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

| Operation | Keystroke | Function | Note |
|---|---|---|---|
| **Input Completion**<br>• **Default**<br>• **Ido**<br>• **Ivy**<br>• **Helm** | On Emacs, input completion is available when Emacs prompts for the name of a file, buffer, variable, function, command (and more). The input completion offers a way to complete your input. Several methods are supported, some are built-in or use a package that comes with Emacs, other require external packages. The available input completion methods are:<br>• Emacs default completion method: use the **<tab>** key to get a list of potential candidates.<br>• Ido (Interactive Do), a much more powerful completion method. Ido is distributed with Emacs.<br>  • By default it support file (**C-x C-f**) and buffer name (**C-x b**) completion.<br>  • PEL provides completion for searching symbols defined in the source code file visited by the current buffer with **M-g h**. See ⅀ Navigation<br>•<br>using several input completion methods. Emacs default completion is based on the **<tab>** key. Just type the beginning of what your search and hit <tab> to get a list of potential candidates in the minibuffer and complete your entry by adding characters. Or after issuing the command type <tab> twice to get the complete list. This is the Emacs standard input completion mechanism. There are several other packages that provide other ways of performing input completion. The Ido package is bundled with Emacs. Helm, Ivy and Counsel are external packages that must be installed and support ELPA compliant package managers (which PEL uses).<br>PEL supports the input completion packages listed below. If you select one or several of the external packages through PEL customization PEL configures them for you and make them available. You select which one you want to use when Emacs starts (via customization) and then when Emacs run you can use a PEL command to select another input completion engine. PEL eases the exploration and you can then use the input completion engine that first your tasks best and change dynamically any time. | | |
| **Input Completion Mode Selection** | PEL supports several input completion modes that kick in with the **M-x**, **C-x b**, **C-x C-f**, **<f1> o** and many other commands. PEL supports the following input completion modes:<br>1. Emacs' default tab completion<br>2. 📦 **Helm mode** completion : ☑ set **pel-use-helm** to **t**.<br>3. **Ido mode** completion : ☑ set **pel-use-ido** to **t**<br>4. 📦 **Ivy mode** completion : ☑ set **pel-use-ivy** to **t**<br>5. 📦 **Ivy mode completion with Counsel mode** : ☑ set **pel-use-counsel** to **t**<br>6. 📦 Ido/Helm mode where Ido is used for dealing with Files and buffers and Helm is used everywhere else (including all Helm specific commands).<br>🎛 Use **<f11> <f2> P M-c**, to customize the PEL completion group user options above.<br>• Set the **pel-initial-completion-mode** user option to select which completion mode is used when Emacs starts.<br><br>As soon as one of the extra completion mode is activated via the corresponding pel-use- user option, PEL makes the following commands available to change the completion mode and to see which one is currently active. | | |
| **CustomizePEL Input Completion control**<br>See also:⅀ **Customize** | **<f11> <f2> P M-c 1** | (**pel-cfg-pkg-completion** &optional OTHER-WINDOW) | Customize PEL Input Completion support: access the customization buffer that holds the PEL user options that activate the input completion packages.<br>• If OTHER-WINDOW is non-nil (use **C-u**), display in other window. |
| **Customize Emacs Input Completion control**<br>See also:⅀ **Customize** | **<f11> <f2> P M-c** | (**pel-cfg-pkg-completion** &optional OTHER-WINDOW) | Customize Emacs Input Completion support: helm, ido, ivy, counsel..<br>• If OTHER-WINDOW is non-nil (use **C-u**), display in other window. |
| **Select the completion mode** | **<f11> M-c** | (**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. |
| **Show what completion mode is currently used.** | **<f11> ? M-c** | (**pel-show-active-completion-mode**) | Display the completion mode currently used. |
| • **Default Input Completion** | Emacs default input completion is available when no other completion mechanism is active.<br>• The keys available to expand or act on the completed name (or symbol) are listed below.<br>• See **Emacs Completion Example** for a simple example of how to use completion keys. | | |
| **Complete word** | **SPC** | (**minibuffer-complete-word**) | Complete the minibuffer contents at most a single word.<br>• After one word is completed as much as possible, a space or hyphen is added, provided that matches some possible completion.<br>• 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.<br>• Return nil if there is no valid completion, else t.<br>• If no characters can be completed, display a list of possible completions.<br>• 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**<br>• **C-j** | (**minibuffer-complete-and-exit**) | Exit if the minibuffer contains a valid completion.<br>• Otherwise, try to complete the minibuffer contents. If completion leads to a valid completion, a repetition of this command will exit.<br>• If 'minibuffer-completion-confirm' is 'confirm', do not try to complete; instead, ask for confirmation and accept any input if confirmed.<br>• 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<br>• '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**<br>• **<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:**<br>• **Select a completion** | • **RET**<br>• **<mouse-2>** | (**choose-completion** &optional EVENT) | Choose the completion at point.<br>• If EVENT, use EVENT's position to determine the starting position. |
| **Move to next completion** | • **Tab**<br>• **<right>** | (**next-completion** N) | Move to the next item in the completion list.<br>• With prefix argument N, move N items (negative N means move backward). |
| **Move to previous completion** | • **S-Tab**<br>• **<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. | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

| Operation | Keystroke | Function | Note |
|---|---|---|---|
| | | | |
| • **Helm Input Completion** | The **Helm** external package is very powerful and comes with a large set of features.<br>• 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.<br>• You can resize the Helm window when it is opened.<br>• 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).<br>• 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.<br>• 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.<br>• See the document title "**A package in a league of its own: Helm**" for a more comprehensive overview with screen shots.<br><br>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)<br><br>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.<br>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.<br>• 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**<br>• **<down>** | (**helm-next-line** &optional ARG) | Move selection to the next ARG line(s).<br>• 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**<br>• **<up>** | (**helm-previous-line** &optional ARG) | Move selection to the ARG previous line(s).<br>• Same behavior as 'helm-next-line' when called with a numeric prefix arg. |
| **Move down 1 page** | • **C-v**<br>• **<PgDn>** | (**helm-next-page**) | Move selection forward with a pageful. |
| **Move up 1 page** | • **M-v**<br>• **<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**<br>• **C-@** | (**helm-toggle-visible-mark** ARG) | Toggle helm visible mark at point ARG times.<br>If ARG is negative toggle backward. |
| **Select all** | **M-a** | (**helm-mark-all** &optional ALL) | Mark all visible unmarked candidates in current source.<br>• With a prefix arg mark all visible unmarked candidates in all sources. |
| **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**<br>• **C-c <tab>** | (**helm-copy-to-buffer**) | Copy selection or marked candidates to 'helm-current-buffer'.<br>• 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.<br>• 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)**<br>• **List possible actions**<br>• **First is the native action**<br>• **Other possible actions follow. The list depends on the original command.** | • **<tab>**<br>• **C-i** | (**helm-select-action**) | Select an action for the currently selected candidate(s).<br>• If action buffer is selected, back to the helm buffer.<br>• 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.<br>• 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…<br>• 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**<br>• **C-M-i** | (**helm-execute-persistent-action** &optional ATTR SPLIT) | Perform the associated action ATTR without quitting helm.<br>• 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**<br>• **C-c ?** | (helm-help) | Generate helm's help according to 'help-message' attribute.<br>• If 'helm-buffer' is empty, provide completions on 'helm-sources' to choose its local documentation.<br>• If source doesn't have any 'help-message' attribute, a generic message explaining this is added instead.<br>• The global 'helm-help-message' is always added after this local help. |
| **Launching Helm Search** | The rest of this table needs to be completed. | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

| Operation | Keystroke | Function | Note |
|---|---|---|---|
| **Helm special commands** | Helm provides the following commands that integrate with other tools.<br>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. | | |
| | | | |
| | | | |
| **Ivy/Counsel/Swiper** | See the **Ivy, Counsel, Swiper Tutorial** | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |