## Shells, Terminal Emulators, REPLs & Applications

Description	Keystroke	Function	Note		
Emacs Shells			nning programming language specialized shells and programming language REPL.		
	It provides multiple terminal emulators and shells. There's also several external packages that provide more.				
See also <b>§ Shells/</b>	This page describe the command	ls available to start these wheels,	terminal emulators and REPLs inside Emacs buffer windows.		
Terminals Comparisons	Note that inside the term, ansi-term and vterm buffers, all Emacs keys are not always available: these major modes operate in to input modes:  • shell input (char) mode: where the shell gets the keys  • Emacs input (line) mode: where Emacs key bindings are available.  See   • Shells/Terminals Comparisons for more information.				
Open this PDF file. See also: <u>Nelp/Info</u>	• <f11> SPC SPC z <f1> • <f11> SPC SPC s <f1> • <f11> SPC SPC t <f1></f1></f11></f1></f11></f1></f11>	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the <u>Shells</u> local PDF. If the prefix argument (like <b>C-u</b> or <b>M</b> ) is used, then it opens the remote GitHub hosted raw PDF instead. If the <b>pel-flip-help-pdf-arg</b> user-option is set it's the other way around.		
	<f12> <f1></f1></f12>		The <f12> <f1> key sequence is available in the following major modes: shell-mode,</f1></f12>		
<u>&gt; Customize</u> PEL shell management control	• <f11> SPC SPC z <f2> • <f11> SPC SPC s <f2> • <f11> SPC SPC t <f2></f2></f11></f2></f11></f2></f11>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL shell support: term.  • If OTHER-WINDOW is non-nil (use <b>C-u</b> ), display in other window.		
	<f12> <f2></f2></f12>		The <f12> <f2> key sequence is available in the following major modes: shell-mode,</f2></f12>		
<u>∑ Customize</u> Emacs shell & term control	<f11> SPC SPC z <f3></f3></f11>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs shell support: shell, term, terminal, term.		
∑ Customize Emacs shell control	<f11> SPC SPC s <f3></f3></f11>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs <b>shell-mode</b> support.  • If OTHER-WINDOW is non-nil (use <b>C</b> - <b>u</b> ), display in another window.		
	<f12> <f3></f3></f12>		In <b>shell-mode</b> the <b><f12></f12> <f3></f3></b> key opens the <b>shell</b> customization group.		
© Customize Emacs shell control See also:	<f11> SPC SPC t <f3></f3></f11>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs <b>term-mode</b> support.  • If OTHER-WINDOW is non-nil (use <b>C-u</b> ), display in another window.		
§ Shells/Terminals Comparisons	<f12> <f3></f3></f12>		In <b>term-mode</b> the <f12> <f3> key opens the <b>term</b> customization group.  • AThe key sequence is only available in term and ansi-term buffers when the buffer is operating in Emacs (line) input mode. Toggle to line input mode by typing <b>C-c C-j</b>.</f3></f12>		
Launch OS	With the following command you	can launch an operating system	application that will run independently of Emacs.		
Application from Emacs	<f11> A</f11>	(counsel-linux-app &optional ARG)	Launch a Linux desktop application, similar to Alt- <f2>.  When ARG is non-nil, ignore NoDisplay property in *.desktop files.  On Linux, requires the counsel external package.  PEL activates it when the pel-use-counsel user option is set to t.</f2>		
		(counsel-osx-app)	Launch a macOS application via ivy interface.  On macOS, requires the <u>counselx-osx-app</u> external package. PEL activates it when the <u>pel-use-counsel-osx-app</u> user option is set to t.		
List Emacs Child Processes	Emacs can run several synchronous and asynchronous processes as child processes. They can be listed, showing the actual command line used to launch them with the following command.				
List processes See also: <u>▼ Help/Info</u>	• <f11> z ? • <f11> ? e C-p</f11></f11>	(list-processes &optional QUERY-ONLY BUFFER)	Display a list of all processes that are Emacs sub-processes.  If optional argument QUERY-ONLY is non-nil, only processes with the query-on-exit flag set are listed.  Any process listed as exited or signalled is actually eliminated after the listing is made.		
Run Commands in system shell	The following commands can be used to quickly execute an external command inside a system shell process and display the result inside an Emacs buffer.  • Use these commands from buffer of any major modes.				
Run a shell command	• M-! • ₩-L	(shell-command COMMAND &optional OUTPUT-BUFFER ERROR-BUFFER)	Prompts for the command in the minibuffer, show the command output in the next window in the *Shell Command Output* buffer in Fundamental mode.		
Run a command on a marked region	м-	(shell-command-on-region START END COMMAND &optional OUTPUT-BUFFER REPLACE ERROR-BUFFER DISPLAY-ERROR-BUFFER)	Execute string COMMAND in inferior shell with region as input.  Normally display output (if any) in temp buffer '*Shell Command Output*';  Prefix arg means replace the region with it. Return the exit code of COMMAND.  Mark the region first. Then type M-I. Emacs prompts for the command to run. Use an argument to replace the region with the command output (ie. type C-u M-I)		
Run a shell command asynchronously	M-&	(async-shell-command COMMAND &optional OUTPUT-BUFFER ERROR- BUFFER)	Execute string COMMAND asynchronously in background.  Like 'shell-command', but adds '&' at the end of COMMAND to execute it asynchronously.  The output appears in the buffer '*Async Shell Command*'.  That buffer is in shell mode.		
Open a shell or terminal buffer	Several terminal-like shells are available. They can be grouped in 3 categories:  1. eshell. Pure Emacs shell with all commands implemented in Emacs Lisp. Supports Unix style commands in any Operating System. Also support evaluation of Lisp expressions. If you know Emacs Lisp this can be extremely useful.  2. The other classical terminal and shells: shell, ansi-term and term. These all have pros and cons. They run slower than vterm but they are built-in. Of those, the ansi-term has more capabilities.  3. vterm. A relatively new shell for Emacs that is very fast. It's an external package that gets installed by PEL when the pel-use-vterm user option is set to t. Fo running Unix commands this is probably what you will want to use. See its installation information on its row below.				
	Each have pros and cons. See the Shells/Terminals Comparisons document.				
Open an eshell	<f11> z e</f11>	(eshell &optional ARG)	Open an eshell buffer. To open another eshell instance: use the <b>C-u</b> prefix  To open a numbered eshell: use the <b>C-u number</b> prefix		
	Implementation:  • eshell is implemented in Emacs Lisp and implements several Unix commands, making them available to OS that do not natively have them (like Windows). If a command is not implemented it runs the one found in PATH.  Extra Features  • Can redirect output into a buffer. The grep command output goes to a grep result buffer which can be used to open the various files.  • Support lisp commands.  Supports  • Cursor lateral cursor line beginning/end, kill, yank.  • Meta-cursor word-move keys, but going left it does not stop at the prompt.  • command tab expansion, command line re-direction  • Is colouring (done by the eshell implementation), columns are aligned.  • Command history (and shows history item # in mini-buffer)  • Can run top, man, less (which start inside separate buffer)  • Can run Python scripts.  Limitations:  • Meta-cursor word-move keys going left does not stop at the prompt.  • Clear screen does not work  • No bash alias, however eshell can remember its own aliases and will prompt for commands often ran & unfound.				

Description	Vouotroko	Function	Note		
Open a shell	Keystroke <f11> z s</f11>	Function (pel-shell)	Note  Opens an inferior shell in the current window or moves point to the *shell* buffer already showing in one of the windows.		
	<f11> z s Implementation • This is the PEL implementation vterm. • The built-in Emacs shell concommands and the PEL implementation vterm. • The Emacs shell command is the does not see the keys until <r p="" supports<=""> • Can run multiple shell, each insection of the constant of the c</r></f11>	(pel-shell)  In which uses the built-in Emacs is mmands creates a window in the plementation fixes that. The oldest one. It uses the comine the plementation fixes that is pressed making it unfit for side its own buffer/name ining/end, kill, yank.  Introl Up/Down) In Python REPL, REPL is OK, expert of the provides the state of the provides the state of the provides the state of the plementations such as top, it (ansi-term PROGRAM & optional NEW-BUFFERNAME)  ult is /bin/bash. Can use others.	Opens an inferior shell in the current window or moves point to the *shell* buffer already showing in one of the windows.  Shell command and ensures it opens inside the current window, like term, ansi-term, ielm and enther window. This is a surprising behaviour compared to the other inferior process it-mode, which makes it quite versatile. Emacs keys are possible, however the sub-process reprograms that directly read the input.  She is OK, no Python colouring, but each command is colored.  2> c or C-c M-o. See below.  Normally operates in character mode, in which up/down navigation and kill/yank is not possible. Change to line mode to do that:  Use C-x C-j to change to line mode an allow movement, mark, saving.  When done use C-c C-k to switch to character mode.		
	<ul> <li>A terminal emulator written in Emacs Lisp.</li> <li>Newer implementation than term.</li> <li>You can even run other editors within it (vi, emacs, others). But use character-mode.</li> <li>Specificities:</li> <li>C-x is mapped to term-escape-char</li> <li>Supports:</li> <li>Scroll up/down with M-<up>, M-<down></down></up></li> <li>Is colouring, columns are aligned</li> <li>bash alias</li> <li>bash alias</li> <li>bash tab expansion</li> <li>command line redirection</li> <li>clear screen</li> <li>Command history</li> <li>Can run Python scripts.</li> <li>Running Python shell:</li> <li>REPL is OK, echo is OK</li> <li>Limitations:</li> <li>Natively runs in character mode, which does not allow movement nor saving.</li> <li><up>, <down> cursor, C-n/C-p do not work as navigating: used as shell command history. Change to line mode (see above) to enable these.</down></up></li> <li>Not yet found a way to control prompt (PS1 setup of .bash _profile does not seem to be used).</li> </ul>				
Open a term shell	<f11> z t</f11>	(term PROGRAM)	Prompts for shell to use. Default is /bin/bash. Can use others. Opens in current window.		
	Implementation:  Shell implemented in Emacs Lisp. The keys are sent directly to the sub-process, which means they are not interpreted by Emacs.  Same access as normal shell: can use the bash alias, tab-autocomplete, clear screen, can use less and indirection, can execute python scripts. Can even run other terminal editors like vim, synaptic, etc  Supports  Cursor lateral cursor line beginning/end, kill, yank.  Meta-cursor keys, but only in terminal Emacs, not in GUI Emacs.  Is colouring, columns are aligned  bash tab expansion  command line redirection  clear screen  Command history  Can run Python scripts.  Running Python shell:  REPL is OK, echo is OK  Limitations:  In GUI Emacs: Meta-left/right cursor word move do not work. Use Esc-b and Esc-f here instead.  Normal Emacs keystrokes does not always work, it depends on the programs that are executed from the shell. When it stops working, either use C-c b to switch to another buffer or exit the shell to gain control to Emacs keys in this buffer.  Vertical cursor history works only with Control-Up and Control-Down  Emacs keys with Meta do not work. The ones with Control do work.  Can run top in the buffer, but then C-c does not stop it. To stop it split the buffer in 2, kill the buffer with C-x k, confirm, close the buffer.				
Open a <u>vterm shell</u>	<f11> z v</f11>	(vterm &optional BUFFER- NAME)	Create a new vterm shell. A fast & full-featured *nix-compliant shell.  d Although vterm is relatively new this is the fastest shell. Highly recommended.		
	Requires the Emacs-libvterm (vterm) external package, the libvterm library. On macOS that can be installed with Homebrew. PEL activates it when the pel-use-vterm user option is set to t.  Use C-c C-t to toggle the Vterm-Copy mode which allows navigation and text copy in the buffer.  While the buffer is in Vterm mode you cannot use the PEL function keys as they are interpreted by the program running in the vterm shell. All other Emacs keys work. In Vterm-Copy the function keys are interpreted by Emacs so the PEL function key mappings do work.  The value used is set by the vterm-max-scrollback user option which defaults to 1000. If you plan to use commands that print a long number of lines, you may want to change this value.  When using this shell please first read the shell-side-configuration notes				
Shell-mode Commands	The following commands are ava	ilable in a shell-mode buffer.			
Resync directories.  • Use to (re-)activate tab completion in shell.	<ul> <li><f12> r (shell-resync-dirs) Resync the buffer's idea of the current directory stack.</f12></li> <li>Use this command to allow tab completion at the shell prompt if it does not work.</li> <li>This command queries the shell with the command bound to 'shell-dirstack-query' (default "dirs"), reads the next line output and parses it to form the new directory stack.</li> <li>DON'T issue this command unless the buffer is at a shell prompt.</li> <li>Also, note that if some other subprocess decides to do output immediately after the query, its output will be taken as the new directory stack you lose. If this happens, just do the command again.</li> </ul>				
Clear shell buffer	• <f12> c • C-c M-o</f12>	(pel-comint-clear-buffer- and-get-prompt @optional BUFFER-OR-NAME)	Clear the command interpreter buffer. Ensure the shell is ready to take input.  • Useful in the *shell* buffer because the clear shell command does not work.  • The C-c M-o binding to the PEL function is local to the buffer. Bindings in buffers with the major modes remain attached to the Emacs built-in comint-clear-buffer command.		
Move point to previous prompt	<f12> <up></up></f12>	(pel-shell-previous-prompt)	Move point to the previous prompt line.  • Use the pel-shell-prompt-line-regexp user-option to identify what to search.		
Move point to next prompt	<f12> <down></down></f12>	(pel-shell-next-prompt)	Move point to the next prompt line.  • Use the pel-shell-prompt-line-regexp user-option to identify what to search.		
prompt			Use the pel-shell-prompt-line-regexp user-option to identity what to search.		

Specialized REPL  You can run several read eval nut loop programming shells in Emacs.  Several of those REPLs, like lelm and run-python are part of Emacs.  PEL makes the other available or adds some functionality to others when the corresponding pel-use-user option variable for the respective language is turned on leaft to 1).  It is also possible to use shells to run other PEPL programs directly from an embedded terminal shell like veterm (see above).  The command for the finess Lisp REPL, lelm, is accessible via the pelregal key prefix ⟨≤11⟩ z    The REPL for the other programming languages are accessible via the pelregal key prefix ⟨≤11⟩ z    All REPL commands are accessible via the <≤12⟩ z key binding of their respective major mode.  Start Shell  See also: 191 - Are  From Arc buffer  ⟨≤11⟩ z    (run-arc CMD)  Run an inferior Arc process, input and output via buffer "arc", switch to that buffer.  (filt) = z    Fun an inferior Arc process, input and output via buffer "arc", switch to that buffer.  (filt) = z    Fun an inferior Arc process, input and output via buffer "arc", switch to that buffer in the process buffer on all stor commands.)  From Arc buffer  ⟨≤11⟩ z    (filt) = z    (f	
If there is a process already running in "arc", switch to that buffer.   With argument, lower you to edit the command line (default is value of 'command').   Requires the arc-mode external package.   Runs the hook 'inferior-arc-mode-hook' (after the 'cominit-mode-hook' is '(Type hi in the process buffer for a list of commands.)   Requires the arc-mode external package.   PEL activates this when the pel-use-arc user-options is set to to the sea also:   1	arc-program-
PEL activates this when the pel-use-arc user-options is set to t.	s run).
See also:  ### - Emacs Lisp    * <f12> z   Switches to the buffer "felm", or creates it fit does not exist.   * Switches to the buffer "felm", or creates it fit does not exist.   * Switches to the buffer "felm", or creates it fit does not exist.   * Switches to the buffer "felm", or creates it fit does not exist.   * Switches to the buffer "felm", or creates it fit does not exist.   * Switches to the buffer "felm", or creates it fit does not exist.   * Switches to the buffer "felm", or creates it fit does not exist.   * Switches to the buffer in emacs-lisp-mode.   * Open or switch to Common-Lisp REPL buffer window.   * Use the Common Lisp REPL selected by the PEL user-options:   * Slime when 'pel-use-slime'is on and 'pel-clisp-ide' is set to slime,   * Slime when 'pel-use-slime'is on and 'pel-clisp-ide' is set to slime,   * The behaviour of the command is affected by the optional argument N:   * With no buffers running REPL:   * N is nil or absent: open REPL in current window   * N is positive: open REPL in current window   * N is negative: create new REPL in current window   * N is nil or absent: open REPL in current window   * N is nil or absent: open REPL in current window   * N is positive: open REPL in current window   * N is nil or absent: open REPL in current window   * N is positive: open REPL in current window   * N is nil or absent: open REPL in current window   * N is positive: open REPL in current window   * N is positive: open REPL in current window   * N is positive: open REPL in current window   * N is positive: open REPL in current window   * N is positive: open REPL in current window   * N is positive: open REPL in current window   * N is positive: open REPL in current window   * N is positive: open REPL in current window   * N is positive: open REPL in current window   * N is positive: open REPL in current window   * N is positive: open REPL in current window   * N is positive: open REPL in current window   * N is positive: open REPL in current window   * N is positive: open REPL in current wi</f12>	
Use the Common Lisp REPL selected by the PEL user-options: SLY when 'pel-use-sly' is on and 'pel-clisp-ide' is set to sly, Slime when 'pel-use-slime' is on and 'pel-clisp-ide' is set to sly, Slime when 'pel-use-slime' is on and 'pel-clisp-ide' is set to sly, Slime when 'pel-use-slime' is on and 'pel-clisp-ide' is set to slime, the inferior lisp mode otherwise. The behaviour of the command is affected by the optional argument N: with no buffers running REPL: N is nil or absent: open REPL in current window N is positive: open REPL in current window N is negative: create new REPL in current window N is negative: open REPL in current window N is positive: open REPL	valuate Emacs
See also: 191 - Elixir  • Show the IEx buffer if an IEx process is already run.  Requires the alchemist package and the Elixir programming language for to t.  Start Erlang Shell  • <f11> z r e  (erlang-shell)  Start a new Erlang shell.  • The variable 'erlang-shell-function' decides which method to use, default Erlang host. It is possible that, in the future, a new shell on an already run started.</f11>	
See also: 191 - Elixir  See also: 191 - Elixir  See also: 191 - Elixir  • Show the IEx buffer if an IEx process is already run.  • Requires the alchemist package and the Elixir programming language for to t.  Start Erlang Shell  • <f11> z r e  (erlang-shell)  Start a new Erlang shell.  • The variable 'erlang-shell-function' decides which method to use, default Erlang host. It is possible that, in the future, a new shell on an already run.  • The variable 'erlang-shell-function' decides which method to use, default Erlang host. It is possible that, in the future, a new shell on an already run.</f11>	
See also: Pi - Erlang  • C-c C-z • <f12> z  • The variable 'erlang-shell-function' decides which method to use, default Erlang host. It is possible that, in the future, a new shell on an already rustarted.</f12>	•
See also: 191 - Erlang   • C-c C-z   Erlang host. It is possible that, in the future, a new shell on an already rustarted.	It is to start a new
• C-c C-z starts the Erlang Shell from the Erlang Mode.     • <f11> z r starts it anytime, as long as it was installed.  Under PEL this command is available only when the pel-use-erlang use t.</f11>	unning host will be
Open a Forth shell	
See also: <u>also: last horizontal last and activated and last and last and activated and last </u>	lation and it will h case Homebrew
• From Forth buffer: <f12> z  use-forth user option is t. It also requires a Forth interpreter (which must be separately)</f12>	
Start Haskell Shell See also: NI - Haskell See also: NI - Haskell Start Haskell (run-haskell) Show the inferior-haskell buffer. Start the process if needed.  Requires the haskell-mode and Haskel installed.	
• From buffer <f12> z PEL activates this when the pel-use-haskell and the pel-use-haskell-options are set to t.</f12>	·mode user-
Start Julia Shell  See also: 11 > z r j  (julia-snail)  Start a Julia REPL and connect to it, or switch if one already exists.  • The following buffer-local variables control it:  • 'julia-snail-port' (default: *julia*)  • 'julia-snail-port' (default: 10011)  • To create multiple REPLs, give these variables distinct values (e.g.: *julia-snail trains and 10012).  Requires the julia-snail Emacs package and the Julia programming language it also requires yterm (see above).	
• From Julia buffer: <f12> z</f12>	
LFE Shell (Lisp Flavoured Erlang)  (run-lfe CMD)  Run an inferior LFE process, input and output via a buffer '*inferior-lfe*'.  • If 'CMD' is given, use it to start the shell, otherwise:  'inferior-lfe-program' 'inferior-lfe-program-options' -env TERM'  Requires the Ife-mode package and LFE (Lisp Flavoured Erlang) install	
• From LFE buffer: <f12> z    From LFE buffer:   Flavoured Erlang) Install    </f12>	ieu.
Start OCaml Shell See also:    Start OCaml Shell	
• From OCaml buffer <f12> z  PEL activates this when the pel-use-ocaml and the pel-use-tuareg us to t.</f12>	er-options are set
Start Python Shell	ue When called
See also: 181 Python interactively with 'prefix-arg', it allows the user to edit such value and che interpreter should be DEDICATED for the current buffer. When numeric put than 0 or 4 do not SHOW.	oose whether the
• From Python buffer:   • For a given buffer and same values of DEDICATED, if a process is already will do nothing. This means that if the current buffer is using a global prostill able to switch it to use a dedicated one.	

<u>Description</u>	<u>Keystroke</u>	Function	<u>Note</u>
Start Chez Scheme Shell See also: • From Chez buffer	<f11> z r C-z</f11>	(pel-chez-repl &optional N)	Run the Chez REPL in window specified by N.  • By default use the other window. If a numeric argument is specified, its value correspond to the direction of a numeric keypad:  8 4 6
			2 That is:  • 8: up  • 4: left  • 6: right  • 2: down  • 0 and 5 identify the current window.  Requires the Chez Scheme installed.
Start Chibi Scheme Shell See also:	<f11> z r C-i</f11>	(pel-chibi-repl &optional N)	Run the Chibi REPL in window specified by N. • See 'pel-chez-repl' for complete description. Requires the Chibi Scheme installed.  PEL activates it when the pel-use-chibi is set to t.
From Chibi buffer	<f12> z</f12>		
Start Chicken Scheme Shell See also:	<f11> z r C-k</f11>	( <b>pel-chicken-repl</b> &optional N)	Run the Chicken REPL in window specified by N.  • See 'pel-chez-repl' for complete description.  Requires the Chicken Scheme installed.
From Chicken buffer	<f12> z</f12>		set to t.
Start Gambit Scheme	<f11> z r C-b</f11>	(pel-gambit-repl &optional N)	Run the Gambit Scheme REPL in window specified by N.  • See 'pel-chez-repl' for complete description.  Requires the gambit.el file and Chicken Scheme installed.  description.
Shell See also: PI - Gambit Scheme • From Gambit buffer	<f12> z</f12>		
Start Gerbil Scheme Shell See also: <u>MI - Gerbil</u>	<f11> z r C-e</f11>	(pel-gerbil-repl &optional N)	Run the Gerbil REPL in window specified by N.  • See 'pel-chez-repl' for complete description.  Requires the gerbil-mode external package and Gerbil Scheme installed.  PEL activates it when the pel-use-gerbil is set to t.
• From Gerbil buffer	<f12> z</f12>		
Start Guile Shell	<f11> z r C-g</f11>	(pel-guile-repl &optional N)	Run the Guile REPL in window specified by N.
From Guile buffer	<f12> z</f12>		See 'pel-chez-repl' for complete description.  Requires Guile Scheme installed.  PEL activates it when the pel-use-guile is set to t.
Start MIT/GNU Scheme Shell • From MIT/GNU Scheme buffer	<f11> z r C-m</f11>	(pel-mit-scheme-repl &optional N)	Run the MIT/GNU Scheme REPL in window specified by N. • See 'pel-chez-repl' for complete description. Requires MIT/GNU Scheme Scheme installed.  ZHEL activates it when the pel-use-mit-scheme is set to t.
	<f12> z</f12>		
Start Racket Shell See also: Pt - Racket	<f11> z r C-r</f11>	(pel-racket-repl &optional N)	Run the Racket REPL in window specified by N.  • See 'pel-chez-repl' for complete description.
From Racket buffer	<f12> z</f12>		Requires the <u>racket-mode</u> external package and Racket installed.  PEL activates it when the <u>pel-use-racket</u> is set to t.
Start Scsh Scheme Shell	<f11> z r</f11>	(pel-scsh-repl &optional N)	Run the Scsh REPL in window specified by N.  • See 'pel-chez-repl' for complete description.
From Scsh buffer	<f12> z</f12>		Requires Scsh Scheme Scheme installed.

## Shells - References

Topic & Link	Extra Notes
GNU Emacs - Running Shell Commands	
Eshell manual	
Difference between various emacs shells	
Difference between various emacs shells	
How to run multiple shells on Emacs	
EmacsWiki: Ansi Term	Quick overview
Emacswiki: Ansi Term Hints	Several hints
Copy/Paste in Ansi Term	Quick overview of the capability for cut/paste.
Launch GUI emacs from command line in OSX	This describes a solution on how to start the GUI emacs in OSX, but not in the background
How to launch GUI Emacs from command line in OSX?	This one describes the solution for handling it in the background
Run commands in background	Describes the & and the disown
Executing commands in background from bash scripts	
Pass command arguments to bash scripts	
explainshell.com	Online application where you can type a shell command: the app explains each argument. Very useful.