

ISPELL

by Pace Willisson and Geoff Kuenning

Version 3.1

December 1993

Copyright © 1988, 1990, 1991, 1992, 1993 Free Software Foundation, Inc.
Published by Geoffrey H. Kuenning
12840 Winthrop Ave.,
Granada Hills, CA 91344-1221 USA

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the Foundation.

1 ISPELL

`IsPELL` is a program that helps you to correct spelling and typographical errors in a file. When presented with a word that is not in the dictionary, `ispell` attempts to find **near misses** that might include the word you meant.

This manual describes how to use `ispell` from within `emacs`. Other information about `ispell` is available from the **Unix** manual pages.

1.1 Using `ispell` from `emacs`

1.1.1 Checking a single word

The simplest `emacs` command for calling `ispell` is `'M-$'` (meta-dollar. On some terminals, you must type `ESC-$`.) This checks the spelling of the word under the cursor. If the word is found in the dictionary, then a message is printed in the echo area. Otherwise, `ISPELL` attempts to generate near misses.

If any near misses are found, they are displayed in a separate window, each preceded by a digit or character. If one of these is the word you wanted, just type its digit or character, and it will replace the original word in your buffer.

If no near miss is right, or if none are displayed, you have five choices:

I

Insert the word in your private dictionary. Use this if you know that the word is spelled correctly.

A

Accept the word for the duration of this editing session, but do not put it in your private dictionary. Use this if you are not sure about the spelling of the word, but you do not want to look it up immediately, or for terms that appear in your document but are not truly words. The next time you start `ispell`, it will have forgotten any accepted words.

SPC

Leave the word alone, and consider it misspelled if it is checked again.

R

Replace the word. This command prompts you for a string in the minibuffer. You may type more than one word, and each word you type is checked again, possibly finding other near misses. This command provides a handy way to close in on a word that you have no idea how to spell. You can keep trying different spellings until you find one that is close enough to get a near miss.

L

Lookup. Display words from the dictionary that contain a specified substring. The substring is a regular expression, which means it can contain special characters to be more selective about which words get displayed. See [section “Reg-exps” in `emacs`](#)

If the only special character in the regular expression is a leading `^`, then a very fast binary search will be used, instead of scanning the whole file.

Only a few matching words can be displayed in the ISPELL window. If you want to see more, use the `look` program directly from the shell.

Of course, you can also type `^G` to stop the command without changing anything.

If you make a change that you don't like, just use emacs' normal undo feature See [section "undo" in emacs](#)

1.1.2 Checking a whole buffer

If you want to check the spelling of all the words in a buffer, type the command `M-x ispell-buffer` . This command scans the file, and makes a list of all the misspelled words. When it is done, it moves the cursor to the first word on the list, and acts like you just typed `M-$` See [Section 1.1.1 \[Word\], page 1](#).

When you finish with one word, the cursor is automatically moved to the next. If you want to stop in the middle of the list type `X` or `^G`

1.1.3 Checking a region

You may check the words in the region with the command `M-x ispell-region` . See [section "mark" in emacs](#)

The commands available are the same as for checking a whole buffer.

1.1.4 Using Multiple Dictionaries

Your site may have multiple dictionaries installed: a default one (usually '`english.hash`'), and several others for different languages (e.g. '`deutsch.hash`') or variations on a language (such as British spelling for English).

`ispell-change-dictionary`

This is the command to change the dictionary. It prompts for a new dictionary name, with completion on the elements of `ispell-dictionary`.

It changes `ispell-dictionary` and kills the old ispell process, if one was running. A new one will be started as soon as necessary.

By just answering `RET` you can find out what the current dictionary is.

`ispell-dictionary`

If non-nil, a dictionary to use instead of the default one. This is passed to the ispell process using the `-d` switch and is used as key in `ispell-dictionary-alist`.

You should set this variable before your first call to ispell (e.g. in your '`.emacs`'), or use the `M-x ispell-change-dictionary` command to change it, as changing this variable only takes effect in a newly started ispell process.

`ispell-dictionary-alist`

An alist of dictionaries and their associated parameters.

Each element of this list is also a list:

```
(dictionary-name
  casechars not-casechars otherchars many-otherchars-p
  ispell-args )
```

dictionary-name is a possible value of variable **ispell-dictionary**, **nil** means the default dictionary.

casechars is a regular expression of valid characters that comprise a word.

not-casechars is the opposite regexp of **casechars**

otherchars is a regular expression of other characters that are valid in word constructs. Otherchars cannot be adjacent to each other in a word, nor can they begin or end a word. This implies we can't check 'Stevens' as a correct possessive and other correct formations.

Hint: regexp syntax requires the hyphen to be declared first here.

many-otherchars-p is non-nil if many otherchars are to be allowed in a word instead of only one.

ispell-args is a list of additional arguments passed to the ispell subprocess.

Note that the **casechars** and **otherchars** slots of the alist should contain the same character set as casechars and otherchars in the **language.aff** file (e.g., 'english.aff').

1.2 Old Emacs

Until ispell becomes part of the standard emacs distribution, you will have to explicitly request that it be loaded. Put the following lines in your emacs init file See [section "init file" in emacs](#)

```
(autoload 'ispell-word "ispell" "Check the spelling of word in buffer." t)
(autoload 'ispell-region "ispell" "Check the spelling of region." t)
(autoload 'ispell-buffer "ispell" "Check the spelling of buffer." t)
(global-set-key "\e$" 'ispell-word)
```

(It will do no harm to have these lines in your init file even after ispell is installed by default.)

1.3 Your private dictionary

Whenever ispell is started the file '**.ispell_words**' is read from your home directory (if it exists). This file contains a list of words, one per line. The order of the words is not important, but the case is. Ispell will consider all of the words good, and will use them as possible near misses.

The **I** command adds words to '**.ispell_words**', so normally you don't have to worry about the file. You may want to check it from time to time to make sure you have not accidentally inserted a misspelled word.

1.4 All commands in emacs mode

DIGIT	Select a near miss
I	Insert into private dictionary
A	Accept for this session
SPC	Skip this time

R	Replace with one or more words
L	Lookup: search the dictionary using a regular expression
M-\$	Check word
M-x ispell-buffer	Check buffer
M-x ispell-region	Check region
M-x ispell-change-dictionary	Select different dictionary.

1.5 Definition of a near miss

Two words are near each other if they can be made identical with one of the following changes to one of the words:

```

Insert a blank space
Interchange two adjacent letters.
Change one letter.
Delete one letter.
Add one letter.

```

Someday, perhaps ispell will be extended so that words that sound alike would also be considered near misses. If you would like to implement this, see Knuth, Volume 3, page 392 for a description of the Soundex algorithm which might apply.

1.6 Where it came from

Ispell has a long and convoluted history. Originally called SPELL, it was written by Ralph E. Gorin in 1971. That version was written in assembly language for the DEC PDP-10 to run under the WAITS operating system at the Stanford Artificial Intelligence Laboratory. Subsequent versions, also in PDP-10 assembly language, were developed for the BBN TENEX, MIT ITS, and DEC TOPS-10 and TOPS-20 operating systems. It was later revised by W. E. Matson (1974), and W. B. Ackerman (1978), changing its name to ISPELL in the process.

In 1983, Pace Willisson (pace@ai.mit.edu) converted this version to the C language and modified it to work under Unix.

In 1987, Walt Buehring revised and enhanced ispell, and posted it to the Usenet along with a dictionary. In addition, Walt wrote the first version of "ispell.el", the emacs interface.

Geoff Kuenning (geoff@ITcorp.com, that's me, and by the way I pronounce it "Kenning") picked up this version, fixed many bugs, and added further enhancements. In 1988 I got ambitious and rewrote major portions of the code, resulting in the table-driven multi-lingual version. Ken Stevens (stevens@hplabs.hp.com) made overwhelming contributions to the elisp support to produce the version you are using now.

Due to a misunderstanding involving the Free Software Foundation, it later became necessary to rename this version to ispell to avoid confusion on the part of users.

Many other enhancements and bug fixes were provided by other people. Although I omit mention here due to space, many of these people have also made significant contributions to the version of ispell you see here. For a full list of people who have contributed to ispell, refer to the file ‘**Contributors**’ which is distributed with the ispell sources.

Geoff Kuenning
geoff@ITcorp.com

Table of Contents

1	ISPELL	1
1.1	Using ispell from emacs	1
1.1.1	Checking a single word	1
1.1.2	Checking a whole buffer	2
1.1.3	Checking a region	2
1.1.4	Using Multiple Dictionaries	2
1.2	Old Emacs	3
1.3	Your private dictionary	3
1.4	All commands in emacs mode	3
1.5	Definition of a near miss	4
1.6	Where it came from	4

