↖ Topic Index

Keyboard Macros

Description	Keystroke	Function	Note
Keyboard Macros	Keyboard macros allow you to quickly record a sequence of keys and then re-execute that same sequence later. Any of the keys can be Emacs bound commands. This is a very useful tool to speed up editing.		
PEL Customization	PEL provides the pel-kbmacro customization group: use **<f11> k <f2>** to quickly access the customization buffer for the group. This holds the following user options: • **pel-kbmacro-prompts**: if **t** the keyboard macro recorder will prompt before overriding an existing keyboard macro. Off (nil) by default. PEL supports the following external packages that extend keyboard macro features: The **centimacro** external package PEL downloads, installs and activates it when the **pel-use-centimacro** user option is set to **t**. The **elmacro** external package. PEL downloads, installs and activates it when the **pel-use-elmacro** user option is set to **t**. The **emacros** external package. PEL downloads, installs and activates it when the **pel-use-emacros** user option is set to **t**.		
Open this PDF file. See also: **∑ Help/Info**	**<f11> k <f1>**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the **∑ Keyboard Macros** local PDF. With prefix argument (like **C-u** or **M--**) open the remote GitHub raw PDF instead. If **pel-flip-help-pdf-arg** is set it's the other way around.
∑ Customize PEL support for keyboard macros	• **<f11> k <f2>** • **<f11> k e <f2>** • **<f11> k l <f2>**	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL keyboard macro external package support: centimacro, emacros, elmacro. • If OTHER-WINDOW is non-nil (use **C-u**), display in other window.
∑ Customize Emacs support for keyboard macros	**<f11> k <f3>**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize the Emacs keyboard macro external package support: kmacro, centimacro.
∑ Customize emacros package	**<f11> k e <f3>**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize the Emacs keyboard macro external package support: emacros.
∑ Customize elmacro package	**<f11> k l <f3>**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize the Emacs keyboard macro external package support: elmacro.
Common C-x C-k prefix	Several of the keyboard macros commands share the same **C-x C-k** prefix. • Once you have typed the **C-x C-k** prefix of one of these commands, you can type just the last part for executing the others. • For example you could type **C-x C-k C-p C-k C-n C-k** to exit the previously defined keyboard macro and then execute the last defined keyboard macro. • Also since you can bind macros to single characters in the range [0-9] and [A-Z] these can also be executed inside that string of characters.		
Record & Play	Use **<f3>** and **<f4>** to record and play keyboard macros, instead of the older **C-x (** and **C-x)** bindings. These are easier to type and easier to use since **<f4>** is used both to stop recording and to execute the macro.		
Start Recording	• **<f3>** • **C-x (**	(**kmacro-start-macro-or-insert-counter** ARG) (**pel-kmacro-start-macro-or-insert-counter** ARG)	Record subsequent keyboard input, defining a keyboard macro. • The commands are recorded even as they are executed. • While already defining a macro (with a previous F3), typing F3 inserts the current value of the keyboard macro counter into the buffer, and increments the counter by 1. • See The Keyboard Macro Counter.
	• **C-u <f3>** executes the last macro then appends the keystrokes to its definition. • **C-u C-u <f3>** appends keys to the last defined macro without executing it. • **C-x (** is an older command binding. By setting the **pel-kbmacro-prompts** user-option to t, the command will prompt if a keyboard macro already exists and would be overwritten. • Use a negative argument (**M- -** or **C-_**) argument or numeric 0 to prevent this prompt and allow overwriting already defined macro.		
End Recording or call last macro	• **<f4>** • **C-x e** • **C-x)**	(**kmacro-end-or-call-macro** ARG &optional NO-REPEAT)	Ends macro recording done with **<f3>**. Typing **<f4>** again runs the last recorded macro. This is the most convenient way to replay a recently recorded macro. Typing **C-u <f4>** runs the *second* macro in the ring.
	• A prefix argument number N specified the number of times to execute the macro. ⚠ If N is 0, the macro will run forever until reaching and error or stopped with **C-g** (or **C-<BREAK>** on DOS/Windows)! During that time the display may not even be updated!! • **C-x)** is an older command binding. It only ends recording and cannot be used to replay the macro, unlike **<f4>** which can be used to replay.		
Execute macro at the head of the macro ring	**C-x C-k C-k**	(**kmacro-end-or-call-macro-repeat** ARG)	Use this instead of **<f4>** to end a macro definition or to execute the *current* macro (the one at the top of the keyboard macro ring). • It's advantageous if you want to execute another command with the **C-x C-k prefix** right after: you won't have to repeat the prefix, so you could type **C-k** again, **C-n** or **C-p** or one of the other commands with the same prefix.
Allow overwrite of recorded macro	**<f11> k k**	(**pel-forget-recorded-keyboard-macro**)	Forget that a keyboard macro was recorded by **<f3>**. • Does not delete the macro from the keyboard macro ring.
Apply macro to selected area (region)	**C-x C-k r**	(**apply-macro-to-region-lines** TOP BOTTOM &optional MACRO)	Apply last defined keyboard macro to each line of a region. • It does it line by line: moves point to the beginning of the line, then executes the macro.
Name & Save to reuse	Use this to create and use **several** keyboard macros. Use one of the following 2 quick methods to store and re-use keyboard macros: • Bind the latest keystroke macro defined to a single key ([0-9] or [A-Z]). Later invoke it with '**C-x C-k** *prefix*' , where *prefix* is that key. • Assign a name to the latest keyboard macro defined. Later invoke it with '**M-x** *name*', where *name* is the selected name.		
Bind the most recent defined macro	**C-x C-k b**	(**kmacro-bind-to-key** ARG)	Emacs will prompt. Use keys [0-9A-Z]. Emacs then binds the last defined keystroke macro to the corresponding command **C-x C-k 0** through **9** and capital **A** to **Z**. To re-run the macro: type **C-x C-k** followed by the character that identifies the macro. For example: **C-x C-k 0** would execute macro bound to 0. ⚠ These bindings **do not** persist after Emacs closes.
Name last defined macro	**C-x C-k n** *<name>*	(**kmacro-name-last-macro** SYMBOL)	The name can be any string as long as it does not conflict with the name of an existing function (in which case it won't be accepted). A good convention is to use underscores or period in the names since most of the emacs functions do not use them. For example, "km_" or "km." as prefix. To execute a name keyboard macro, use **M-x** *<name>* ⚠ The names **do not** persist after Emacs closes. If you want to retain the named keyboard macro convert it into Lisp code with **insert-kbd-macro** in a file and save that file.
Insert Lisp definition in buffer	**<f11> k i**	(**insert-kbd-macro** MACRONAME &optional KEYS)	Insert in buffer the definition of kbd macro MACRONAME, as Lisp code. MACRONAME should be a symbol. • Optional second arg KEYS means also record the keys it is on (this is the prefix argument, when calling interactively). • This Lisp code will, when executed, define the kbd macro with the same definition it has now. If you say to record the keys, the Lisp code will also rebind those keys to the macro. Only global key bindings are recorded since executing this Lisp code always makes global bindings. • To save a kbd macro, visit a file of Lisp code such as your '~/.emacs', use this command, and then save the file.
Name last defined macro		(**name-last-kbd-macro** SYMBOL)	An older implementation, similar to **kmacro-name-last-macro** but which does not put the 'kmacro property to the symbol. I suspect this will eventually go away or become an alias for the other one. Use **kmacro-name-last-macro** instead.

Description	Keystroke	Function	Note
Keyboard Macro Ring	The macro at the head of the macro ring can be executed with **<f4>** and **C-x C-k n** will name it. The maximum number of macros in the keyboard macro ring is determined by the customizable variable '***kmacro-ring-max***'.		
Show keyboard macro ring status	**<f11> k ?**	(**pel-kmacro-ring-show-status** &optional PRINT-IN-BUFFER)	Show the 'kmacro-ring' status information. • Print a message unless PRINT-IN-BUFFER is non-nil in which case it prints the information in a dedicated buffer.
Rotate the macro ring to the next (defined earlier) macro	**C-x C-k C-n**	(**kmacro-cycle-ring-next** &optional ARG)	Move to next keyboard macro in keyboard macro ring. • Displays the selected macro in the echo area. • The ARG parameter is unused. • You can continue to rotate the ring with a single **C-n** or **C-p** until the desired macro is at the head of the ring, and then execute it with a single **C-k**
Rotate the macro ring to the previous (defined later) macro	**C-x C-k C-p**	(**kmacro-cycle-ring-previous** &optional ARG)	Move to previous keyboard macro in keyboard macro ring. • Displays the selected macro in the echo area. • The ARG parameter is unused. • You can continue to rotate the ring with a single **C-n** or **C-p** until the desired macro is at the head of the ring, and then execute it with a single **C-k**
Delete current macro from Keyboard macro ring	**C-x C-k C-d**	(**kmacro-delete-ring-head** &optional ARG)	Delete current macro from keyboard macro ring. • The ARG parameter is unused.
Keyboard macro counter	A counter is associated with each keyboard macro. A macro can be defined to insert the integer value of the counter in the text. While defining the macro, you can type either <f3> or C-x C-k C-i to insert the counter. You can also define a way to format the counter : use **C-x C-k C-f** before defining the macro. With these facilities you can easily create a macro to number lines. For example, consider the following commands: • **C-x C-k C-f %02d** • **<f3> C-a <f3> . SPC <f4>** • That will create lines with: "00.", "01. ", "02. ", etc… By default the counter start at 0. If you want another value, initialize it to another value specify it as a numeric argument to <**f3**>.		
Insert current counter & increment by 1	**<f3>**	• (**kmacro-start-macro-or-insert-counter** ARG) • (**pel-kmacro-start-macro-or-insert-counter** ARG)	While defining a macro typing **<f3>** inserts the macro counter in the buffer.
Insert keyboard macro counter value in the buffer, then increment it by 1 (or ARG)	**C-x C-k C-i**	(**kmacro-insert-counter** ARG)	Insert current value of 'kmacro-counter', then increment it by ARG. • Interactively, ARG defaults to 1. • With **C-u,** insert the previous value of '***kmacro-counter***', and do not increment the current value. The previous value of the counter is the one it had before the last increment. • Can be typed while defining a macro, but also outside.
Set keyboard macro counter	**C-x C-k C-c**	(**kmacro-set-counter** ARG)	Set the value of '***kmacro-counter'*** to ARG, or prompt for value if no argument. With **C-u** prefix, reset counter to its value prior to this iteration of the macro.
Add prefix arg to the keyboard macro counter	**C-x C-k C-a**	(**kmacro-add-counter** ARG)	Add the value of numeric prefix arg (prompt if missing) to '***kmacro-counter***'. With **C-u**, restore previous counter value.
Specify the format for inserting the keyboard macro counter	**C-x C-k C-f**	(**kmacro-set-format** FORMAT)	Set the format of '***kmacro-counter***' to FORMAT. See formatting strings. This allows controlling the string inserted when the macro counter is inserted. The text provided should contain a format entry for **one** integer. The enclosing double quotes are ignored. ☞ The format string must be defined **before** defining the macro.
Macros with variations	You can force macro execution to stop, query for action. Use C-x q during macro definition to identify that point. When the macro runs, it will prompt at that point with **C-l**, **C-r**, **y**, **n**, **q**.		

Insert a query in a macro	**C-x q**	(**kbd-macro-query** FLAG)	Used when defining a macro to force a query when the macro will be executed.
			• With prefix argument (**C-u**), enters recursive edit, reading keyboard commands even within a kbd macro. You can give different commands each time the macro executes.
			• Without prefix argument, asks whether to continue running the macro. Your options are:
			  • **y** Finish this iteration normally and continue with the next.
			  • **n** Skip the rest of this iteration, and start the next.
			  • **RET** Stop the macro entirely right now.
			  • **C-l** Redisplay the screen, then ask again.
			  • **C-r** Enter recursive edit; ask again when you exit from that.
			Later during execution of that macro, use **C-M-c** to exit the recursive edit.

Editing macros	You can edit the content of a keyboard macro with the following commands. They format the macro definition in a buffer and enter a specialized mode to edit it. Type **C-c C-c** to complete the editing. ☞These work better in **graphics** mode; in terminal mode the cursor and meta keys are recorded as escape sequences and that's what is shown in the key lossage. It makes the macro actions more difficult to read.		
Edit the last defined keyboard macro	**C-x C-k C-e**	(**kmacro-edit-macro-repeat** &optional ARG)	Edit last keyboard macro.
Edit the last defined keyboard macro	**C-x C-k RET**	(**kmacro-edit-macro** &optional ARG)	As edit last keyboard macro, but without kmacro-repeat property.
Edit a previously defined macro name	**C-x C-k e** *<name>*	(**edit-kbd-macro** KEYS &optional PREFIX FINISH-HOOK STORE-HOOK)	Edit a keyboard macro. • At the prompt, type any key sequence which is bound to a keyboard macro. • Or, type **C-x e** or **RET** to edit the last keyboard macro, **C-h l** to edit the last 300 keystrokes as a keyboard macro, or **M-x** to edit a macro by its command name. • With a prefix argument, format the macro in a more concise way.
Edit the last 300 keystrokes as a keyboard macro	**C-x C-k l**	(**kmacro-edit-lossage**)	Edit most recent 300 keystrokes as a keyboard macro. No mouse click allowed in the last 300 events for this to work.
Stepwise macros editing	You can also interactively execute and edit a keyboard macro with the following command.		
Stepwise Replay/Edit	**C-x C-k SPC**	(**kmacro-step-edit-macro**)	Step edit and execute last keyboard macro.
		Executes the keyboard macro but before each command prompts the user for action, including: • **y** execute current command, go to next one • **n, d** skip & delete current command • **f** skip current command (don't delete) • **<tab>** execute current commands and all similar in sequence • **c** continue execution without further editing • **C-k** skip and deletes rest of macro, terminate editing and replace • **q, C-g** cancel editing, ignore any editing changes • **i** insert/execute following keys in macro up until **C-j** in macro • **I** insert/execute one key sequence in macro • **r** replace current command with read/executed keys in macro up until **C-j**, advance over inserted key sequence. • **R** read & execute key sequence, replace current command with it, advance over inserted key sequence. • **a** append & execute key sequence up until **C-j**. Append to end of macro & advance overt inserted key sequence. • **A** Same as 'a' but after appending stops editing & replace original macro.	

Description	Keystroke	Function	Note

centimacro
 	
Bind other keys to keyboard macros.
	📦 Requires the **centimacro** external package 🔄 PEL downloads, installs and activates it when the **pel-use-centimacro** user option is set to **t**.
	• Quick access to the pel-kmacro group is available via <**f11> k <f2>** .
	• The centimacro package provides the ability to create keyboard macros that are bound to keys other than **<f4>**.
	• To create a keyboard macro binding you use the **centi-assign** command, identify the key that will playback the keyboard macro, type the keys that constitute the keyboard macro and terminate it by the same playback key.
	• The bindings can override another key binding and the centimacro bindings can be deleted, restoring the original bindings by executing the **centi-restore-all** command.
	• The centimacro bindings are not persistent: they are not kept once Emacs is closed.
	• By default, centimacro maps the **<f5>** key to centi-assign. It's a problem for PEL as PEL uses that key as the repeat key.
	  • 🎛️ To solve that PEL issue, PEL provides the **pel-centi-assign-key** user option to identify the default key for centi-assign and reassigns the key on startup. By default it uses the **<C-f5>** key instead. But also not the other key binding below, which is always available with PEL.
	  • Therefore, when using PEL, just leave the centi-assign-key user option unchanged to **<f5>** and control the key binding with pel-centi-assign-key instead. PEL code restores the binding of **<f5>** to repeat and uses pel-centi-assign-key to specify the key binding used for centi-assign.
	☝ With PEL, the **<C-f5>** key binding becomes active only once you invoked a centimacro command through one of the the **<f11> k** keys.		

Record a keyboard macro bound to a specified key See also: ⌨**Keys - Fn**	• **<C-f5>** • **<f11> k M-=**	(**centi-assign**)	Record a keyboard macro for a specified KEY. • Read a KEY and start recording a keyboard macro for it. • Pressing KEY again stops recording and assigns the macro to KEY. • Aborts if KEY belongs to a minor mode. • Use 'centi-summary' to list bound macros. • Use 'centi-restore-all' to un-bind macros and restore the old key bindings.
	☛See ⌨**Keys - Fn** for a list of function key bindings that can be used with PEL without disrupting PEL key bindings. The available keys include: • **<C-f7>** , **<M-f7>** , **<C-M-f7>** , **<M-S-f7>** • **<C-f8>** , **<M-f8>** , **<C-M-f8>** , **<M-S-f8>** • **<C-f9>** , **<M-f9>** , **<C-M-f9>** , **<M-S-f9>** • **<C-f11>** , **<M-f11>** , **<C-M-f11>** • **<C-f12>** , , **<C-M-f12>** • Note that **<M-S-f9>** is used to display Emacs startup benchmark, but this key can easily be spared. The **<M-f12>**, **<M-S-f11>** and **<M-S-f12>** cannot be used because PEL uses **<M-f11>** and **<M-f12>** as prefix keys. • No combination mixing function key and control key can be used in terminal mode, but they can be used in graphics mode.		
Show all keyboard macros created by centi-assign	**<f11> k M-?**	(**centi-summary**)	Show a summary of bound macros created by centi-assign.
Unbind all keyboard macros created by centi-assign	**<f11> k M-DEL**	(**centi-restore-all**)	Unbind all bound macros created by centi-assign, restoring the previous key bindings.
elmacro Generate Emacs Lisp code from recorded macros or command history	📦 Requires the **elmacro** external package. 🔄 PEL downloads, installs and activates it when the **pel-use-elmacro** user option is set to **t**. 🎛️ Quick access to the pel-kmacro group is available via <**f11> k <f2>** . The elmacro customization group that provides a set of user options that can help the generation of Emacs Lisp code. • The elmacro package provide the ability to generate Emacs Lisp commands from the recorded keyboard macros. • Once the elmacro-mode is active, you can show and save generated Emacs Lisp code for keyboard macros recorded while elmaco-mode is active. This help you see what is being executed by the keyboard macro, specially in terminal mode. You can also save the buffer in a file if you want to use the command permanently. PEL provides the following key bindings. The key binding for elmacro-mode is available when **pel-use-elmacro** is **t**. The other keys are made available only when the elmacro-mode has been activated once.		
Toggle activity recording	**<f11> k l l**	(**elmacro-mode** &optional ARG)	Toggle emacs activity recording (elmacro mode). • With a prefix argument ARG, enable elmacro mode if ARG is positive, and disable it otherwise.
Generate Emacs Lisp code for the last executed commands	**<f11> k l c**	(**elmacro-show-last-commands** &optional COUNT)	Show the last COUNT commands as Emacs lisp code. • COUNT default is '**elmacro-show-last-commands-default**' user option. • You can also modify this number by using a numeric prefix argument or by using the universal argument, in which case it'll ask for how many in the minibuffer. • The command writes the code in a * elmacro - last-X-commands * buffer. • This is basically a better version of 'kmacro-edit-lossage'.
Generate Emacs Lisp code for the commands executed by the last invoked keyboard macro.	**<f11> k l m**	(**elmacro-show-last-macro** NAME)	Show the last macro as emacs lisp with NAME. • Prompts for a Emacs Lisp function name. • It opens a * elmacro -last-macro * buffer and write the Emacs Lisp there. ⚠️ Check the generated code, there's often several lines after the (interactive) line that represent past commands that you will not want. Just erase the lines.
Clear the list of recorded commands	**<f11> k l DEL**	(**elmacro-clear-command-history**)	Clear the list of recorded commands.
emacros Store keyboard macros in files associated with major mode and location of edited files.	📦 Requires the **emacros** external package. 🔄 PEL downloads, installs and activates it when the **pel-use-emacros** user option is set to **t**. • Quick access to the pel-kmacro group is available via <**f11> k <f2>** . • With the **emacros** package you can store recording of keyboard macros associated to names. The scope of the macros loaded from files is restricted to the mode of the buffer from which you load them. You can also store them in global and local files. The definitions stored in global files are accessible for all buffer of the specified mode. The definitions stored in local files are accessible while editing files in the same directory. 🎛️ emacros customization group allows selection of: • **emacros-global-dirpath** : directory where global macros definitions are stored. • **emacros-subdir-name** : specifies whether local emacros definition files are stored inside a sub-directory of specified name (defaults to ".emacros") or not. The definition of the emacros are stored in a file specific to a major mode with a name that identifies the major mode.		
Load keyboard macros definitions for the current mode from files.	**<f11> k e L**	(**emacros-load-macros**)	Load existing keyboard macros for the current mode. • Attempt to load from the global directory and the current local directory if the files exist for the current mode. • Remember the loaded directories inside 'emacros--already-loaded-dirs'.
List names of recorded key macros	**<f11> k e /**	(**emacros-show-macro-names** ARG)	Display the names of the kbd-macros that are currently defined. • With prefix ARG, display macro names in a single column instead of the usual two column format.
Show recorded key macro names and their code	**<f11> k e ?**	(**emacros-show-macros**)	Displays the kbd-macros that are currently defined.
• **define emacros**	Use the following commands to name the last defined keyboard macro		
Assigns a name for the last keyboard macro created with <f3> and <f4>	**<f11> k e =**	(**emacros-name-last-kbd-macro-add** &optional ARG)	Assigns a name to the last keyboard macro defined. • Accepts letters and digits as well as "_" and "-". • Requires at least one non-numerical character. • Prompts for a choice between local and global saving. • With ARG, prompt the user for the name of a file to save to. Default is the last location that was saved or moved to in the current buffer.
• **execute emacros**	Use the following commands to execute a named macro that has already been defined or loaded. • These commands are only available once one of the first 4 commands above has been executed.		
Execute keyboard macro by name	**C-c x**	(**emacros-auto-execute-named-macro**)	Prompts for the name of a macro and execute when a match has been found. Accepts letters and digits as well as "_" and "-". Backspace acts normally, **C-g** exits, **RET** does rudimentary completion. Default is the most recently saved, inserted, or manipulated macro in the current buffer.

Description	Keystroke	Function	Note
Execute keyboard macro by name read from mini buffer. Supports completion.	• `C-c e` • `<f11> <f4>` • `<f11> k e e`	(**emacros-execute-named-macro**)	Prompts for the name of a macro and execute it. Does completion. Default is the most recently saved, inserted, or manipulated macro in the current buffer.
• **manage emacros**	Use the following commands to reset the macros to their original meaning, delete, and rename macros. • The macros can also be moved from local or global scope or vice versa. • These commands are only available once one of the first 4 commands above has been executed.		
Erases all keyboard macros and reload only the ones for the current mode.	`<f11> k e R`	(**emacros-refresh-macros**)	Erases all macros and then reloads for current buffer. • When called in a buffer, this function produces, as far as kbd-macros are concerned, the same situation as if Emacs had just been started and the current file read from the file system.
Delete a recorded keyboard macro from file.	`<f11> k e DEL`	(**emacros-remove-macro**)	Remove macro from current session and from current macro files. The macroname defaults to the name of the most recently saved, inserted, or manipulated macro in the current buffer.
Rename a keyboard macro	`<f11> k e r`	(**emacros-rename-macro**)	Renames macro in macrofile(s) and in current session. • Prompts for an existing name of a keyboard macro and a new name to replace it. • Default for the old name is the name of the most recently named, inserted, or manipulated macro in the current buffer.
Move definition of keyboard macro between local and global files.	`<f11> k e m`	(**emacros-move-macro**)	Move macro from local to global macro file or vice versa. • Prompts for the name of a keyboard macro and a choice between "from local" and "from global", then moves the definition of the macro from the current local macro file to the global one or vice versa. • Default is the name of the most recently saved, inserted, or manipulated macro in the current buffer.

Keyboard Macros — References

Topic & Link	Description
Emacs Keyboard Macros	
GNU Emacs manual - Keyboard Macros	
GNU Emacs manual - Keyboard Macros - Basic Usage	
GNU Emacs - naming, saving macros	
GNU Emacs - macros with variations	
GNU Emacs - keyboard macro counter	
GNU Emacs- editing keyboard macro	
Emacs Wiki - Keyboard Macros	
Emacs Wiki - Keyboard Macros Tricks	
Code example 2 - adding more keys to run macros	
Introduction - very basic	
Example @ Ergoemacs	
Stepwise Editing a Keyboard Macro	
Building macros with pause, prompts, etc..	
Define, name and run keyboard macros	
Extra Notes	2 elisp files implement these functions: macros.el and kmacro.el . The latter appears to be newer and provides more functionality. For example: it provides the (**kmacro-name-last-macro** SYMBOL) similar to the (**name-last-kbd-macro** SYMBOL). It's almost the same code except that it puts the macro property to the symbol, which the older function does not do. I would think that these 2 files should be merged in future versions of Emacs to reduce code bloat.
External Packages	The following external packages extend keyboard macro support
centimacro	Provides the ability to record keyboard macros assigned to another key (instead of F4).
emacros	Provides the ability to store keyboard macros inside files.
elmacro	Provides ability to generate Emacs Lisp code from recorded macro or last executed commands.