↖ Topic Index

Input Completion (Emacs default, Helm, Ido, Ivy, Ivy/Counsel) 🚧

Operation	Keystroke	Function	Note

Input Completion
• **Default**
• **Ido**
• **Ivy**
• **Helm**
See also: 🚦 **Completion Modes Availability**
See also: ⨌ **Navigation**
			Input completion := automated completion of user input on prompts.
			On Emacs, input completion is available when Emacs prompts for the name of a file, buffer, variable, function, command, and more.
			• It offers help to complete your input.
			• Several methods are supported, some are built-in or use a package that comes with Emacs, others require external packages.
			
			The available input completion methods are:
			• **Emacs default completion** method. The simplest method. Type text then use the **<tab>** key to get a list of potential candidates displayed inside the *completion* buffer.
			• **Ido (Interactive Do)**, included with Emacs, is a much more powerful completion method. Several packages extend it further in various ways. PEL supports several of them.
			  • When basic **ido** activated, Ido completion is available in file (**C-x C-f**) and buffer name (**C-x b**) prompts. Extension are available:
			  • With the **smex** external package, command prompt down with **M-x** and **M-X** provide specialized completion.
			  • With the **Ido ubiquitous mode** external package, several other commands provide completion, like the help for objects (**<f1> o**), help for variable (**<f1> v**) and several other commands.
			• **Ivy** , which provides simpler completion with very good visual feedback, using vertical lines menu at the bottom of the Emacs frame. Ivy, like Helm provides completion for more commands than the basic Ido.
			• **Helm**, a powerful incremental and narrowing completion system. It uses a larger part of the frame than the others but provide commands that can be applied to the selection. More info on this **Helm introduction page**.
			• **Ido/Helm** is a hybrid mode implemented by PEL where Ido is used for buffer and file prompts and Helm is used for the others.
			
			PEL provides completion for searching symbols defined in the source code file(s):
			• The command **pel-goto-symbol** (bound to **M-g M-h**) prompts for a symbol defined in the current buffer.
			• The command **pel-imenu-anywhere** (bound to **M-g M-y**) prompts for the symbols defined in all buffers of the same type as the current one.

Open this PDF file. See also: ⨌ **Help/Info**	**<f11> M-c <f1>**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the ⨌ **Completion/Input** 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. ☝ You can also type **<f1>** during completion prompt to open this PDF file. See below.
Customize PEL Input Completion Support See also:⨌ **Customize**	**<f11> M-c <f2>**	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL input completion support: open PEL buffer support specific group. • If OTHER-WINDOW is non-nil (use **C-u**), display in other window.
Customize Emacs & external package input completion support See also:⨌ **Customize**	**<f11> M-c <f3>**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs and external packages related to input completion, including: 1. helm 2. ido 3. ido-completing-read-plus 4. ido-grid-mode 5. ivy 6. counsel 7. minibuffer : where the completion-style user-variable is defined. 8. smex • When a prefix argument (like **C-u**) opens the buffer inside another window.
			☝ Some of the available groups are only available when the corresponding feature is installed and loaded, see below. • PEL will prompt if a required feature is not loaded and will attempt to load it if its installed. If it is not installed, it will request it to be installed. • Emacs will simply refuse to load a group when its related feature has not previously loaded since Emacs won't know about it. PEL simplifies this.

Input Completion Mode Selection
See also: 🚦 **Completion Modes Availability**
See also: ⨌ **Navigation**
			PEL provides, via the **pel-pkg-for-completion** customization (accessible via **<f11> M-c <f2>**) the following:
			• activation of external package that extend Emacs completion mechanisms via several customization **pel-use-** user-option variables,
			• selection of Emacs initial behaviour of various aspects of completion via the customization **pel-initial-** user-options variables,
			• a set of commands to modify the state of the selection during an Emacs editing session. The modification does not persist across Emacs editing sessions (like the customization selection do) but allow you to quickly change the input completion behaviour and adapt it during an editing session.
			• A command that displays the current state of PEL controlled input completion mechanisms.
			
			The **pel-pkg-for-completion** customization group provides the ability to activate the following features:
			1. **Ido mode** completion. Built-in Emacs but not activated by default. PEL activates it when **pel-use-ido** user-option is set to **t.**
			  • Several other packages attempt to extend Ido or provide Ido in more prompts, including the following supported by PEL:
			  • 📦 **Ido ubiquitous mode**. PEL activates it when **pel-use-ido-ubiquitous** user-option is set to **t** or **use-from-start**. Extends Ido to more commands.
			  • 📦 **smex**: Ido completion for **M-x** PEL activates it when **pel-use-smex** user-option is set to **t**. Extends Ido to **M-x** , and provides **M-X** for Ido restricted to major modes.
			  • 📦 **flx-ido** : Ido and ivy fuzzy completion. PEL activates it when **pel-use-flx** user-option is set to **t** or **use-from-start**. Specializes pattern matching specially useful for long names or file paths.
			  • **Ido presentation geometries:** by default Ido prompt is linear. That can be changed with the following packages:
			  • 📦 **ido-vertical-mode**. PEL activates it when **pel-use-ido-vertical-mode** user-variable is set to **t**. Shows Ido matches vertically.
			  • 📦 **ido-grid**. PEL activates it when **pel-use-ido-grid** user-variable is set to **t**. Provides a grid geometry for Ido. ido-grid-mode's successor.
			  • 📦 **ido-grid-mode**. PEL activates it when **pel-use-ido-grid-mode** user-variable is set to **t** and **pel-use-ido-grid** is set to **nil**. This provides 2 geometries collapsed and expanded. This package is not compatible with ido-grid.
			2. 📦 **Ivy mode** completion : set **pel-use-ivy** to **t**
			  • 📦 **Ivy mode completion with Counsel mode** : set **pel-use-counsel** to **t**
			3. 📦 **Helm mode** completion : set **pel-use-helm** to **t**.
			  • Ido/Helm mode, implemented by PEL, where Ido is used for dealing with Files and buffers and Helm is used everywhere else.
			🎛 The customization group also provides these user-options that select Emacs initial behaviour:
			  • **pel-initial-completion-mode** : identifies the completion mode used when Emacs starts.
			  • **pel-initial-ido-geometry**: identifies the geometry of the Ido prompt. It is used when Ido completion is used.
			PEL makes the following commands available to change the completion mode and to see which one is currently active.
			• PEL also provide extended completion selection of commands for navigation across imenu extracted symbols in current buffer and all buffers that use the same major mode as the current buffer.
			  • **pel-initial-goto-symbol-completion-mode** user-option identifies the completion mode used by **pel-imenu-anywhere**.

Select the completion mode	**<f11> M-c <f4>**	(**pel-select-completion-mode**)	Prompt user for completion mode to activate. The available modes depend on what is currently activated by customization. See the list above.

Toggle Ido ubiquitous mode	• **<f11> M-c M-u**
	• **M-g <f4> M-u**
		(**pel-ido-ubiquitous** &optional ACTIVATE SILENT)	Activate, deactivate or toggle the 'ido-ubiquitous-mode'.
			  • By default: toggle mode.
			  • With the **C-u** prefix or positive numeric prefix: activate
			  • With negative numeric prefix: deactivate.
			📦 Requires the **Ido ubiquitous mode** external package. PEL activates it when **pel-use-ido-ubiquitous** user-option is set to **t** or **use-from-start**.

			• ☝Once this is activated, the new commands providing Ido completion will also provide ivy and helm completion when those modes are selected. • You can add more commands with completion (eg. describe-variable) via the **ido-completing-read-plus** customization group by adding them to the **ido-cr+-function-whitelist** user-option or prevent some by adding them in the **ido-cr+function-blacklist** user-option.
Toggle flx-ido	• **<f11> M-c M-f** • **M-g <f4> M-f**	(**pel-flx-ido** &optional ACTIVATE SILENT)	Toggles use of **flx-ido** where Ido completion is available. 📦 Requires the **flx-ido** external package. PEL activates it when **pel-use-flx** user-option is set to **t** or **use-from-start**.

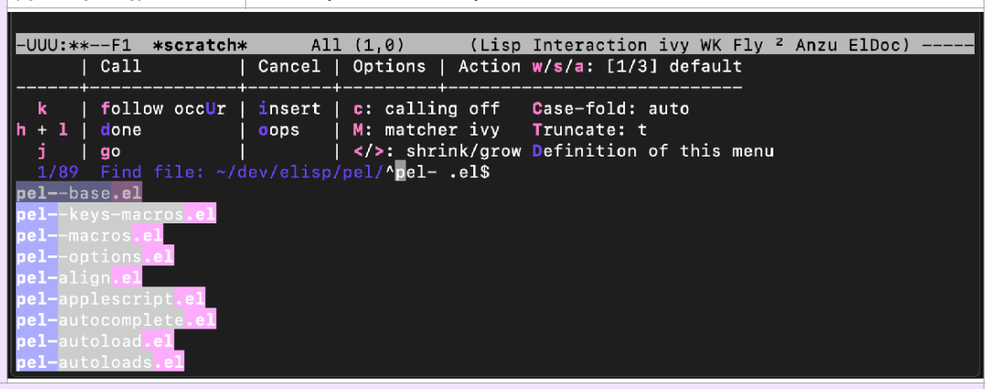
Operation	Keystroke	Function	Note
Change Ido Presentation Geometry: • **default - linear** • **grid -collapsed** • **grid - expanded** • **vertical**	• **<f11> M-c M-g** • **M-g <f4> M-g**	**(pel-select-ido-geometry)**	Select the way Ido shows the match selection: • emacs default : a linear selection • grid-collapsed : ido-grid-mode starting collapsed: type Tab to expand grid • grid-expanded : ido-grid-mode already expanded • vertical At first the geometry used is determined by the value of the **pel-initial-ido-geometry** user-option, this command changes what is sued by Emacs during the editing session but does not change the user-option value which persists across editing sessions.
			Requires **ido-grid** and pel-use-ido-grid set to **t** for the ido-grid selection. Requires **ido-grid-mode** and pel-use-ido-grid-mode set to **t** and pel-use-ido-grid set to nil for the ido-grid-mode selections: collapsed-gid and expanded-grid. Requires **Ido ubiquitous mode** and pel-use-ido-vertical-mode set to **t** for the vertical selection.
Show what completion mode is currently used.	**<f11> M-c ?**	**(pel-show-active-completion-mode)**	Display the completion mode currently used, and the Ido prompt geometry.
• **Default Input Completion**			
			Emacs default input completion is available when no other completion mechanism is active. • The keys available to expand or act on the completed name (or symbol) are listed below. • See **Emacs Completion Example** for a simple example of how to use completion keys. The **completion-style** user-option variable from the ido group controls the types of matching styles supported: • basic: complete with the same beginning • partial-completion: agressive completion: *em-l-m* matches emacs-lisp-mode. • emacs22: same as basic but ignores text in minibuffer after point • substring: must contain text in minibuffer & point position controls matching extension added to beginning, end and where point is located. • initials: aggressive completion style: attempt to complete acronyms and initialisms: for example: *lch* matches list-command-history. The first 3 are available in the default value of completion-style. They can be added by customization: • **M-x customize-option RET completion-style RET** to customize this variable, • **M-x customize-group RET minibuffer RET** access its group or • the PEL **<f11> M-c <f2>** key sequence.
Complete word	**SPC**	(**minibuffer-complete-word**)	Complete the minibuffer contents at most a single word. • After one word is completed as much as possible, a space or hyphen is added, provided that matches some possible completion. • Return nil if there is no valid completion, else t.
Complete input	**Tab**	(**minibuffer-complete**)	Complete the minibuffer contents as far as possible. Type it twice if no input to list all choices. • Return nil if there is no valid completion, else t. • If no characters can be completed, display a list of possible completions. • If you repeat this command after it displayed such a list, scroll the window of possible completions.
List all possible choices	**?**	(**minibuffer-completion-help** &optional START END)	Display a list of possible completions of the current minibuffer contents.
Complete and exit	• **RET** • **C-j**	(**minibuffer-complete-and-exit**)	Exit if the minibuffer contains a valid completion. • Otherwise, try to complete the minibuffer contents. If completion leads to a valid completion, a repetition of this command will exit. • If 'minibuffer-completion-confirm' is 'confirm', do not try to complete; instead, ask for confirmation and accept any input if confirmed. • If 'minibuffer-completion-confirm' is 'confirm-after-completion', do not try to complete; instead, ask for confirmation if the preceding minibuffer command was a member of • 'minibuffer-confirm-exit-commands', and accept the input otherwise.
Escape	**C-g**	(**abort-recursive-edit**)	Abort the command that requested this recursive edit or minibuffer input.
Select completion list window	• **M-v** • **<Pgup>**	(**switch-to-completions**)	Select the completion list window: move point to the window listing all possible completions.
In Completion Window	The following commands are available *inside* the completion window listing all possible completions.		
From completion window: • **Select a completion**	• **RET** • **<mouse-2>**	(**choose-completion** &optional EVENT)	Choose the completion at point. • If EVENT, use EVENT's position to determine the starting position.
Move to next completion	• **Tab** • **<right>**	(**next-completion** N)	Move to the next item in the completion list. • With prefix argument N, move N items (negative N means move backward).
Move to previous completion	• **S-Tab** • **<left>**	(**previous-completion** N)	Move to the previous item in the completion list.
Quit completion window	**q**	(**quit-window** &optional KILL WINDOW)	Quit the window showing it and selects the window showing the minibuffer.
Kill completion buffer	**z**	(**kill-current-buffer**)	Kill completion buffer it and delete the window showing it.
• **Ido Input Completion**			
			Emacs also provides the Ido (Interactive Do) completion mechanism in a separate package, part of Emacs distribution but not activated by default. PEL activates the following extra Ido features when Ido mode is selected: • Ido mode used *everywhere*, which for Ido mode means for both file and buffer prompts. It does: `(ido-everywhere 1)` • Flex matching is enabled. It does: `(setq ido-enable-flex-matching t)` • Disable need for confirmation when creating new buffers with C-x b It does: `(setq ido-create-new-buffer 'always)` Ido mode supports different prompts and has different keys valid for these prompts. There are several key sequences valid for all prompts and some valid for each prompt. The prompt types are listed following the key sequence that opens it: • **C-x C-b** Prompt for buffers • **C-x C-f** Prompt for files. • **C-x C-d** Prompt for directory. • **C-x d** Prompt for directory. • For file prompting, Ido propose files in the current directory but can propose others. The prompt in Ido mode remembers directories that have been visited in the past (even in previous Emacs sessions) so it is able to propose files in different directories than the one holding the file visited in the current buffer. • ☝When trying to open a new file and want to avoid ido from proposing matches, type **C-f** to exit Ido mode, then type your new file name. • ☝However, if you want to open a file in a directory previously visited use completion: ido will be able to find it! Ido has a large number of key bindings, listed in the next 4 sections.
• **all prompts**	Commands available in all Ido prompts		
• **Help**	Open this PDF help. Does not modify state of the Ido prompt even though focus might move to the new PDF display window.		
Open this Completion/ Input PDF file or the web page See also: **Help/Info**	**<f12> <f1>**	(**pel-help-on-completion-input** &optional OPEN-WEB-PAGE)	Open the input completion help PDF file. • With a prefix (like **C-u**) use the default browser to open the GitHub web page. • ☝Note that if your browser can render PDF content you will then be able to easily navigate across PDF pages that have several web links.

Operation	Keystroke	Function	Note
• **Matcher control**	The following commands control various aspects of Ido matching mechanism. • These commands are available for all Ido geometries, except **C-p** not available in grid. The **<f12> C-p** is provided to replace it.		
Toggle inclusion of ignored files.	**C-a**	(**ido-toggle-ignore**)	Toggle ignoring items for the current prompt: • In ido-buffer: toggle ignoring special buffers identified in the '**ido-ignore-buffers**' user-option. • In ido-find-file: toggle ignoring files specified with '**ido-ignore-files**' user-option, files with extensions listed in '**completed-ignored-extensions**' **dired** user-option (when '**ido-ignore-extension**' user-option is non-nil) For example, traditional behaviour is: • not to list buffers whose names begin with a space character, • not to list files whose names begin with a #, for which the regexp is '\'#', Customize these user-option variables in the **ido** customization group.
Toggle case folding • **when on: lower case letter match uppercase**	• **C-c** • **<f12> c**	(**ido-toggle-case**)	Toggle the value of 'ido-case-fold' which controls whether searching for buffer or file name should ignore case. ⚠ The **C-c** binding is often hidden by the **C-c** key prefix used by various packages. PEL adds the **<f12> C-c** key binding to the common ido key map for that purpose. ☝ This is quite useful in OS that treat their file names are can sensitive names (like Unix and Linux, but unfortunately not MacOS!) or just if you want to quickly access names that have specific upper or lower case letters in it. For example using case sensitive matching will help select a Makefile.
Toggle prefix matching method	• **C-p** • **<f12> p**	(**ido-toggle-prefix**)	Toggle the value of user-option variable '**ido-enable-prefix**'. It's nil by default. • Non-nil means only match if the entered text is a prefix of file name. • This behavior is like the standard Emacs completion. • If nil, match if the entered text is an arbitrary substring. For example: "base" will match pel—base.el if ido-enable-prefix is nil, but no it is is **t.** ⚠ The **C-p** key is hidden, reused for something else in ido-grid-mode. PEL provides the **<f12> C-p** to make the command available everywhere.
Toggle regular expression matching See also: ⟫ **Search/ Replace**	**C-t**	(**ido-toggle-regexp**)	Toggle the value of '**ido-enable-regexp**' to enable Ido to perform matching using regular expressions. This is nil (off) by default. You can customize this user-option variable. ☝ Regular expression matching is useful to select file with specific extensions. • See ⟫ **Search/Replace** for Emacs regular expression meta characters.
• **Completion**	The following commands control various aspects of Ido matching mechanism. • These commands are available for all Ido geometries.		
Show all possible completions	**?**	(**ido-completion-help**)	Show all possible completions in a completion list buffer.
Complete current selection	**Tab**	(**ido-complete**)	Try and complete the current pattern amongst the item names. • If several candidates, show the list in the *Ido Completions* buffer.
Complete current selection or insert space	**SPC**	(**ido-complete-space**)	Try completion unless inserting the space makes sense. • When space cannot be accepted as input, open a completion list buffer. • With list buffer already opened, scroll one page down and roll back to top.
Narrow list of candidates to current list of matching items.	• **C-SPC** • **C-@**	(**ido-restrict-to-matches** &optional REMOVEP)	Set current item list to the currently matched items. Further match only inside this narrowed list. • With prefix argument, remove the currently matched items instead and start matching against the remaining items.
Undo/redo last Ido "*directory merge*" proposing file in another directory	**C-z**	(**ido-undo-merge-work-directory** &optional TEXT TRY REFRESH)	Undo or redo last Ido directory merge operation. • If no merge has yet taken place, toggle automatic merging option. Ido "*directory merge*" occurs when trying to match a file name found in other directory.
Select match, create buffer/file if none	**C-j**	(**ido-select-text**)	Select entered name without attempt for completion. • If no buffer or file exactly matching the prompt exists, create a new one. • buffer: prompt for confirmation as controlled by `**confirm-nonexistent-file-or-buffer**' and `**ido-create-new-buffer**' user-options. • file: prompt for confirmation as controlled by `**confirm-nonexistent-file-or-buffer**' user-option.
Select first match	• **C-m** • **RET**	(**ido-exit-minibuffer**)	Exit minibuffer, but make sure we have a match if one is needed. • Select the first element in the list of possible match.
• **Match history**	Once a match is selected it is stored in the history. Elements from the history can be retrieved with the following commands. • These commands are available for all Ido geometries.		
Select next match	• **C-.** • **C-s** • **<right>**	(**ido-next-match**)	Move to next match element. Put first element of 'ido-matches' at the end of the list.
Select previous match	• **C-,** • **C-r** • **<left>**	(**ido-prev-match**)	Move to last match element. Put last element of 'ido-matches' at the front of the list.
Get previous selection using regexp	**M-r**	(**previous-matching-history-element** REGEXP N)	Find the previous history element that matches REGEXP. • (Previous history elements refer to earlier actions.) • With prefix argument N, search for Nth previous match. • If N is negative, find the next or Nth next match. • Normally, history elements are matched case-insensitively if 'case-fold-search' is non-nil, but an uppercase letter in REGEXP makes the search case-sensitive. • See also 'minibuffer-history-case-insensitive-variables'.
• **Edit user input** • **Switch mode**	Edit and move point inside user input used as the basis for the match search. Use these to modify the user input to update matching request. • These do not change the order of matches or their position. • These commands are available for all Ido geometries.		
• **Delete next char or** • **enter Dired**	**C-d**	(**ido-magic-delete-char** ARG)	Delete following char in user input or perform magic action. • Before any user entry for item matching in the following file and directory prompt ido functions, preform the following: • ido-dired **... C-d** enter 'dired' on current directory.' • ido-find-file **... C-d** enter 'dired' on current directory. • ido-list-directory **... C-d** enter 'dired' on current directory.
• **Move backward in user-input or** • **change to buffer prompt**	**C-b**	(**ido-magic-backward-char** ARG)	Move backward in user input. Inside directory path move up one directory level. • Before any user entry for item matching, on the left-most character switch to a buffer prompt selected depending on the currently executing command: • ido-buffer **... C-b** fallback to 'switch-to-buffer' • ido-dired **... C-b** switch to 'ido-buffer' • ido-find-file **... C-b** switch to 'ido-buffer' • ido-list-directory **... C-b** switch to 'ido-buffer'
• **Move forward in user input or** • **change to non-Ido find-file**	**C-f**	(**ido-magic-forward-char** ARG	Move forward on user-input. • Before any user entry item matching, switch to a buffer prompt selected depending on the currently executing command: • ido-buffer **... C-f** switch to 'ido-find-file'. • ido-dired **... C-f** fallback to non-Ido 'dired'. • ido-find-file **... C-f** fallback to non-Ido 'find-file'. • ido-list-directory **... C-f** fallback to non-Ido brief 'dired'.

Operation	Keystroke	Function	Note
Enter non-matching edit mode	**C-e**	(**ido-edit-input**)	Switch to a temporary non-matching edit mode for editing the absolute buffer/file/directory name entered so far with Ido; terminate by RET to return to matching mode. • If cursor is not at the end of the user input, move to end of input. • When this is selected the matching mechanism is paused. It restarts with **RET**.
Take/edit first match	**M-SPC**	(**ido-take-first-match**)	Use first matching item as input text. Leave the cursor at the end of input text. ☝ Useful, like **C-e** to edit a match and create a new file with similar name. Then type **C-j** to force Ido to use that name and open a new file.
• **Escape**	Exit the Ido prompt.		
Escape prompt	**C-g**	(**minibuffer-keyboard-quit**)	Abort the command that requested this recursive edit or minibuffer input.
• **buffer prompts**	In **C-x b** buffer prompts, all common commands are available plus the commands listed below. See ⟫ **Buffers** • These commands are available for all Ido geometries.		
Change to file prompt	**C-x C-f**	(**ido-enter-find-file**)	Drop into 'find-file' from buffer switching.
Change to standard buffer prompt See also: ⟫ **Buffers**	**C-x C-b**	(**ido-fallback-command** &optional FALLBACK-COMMAND)	Fallback to 'switch-to-buffer': standard Emacs prompt for buffer. See ⟫ **Buffers** for more information on switch-to-buffer command.
Bury buffer	• **C-S-b** • **<f12> b**	(**ido-bury-buffer-at-head**)	Bury the buffer at the head of the Ido matches, moving it a the end of the list of matching buffers, before the name of current buffer.
Kill buffer identified as the first match	**C-k**	(**ido-kill-buffer-at-head**)	Kill the buffer at the head of 'ido-matches'. • If cursor is not at the end of the user input, delete to end of inpu
Toggle use of virtual buffers	**C-o**	(**ido-toggle-virtual-buffers**)	Toggle the use of virtual buffers. • With virtual buffers on, you see names of buffers that have been opened recently even if they have been closed since then. This includes files recently opened. • This does not include special buffers. Use **C-a** to toggle visibility of special buffers.
• **dir prompts**	In **C-x d** or **C-x C-d** directory related prompts, all common commands are available plus the commands listed below. See ⟫ **File-mngt** • These commands are available for all Ido geometries.		
Change to buffer prompt	**C-x C-b**	(**ido-enter-switch-buffer**)	Drop into 'ido-switch-buffer' from file switching.
Change to non-ido listing directory	**C-x C-f**	(**ido-fallback-command** &optional FALLBACK-COMMAND)	Drop into a non-ido directory listing command: no interpretation, no proposal, accept input as typed.
Enter Dired buffer	**C-x C-d**	(**ido-enter-dired**)	Drop into 'dired' from file switching: open the Dired buffer with current directory name.
Next directory in search	**<down>**	(**ido-next-match-dir**)	Find next directory in match list. If work directories have been merged, cycle through directories for first matching file.
Previous directory in search	**<up>**	(**ido-prev-match-dir**)	Find previous directory in match list. • If work directories have been merged, cycle through directories for first matching file.
To next directory in list	• **<M-up>** • **M-p**	(**ido-prev-work-directory**)	Change to next working directory in list.
To previous directory in list	• **<M-down>** • **M-n**	(**ido-next-work-directory**)	Change to previous working directory in list.
Delete char backwards to go up 1 directory level	• ⌫ • **DEL** • **<backspace>**	(**ido-delete-backward-updir** COUNT)	Delete char backwards, or at beginning of buffer, go up one level.
Delete chars backwards to go up 1 directory level	**M-**⌫	(**ido-delete-backward-word-updir** COUNT)	Delete all chars backwards, or at beginning of buffer, go up one level.
Go up 1 directory level	**C-**⌫	(**ido-up-directory** &optional CLEAR)	Go up one directory level.
Re-read directory	**C-l**	(**ido-reread-directory**)	Read current directory again. • May be useful if cached version is no longer valid, but directory timestamp has not changed (e.g. with FTP or on Windows).
Wide directory	**M-d**	(**ido-wide-find-dir-or-delete-dir** &optional DIR)	Prompt for DIR to search for using 'find', starting from current directory. • If input stack is non-empty, delete current directory component.
Move to previous directory, push it in list of choices.	**M-b**	(**ido-push-dir**)	Move to previous directory in file name, push current input on stack.
Previous directory in search	**M-v**	(**ido-push-dir-first**)	Move to previous directory in file name, push first match on stack.
Expand absolute path and leave Ido until RET	**M-f**	(**ido-wide-find-file-or-pop-dir** ARG)	Expand absolute path of the directory and leave Ido prompting mode until RET is typed.
Remove directory from history	**M-k**	(**ido-forget-work-directory**)	Remove current directory from the history
Make directory	**M-m**	(**ido-make-directory** &optional DIR)	Prompt for DIR to create in current directory.
Show next element in search history	**<PgDn>**	(**next-history-element** N)	Puts next element of the minibuffer history in the minibuffer. With argument N, it uses the Nth following element.
Previous file name	**M-o**	(**ido-prev-work-file**)	Change to previous working file name in list. 🚧 These 2 commands seem to have invalid docstrings, I assume the function names are correct. They move from file to file but i's not obvious what the direction is. More investigation is needed.
Next file name	**M-C-o**	(**ido-next-work-file**)	Change to next working file name in list.
Show previous element in search history	**<Pgup>**	(**previous-history-element** N)	Puts previous element of the minibuffer history in the minibuffer. With argument N, it uses the Nth previous element.
Search for file matching input	**M-s**	(**ido-merge-work-directories**)	Search (and merge) work directories for files matching the current input string. • This searches in all Ido remembered directories, supporting the match mechanisms. ☝ For example, if you want to open the file std_base_type.h while editing some Python file but you know you visited a C directory that had that file, type "_base" followed by M-s
• **file prompts**	In **C-x C-f** file prompts, all common commands, dired commands and the following commands are available. See ⟫ **File-mngt** • These commands are available for all Ido geometries.		
Delete disk file at head	**C-k**	(**ido-delete-file-at-head**)	Delete disk file identified at the head of the matches. Prompt for deleting the file. Then return to the prompt.
Insert word at point in file-name prompt	**C-o**	(**ido-copy-current-word** ALL)	Append the word located at the location of the currently edited buffer (used just before the prompt stated) to the file name used in the prompt. • This is a way to quickly build a file name using the current directory and the new word (which will include the file extension if there is any in the word at point).

Operation	Keystroke	Function	Note
Insert file name of current buffer to prompt	**C-w**	(**ido-copy-current-file-name** ALL)	Insert file name of current buffer. • If repeated, insert text from buffer instead. ☝Use this to create or search for file with the same name as but with a different extension than current one.
Toggle literal reading fo file	**M-l**	(**ido-toggle-literal**)	Toggle literal reading of this file. • Affects next selection of file with RET. • When reading a file literally, Emacs visit it in Fundamental mode and does not interpret the type of file, does not encode or decode, and does not use the major mode normally associated with the file.
Ivy/Counsel/Swiper			
See: • **Ivy Manual** • **Ivy, Counsel, Swiper Tutorial**	Ivy is another powerful completion engine, with a lot of support over a large number of packages. • Ivy is easy and intuitive to use; it shows all potential candidates on several lines. • It supports several commands. The commands and key bindings are listed in the following sections. • PEL adds the <f1> key binding to open this page, although Ivy provides a well written help page that is accessible with **C-h m** and contains most of this information. 📦 Requires **Ivy** external package. PEL activates it when the **pel-use-ivy** user-option is set to **t.**		
Open this ∑ Completion/ Input PDF file or the web page See also: **∑ Help/Info**	**<f12><f1>**	(**pel-help-on-completion-input** &optional OPEN-WEB-PAGE)	Open the input completion help PDF file. • With a prefix (like **C-u**) use the default browser to open the GitHub web page. • ☝Note that if your browser can render PDF content you will then be able to easily navigate across PDF pages that have several web links. • Use the **C-h** key prefix to access Emacs help commands, as PEL uses **<f1>** for this command.
Help for Ivy	**C-h m**	(**ivy-help**)	Open a window with the help for 'ivy'.

• **Using ivy Hydra**	• Enter Hydra with **C-o**
	• When in Hydra, **C-o** or **i** resumes editing.
	• Hydra simplify key strokes, for example: **C-n C-n C-n C-n** is **C-o jjjj** in Hydra. Besides certain shorter keys, Hydra shows useful info such as case folding and the current action.
	• Additionally, here are the keys that are otherwise not bound:
	
	  • **<** and **>** adjust the height of the minibuffer.
	  • **c** (ivy-toggle-calling) - toggle calling the current action each time a different candidate is selected.
	  • **M** (ivy-rotate-preferred-builders) - rotate regex matcher.
	  • **w** and **s** scroll the actions list.
	
	Minibuffer editing is disabled when Hydra is active.
	
	📦Requires the **hydra** external package PEL provides Hydra when **pel-use-hydra** user option is set to **t**.		

Start ivy Hydra	**C-o**	(**hydra-ivy/body**)	Invokes Hydra menus with key shortcuts.
			
• **Ivy Matcher control**	Ivy supports several match control engines (matchers). They are all regexp-based: • **ivy-regex-plus** : spaces separating words match any characters • **ivy-regex-ignore-order** : ignore order of regexp tokens • **ivy–regex-fuzzy** : splits each character with a wildchar and uses **flx** package scoring mechanism if it is installed.		
Change ivy matcher	**C-o M**	(**hydra-ivy/ivy-rotate-preferred-builders**)	Changelvy matcher, rotating from regex-plus, regexp-ignore-order and regexp-fuzzy as described above. • Type **C-o** or **i** to exit Ivy Hydra and continue typing selection.
• **Keys for selection navigation**	Ivy display selections over a set of lines. Use the following keys to navigate through them.		
Select next candidate	• **C-n** • **<down>**	(**ivy-next-line**)	Select next candidate
Select previous candidate	• **C-p** • **<up>**	(**ivy-previous-line**)	Select previous candidate
Scroll to next page	**C-v**	(**ivy-scroll-up-command**)	Next page of candidates
Scroll to previous page	**M-v**	(**ivy-scroll-down-command**)	Previous page of candidates
Select first candidate	**M-<**	(**ivy-beginning-of-buffer**)	Select first candidate
Select last candidate	**M->**	(**ivy-end-of-buffer**)	Select last candidate
• **Keys for single selection**	The following commands act on a single item selection.		
Select candidate end exit	• **C-m** • **RET**	(**ivy-done**)	Exit the minibuffer with the selected candidate. • Exit with the current action.
Select item and execute single action	**M-o**	(**ivy-dispatching-done**)	Select item and prompt for a specific action. One of: • **o**: default operation for the context. • **i**: insert copied textin buffer • **w**: copy selection text
Select candidate/ continue directory completion	**C-j**	(**ivy-alt-done** &optional ARG)	Exit the minibuffer with the selected candidate. • When the candidate is a directory, enter it. Otherwise, exit with the current action. • When ARG is t, exit with current text, ignoring the candidates.
Complete as much as possible	**Tab**	(**ivy-partial-or-done**)	Complete the minibuffer text as much as possible. • Attempt partial completion, extending the current input as much as possible. • **Tab Tab** is the same as **C-j**.

Operation	Keystroke	Function	Note
Use current selection as is	**C-M-j**	(**ivy-immediate-done**)	Exit the minibuffer with current input instead of current candidate. • Exit with the current action, calling it on the current input instead of the current candidate. This is useful especially when creating new files or directories - often the input will match an existing file, which you don't want to select.
Select a candidate with avy	• **C-'** • **M-H**	(**ivy-avy**)	Select a candidate from the current page with **avy** and exit with the current action. Any works by typing 2 characters that are replaced with a highlight to use. ⚠ The **C-'** binding only works in graphics mode. When **pel-use-avy** is t, PEL adds the **M-H** key binding that works also in terminal mode. 📦 Requires the **avy** and **ivy-avy** external packages PEL activates both of them when the **pel-use-avy** user option is set to **t**.
• **Keys for multiple selections**	The following commands act on a multiple item selection, which is used in some context.		
	C-M-m	(**ivy-call**)	Non-exiting version of **C-m**
	C-M-n	(**ivy-next-line-and-call**)	Combines **C-n** and **C-M-m**.
	C-M-p	(**ivy-previous-line-and-call**)	Combines **C-p** and **C-M-m**.
	C-M-o	(**ivy-dispatching-call**)	Non-exiting version of **M-o**
• **Keys that alter minibuffer input**	The following commands modify the user input, using the history of previously selected items.		
Next input in history	**M-n**	(**ivy-next-history-element**)	Select the next history element or symbol/URL at point.
Previous input in history	**M-p**	(**ivy-previous-history-element**)	Select the previous history element or symbol/URL at point.
	C-r	(**ivy-reverse-i-search**)	Start a recursive completion session to select a history element.
	M-j	(**ivy-yank-word**)	Insert the sub-word at point into the minibuffer.
Narrow selection to current matches	• **S-SPC** • **C-SPC**	(**ivy-restrict-to-matches**)	Deletes the current input, and resets the candidates list to the currently restricted matches. • This is how Ivy provides narrowing in successive tiers. ⚠ The **S-SPC** binding only works in graphics mode. PEL adds the **C-SPC** key binding that works also in terminal mode.
• **Other**			
Copy selections in kill ring	**M-w**	(**ivy-kill-ring-save**)	Copies the selected candidates to the kill ring; when the region is active, copies the active region.
• **Saving current completion in buffer**			
List selection in separate buffer	**C-c C-o**	(**ivy-occur**)	Saves the current candidates to a new buffer; the list is active in the new buffer.
Select item from buffer	• **RET** • **<mouse-1>**	(**ivy-occur-press-and-switch**)	Used in the new buffer calls the appropriate action on the selected candidate.
• **Helm Input Completion**			
	The **Helm** external package is very powerful and comes with a large set of features. • It does not use the minibuffer and does not use the <tab> key for completion; you just need to type some part of the text you search for and Helm will pattern match it. Once you enter a command with Helm input completion a Helm buffer shows a list of potential match with the most probable on top. The list is updated as you type and refine your search pattern. • You can resize the Helm window when it is opened. • You can navigate the pattern match list, select one or several matches (for some of the commands that open the Helm buffer like when you type **C-x b** to switch/open other buffer or when you type **C-x C-f** to find/open file(s). • You can also perform other actions on the selections such as opening a file as root. And you can perform a Helm action and keep the Helm window open (as it normally closes right after you made your selection for the command you were executing. • And Helm comes with extensions of other commands, like running top and allowing pattern match to filter the list of processes you want to see. • See the document title "**A package in a league of its own: Helm**" for a more comprehensive overview with screen shots. This is the 📦 **Helm mode** completion external package, activate it with the **pel-use-helm** user-option set to **t**. • PEL provides a basic configuration for Helm that is similar to the extended config described in that document. But it does not set Helm values that can be customized. • Customize Helm with **M-x customize-group helm** or with **<f11> <f2> g helm**. (See also: **Customize**) PEL sets the Helm global prefix to be **C-c h**. Once helm mode is active (or ido/helm mode) you can execute global Helm commands via that prefix key.		
Operation inside Helm buffer	Helm buffer windows opens up as soon as you launch a Helm session. The following sections describe the commands available inside Helm buffer window.		
Resize Helm Window	Use the following command to reposition the Helm buffer window from horizontal to vertical, going through the 4 possible quadrants of the frame.		
Resize Helm window	**C-t**	(**helm-toggle-resplit-and-swap-windows**)	Multi key command to re-split and swap helm window. • First call runs 'helm-toggle-resplit-window', and second call within 1s runs 'helm-swap-windows'.
Navigate Helm Pattern buffer	The following commands move the currently selected pattern line in the Helm pattern buffer list		
Move to next pattern	• **C-n** • **<down>**	(**helm-next-line** &optional ARG)	Move selection to the next ARG line(s). • When numeric prefix arg is > than the number of candidates, then move to the last candidate of current source (i.e. don't move to next source).
Move to previous pattern	• **C-p** • **<up>**	(**helm-previous-line** &optional ARG)	Move selection to the ARG previous line(s). • Same behavior as 'helm-next-line' when called with a numeric prefix arg.
Move down 1 page	• **C-v** • **<PgDn>**	(**helm-next-page**)	Move selection forward with a pageful.
Move up 1 page	• **M-v** • **<PgUp>**	(**helm-previous-page**)	Move selection back with a pageful.
Move to top of list	**M-<**	(**helm-beginning-of-buffer**)	Move selection at the top of helm buffer list.
Move to end of list	**M->**	(**helm-end-of-buffer**)	Move selection at the bottom of helm buffer list.
Select patterns in Helm Pattern buffer	The following commands, available only for some input lists, allow you to mark several patterns to be processed.		
Toggle line selection	• **C-SPC** • **C-@**	(**helm-toggle-visible-mark** ARG)	Toggle helm visible mark at point ARG times. If ARG is negative toggle backward.
Select all	**M-a**	(**helm-mark-all** &optional ALL)	Mark all visible unmarked candidates in current source. • With a prefix arg mark all visible unmarked candidates in all sources.

Operation	Keystroke	Function	Note
Operate on selection	The following commands are used to act on the selected items from the Helm list		
Copy Helm selection to current buffer	• **C-c C-i** • **C-c <tab>**	(**helm-copy-to-buffer**)	Copy selection or marked candidates to 'helm-current-buffer'. • Note that the real values of candidates are copied and not the display values.
Act on current selection(s)	**RET**	(**helm-maybe-exit-minibuffer**)	If Helm session has completed the search and is displaying the result, exit the helm session and act on the current selection, doing what corresponds to the command that launched the Helm session. • The action applies to all selected candidates and is applied inside the window that was current when the Helm session started. so if point is inside window A when you issue a C-x C-f command to find a file and select several files then these files will be opened in buffers whose window will split the area of the previous window A.
Act on current selection(s) • **List possible actions** • **First is the native action** • **Other possible actions follow. The list depends on the original command.**	• **<tab>** • **C-i**	(**helm-select-action**)	Select an action for the currently selected candidate(s). • If action buffer is selected, back to the helm buffer. • If several actions are possible, display a menu of possible actions, their assigned function key (for the first 12 possible action), a short descriptive link that may include possible key binding for the action. • The list of possible actions can be quite long. For example, the list of actions shown in a Helm session opened to visit a file can include about 50 different actions that range from just visiting the file to diffing it, making a backup, compiling it, opening in hexadecimal editing, etc… • The action applies to all selected candidates and is applied inside the window that was current when the Helm session started. so if point is inside window A when you issue a C-x C-f command to find a file and select several files then these files will be opened in buffers whose window will split the area of the previous window A.
Perform action on current pattern without quitting Helm	• **C-j** • **C-M-i**	(**helm-execute-persistent-action** &optional ATTR SPLIT)	Perform the associated action ATTR without quitting helm. • The action applies to the current pattern, not lines that might have been selected.
Helm Help	Once Helm is running the following command open Helms manual.		
Open Helm Manual (in Org mode format)	• **C-h m** • **C-c ?**	(helm-help)	Generate helm's help according to 'help-message' attribute. • If 'helm-buffer' is empty, provide completions on 'helm-sources' to choose its local documentation. • If source doesn't have any 'help-message' attribute, a generic message explaining this is added instead. • The global 'helm-help-message' is always added after this local help.
Launching Helm Search	The rest of this table needs to be completed.		
Helm special commands	Helm provides the following commands that integrate with other tools. With PEL, when Helm or Ido/Helm mode is active the **<F11> h** key prefix is active giving quick access to these useful helm commands.		