Spell Checking

Description	Keystroke	Function	Note
Spell Checking in Emacs	Spell checking is not performed by Emacs itself; Emacs uses an external process for that. On Unix-like system it's either aspell or ispell; aspell is preferred because it is faster and is more modern. The ispell process is not able to handle UTF-8 documents, but the aspell process does. These programs are not bundled with Emacs; you may have to install these programs independently.		
Using Ispell	Once Ispell (or Flyspell) is activated the Ispell commands are available. If Flyspell mode is active the following 2 key bindings are instead bound to the Flyspell functions (see the section below on how to activate Flyspell). But even then all other ispell commands are available.		
Ispell - complete a word	• `M-<tab>` • `C-M-i` • `C-.`	(**ispell-complete-word** &optional INTERIOR-FRAG)	Try to complete the word before or at point. • If optional INTERIOR-FRAG is non-nil, then the word may be a character sequence inside of a word. • Standard ispell choices are then available. • Notes: 1. this also works in Org-Mode, even though the binding is not available. 2. If flyspell is activated, the keys are bound to flyspell-auto-correct-word (see below).
Ispell - Check a single word	`M-$`	(**ispell-word** &optional FOLLOWING QUIETLY CONTINUE REGION)	Check spelling of word under or before the cursor. Several options are available at that moment: see the following "**Ispell operation**" lines below for the single line command that can then be used. A list of replacement is shown in a buffer. Use the letter **i** to include the word into the dictionary.
Interactive spell checking	The following commands perform interactive spell checking. The commands spell check a portion of text and stops at the first misspelled word, opening a *Choices* buffer to prompt for replacement. Your response can be one of several characters, described in the row below. The related commands are shown in the following rows.		
Ispell *Choices* buffer keys **Response characters.**	• `?` : Ispell prints more options in the minibuffer. These extra options are over the correction characters shown in *Choices* • *digit* : Replace the word with the one identified by the choice digit. • `i` : insert the "*misspelled*" word inside the private dictionary file located in ~/.ispell_*<language>* • `m` : same as i but you can also specify dictionary completion information. • `l word` : look in the dictionary for words that match *word*. These words become the new list of replacement proposals. You can use '*' in *word* as wildcards. • `u` : uncapitalized the "*misspelled*" word inside the private dictionary file located in ~/.ispell_*<language>* • `r` : prompt for the correct spelling, replace this instance (the replacement string is then accepted later in the text) • `R` : query replace in buffer • `<space>` : do not replace, skip • `a` : accept word, treat it as correct; do not replace, skip over this word now and later in all buffer for this session • `A` : accept word, treat it as correct; do not replace, skip over this word now and after in this buffer only for this session. • `X` : quit interactive spell-checking, leaving point at the word that was being checked. Resume checking afterward with **C-u M-$**. • `x` : quit interactive spell-checking and move point back to where it was when you started spell-checking. • `q` : quit interactive spell-checking and kill the Ispell process. • `C-q` : Stop Ispell. When this is used, it is possible to resume Ispell later with **C-u M-$** or via the menu.		
Ispell - spell check buffer or region	`<f11> $ .`	(**ispell)**	Interactively check a region or buffer for spelling errors. • If 'transient-mark-mode' is on, and a region is active, spell-check that region. Otherwise spell-check the buffer.
Ispell - spell check buffer	`<f11> $ b`	(**ispell-buffer**)	Check the current buffer for spelling errors interactively. Disregard presence of region.
Ispell - spell-check region	`<f11> $ r`	(**ispell-region** REG-START REG-END &optional RECHECKP SHIFT)	Interactively check a region for spelling errors.
Ispell - spell-check email body	`<f11> $ m`	(**ispell-message**)	Check the spelling of a mail message or news post. • Don't check spelling of message headers except the Subject field. • Don't check included messages. • To abort spell checking of a message region and send the message anyway, use the 'x' command. (Any subsequent regions will be checked.) • The 'X' command aborts sending the message so that you can edit the buffer.
Ispell - spell-check comment and strings	`<f11> $ ;`	(**ispell-comments-and-strings**)	Check comments and strings in the current buffer for spelling errors.
Ispell - continue spell checking	`C-u M-$`	(**ispell-continue**)	Continue a halted spelling session beginning with the current word.
Ispell Process Control			
Ispell - kill the ispell process	`<f11> $ K`	(**ispell-kill-ispell** &optional NO-ERROR CLEAR)	Kill current Ispell process (so that you may start a fresh one). The spell check program runs as a background task connected via a pipe. It's not taking much CPU when no spelling is done, so it's normally not necessary to kill it; you can leave it running. However, it may become necessary to kill it when you want to change the dictionary or want to reduce the overhead.
Change Language Dictionary	`<f11> $ D`	(**ispell-change-dictionary** DICT &optional ARG)	Change to dictionary DICT for ispell/aspell. With a prefix arg, set it "globally", for all buffers. Without a prefix arg, set it "locally", just for this buffer. By just answering RET you can find out what the current dictionary is.

Activating Flyspell	Flyspell is a minor mode that performs automatic spell-checking.
	Flyspell can be activated without having to activate ispell, even though several of the ispell customizations also affect Flyspell. However ispell must be installed. Flyspell processes text continuously, just like a word processor, and it highlight misspelled words.
	
	☞ It's best to activate Flyspell mode for text buffers and Flyspell Prog mode for programming language buffers with hooks using the following code:
	`(add-hook 'text-mode-hook 'flyspell-mode)`
	`(add-hook 'prog-mode-hook 'flyspell-prog-mode)`
	
	The spell check command bindings are only available when Flyspell (or ispell) mode is active.
	
	⚠ A 3-button mouse is needed for Flyspell to access the pop-up menu of provided replacements suggestions, and the pop-ip menu does not work in terminal mode, unless code is used to fix this problem. Example of this code is shown in the FlySpell page of EmacsWiki. The PEL package incorporates that code and activates it in terminal mode using the following code:
	`(when (not (display-graphic-p))`
	`(eval-after-load "flyspell"`
	`'(progn`
	`(fset 'flyspell-emacs-popup 'pel-spell-flyspell-emacs-popup-textual))))`		

Description	Keystroke	Function	Note
Enter/Leave Flyspell mode	`<f11> $ F`	(**flyspell-mode** &optional ARG)	Toggles the use of Flyspell mode. • Mode line shows “Fly” when Flyspell mode is active. • Flyspell mode works like word processors; misspelled words are highlighted. • Use Flyspell Prog mode for code; Flyspell processes all text. • With a prefix argument ARG, enable Flyspell mode if ARG is positive, and disable it otherwise. • Flyspell mode is a buffer-local minor mode. When enabled, it spawns a single ispell/aspell process and checks each word. The default flyspell behavior is to highlight incorrect words. ☞ If a hook already activate Flyspell you probably won’t need this command ☞ unless you want to disable it. If you do that in a buffer for programming ☞ language you probably will want to re-activate it using the next command.
Enter Flyspell Prog mode	`<f11> $ p`	(**flyspell-prog-mode**)	Turn on Flyspell prog mode: turn on Flyspell but restricts it to comments and strings, do not spell check source code itself. Highlight misspellings only in comments or strings. ☝ If a hook activates Flyspell Prog mode, you won’t need this command. ⚠ Note that the command always enables the mode, it does not toggle it. If you want to turn spell checking off, you must use the flyspell-mode command. To re-enable Flyspell Prog mode you then use this one.
Using Flyspell	With Flyspell mode activated, the following key bindings are active and can be used to fix spelling of misspelled or incomplete words.		
Flyspell - complete a word	• `M-<tab>` • `C-M-i` • `C-.`	(**flyspell-auto-correct-word**)	Correct the current word in place. • This command proposes various successive corrections for the current word. If invoked repeatedly on the same position, it cycles through the possible corrections of the current word. • In most cases this is much faster than using M-$ which always proposes choices.
Ispell - Check a single word	`M-$`	(**ispell-word** &optional FOLLOWING QUIETLY CONTINUE REGION)	Check spelling of word under or before the cursor. • Opens a *Choices* buffer showing all available corrections/suggestions, similar to the way ispell does it. • Several options are available at that moment: see the following “**Ispell operation**” lines in the above table for the single line commands that can then be used.
Flyspell - correct word	• `C-c $` • **`4r`**	(**flyspell-correct-word-before-point** &optional EVENT OPOINT)	Pop up a menu of possible corrections for misspelled word before point. 📦🛂 With PEL, the **`4r`** key-chord is also available when key-chord is available and active. See ∑ Key-Chords. ⚠ To activate this in terminal mode you must write some code. See the note in the “Activating Flyspell” row above.
Using Flyspell when not activated	The following command can be used even when Flyspell mode is not activated.		
Check all text in buffer		(**flyspell-buffer**)	Flyspell whole buffer. This command is marginally useful. You can use it when Flyspell mode is not active to highlight misspelled words in the buffer. Since the other Flyspell commands bindings are not available you have to fix spelling of the words manually and re-run the command. A better way is to simply activate Flyspell and use the commands. To use this command, type: **M-x flyspell-buffer**
Querying Information	The following commands print status information inside the mini-buffer about the spell check programs being used.		
Ispell - check version	`<f11> $ v`	(**ispell-check-version** &optional INTERACTIVEP)	Display Ispell process version as well as the version of ispell.el
Show information about used spell program	`<f11> $ ?`	(**pel-spell-show-use**)	Display what spell checking program is being used, its version, the status of the spell modes and the dictionary used.

PEL: Selecting/Configuring the Spell Checker Program	You can identify the spelling program used by PEL by scheduling the call of the **pel-spell-init** function inside your init file. This is done by writing code like the following, which selects the program ‘aspell’ and specify the location of the personnel dictionary:
	
	`(eval-after-load "ispell"`
	`  '(pel-spell-init “aspell" "~/.emacs.d/.ispell"))`		

Set the spell checker program		(**pel-spell-init** SPELL-PROGRAM-NAME &optional SPELL-PATH PERSONAL-DICTIONARY)	Initialize spell checking. • Use specified SPELL-PROGRAM-NAME as the spell checking process. A string. It must be a Ispell compatible program, like: "ispell", "aspell", "hunspell", "enchant". • Specify the directory where the program is found in SPELL-PATH when that program is not already found in the ‘exec-path’. • To be used, the value must be a string. • If no path is needed use nil. • Any other type raises an error. • Optionally identify the PERSONAL-DICTIONARY to use. Activates flyspell-mode and fix issues in terminal mode. When running in terminal mode, the function modifies ‘**flyspell-emacs-popup**’ with ‘**pel-spell-flyspell-emacs-popup-textual**’ to allow the flyspell pop-up menu to work in terminal mode.

Spell Checking — References

Topic & link	Description
Make ispell automatically clear minibuffer when replacing word	
How can I change the language in Emacs when using ispell?	
Enabling spell-checking in comments	
in Emacs flyspell-mode, how to add new word to dictionary?	
Aspell Windows @ EmacsWiki	In Setup for 64-bit Windows 7
GNU Aspell (Win32 version)	