ARG)	Insert or realign comment on current line (or region if a region is active). If line/region is already commented, uncomment it. • On a single line, the comment is placed *after* the code. • `C-u M-;` executes comment-kill
		(**pel-comment-dwim** ARG)	Same as **comment-dwim** but comments the current line with a numeric ARG or 0.
Select C3 comment style	`<f12> <f4> M-;`	(**pel-select-c3-comment-style**)	Select from the following C3 comment styles generated by **comment-dwim** and **pel-comment-dwim** commands: • `//` • `/* */` • `<* *>`
Navigation More in Σ **Navigation**	• **Shift selection** is supported by some commands, not all. The following symbols are used to identify whether the command supports shifts selection: • ⇓ This command supports shift selection in GUI and terminal mode. • ⇓ This command supports shift selection only in GUI mode. • ⇓ This command supports shift selection in GUI mode and also in terminal mode under some conditions (described in the description cell for the command). • ǂ This command does **not** support shift selection. Sometimes for this you can first set the mark before moving. • Pressing the Shift key when using the key binding for commands that do not show any of these 3 arrows have no impact on the shift selection (and may be inappropriate for the command).		
• **by defun : C3 definitions** 🌲	The commands move point by C3 definitions: **functions**, **macros**, **structs**, **bitstructs**, **enums**, **unions**, constants, **alias**, **typedef**, • The `<f6>` cursor key mappings use `<up>` and `<down>` to move to the beginning of the defun, and `<left>` and `<right>` to the end of the defun. • In this context the word *defun* corresponds to any of the C3 definitions listed above. 🌲 These commands are all enhanced by the use of Σ **Tree Sitter** .		
Backward to beginning of C3 definition 🌲⇓	• `<f6> <up>` ⇓ • `C-M-a` • `C-M-<home>` • `C-[ C-a` • `Esc C-a`	(**beginning-of-defun** &optional ARG)	Move backward to the beginning of a C3 definition. • With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun. ⚠ This command moves to the beginning go the next definition of the same nesting level of the current location. It skips the nested definitions.
Forward to end of C3 definition 🌲⇓	• `<f6> <right>` ⇓ • `C-M-e` • `C-M-<end>` • `C-[ C-e` • `Esc C-e`	(**end-of-defun** &optional ARG)	Move forward to next end of C3 definition. • With argument, do it that many times. Negative argument -N means move back to Nth preceding end of defun. ⚠ This command moves to the end of the next **top-level** function or class. It skips the nested definitions.
Forward to start of next C3 definition 🌲⇓	`<f6> <down>`	(**pel-beginning-of-next-defun** &optional SILENT DONT-PUSH_MARK)	Move forward to the beginning of the next C3 definition. • Beeps if does not find beginning of next function unless SILENT is non-nil. • If the beginning of next function is found, push the start location to the mark ring unless DONT-PUSH_MARK is non-nil. • Move back to previous position with `M-`` or `<f6><f6>`
Backward to end of previous C3 definition 🌲⇓	`<f6> <left>`	(**pel-end-of-previous-defun** &optional SILENT DONT-PUSH_MARK)	Move backwards to the end of the previous C3 definition. • Beeps if does not find end of previous function unless SILENT is non-nil. • If the end of previous function is found, push the start location to the mark ring unless DONT-PUSH_MARK is non-nil. • Move back to previous position with `M-`` or `<f6><f6>`
• **by blocks**	Blocks can be: pairs of brackets: (),[],{},<>,"", '', Blocks using parentheses correspond to Lisp S-Expressions (sexp). ☝The commands move across C3 blocks but also to the next/previous syntax element.		
block backward 🌲⇓	• `C-M-<left>` ⇓ • `Esc C-<left>` ⚠ ⇓ • `C-M-b` • `C-[ C-b` • `Esc C-b`	(**backward-sexp** &optional ARG)	Move backward across one balanced expression (sexp). • With ARG, do it that many times. Negative arg -N means move forward across N balanced expressions. This command assumes point is not in a string or comment. • ⚠ With PEL: if you want to use `Esc C-<left>` binding you must ensure that **pel-windmove-on-esc-cursor** user option is set to nil. ❖ `C-M-<left>` does not work on Windows, but `H-<left>` works.

Description	Keystroke	Function	Note
	🐧 Several Linux distros map **C-M-<left>** to desktop workspace operation. In that case you can either use another key binding or change Linux key binding in Systems->settings->keyboard->shortcuts to prevent it from using that key sequence.		
block forward	• **C-M-<right>**	(**forward-sexp** &optional ARG)	Move forward across one balanced expression (sexp). • With ARG, do it that many times. Negative arg -N means move backward across N balanced expressions. This command assumes point is not in a string or comment. • ⚠️ With PEL: if you want to use **Esc C-<right>** binding you must ensure that **pel-windmove-on-esc-cursor** user option is set to nil. ❖ **C-M-<right>** does not work on Windows, but **H-<right>** does.
	• **Esc C-<right>** ⚠️		
	• **C-M-f** • **C-[C-f** • **Esc C-f**		
	🐧 Several Linux distros map **C-M-<right>** to desktop workspace operation. In that case you can either use another key binding or change Linux key binding in Systems->settings->keyboard->shortcuts to prevent it from using that key sequence.		
Up/inside sexp hierarchy	• **C-M-<up>**	(**backward-up-list** &optional ARG ESCAPE-STRINGS NO-SYNTAX-CROSSING)	Move backward out of one level of parentheses. • This command will also work on other parentheses-like expressions defined by the current language mode. With ARG, do this that many times. • A negative argument means move forward but still to a less deep spot. • ⚠️ With PEL: if you want to use **Esc C-<up>** binding you must ensure that **pel-windmove-on-esc-cursor** user option is set to nil. ❖ **C-M-<up>** does not work on Windows, but **H-<up>** does.
	• **Esc C-<up>**		
	• **C-M-u** • **C-[C-u** • **Esc C-u**		
Down/inside sexp/block	• **C-M-<down>**	(**down-list** &optional ARG)	Move forward down one level of parentheses. • This command will also work on other parentheses-like expressions defined by the current language mode. • With ARG, do this that many times. A negative argument means move backward but still go down a level. • This command assumes point is not in a string or comment. • ⚠️ With PEL: To use **Esc C-<down>** binding you must ensure that **pel-windmove-on-esc-cursor** user option is set to nil. ❖ **C-M-<down>** does not work on Windows, but **H-<down>** does.
	• **Esc C-<down>**		
	• **C-M-d** • **C-[C-d** • **Esc C-d**		

Emacs & C3— References

Document	Notes		
The C3 Programming Language	• **C3 home** • c3c: C3 Compiler @ Github	Github repos: • Awesome C3 Projects @ Github	• c3 vendor libraries @ Github
Learning C3	• What is C3?		
C3 blogs	• C evolved: The C3 programming Language, by Cristopher Lernö		
C3 LSP servers	• **c3-lsp**: LSP-server for the C3 language. This must be installed manually. See the installation instructions.		
Emacs support	• **c3-ts-mode @ Github** : tree-sitter-based major-mode for C3. • **tree-sitter-c3 @ Github** : tree-sitter language grammar for C3.		