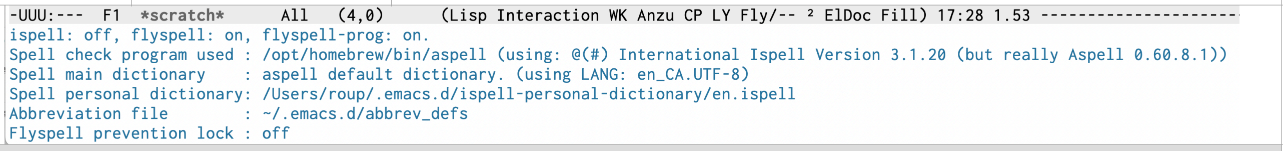
↖ Topic Index

Spell Checking in Emacs

Description	Keystroke	Function	Note
Spell Checking ○ Help & Customization • Spell checker programs • Query version and setup • Spell process control • Manual word spell check ○ Spell checking of text • Automatic spell checking	Emacs support spell checking inside text files and inside source code comments and docstrings! Two main modes of operation are supported and both are built-in Emacs: • **ispell**: a mode where you request an explicit spell-check verification of a word or an area of the current buffer. • **flyspell**: an active mode that runs in the background and detects spelling error on the fly, highlighting errors. • There is also a program-mode flyspell which activates automatic spell check of source code comments and docstrings. The actual spell checking is performed by a ispell-compatible spell checking process: **ispell**, **aspell**, **hunspell**, or **enchant** programs are supported. • **ispell** does not support UTF-8 encoding. It was the first spell checking program used by Emacs. Prefer aspell to ispell. • **aspell** is preferred to ispell because it is more modern (it supports UTF-8), faster that ispell and hunspell, and quite good for suggestions. • **hunspell** is actively maintained, used by several open source project. It support UTF-8 as well. • **enchant** is a front-end that attempt to unify spell-checkers. It supports aspell, hunspell and several others, broadening the choice further. ⚠️These programs are not bundled with Emacs; you may have to install these programs independently.		
Last updated on:	2025-12-19		
Open this PDF file. See also: **Help/Info**	`<f11> $ <f1>`	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the **Spell Checking** 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.
Customize PEL spell checking control	`<f11> $ <f2>`	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL support for: spell checking. Identify which major modes will automatically activate either flyspell-mode or flyspell-prog-mode. • If OTHER-WINDOW is non-nil (use **`C-u)`**, display in other window.
Customize Emacs spell checking control	`<f11> $ <f3>`	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs spelling support. Opens the following customization groups: ispell, flyspell.
Using Emacs Ispell	Once Ispell (or Flyspell) is activated the Ispell commands are available. But you must first select the spell checking program:		
Select spell checker program • **ispell man page** • **ispell file format** • **aspell man page** • **hunspell man page** • **enchant man page**	**Without PEL:** See also: **Customize** • Identify the full path of the spell checking program in the **ispell-program-name** user-option. **When using PEL:** • Set the name only of the spell checking program accessible via PATH in the **pel-spell-check-tool** user-option. • PEL sets **ispell-program-name** from it. • Use **`<f11> $ <f2>`** key sequence to gain access to the pel-pkg-for-spelling customization group where you can set **pel-spell-check-tool.** • Use **`<f11> $ <f3>`** to open the customization buffer of Ispell and Flyspell. • For the changes to take effect, save the changes and execute pel-init (with **`M-x pel-init`**) or restart Emacs.		
Select Spell Checking dictionaries • **Base dictionary** • **Personal dictionary** ⚠️ Use the **same** natural language **in both** dictionaries!	Emacs ispell supports two different dictionary at all times: the base dictionary and the personal dictionary: 1. The **base dictionary** is identified by either **ispell-dictionary** or by **ispell-local-dictionary** user-options. • The **ispell-local-dictionary**, if defined, takes precedence. • Normally you define **ispell-dictionary** user-option only. You can override it with a file-local definition of ispell-local-dictionary. • See **File/Directory Variables**. • Both base dictionaries are identified by their natural language code name. • The list of available dictionaries depend on the spell checking tool you are using. Execute **ispell-change-dictionary** to see the list. • See below. 2. The **personal dictionary**, identified by the **ispell-personal-dictionary** user-option, but PEL uses **pel-spell-personal-dictionary-directory**. • This is identified by its filename with a complete path. • The file is created when you add a word to your personal dictionary during spell checking operation. • The name of the natural language is written on the first line of the dictionary file. It is best to identify the language name in the file name. • ⚠️ Spell checking done with aspell will fail if the natural language used by the base dictionary differs from the one inside the personal dictionary!		
	• PEL sets the ispell-personal-dictionary from **pel-spell-personal-dictionary-directory** and the language code selected by **pel-spell-change-dictionary** command. Please select the same language code as the main one. This should avoid the mismatch problem. On startup Emacs checks the **LANG** environment variable to identify the default natural language and the base dictionary. • PEL 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.		
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 spell checking programs and status	• `<f11> $ ?` • `<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. It displays something like what is shown below:



```
-UUU:---  F1  *scratch*      All   (4,0)      (Lisp Interaction WK Anzu CP LY Fly/-- ² ElDoc Fill) 17:28 1.53 ---------------------
ispell: off, flyspell: on, flyspell-prog: on.
Spell check program used : /opt/homebrew/bin/aspell (using: @(#) International Ispell Version 3.1.20 (but really Aspell 0.60.8.1))
Spell main dictionary    : aspell default dictionary. (using LANG: en_CA.UTF-8)
Spell personal dictionary: /Users/roup/.emacs.d/ispell-personal-dictionary/en.ispell
Abbreviation file        : ~/.emacs.d/abbrev_defs
Flyspell prevention lock : off
```

Description	Keystroke	Function	Note
• **Process Control** **Change Language Dictionary**	`<f11> $ D`	(**pel-spell-change-dictionary** DICT &optional ARG)	Change to dictionary. Prompt for language code for the main dictionary and for the personal one. • By default set the dictionary for the current buffer (locally). • With a prefix arg (**`C-u`** or **`M--`** or something else), set it "globally", for all buffers.
⚠️ Use the **same** natural language **in both** dictionaries!	💻 PEL implements pel-spell-change-dict. It calls the **ispell-change-dict** to perform the change but also prompts and change the personal dictionary. 👆 The first prompt supports completion: hit **`TAB`** to see the complete list of available dictionaries. Type **`RET`** to only see current dictionary. ⚠️ The list of available dictionaries is not alway correct. It may include names that are not available. Check the available list with the spell checker tool on the command line: • **`aspell dump dicts`** lists the name of the dictionaries available to aspell. • **`hunspell -D`** lists the names of dictionaries available to hunspell. ⚠️ If you use a personal dictionary, it **must** use the same natural language as the base dictionary you select with this command, otherwise the next spell checking operation may fail: • Aspell will fail, issuing a message like this: `Error: Expected language "fr" but got "en".` • The above error occurs when the base dictionary language code is "fr" but the personal dictionary has word in the language code "en".		
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, normally blocked.
	ispell kill and restart it when changing dictionary. You may only need this when debugging problems with dictionary.		
• **Manual spell check** **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. • this also works in Org-Mode, even though the binding is not available. • 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.
Fix spelling mistake before point • **Add old->new in the abbreviation table** See also: **Abbreviations**	• `<f11> a $` • `<f11> M-$`	(**pel-ispell-word-then-abbrev** &optional LOCALLY)	Fix spelling mistake in text before point and remember the change in abbreviations for auto fix in future. 👆A similar operation is possible with flyspell. See **flyspell-auto-correct-word.**
	• This create an 'abbrev' abbreviation for the spelling mistake. Store the abbreviation globally unless the LOCALLY argument is non-nil. • If there's nothing wrong with the word at point, keep looking for a typo until the beginning of buffer. You can skip typos you don't want to fix with '**`SPC`**', and you can abort completely with '**`C-g`**'.		

Description	Keystroke	Function	Note
• **Interactive spell checking of text**	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.

Automatic Spell Checking: Flyspell
Activating Flyspell
See also: Σ **Customize**
	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)`
	☝🎛 PEL provides 2 user options that identify the modes where flyspell-mode and flyspell-prog-mode are automatically activated: **pel-modes-activating-flyspell-mode** and **pel-modes-activating-flyspell-prog-mode**. PEL code comes with defaults, activating flyspell and flyspell-prog modes for several major modes.
	• The **pel-spell-prevent-flyspell** user-option can also be set to prevent automatic activation of flyspell-mode and flyspell-prog-mode.
	  • Turned off by default to all activation of automatic spell checking.
	  • Turn it on if you want to reduce CPU load or when you want to debug spell checking code.
	  • Toggle unsaved value by typing **`<f11> $ M-f`** to invoke the **pel-spell-toggle-prevent-flyspell** command.
	• Use **`<f11> $ <f2>`** to quickly open the PEL Spelling customization group and configure these 2 user-options to add or remove groups. Then ‘Apply and Save’ and restart Emacs for the changes to take effect.
	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))))`		

Enter/Leave Flyspell mode	`<f11> $ F`	(**flyspell-mode** &optional ARG)	Toggles the use of Flyspell mode. • With a prefix argument ARG, enable Flyspell mode if ARG is positive, and disable it otherwise.
		• Mode line shows “Fly” when Flyspell mode is active. • Flyspell mode works like word processors; misspelled words are highlighted (by default). • Use Flyspell Prog mode for code to spell check only comments and strings. Flyspell processes all text. • Flyspell mode is a buffer-local minor mode. When enabled, it spawns a single ispell-compatible process. ☞ Use hooks to activate Flyspell automatically in major modes. PEL provides user-options for that. See above. ☞Use this command to toggle activation of Flyspell dynamically.	
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.
Toggle ability to automatically activate Flyspell-mode and Flyspell-prog-mode	`<f11> $ M-f`	(pel-spell-toggle-prevent-flyspell)	Toggle lock preventing flyspell-mode and flyspell-prog-mode activation. • By default, PEL activates flyspell-mode when Emacs opens a buffer in one of the major modes identified by the **pel-modes-activating-flyspell-mode** user-option, and activates flyspel-prog-mode when Emacs opens a buffer in one of the major modes identified by the **pel-modes-activating-flyspell-prog-mode** user-option. • This automatic activation is disabled when the **pel-spell-prevent-flyspell** user-option is turned on by this command and re-activated when is is toggled back off. • Use this command to prevent automatic activation of the automatic spell checking. This can help in various situations: you may want to reduce computation overhead or debug the spell checking mechanism.
Using Flyspell	With Flyspell mode activated, the following key bindings are active and can be used to fix spelling of misspelled or incomplete words.		
Auto correct previous word See Also: Σ **Cursor**	• `C-;` • `<f11> \`	(**flyspell-auto-correct-previous-word** POSITION)	Auto correct the first misspelled word that occurs before point. ⚠ Both iEdit and Flyspell use the C-; key as their default binding. PEL detects and reports that situation. If you see this warning modify the binding of one of the two user options.
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. ☝🎛If you want to include flyspell corrections inside the abbreviation table to automatically correct future typos you can modify the following flyspell user-options: • **flyspell-abbrev-p** : set it to t to automatically store flyspell corrections in local abbrev table. • **flyspell-use-global-abbrev-table-p** : set it to t to have it store in the global abbrev table instead.

Description	**Keystroke**	**Function**	**Note**
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 See also: • **∑Highlight** • **∑ Input Method** • **∑ Key-Chords**	• **`C-c $`** • **`<f11> $ $`** • **`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 a available when **pel-use-key-chord** user-option is t. • ⚠ Key-chord may not work properly when a different input-method is used. • ⚠ To activate this in terminal mode you must write some code. See the note in the “Activating Flyspell” row above. • ⚠ **fci-mode** interferes with pop-up menu displays in terminal-mode, at least with the one used by **flyspell-correct-word-before-point:** the menu lines become all jagged, they do not line up vertically. The problem does not affect Emacs running in graphics mode.
Using Flyspell when not activated	The following command can be used even when Flyspell mode is not activated.		
Check all text in buffer	`M-x flyspell-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.

Identify Ispell dictionary inside the text file:
• **Select a dictionary for the file using specialized text lines:**
	To select a local dictionary, use the following, followed by the language code on the same line:
	` Local IspellDict:`
	To select a local personal dictionary, use the following, followed by language code on the same line:
	` Local IspellPersDict:`		

• **Or use file local variables:**	`;; Local Variables:`
	`;; mode: emacs-lisp`
	`;; comment-column: 40`
	`;; ispell-check-comments: exclusive`
	`;; ispell-local-dictionary: "american"`
	`;; End:`		

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?	
GNU Aspell - latest version: 0.60.8	Aspell is a very good spell checking program and library. Unfortunately maintenance has severely slowed down. See: • **Aspell and Hunspell: The Tale of Two Spell Checkers**, by Sumit Khanna, Sep 27, 2016.
	• **GNU Aspell @ GNU** • **GNU Aspell @ Github** • **GNU Aspell @ Wikipedia**
Aspell 0.61 Manual	The latest version of the Aspell manual as of Nov 2021. Formatting is not as nice as the manual for version 0.60.9
Gnu Apell 0.60.9 Manual	The manual of the version currently available under Homebrew (as of Nov. 2021).
GNU Aspell - Mailing Lists	The place to get support. The following lists are available: • **aspell-announce** - **archives** • **aspell-devel** - **archives** • **aspell-user** - **archives**
Aspell Dictionaries	• **Aspell dictionary files @ GNU**, list organized by the ISO 639-1 2-letter language codes. Files are .tar.bz2 • Note however that aspell dictionary files are environment dependent compiled files. Read the section on aspell dictionary files listed below first. • On macOS use homebrew to install aspell. it also installs the dictionary files.
Aspell directory files **See:** • **Aspell Manual - Working With Dictionaries**	To list aspell configuration, use the following command: **`aspell config`** • This lists all aspell configuration information, including the **`data-dir`** that identifies the **location of the aspell dictionary files**. • On a macOS system with aspell installed with homebrew, the dictionary files are stored inside the following directory: /usr/local/Cellar/aspell/0.60.8/lib/aspell-0.60 File types: • `.alias`: aspell dictionary alias name, a list of aspell commands identifying another .rws or .multi file. • `.amf` : aspell mode filter control file • `.cmap` : aspell character map file • `.cset` : aspell character set data file • `.dat` : **language data file**, uses the same format as aspell configuration file. • The *-phonet.dat files are the *soundlike* files used for **phonetic comparisons**. • The *-affix.dat files are **affix compression** files. • `.info` : aspell filter option files • `.kbd` : keyboard layout files (identifies side-by-side keys that may cause mis-typing). • `.multi`: multi-dictionary compound instructions which refer to multiple .rws files. • `.rws` : compiled dictionary platform dependent file. Created by the **aspell create master** command.

Testing aspell on the command line with the available dictionaries:
 	
Aspell produces better results than hunspell:
Note that the aspell results for the French language is superior to what hunspell is able to detect (see the results for the same test run with hunspell below).
	Testing in English:
	`> echo htink | aspell -a --sug-mode=ultra --lang=en_US`
	`@(#) International Ispell Version 3.1.20 (but really Aspell 0.60.8)`
	`& htink 4 0: think, stink, ht ink, ht-ink`
	
	Test en français:
	`> echo francais | aspell -a --sug-mode=ultra --lang=fr_CA`
	`@(#) International Ispell Version 3.1.20 (but really Aspell 0.60.8)`
	`& francais 7 0: français, française, fiançais, François, fronçais, franc ais, franc-ais`
	
	`> echo francias | aspell -a --sug-mode=ultra --lang=fr`
	`@(#) International Ispell Version 3.1.20 (but really Aspell 0.60.8)`
	`& francias 5 0: francisa, francisas, français, franciens, francien`

Aspell Windows @ EmacsWiki	In Setup for 64-bit Windows 7

Topic & link	Description
GNU Aspell (Win32 version)	
Hunspell	Hunspell is more popular than aspell because it is currently (in 2021) actively maintained and used in several Open Source programs such as LibreOffice, Firefox, Chrome, and several others. Unfortunately it is not as good as aspell in some respect. The two sets of tests in French here show one situation where aspell is better.
	• **Hunspell Home Page** • **Hunspell @ Github** • **Hunspell @ Wikipedia**
Hunspell-compatible dictionary files	• **libreoffice/dictionaries** - **libre-office dictionary wiki** - **git repository** • **French: Grammalecte·dic(fr)** • **Dictionnaires Hunspell 7.0 , Lexique 7.0, Thésaurus et Césures** (téléchargement)
Hunspell files: dictionary and affix files.	The document titled "**Editing the spell checking dictionaries**" from the Chromium Project, describes the format and purpose of the files used by hunspell: • the .dic files: dictionary files: the list of words. • the .aff files: the affix rules files: a list of rules and other options.
Location of Hunspell directories	The **`hunspell -D`** command lists the hunspell directories it is able to find and lists the searched directories. • On my macOS system the directories listed include the following: • /usr/share/hunspell • /usr/share/myspell • /usr/share/myspell/dicts • /Library/Spelling • ~/Library/Spelling • … and several directories for OpenOffice, even though I have LibreOffice and several files are stored inside the ~/Library/Application Support/LibreOffice/… directory tree. I installed several dictionaries using LibreOffice and they are not listed by **`hunspell -D`**. • So I searched for them using the **`fd -g *.aff`** and the **`fd -g *.dic`** commands. • Then I copied the files into my ~/Library/Spelling directory. Now the **`hunspell -D`** command lists the directories available.

Testing hunspell com the command line wit available dictionaries:	Testing in English:
	`> echo htink | hunspell -a -d en_US`
	`@(#) International Ispell Version 3.2.06 (but really Hunspell 1.7.0)`
	`& htink 4 0: think, stink, ht ink, ht-ink`
	
	Test en français:
	`> echo francais | hunspell -a -d fr-classique`
	`@(#) International Ispell Version 3.2.06 (but really Hunspell 1.7.0)`
	`& francais 5 0: français, francisa, franchis, franc ais, franc-ais`
	`>`
	`> echo francias | hunspell -a -d fr-classique`
	`@(#) International Ispell Version 3.2.06 (but really Hunspell 1.7.0)`
	`& francias 5 0: francisa, francisas, franciens, franchisas, francs`

Language Codes	
ISO 639 Language Codes	• **ISO 639-1 @ Wikipedia**. ISO 639-1 : the 2-letter language codes • **ISO 639.2 Language Code List**