↖ Topic Index

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 Σ **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.		

Open this PDF file. See also: Σ **Help/Info**	`<f11> SPC z t <f1>` `<f12> <f1>`	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the Σ **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.
Σ **Customize PEL term management control**	`<f11> SPC z t <f2>` `<f12> <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.
Σ **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 Σ **Help/Info** for more info.

• **Switch to char or line Mode**	The **term-mode** has 2 different *sub-modes*:
	  • **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 *sub-modes*.		
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 *page-at-a-time* feature	`C-c C-q`	(**term-pager-toggle**)	Toggle the *page-at-a-time* 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** s*ub-mode* of term-mode and ansi-mode. Other global navigation commands are available and described in the Σ **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.