

Inside term-mode *term* & *ansi-term* Buffers

Description	Keystroke	Function	Note
term-mode ansi-term	Emacs term-mode is one of several terminal emulator modes provided by Emacs. - 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. In term-mode, Emacs provides two line modes. By default term-mode operates in char (raw) mode where most key sequences are sent directly to the inferior (sub) shell process. In the other mode, the line (cooked) mode, all Emacs key bindings are available. Using the default (char/raw) mode allows using a larger number of command line programs than what can be used in the shell-mode. Last updated on: 2025-04-08		
Open this PDF file. See also: 🔗 Help/Info	<f11> SPC z t <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.
	<f12> <f1>		
🔗 Customize PEL term management control	<f11> SPC z t <f2>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL term support. If the prefix argument is non-nil (like C-u or M--), display in other window.
	<f12> <f2>		
🔗 Customize Emacs term control	<f11> SPC z t <f3>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs term support. If the prefix argument is non-nil (like C-u or M--), display in other window.
Describe active major/minor(s) modes and the key bindings	C-h m <f1> m <f11> ? k m	(describe-mode &optional BUFFER)	Lists the active major mode, all active minor modes and the bound keystrokes. In various shell modes this is particularly useful. See 🔗 Help/Info for more info.
Switch to char or line Mode	The term-mode has 2 different <i>sub-modes</i> : char-mode, the default, where most characters are sent directly to the inferior (sub) shell process, except for key sequences that start with C-c, the char sub-mode escape character term-escape-char. Use the term-set-escape-char function to change it. The C-c prefix replaces the C-x prefix normally available. line-mode, where keys are passed to Emacs enabling you to use Emacs editing commands. Use the following key sequences to switch from one mode to another. The ansi-term is almost like term-mode, except that the escape character is C-x instead of C-c.		
Switch to line (cooked) mode	C-c C-j	(term-line-mode)	Switch to line ("cooked") sub-mode of term mode. This means that Emacs editing commands work as normally, until you type: RET which sends the current line to the inferior (the shell process), or C-c C-k
Switch to char (raw) mode	C-c C-k	(term-char-mode)	Switch to char ("raw") sub-mode of term mode. Each character you type is sent directly to the inferior without intervention from Emacs, except for the escape character (usually C-c).
Job control	The following commands can be used in both raw and line <i>sub-modes</i> .		
Interrupt current subjob	C-c C-c	(term-interrupt-subjob)	Interrupt the current subjob. Use this to pass Ctrl-C to the shell.
Toggle the <i>page-at-a-time</i> feature	C-c C-q	(term-pager-toggle)	Toggle the <i>page-at-a-time</i> feature which operates like Unix more. When active, the mode-line displays 'page', and when the output of a command is longer than a page worth of lines, it pauses, displays ***MORE*** and waits for the SPC key or some other control key (type ? to list them).
Navigation (in line-mode)	The following commands are available in the line-mode <i>sub-mode</i> of term-mode and ansi-mode. 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.
	C-c C-p	(term-previous-prompt N)	Move to end of Nth previous prompt in the buffer. Uses 'term-prompt-regexp' to search the prompt. By default, PEL sets it to the value specified by pel-shell-prompt-line-regexp, allowing customization of the behaviour. Change this by setting pel-term-use-shell-prompt-line-regexp user option to nil.
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.
	C-c C-n	(term-next-prompt N)	Move to end of Nth next prompt in the buffer. Uses 'term-prompt-regexp' to search the prompt. By default, PEL sets it to the value specified by pel-shell-prompt-line-regexp, allowing customization of the behaviour. Change this by setting pel-term-use-shell-prompt-line-regexp user option to nil.
Open file at point See: 🔗 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 🔗 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: 🔗 File-mngt	<f11> f ;	(pel-set-open-at-point-dir) - 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.	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.
Open file or web-page whose name is at point ★★ See: 🔗 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 🔗 Key-Chords.