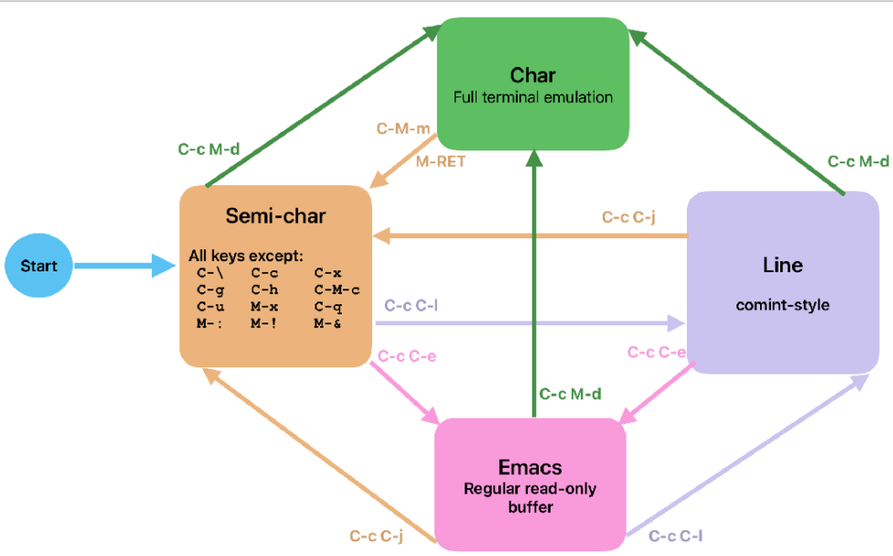
↖ Topic Index

Inside eat-mode - Emulate A Terminal 🚧

Description	Keystroke	Function	Note

eat-mode
• **eat User Manual**
• Emacsconf 2023 Screen cast **presentation of eat** from its author.
 	
Last updated on:
	📦 Emacs external **emacs-eat** provides, eat-mode a terminal emulator that is *"pretty fast, more than three times faster than Term, despite being implemented entirely in Emacs Lisp."*
	PEL activates it when the **pel-use-emacs-eat**user-option is set to **t**.
	• It requires Emacs >= 26.1
	• Also see:
	  • The **Shells** page for information on how to launch the various terminal shells.
	  • The **Shells/Terminals Comparisons** which compares the features of these terminal shells.
	🚧 This is a very early drafted page. I will complete it once I have time to test eat-mode in several environments and test its various features.
	
	2025-04-08		

Open this PDF file. See also: **Help/Info**	`<f11> SPC z f <f1>` `<f12> <f1>`	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the **Shells** 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 eat-mode management control	`<f11> SPC z f <f2>` `<f12> <f2>`	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL eat-mode support. • If the prefix argument is non-nil (like `C-u` or `M--`), display in other window.
Customize Emacs eat-mode control	`<f11> SPC z f <f3>`	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs shell support: shell, comint. • If the prefix argument is non-nil (like `C-u` or `M--`), display in other window.
eat input-mode **Changing modes**	The eat-mode has four distinct input modes: • **char mode** : full terminal emulation similar to Emacs term. • **line mode** : line -oriented input, similar to Emacs shell. • **semi-char mode** : something close to full terminal emulation but allows almost all Emacs key bindings. • **emacs mode**: the buffer is read only and all Emacs key bindings are allowed. The currently active mode is shown on the window's modeline. The key bindings used for switching from a mode to another are show in the diagram. Note: once in char mode, you can only change to semi-char mode.		
Change to char mode	`C-c M-d`	(**eat-char-mode**)	Switch to char mode. Allowed in all other 3 modes.
Change to semi-char mode	• `C-M-m` • `M-RET`	(**eat-semi-char-mode**)	Switch to semi-char mode, when in char mode.
	`C-c C-j`		Switch to semi-char mode, from line or emacs mode.
Change to line mode	`C-c C-l`	(**eat-line-mode**)	Switch to line mode, from semi-char or emacs mode.
Change to emacs mode	`C-c C-e`	(**eat-emacs-mode**)	Switch to Emacs keybindings mode.
• Navigation	The **shell-mode** provides these mode specific navigation commands. Other global navigation commands are available and described in the **Navigation** page.		
Move point to previous prompt	`<f12> <up>`	(**pel-shell-previous-prompt N**)	Move point to the previous prompt line. Repeat N times. With negative N: reverse direction. • Use the **pel-shell-prompt-line-regexp** user-option to identify what to search. This places the point after the prompt at the beginning of the command.
Move point to next prompt	`<f12> <down>`	(**pel-shell-next-prompt N**)	Move point to the next prompt line. Repeat N times. With negative N: reverse direction. • Use the **pel-shell-prompt-line-regexp** user-option to identify what to search. This places the point after the prompt at the beginning of the command.
Move backward command	`C-c C-b`	(**shell-backward-command** &optional ARG)	Move backward across ARG shell command(s). Does not cross lines. • The variable **shell-command-regexp** specifies how to recognize the end of a command.
Move forward command	`C-c C-f`	(**shell-forward-command** &optional ARG)	Move forward across one shell command, but not beyond the current line. • The variable **shell-command-regexp** specifies how to recognize the end of a command.
Move anywhere	To navigate inside the command you can use any of the navigation keys. Some of them are very similar to what the various shells (sh, bash, zsh, etc..) provide since they normally support Emacs navigation keys. The navigation is not restricted to the current command and point can move inside the prompt or above it.		

Move word forward	`C-<right>`	(**pel-forward-token-start** &optional N)	Move forward to the start of the next token. ⚠️ Point can move outside of command.
			A token being identified by:
			  • any word (with all characters allowed by syntax table)
			  • punctuation
			  • first character after whitespace.
			Move over whitespace but stop at comments, operators, punctuation.
			• Argument N is a numeric argument identifying the number of times the operation is done. If N is negative the move is reversed (and goes backward).

Move word backward	`C-<left>`	(**pel-backward-token-start** &optional N)	Move backward to the start of the previous token. • Argument N is a numeric argument identifying the number of times the operation is done. If N is negative the move is reversed (and goes forward).
Move point to beginning of line (after prompt)	`C-c C-a`	(**comint-bol-or-process-mark**)	Move point to beginning of line (after prompt) or to the process mark. • The first time you use this command, it moves to the beginning of the line (but after the prompt, if any). If you repeat it again immediately, it moves point to the process mark. • The process mark separates the process output, along with input already sent, from input that has not yet been sent. Ordinarily, the process mark is at the beginning of the current input line; but if you have used C-c SPC to send multiple lines at once, the process mark is at the beginning of the accumulated input.

Description	Keystroke	Function	Note
Open file at point See: Σ **File-mngt**	Then result of some commands, like compilation, build, or other, may generate a message that identifies a file with line number and possibly column number. With. PEL, you can use the **pel-open-at-point** command to open the file at the specified location and the **pel-set-open-at-point-dir** to establish the root directory. These commands are described in the Σ File-mngt page. A partial copy of the information is placed here for convenience.		

Set base directory for pel-open-at-point relative file names
 	
See: Σ **File-mngt**
	<f11> f ;	(**pel-set-open-at-point-dir**)	Set the behaviour of '**pel-open-at-point**' in **current buffer**. Which defaults to value selected by **pel-open-file-at-point-dir** user-option.
			• Select method used to determine the directory from which a relative file name is built from following methods:
			  • Use visited file parent directory (the default).
			  • Use buffer's current working directory.
			  • Use a specified directory. Prompts for the directory name. Supports completion.

Open file or web-page whose name is at point ★★ See: Σ **File-mngt**	• **M-*** • **<f11> f .** • **6y**	(**pel-open-at-point** &optional N)	Open the file, library or the URL, named at point, with potential line & column #s. • If necessary will search source code files in current project as specified by **pel-filename-at-point-finders** user-option. Type **<f12> <f4> ?** to **show used file search method** in supporting modes. 🎛 Supports glob characters, partial directory path. When multiple files are found it prompts using the method selected by **pel-prompt-read-method** user-option. 📦 The **6y** key-chord is available if **pel-use-key-chord** is non-nil. See Σ **Key-Chords**.