Tags-based Cross Reference Creation and Navigation 🚧

Description	Keystroke	Function	Note
Setup Tags			
Specify a new tags file	`M-x visit-tags-table`	(**visit-tags-table** FILE &optional LOCAL)	Identify the TAGS file that contains the symbol information. • FILE should be the name of a file created with the 'etags' program. • A directory name is ok too; it means file TAGS in that directory. • Normally M-x visit-tags-table sets the global value of 'tags-file-name'. With a prefix arg, set the buffer-local value instead. When called from Lisp, if the optional arg LOCAL is non-nil, set the local value. When you find a tag with M-x find-tag, the buffer it finds the tag in is given a local value of this variable which is the name of the tags file the tag was in. • If this is not done before attempting to jump to a tag-based location Emacs will prompt for the location of the TAGS file.
Customization Groups			
Group: Etags			
Group: Speedbar (tag browser)			
Group: Helm Tags			
Group: Pel Tags			
Looking Up Identifiers	• The first 4 commands find or prompt for identifier or patterns and request the backed to perform the search. • The search backed depends on the major mode. Elisp, for example, uses info from compiler and load path by default. • If the identifier is not found, you can force search for buffer to use a TAGS file created by tags (or equivalent tool) by executing `xref-etag-mode`. • If multiple identifiers are found they are listed inside the *xref* buffer for selection. • To move back to the original location use the `xref-pop-marker-stack` command, with the **`M-,`** key.		
Find definition of identifier at point	**`M-.`**	(**xref-find-definitions** IDENTIFIER)	Grab symbol at point and move cursor to its definition. • If there are more than one match, prompt in the *xref* buffer. • To search for a symbol entered manually, type **`C-u M-.`**
Find definition of identifier at point, display in other window	**`C-x 4 .`**	(**xref-find-definitions-other-window** IDENTIFIER)	Same as **`M-.`** but opens inside another window.
Find definition of identifier at point, display in other frame	**`C-x 5 .`**	(**xref-find-definitions-other-frame** IDENTIFIER)	Same as **`M-.`** but opens inside another frame.
Find all identifiers that match a regex pattern	• **`C-M-.`** • **`<f11> X .`**	(**xref-find-apropos** PATTERN)	Find all meaningful symbols that match PATTERN. • PATTERN is a regex. • The argument has the same meaning as in 'apropos'.
Go back to where M-. was last issued	**`M-,`**	(**xref-pop-marker-stack**)	• Pop back to where M-. was last invoked. • Marker depth is controlled by the **xref-marker-ring-length** user option.
Toggle the Xref-Etags mode on/off	**`<f11> X X`**	(**xref-etags-mode** &optional ARG)	Toggle etags-based search mode on/off. • Certain major modes install their own mechanisms for listing identifiers and navigation. Turn this on to undo those settings and just use etags.
Display state of the Xref-Etags mode (See also ∑ Help/Info)	• **`<f11> X ?`** • **`<f11> ? X`**	(**pel-show-etags-mode-status**)	Display current state of tags-based search mode. • Essentially display the state of the xref-etags-mode variable for the current buffer.
List/navigate-through all xref			
List all references to symbol at point	**`M-?`**	(**xref-find-references** IDENTIFIER)	Grab the symbol at point, prompt and then display ll its use in a *xref* buffer where user can select each line.
		(first-error &optional N)	Restart at the first error. Visit corresponding source code. With prefix arg N, visit the source code of the Nth error.
Move to next reference found	• **`C-`` `** • **`M-g n`** • **`M-g M-n`**	(**next-error** &optional ARG RESET)	A prefix ARG specifies how many error messages to move; negative means move back to previous error messages. Just C-u as a prefix means reparse the error message buffer and start at the first error.
Move to previous reference found	• **`M-g p`** • **`M-g M-p`**	(**previous-error** &optional N)	Prefix arg N says how many error messages to move backwards (or forwards, if negative).
Using CTags			
List all tags of a source file	`M-x list-tags`	(**list-tags** FILE &optional NEXT-MATCH)	Emacs prompts for the name of the source code file. <tab> completion works at prompt. When source file selected, the list of all tags for this file are shown inside a *Tags List* buffer opened in apropos-mode: type **<ret>** on a line to move to the definition, **q** to close the window.
Regexp search on all files in the tags table	`M-x tags-search`	(**tags-search** REGEXP &optional FILE-LIST-FORM)	Regular expression search. I am not sure I understand this. I could not get this to work.
Run query-replace on all the files	`M-x tags-query-replace`	(**tags-query-replace** FROM TO &optional DELIMITED FILE-LIST-FORM)	Does a regular expression query-replace on all files listed in the tag table. - **C-u M-x tags-query-replace** replaces only whole words.
		(**tags-loop-continue** &optional FIRST-TIME)	Continue last M-x tags-search or M-x tags-query-replace command. Used noninteractively with non-nil argument to begin such a command (the argument is passed to 'next-file', which see). Two variables control the processing we do on each file: the value of 'tags-loop-scan' is a form to be executed on each file to see if it is interesting (it returns non-nil if so) and 'tags-loop-operate' is a form to evaluate to operate on an interesting file. If the latter evaluates to nil, we exit; otherwise we scan the next file.

References — Tags

Topic & Link	Description	Notes
Learning GNU Emacs - Ch 9 - Computer Language Support		
Using CTags		
CTags - wikipedia	Lists various tags processing programs, including the various CTags and Etags (the emacs tags)	
CTags - A maintained ctags implementation https://ctags.io		
CTags - Universal-ctags Hacking Guide	Universal Ctags continues the development of the now-defunct Exuberant CTags. Universal CTags is maintained.	
Tag Tools pages		
ctags	help available in man page. in /usr/bin : restricted.	
etags	Comes with GNU emacs; info available in man page.	
ExuberantCTags	According to the EmacsWiki (https://www.emacswiki.org/emacs/ExuberantCtags) this supports more languages than etags	Apparently this project is no longer maintained; Universal CTags is a fork and is maintained.
Universal CTags	Homebrew has a tap for installing Universal CTags: https://github.com/universal-ctags/homebrew-universal-ctags	I installed universal ctags which is named ctags and placed inside /usr/local/bin (which is before /usr/bin where the original ctags is located. Homebrew removed the man page for the original ctags . It would have been better to have a different name for that universal cats (like uctags) but they did not do that. The ctags man page is now the page for universal ctags… Universal cat's has a mode for emacs. Also note that tags was not removed by the installation of Universal ctags. SO I renamed Universal ctags, which is a symlink in /usr/local/bin to **uctags**, so that I can still access the original ctags if needed. To access the original ctags man page use : "man -a ctags" this will open all ctags man pages one after the other (when one is closed) and after closing the universal ctags page, the original cat page is opened.
Hasktags		
Emacs and CTags		
Using CTags		
CTags - wikipedia	Lists various tags processing programs, including the various CTags and Etags (the emacs tags)	
CTags - A maintained ctags implementation https://ctags.io		
CTags - Universal-ctags Hacking Guide	Universal Ctags continues the development of the now-defunct Exuberant CTags. Universal CTags is maintained.	
CTag Tools		
ctags	help available in man page. in /usr/bin : restricted.	
etags	Comes with GNU emacs; info available in man page.	
ExuberantCTags	According to the EmacsWiki (https://www.emacswiki.org/emacs/ExuberantCtags) this supports more languages than etags. However, apparently this project is no longer maintained; Universal CTags is a fork and is maintained.	
Universal CTags	Homebrew has a tap for installing Universal CTags: https://github.com/universal-ctags/homebrew-universal-ctags	
Notes on installing Universal Ctags on a macOS system	On my macOS system, I installed universal ctags which has an executable that is named ctags and placed inside /usr/local/bin (which is before /usr/bin where the original ctags is located. • Homebrew removed the man page for the original ctags. I would have preferred hey used a different name for universal ctags (something like uctags) but they did not do that. The ctags man page is now the page for universal ctags… • Universal ctags has a mode for emacs. Also note that tags was not removed by the installation of Universal ctags. So I manually renamed Universal ctags, which is a symlink in /usr/local/bin to **uctags**, so that I can still access the original ctags if needed. To access the original ctags man page use : "man -a ctags" this will open all ctags man pages one after the other (when one is closed) and after closing the universal ctags page, the original cat page is opened.	
Using Tags with Erlang		
Etags with Erlang @ erlang.org	Describes how to use tags with Erlang source code and how to create the TAGS file.	
Hasktags		